HITACHI
Inspire the Next

ノンストップデータベース

HiRDB Version 10 SQL リファレンス

文法書

3020-6-561-90

前書き

■ 対象製品

●適用 OS：AIX V7.2，AIX 7.3

P-1M62-35A1 HiRDB Server Version 10 10-09

P-1M62-1BA1 HiRDB/Run Time Version 10 10-09

P-1M62-1CA1 HiRDB/Developer's Kit Version 10 10-09

P-1M62-1DA1 HiRDB/Run Time Version 10(64) 10-09

P-1M62-1EA1 HiRDB/Developer's Kit Version 10(64) 10-09

P-F1M62-11A13 HiRDB Staticizer Option Version 10 10-00

P-F1M62-11A15 HiRDB Non Recover Front End Server Version 10 10-00

P-F1M62-11A16 HiRDB Advanced High Availability Version 10 10-00

P-F1M62-11A18 HiRDB Disaster Recovery Light Edition Version 10 10-00

P-F1M62-11A1A HiRDB Accelerator Version 10 10-00

●適用 OS：Red Hat Enterprise Linux 8 (64-bit x86_64)，Red Hat Enterprise Linux 9 (64-bit x86_64)

P-8462-35A1 HiRDB Server Version 10 10-09

P-8462-1DA1 HiRDB/Run Time Version 10(64) 10-09

P-8462-1EA1 HiRDB/Developer's Kit Version 10(64) 10-09

P-F8462-11A13 HiRDB Staticizer Option Version 10 10-00

P-F8462-11A15 HiRDB Non Recover Front End Server Version 10 10-00

P-F8462-11A16 HiRDB Advanced High Availability Version 10 10-00

P-F8462-11A18 HiRDB Disaster Recovery Light Edition Version 10 10-00

P-F8462-11A1A HiRDB Accelerator Version 10 10-00

●適用 OS：Red Hat Enterprise Linux 8 (64-bit x86_64)，Red Hat Enterprise Linux 9 (64-bit x86_64)

P-8362-1BA1 HiRDB/Run Time Version 10 10-09

P-8362-1CA1 HiRDB/Developer's Kit Version 10 10-09

P-8362-3CA1 HiRDB Developer's Suite Version 10 10-09

●適用 OS：Windows Server 2019，Windows Server 2022，Windows 10 Pro (x64)，Windows 10 Enterprise (x64)，Windows 11

P-2962-91A4 HiRDB Server Version 10 10-09

P-2962-7PA4 HiRDB Accelerator Version 10 10-00

P-2962-7HA4 HiRDB Non Recover Front End Server Version 10 10-00
P-2962-7JA4 HiRDB Advanced High Availability Version 10 10-00

●適用 OS：Windows Server 2019，Windows Server 2022，Windows 10，Windows 11
P-2662-11A4 HiRDB/Run Time Version 10 10-09
P-2662-12A4 HiRDB/Developer's Kit Version 10 10-09
P-2662-32A4 HiRDB Developer's Suite Version 10 10-09

●適用 OS：Windows Server 2019，Windows Server 2022，Windows 10 Home (x64)，Windows 10 Pro (x64)，Windows 10 Enterprise (x64)，Windows 11
P-2962-11A4 HiRDB/Run Time Version 10(64) 10-09
P-2962-12A4 HiRDB/Developer's Kit Version 10(64) 10-09

これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。
なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

記載の会社名、製品名などは、それぞれの会社の商標もしくは登録商標です。

■ 発行

2025 年 1 月　3020-6-561-90

■ 著作権

変更内容

変更内容(3020-6-561-90)　HiRDB Version 10 10-09

追加・変更内容	関連する SQL	変更個所
HiRDB 用 Tableau コネクタを使用することで，Tableau から HiRDB に接続できるようにしました。 これによって，HiRDB に格納されているデータを Tableau で分析及び可視化できます。	XMLAGG 集合関数	表 2-25，表 2-26，2.14.1(5)
データベースの内容を複製した参照専用のデータベースを構築できるようにしました。 これによって，アプリケーション側で更新用 DB と参照用 DB（リードレプリカ）を使い分け，負荷分散及びシステム全体のスループット向上につながります。	定義系 SQL，操作系 SQL	3.1.1(3)，4.1.1(2)，付録 E
SQL の予約語に次の予約語を追加しました。 • CURRENT_ROLE	予約語	表 A-3，表 A-29

単なる誤字・脱字などはお断りなく訂正しました。

変更内容(3020-6-561-80)　HiRDB Version 10 10-08

追加・変更内容	関連する SQL
PURGE TABLE 文実行時にアクセスする RD エリア名称を SQL 文中に指定できるようにしました。 これによって，UAP から発行する SQL 文で RD エリア単位の一括削除ができます。	PURGE TABLE 文

変更内容(3020-6-561-70)　HiRDB Version 10 10-07

追加・変更内容	関連する SQL
HiRDB の適用 OS に次の OS を追加しました。 • AIX V7.3 • Red Hat Enterprise Linux 9	−

変更内容(3020-6-561-60)　HiRDB Version 10 10-06

追加・変更内容	関連する SQL
HP-UX に関する説明を削除しました。	−

はじめに

このマニュアルは，プログラムプロダクト　ノンストップデータベース　HiRDB Version 10 のデータベース操作に使用する，SQL の文法について説明したものです。なお，ここに記載されていない前提情報については，マニュアル「HiRDB Version 10 解説」を参照してください。

■ 対象読者

HiRDB Version 10（以降，HiRDB と表記します）で表を設計，作成する方，および UAP を作成，実行する方を対象にしています。

このマニュアルは次に示す知識があることを前提に説明しています。

- Windows のシステム管理の基礎的な知識（Windows 版の場合）
- UNIX または Linux のシステム管理の基礎的な知識（UNIX 版の場合）
- SQL の基礎的な知識
- C 言語のプログラミング，COBOL 言語のプログラミング，または Java のプログラミングの知識

なお，このマニュアルは次に示すマニュアルを前提としていますので，あらかじめお読みいただくことをお勧めします。

- 「HiRDB Version 10 システム導入・設計ガイド」
- 「HiRDB Version 10 UAP 開発ガイド」

■ パス名の表記

- パス名の区切りは「¥」で表記しています。UNIX 版 HiRDB を使用している場合はマニュアル中の「¥」を「/」に置き換えてください。ただし，Windows 版と UNIX 版でパス名が異なる場合は，それぞれのパス名を表記しています。
- HiRDB 運用ディレクトリのパスを%PDDIR%と表記します。ただし，Windows 版と UNIX 版でパス名が異なるため，それぞれを表記する場合，UNIX 版は$PDDIR と表記します。例を次に示します。

  Windows 版：%PDDIR%¥CLIENT¥UTL¥

  UNIX 版：$PDDIR/client/lib/

- Windows のインストールディレクトリのパスを%windir%と表記します。

■ 図中で使用している記号

このマニュアルの図中で使用している記号を，次のように定義します。

●データの流れ　　●制御の流れ

■ このマニュアルで使用している記号

形式および説明で使用している記号を次に示します。ここで説明する文法記述記号は，説明のための記号なので実際には記述しないでください。

記　号	意　味	例
{　}	この記号で囲まれた複数の項目のうちから一つを選択することを示します。	{：埋込み変数｜？パラメタ} 埋込み変数，または？パラメタのどちらかを選択して記述します。
〔　〕	この記号で囲まれた項目は省略できることを意味します。 複数の項目が並べて記述されている場合は，すべてを省略するか，記号｛　｝と同じくどれか一つを選択します。	〔{ALL ｜ DISTINCT}〕 すべてを省略するか，ALL，またはDISTINCTのどちらかを選択して指定します。すべてを省略した場合は，ALLを指定したときと同じ処置をします。
__（下線）	記号〔　〕で囲まれた複数項目のうち1項目に対して使用し，括弧内のすべての項目を省略したときシステムがとる標準値を示します。	
…	この記号の直前に示された項目を繰り返し複数個指定できることを示します。	（列名〔，列名〕…） 列名を繰り返し複数個指定できます。そのとき，列名の前と後ろを記号（　）で囲みます。
（　）	記号（　）で囲まれた項目は，（　）を省略しないでそのまま記述することを示します。	
△	一つの空白を示します。	＊DC△
▲	一つ以上の区切り文字を示します。	WHERE▲GNO ＝ 1
::=	::=の左にあるものを右にあるもので定義することを示します。	表名::=〔認可識別子.〕表識別子

■ HiRDB のデータベース言語の出典

このマニュアルで記述するHiRDBのデータベース言語仕様は，次に示す規格を基に日立製作所独自の解釈と仕様を追加したものです。原開発者に謝意を表するとともに，仕様の出典を示します。

- JIS X 3005 規格群　データベース言語 SQL
- ISO/IEC 9075　Information technology - Database languages - SQL -

注

JIS：日本工業規格（Japanese Industrial Standard）

ISO：国際標準化機構（International Organization for Standardization）

IEC：国際電気標準会議（International Electrotechnical Commission）

目次

1 基本項目

この章では，SQL を使用する場合の基本事項について説明します。

1.1 SQL の記述形式

1.1.1 オペランドの指定順序

オペランドは，各 SQL の形式で記述している順序に従って指定します。

1.1.2 キーワードの指定

SQL 文の名称（SELECT，UPDATE など）のように，機能を使用するために指定する語をキーワードといいます。ほとんどのキーワードは，システムの予約語として登録されていますので，ユーザは決められた位置以外に指定できません。

ただし，予約語として登録されていないキーワードは，名前として使用できます。

キーワードの例を次に示します。予約語については，「予約語一覧」を参照してください。

```
CREATE      TABLE      ID.  T1  ( C1  CHAR,     C2      INTEGER )
キーワード  キーワード                 キーワード       キーワード
```

1.1.3 数値の指定

SQL 中に記述する数定数以外の数値は，符号なし整数の表記法，及び制限に従って指定します。指定方法については，「定数の表記法」又は「数定数の使用上の制限」を参照してください。

数定数以外の数値には，次に示す数値があります。

- ソート項目指定番号（ORDER BY 句のソート項目指定番号）
- 長さ及び最大長（文字データの長さ，及び最大長）
- 最大リソース数（プログラムで使用する表，及びインデクスの数の合計）
- 未使用領域の比率（表，及びインデクス定義の未使用領域の比率）
- 精度（10 進数データのけた数）
- 位取り（10 進数データの小数点以下のけた数）
- データ保証レベル（ALTER PROCEDURE，ALTER ROUTINE，ALTER TRIGGER，CREATE PROCEDURE，CREATE TRIGGER，及び CREATE TYPE）
- 添字（繰返し列の添字）
- 最大要素数（繰返し列の最大要素数）

- 小数秒精度（時刻印データの小数点以下のけた数）

1.1.4　区切り文字の挿入

区切り文字には，次の文字を指定できます。

- 空白（X'20'）
- TAB（X'09'）
- NL（X'0a'）
- CR（X'0d'）
- 全角空白
- 注釈（コメント）

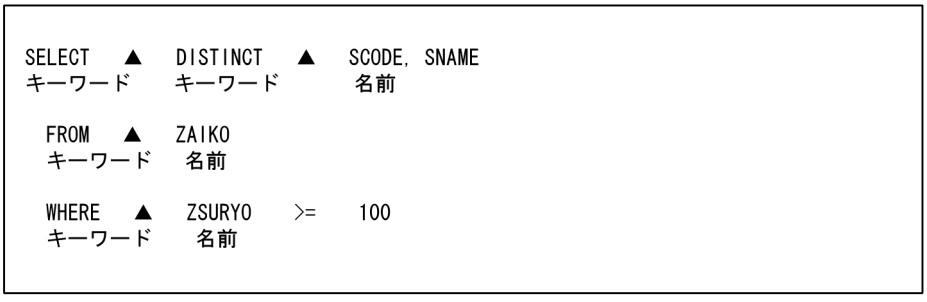
(1)　区切り文字を挿入する位置

区切り文字を挿入する位置を次に示します。

- キーワードとキーワードの間
- キーワードと名前の間
- 名前と名前の間
- キーワードと数値の間
- 名前と数値の間

区切り文字の挿入例を次の図に示します。

図 1-1　区切り文字の挿入例



(2)　区切り文字を挿入してはいけない位置

区切り文字を挿入してはいけない位置を次に示します。

- キーワードの中
- 引用符（"）で囲まれていない名前の中

- 名前を囲む初めの引用符（"）の後ろ
- 名前を囲む終わりの引用符（"）の前
- 数定数の中
- 演算子の中
- 各国文字列定数を表す「N'…'」の「N」と初めの「'」の間
- 混在文字列定数を表す「M'…'」の「M」と初めの「'」の間
- 16 進文字列定数を表す「X'…'」の「X」と初めの「'」の間
- コンポネント指定で「..」と書く場合，ピリオドの間

区切り文字を挿入してはいけない位置の例を次の図に示します。

図 1-2　区切り文字を挿入してはいけない位置の例

```
S▲ELECT      678▲9    ″ZAIKO▲″    ″▲ZAIKO″    <▲=
キーワード    数定数      名前                    演算子
```

(3)　区切り文字を挿入してもよい位置

区切り文字を挿入してもよい位置を次に示します。

- 次に示す特殊記号の前後で，かつ（2）で挿入を禁止されていない位置
  「,」「.」「−」「+」「*」「’」「"」「(」「)」「<」「>」「=」「^」「!」「／」「?」「:」「;」「|」「[」「]」「TAB」「NL」「CR」「空白」「全角空白」

区切り文字を挿入してもよい位置の例を次の図に示します。

図 1-3　区切り文字を挿入してもよい位置の例

```
SCORE▲, SNAME      （特殊記号「, 」の前に空白を挿入）
SCORE = ▲1         （特殊記号「=」の後ろに空白を挿入）
```

(4)　注釈（コメント）

注釈には，囲み注釈と単純注釈があります。それぞれの注釈について説明します。

囲み注釈

SQL 文中の「/*」から，それ以降に現れる最初の「*/」までのすべての文字を注釈とみなします。

単純注釈

システム共通定義 pd_sql_simple_comment_use オペランドが Y の場合に使用できます。SQL 文中の「--」から，それ以降に現れる最初の改行コード（0x0a）までのすべての文字を注釈とみなします。

注釈を挿入する場合の注意事項を次に示します。

- 引用符（"）やアポストロフィ（'）で囲まれた「/*～*/」，及び「--～改行コード」は注釈とみなされません。
- 注釈だけの SQL 文を実行すると，エラーになります。
- 注釈は入れ子にできません。
- 「/*>>～<<*/」の形式の場合，SQL 最適化指定とみなされます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

注釈の指定例を次に示します。

（正しい例）

```
CREATE TABLE T1(C1 INT) /* COMMENT */

CREATE /* COMMENT */ TABLE T1(C1 INT)
```

（誤った例）

```
SELECT * FROM T1 /* COMMENT
…終了記号（*/）がないため，エラーとなります。

CREATE TABLE T1 /* COMMENT1 /* COMMENT2 */ COMMENT3 */(C1 INT)
…注釈の入れ子はできないため，エラーとなります。
```

1.1.5 SQL で使用できる文字

SQL で使用できる文字を次の表に示します。

表 1-1 SQL で使用できる文字

種 別	SQL で使用できる文字
文字列定数	半角文字コード（X'00'を除く）
各国文字列定数	全角文字コードのすべての文字
混在文字列定数	半角文字コード（X'00'を除く），及び全角文字コードのすべての文字
上記以外	• 次に示す半角文字コードの文字 英大文字（A～Z，¥，@，#） 英小文字（a～z） 数字（0～9） 空白

種　別	SQL で使用できる文字
	下線文字（__） 片仮名文字 • 全角文字コードのすべての文字 • 次に示す特殊記号（半角文字コード） コンマ（,） ピリオド（.） ハイフン又は負符号（−） 正符号（＋） アスタリスク（＊） アポストロフィ（'） 引用符（"） 左括弧（(） 右括弧（)） 小なり演算子（＜） 大なり演算子（＞） 等号演算子（＝） サーカムフレックス（^） 感嘆符（！） 斜線（／） 疑問符（？） コロン（：） セミコロン（；） パーセント（%） 垂直棒（｜） 左角括弧（[） 右角括弧（]） TAB（X'09'） NL（X'0a'） CR（X'0d'）

SQL で使用できる文字は，pdsetup コマンドで指定した文字コード種別によって異なります。pdsetup コマンドについては，マニュアル「HiRDB コマンドリファレンス」を参照してください。

使用できる文字には，半角文字と全角文字があって，それぞれ使用できるコードが異なります（UNIX 版の場合，単一バイト文字コードには全角文字はありません）。指定した文字コード種別と使用する文字の関係は，次のとおりです。

指定した文字コード		半角文字	全角文字	備　考
複数バイト文字コード	sjis※3 （シフト jis 漢字）	JISX0201	JISX0208	全角文字に外字を含みます。
	ujis※2 （EUC 日本語漢字）	JISX0201	JISX0208	全角文字に外字を含みません。※1

指定した文字コード		半角文字	全角文字	備　考
	chinese （EUC 中国語漢字）	ISO-8859-1 （80～FF を除きます）	GB2312-80	全角文字に外字は含みません。※1
	utf-8※3※4※6 （Unicode(UTF-8)）	JISX0221	JISX0221	全角文字に外字を含みます。ASCII コードの範囲では，ほかの文字コードと差異はありませんが，1 文字が 6 バイト※5 となることがあります。
		MS-Unicode	MS-Unicode	
	utf-8_ivs※3※4※6 （Unicode（IVS 対応 UTF-8））	JISX0221	JISX0221	全角文字に外字を含みます。ASCII コードの範囲では，ほかの文字コードと差異はありませんが，1 文字が 10 バイト※5 となることがあります。
		MS-Unicode	MS-Unicode （異体字を含む）	
	chinese-gb18030 （中国語漢字 GB18030）	ISO-8859-1 （80～FF を除きます）	GB18030-2000	全角文字に外字を含みます。ASCII コードの範囲では，ほかの文字コードと差異はありませんが，1 文字が 4 バイトになることがあります。
単一バイト 文字コード	lang-c※2 （8 ビットコード）	各コードに従います。	－	US ASCII，及び 8 ビットコードの場合に使用できます。

（凡例）

－：該当する内容はありません。

注※1

EUC コードセット 3（$(8F)_{16}$（xxxx）$_{16}$ の 3 バイトで表現される文字コード）に割り当てられた外字コードは使用できません。

注※2

Windows 版の場合は使用できません。

注※3

Java の UAP と HiRDB，又は HiRDB と Java ルーチンの間で，日本語データを String クラス及びそれを継承したクラスを介して受け渡す場合，Java の文字コードのマッピング規則（該当する文字コードと Unicode）に従います。このとき，外字コードが正しく変換されないことがあるため，注意してください。

注※4

HiRDB では，UTF-8 のエンコードルールだけを意識し，コードと文字のマッピングについては意識していません。このため，UTF-8 のエンコードルールに従った文字を使用できます。ただし，文字コード変換では文字セットとエンコードルールの関係を意識する必要があるため，HiRDB クライアントの

文字コードが SJIS で HiRDB サーバの文字コードが UTF-8 の場合，クライアント環境定義の PDCLTCNVMODE を指定するときは，JISX0221，又は MS-Unicode のどちらを使用しているか意識する必要があります。PDCLTCNVMODE については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注※5

4 バイト以上の文字を使用する場合は，システム定義の pd_substr_length オペランド及びクライアント環境定義 PDSUBSTRLEN の設定が必要な場合があります。pd_substr_length オペランドについては，マニュアル「HiRDB システム定義」を参照してください。PDSUBSTRLEN については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

なお，ISO/IEC 10646 では，文字が割り当てられているのは 1 文字 1～4 バイト（utf-8_ivs の場合は，1～4，7 及び 8 バイト）の範囲です。5 及び 6 バイト（utf-8_ivs の場合は，5，6，9 及び 10 バイト）の範囲は将来の規格のために予約されており，文字が割り当てられていません。したがって，文字が割り当てられていない 1 文字 5，6 バイト（utf-8_ivs の場合は，5，6，9 及び 10 バイト）の範囲を使用した場合は，将来発生するかもしれない問題について保証できません。

注※6

utf-8 と utf-8_ivs では，字形の種類を表すコード（VS（Variation Selector）：字形選択子）の扱いが異なります。utf-8 の場合，基底文字と VS はそれぞれ別の一つの文字として扱い，utf-8_ivs の場合，基底文字と VS を合わせて 1 文字として扱います。そのため，utf-8 の場合は 1 文字の長さが 1～6 バイトになりますが，utf-8_ivs の場合は 1 文字の長さが 1～10 バイトになります。

1.1.6 SQL の最大長

一つの SQL 文は，2,000,000 バイトまで記述できます。

1.1.7 名前の指定

名前の指定には，引用符（"）で囲んで指定する方法と，引用符（"）で囲まないで指定する方法があります。

参考

名前を指定する場合，引用符（"）で囲んで指定することを推奨します。なお，名前を引用符（"）で囲んだ場合，半角英小文字，及び半角英大文字は区別して扱います。

理由

名前には，予約語と同じ名前を指定できませんが，引用符（"）で囲んだ場合は，予約語と同じ名前を指定できます。SQL の拡張に伴って，システムに登録する予約語を追加する場合がありますので，名前はあらかじめ引用符（"）で囲んで指定しておくと，追加した予約語と重複する問題が発生しません。

ただし，手続き中又は関数中に指定するカーソル名以外で，次に示す名前を UAP で使用する場合は，引用符（"）で囲まないで指定してください。

- カーソル名
- SQL 文識別子
- 埋込み変数名，標識変数名，ホスト識別子

(1) 共通規則

- 名前に使用できる文字と長さ（バイト数）の制限については，表「名前に使用できる文字と制限」を参照してください。
- 名前に使用する文字は，半角と全角を混在させて使用できます。
- 引用符（"）で囲まないで指定した場合，半角英小文字は，半角英大文字として扱います。
- 引用符（"）で囲んで指定した場合，半角英小文字，及び半角英大文字を区別して扱います。
- 名前の先頭の文字は，半角英大文字，半角英小文字，又は半角片仮名大文字（ア～ン，ヲ），又は全角文字で始めてください。
- 全角空白を含むことはできません。
- 以下の文字を含む場合は，引用符（"）で囲んで指定してください。
  - 半角空白
  - 半角ハイフン

    ただし，半角ハイフンを含むカーソル名は，手続き又は関数の中に指定する場合だけ引用符（"）で囲んでください。

表 1-2 名前に使用できる文字と制限

名前の種類	制限バイト数，又は制限文字数	半角文字の使用可否						全角文字の使用可否
		英大文字 数字	英小文字	片仮名文字	下線（_）	空白	ハイフン（−）	
インデクス型識別子	30 バイト	○	○	○※2	○	○	○	○※2
インデクス識別子		○	○	○※2	○	○	○	○※2
埋込み変数名	21 文字，22 文字，30 文字，31 文字，63 文字 ※1	○※4	○	○※4	○	×	○※4	○※4
カーソル名	30 バイト	○	○	○※2	○	×	○	○※2
外部ルーチン名	255 バイト	○	○	○※2	○	×	○	○※2
条件名	30 バイト	○	○	○※2	○	○	○	○※2

名前の種類	制限バイト数，又は制限文字数	半角文字の使用可否						全角文字の使用可否
		英大文字 数字	英小文字	片仮名文字	下線（_）	空白	ハイフン（－）	
相関名		○	○	○※2	○	○	○	○※2
属性名		○	○	○※2	○	○	○	○※2
データ型識別子		○	○	○※2	○	○	○	○※2
問合せ名		○	○	○※2	○	○	○	○※2
トリガ識別子		○	○	○※2	○	○	○	○※2
認可識別子	30 バイト※5	○	○	×	×	×	×	×
パスワード※3	30 バイト	○	○	×	×	×	×	×
表識別子		○	○	○※2	○	○	○	○※2
標識変数名	30 文字，31 文字，63 文字※1	○※4	○	○※4	○	×	○※4	○※4
文ラベル	30 バイト	○	○	○※2	○	○	○	○※2
ループ変数名		○	○	○※2	○	○	○	○※2
ホスト識別子	30 文字，31 文字，63 文字※1	○※4	○	○※4	○	×	○※4	○※4
リスト名	30 バイト	○	○	○※2	○	○	○	○※2
ルーチン識別子		○	○	○※2	○	○	○	○※2
列名		○	○	○※2	○	○	○	○※2
RD エリア名		○	○	×	○	○	○	×
SQL パラメタ名		○	○	○※2	○	○	○	○※2
SQL 文識別子		○	○	○※2	○	×	○	○※2
SQL 変数名		○	○	○※2	○	○	○	○※2
制約名		○	○	○※2	○	○	○	○※2
XQuery 変数識別子		○	○	○※2	○	○	○	○※2
順序数生成子識別子		○	○	○※2	○	○	○	○※2
接続制約名		○	○	○※2	○	○	○	○※2

（凡例）

○：使用できます。

×：使用できません。

注※1

埋込み変数名，標識変数名，及びホスト識別子の長さは次のとおりです。

名前の種類		ホスト言語		
		COBOL85	COBOL2002	C 言語
埋込み変数名	BLOB 型	21 文字まで	22 文字まで	63 文字まで
	BLOB 型以外	30 文字まで	31 文字まで	
標識変数名，及びホスト識別子				

COBOL85 及び COBOL2002 の場合，全角文字は 1 文字としてカウントします。C 言語の場合，ユニバーサル文字名は 1 文字としてカウントします。

注※2

使用する文字コードによって制限されます。pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，utf-8_ivs，又は chinese-gb18030 を指定した場合は使用しないでください。

注※3

パスワードについての規則を次に示します。

- 先頭は英字にしてください。
- 数字だけのパスワードは指定できません。
- 「単一文字種の指定禁止」を使用した場合，英大文字だけのパスワードは指定できません。また，英小文字だけのパスワードも指定できません。

注※4

- COBOL85 及び COBOL2002 の場合

  埋込み変数名，標識変数名，及びホスト識別子には，a～z，A～Z，0～9，下線(_)，ハイフン(－)，半角片仮名，及び全角文字が使用できます。

  全角文字と半角文字を混在させて使用できます。

- C 言語の場合

  埋込み変数名，標識変数名，及びホスト識別子には，a～z，A～Z，0～9，及び下線（_）が使用できます。

  コンパイラがユニバーサル文字名（¥uxxxx 及び¥Uxxxxxxxx）をサポートしていれば，ユニバーサル文字名を使用できます。

  マルチバイト文字は使用できません。

注※5

認可識別子の制限バイト数 30 バイトは HiRDB の仕様です。HiRDB と連携する関連製品では認可識別子の制限バイト数が異なりますので，ご使用の際は各製品の仕様を確認してください。

(2) SQL の予約語と重複したときの対応

名前が SQL の予約語と重複したときの対応方法を次に示します。(a)の方法を推奨します。(b)の方法は，アプリケーションの修正ができない場合など，SQL が修正できないときに使用してください。

(a) SQL を修正する

予約語と重複する名前を引用符（"）で囲むように SQL を修正してください。なお，名前を引用符（"）で囲んだ場合，半角英小文字と半角英大文字は区別して扱われるので注意してください。

(b) SQL 予約語削除機能を使用する

SQL 予約語削除機能とは，SQL の予約語として登録してあるキーワードを予約語から削除する機能です。名前と重複した予約語が SQL 予約語削除機能で削除できる予約語の場合，予約語から削除することで，名前を引用符（"）で囲まなくても使用できるようになります。SQL 予約語削除機能で削除できる予約語については，「SQL 予約語削除機能で削除できる予約語」を参照してください。SQL 予約語削除機能で削除できない予約語の場合は，(a)の方法で SQL を修正してください。

削除した予約語は，名前として使用できるようになりますが，同時にキーワードではなくなるため，削除した予約語を使う SQL の機能が使用できなくなる場合があります。使用できなくなる SQL の機能については，「SQL 予約語削除機能で削除できる予約語」を参照してください。

使用方法

SQL 予約語削除機能で削除する予約語を記述した SQL 予約語削除ファイルは，システム共通定義 pd_delete_reserved_word_file オペランドであらかじめ指定する必要があります。システム共通定義 pd_delete_reserved_word_file オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

また，SQL 予約語削除機能を使用する場合は，クライアント環境変数 PDDELRSVWDFILE によって，使用する SQL 予約語削除ファイルを指定する必要があります。クライアント環境変数 PDDELRSVWDFILE については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注意事項

1. SQL オブジェクトを再作成する SQL を実行する場合，クライアント環境変数 PDDELRSVWDFILE には，定義時に指定した予約語削除ファイルを指定してください。
2. ディクショナリ搬出入ユティリティ（pdexp）で表の搬出入を行う場合，クライアント環境変数 PDDELRSVWDFILE には，定義時に指定した予約語削除ファイルを指定してください。また，搬出元と搬入先で以下の内容を合わせておいてください。
   - %PDDIR%¥conf¥pdrsvwd 下の全ファイル
   - システム共通定義 pd_delete_reserved_word_file オペランドの指定値

1.1.8　名前の修飾

名前の修飾は，ピリオド（.）によって，認可識別子，表識別子などを連結して，認可識別子を明示したり，名前を一意にしたりするときに使用します。

(1)　表名，インデクス名，インデクス型名，ユーザ定義型名，ルーチン名，トリガ名，及び順序数生成子名

それぞれの名前の内容と形式を次に示します。

表名：
　表識別子を認可識別子で修飾したもの

インデクス名：
　インデクス識別子を認可識別子で修飾したもの

インデクス型名：
　インデクス型識別子を認可識別子で修飾したもの

ユーザ定義型名：
　データ型識別子を認可識別子で修飾したもの

ルーチン名：
　ルーチン識別子を認可識別子で修飾したもの

トリガ名：
　トリガ識別子を認可識別子で修飾したもの

順序数生成子名：
　順序数生成子識別子を認可識別子で修飾したもの

表名：：＝〔認可識別子.〕表識別子

インデクス名：：＝〔認可識別子.〕インデクス識別子

インデクス型名：：＝〔認可識別子.〕インデクス型識別子

ユーザ定義型名：：＝〔認可識別子.〕データ型識別子

ルーチン名：：＝〔認可識別子.〕ルーチン識別子

トリガ名：：＝〔認可識別子.〕トリガ識別子

順序数生成子名：：＝〔認可識別子.〕順序数生成子識別子

指定項目	指定内容	規　則
認可識別子	指定する表識別子，インデクス識別子，インデクス型識別子，データ型識別子，	認可識別子を省略※3すると，次のとおりに仮定されます。 • ユティリティの実行時

指定項目	指定内容	規　則
	ルーチン識別子，及びトリガ識別子が自分のものであれば，自分の認可識別子を指定します。他ユーザが所有するものであれば，そのユーザの認可識別子を指定します。ただし，表識別子にパブリックビュー，ルーチン識別子にパブリック手続き，パブリック関数の名前を指定する場合は，PUBLIC を指定します。 なお，PUBLIC の指定は，引用符で囲んだ指定，及び引用符で囲まない指定のどちらでも指定できます。	ユティリティを起動するユーザの認可識別子 • UAP の実行時※1 UAP で認可識別子を省略すると，次の順で仮定されます。 1. プリプロセスで指定した認可識別子 2. CONNECT 文のオペランドで指定した認可識別子 3. CONNECT 文のオペランドで指定がなかった場合は，クライアント環境定義 PDUSER で指定した認可識別子 4. PUBLIC（表識別子，ルーチン識別子※2 を指定した場合）
表識別子	実表又はビュー表の名前を指定します。	
インデクス識別子	インデクスの名前を指定します。	
インデクス型識別子	インデクス型の名前を指定します。	
データ型識別子	ユーザ定義型の名前を指定します。	
ルーチン識別子	手続き又は関数の名前を指定します。	
トリガ識別子	トリガの名前を指定します。	
順序数生成子識別子	順序数生成子の名前を指定します。	

注※1

定義系 SQL，PREPARE 文，又は EXECUTE IMMEDIATE 文で指定した SQL の文字列中で認可識別子を省略した場合，次に示す順位で認可識別子が仮定されます。

1. CONNECT 時の認可識別子
2. クライアントの環境変数に設定された認可識別子
3. UAP 実行ユーザ
4. PUBLIC（定義系 SQL 以外で表識別子，ルーチン識別子を指定した場合）

注※2

ルーチン識別子の場合の認可識別子 MASTER に関しては，「スキーマパス」を参照してください。

注※3

定義系 SQL で他ユーザが所有するものを操作する場合，SQL の文字列中で認可識別子を省略したときは，所有者の認可識別子が仮定されます。

例えば，次の例では，1 の SQL 文は 2 の SQL 文のように認可識別子が仮定されます。

（例）

ユーザ（認可識別子：USER007）のインデクス（I001）を表（T002）に作成します。

```
1.CREATE INDEX USER007.I001 ON T002(C001)
2.CREATE INDEX USER007.I001 ON USER007.T002(C001)
```

(2) 表指定

表指定は，一つの SQL 文中に二つ以上の表を指定する場合に，指定する列，＊，又は ROW がどの表に対応するかを一意にするための修飾子で，表名，又は相関名を指定します。

相関名とは，同じ表同士の結合をしたい場合，及び同じ表を副問合せ中で指定し，その問合せ中で外側の問合せの表の列も参照する場合に，指定するそれらの表の別名として使用します。相関名を指定すると，一つの表を二つの異なる表として使用できます。

表指定：：＝ {〔認可識別子.〕表識別子｜相関名｜問合せ名}

WITH 句を用いた問合せ式の問合せ式本体中の一つの FROM 句に，WITH 句に指定した問合せ名と，その問合せ名と同じ表識別子を指定する場合は，表識別子を認可識別子で明示的に修飾し，更にその問合せ名，及び表識別子に相関名を指定して名前を区別する必要があります。

WITH 句を用いた問合せ式の問合せ本体中の表指定では，名前を認可識別子で修飾した場合，その名前を表識別子として扱い，名前を認可識別子で修飾しない場合，その名前を問合せ名，又は表識別子として扱います。ただし，名前を認可識別子で修飾しない場合の名前の優先順位は，問合せ名，表識別子の順になります。

表指定での修飾例を次に示します。

(例 1)

複数の表（ZAIKO，JUTYU）にある同じ名前の列（SCODE）を参照するために，表名（ZAIKO）で修飾する場合

```
SELECT ZAIKO.SCODE,SNAME,TCODE
    FROM ZAIKO,JUTYU
    WHERE ZAIKO.SCODE = JUTYU.SCODE
```

(例 2)

同じ表（ZAIKO）同士の結合で，相関名（X，Y）を使用して修飾する場合

（商品コード 101M と同じ色の商品を検索します）

```
SELECT X.* FROM ZAIKO X,ZAIKO Y
    WHERE X.COL = Y.COL AND Y.SCODE = '101M'
```

(例 3)

相関名を使用して，長い名称の表名に対する記述を簡略化する場合

（ディクショナリ表から，自分の所有する表（ZAIKO）が格納されている RD エリアの名称を検索します）

```
SELECT X.RDAREA_NAME
    FROM MASTER.SQL_RDAREAS X,
    MASTER.SQL_TABLES Y
    WHERE Y.TABLE_SCHEMA = 'U'
    AND Y.TABLE_NAME = 'ZAIKO'
    AND X.RDAREA_NAME = Y.RDAREA_NAME
```

（例４）

WITH 句を用いた問合せ式で，問合せ式本体中の一つの FROM 句に WITH 句中の問合せ名（ZAIKO）と，その問合せ名と同じ名称の表識別子（ZAIKO）を指定する場合

```
WITH ZAIKO(QC1,QC2) AS (SELECT SCODE,TANKA*ZSURYO FROM ZAIKO)
    SELECT * FROM ZAIKO X,USER1.ZAIKO Y
```

(3) 列指定

次の場合，列名又は繰返し列名を表指定で修飾する必要があります。この修飾された列名又は繰返し列名を列指定といいます。

- 一つの FROM 句に複数の表を指定した検索（二つ以上の表の結合）のときに，検索対象の複数の表に同じ列名がある場合（修飾しないと，指定した列がどの表の列なのか分かりません）
- WITH 句を用いた問合せ式の問合せ式本体中の一つの FROM 句に，複数の問合せ名や表名を指定（二つ以上の表の結合）したときに，検索対象の複数の表，又は WITH 句中の導出問合せ式によって導出された表に同じ列名がある場合（修飾しないと，指定した列がどの表の列なのか分かりません）

  （誤った例）

  選択式に指定した CLM1 は，問合せ名 QRY1 の列名なのか，問合せ名 QRY2 の列名なのかが分からないため，列名を修飾してください。

```
    WITH QRY1(CLM1) AS (SELECT SNAME FROM ZAIKO),
         QRY2(CLM1) AS (SELECT SCODE FROM ZAIKO)
         SELECT CLM1 FROM QRY1,QRY2
```

  （正しい例）

  指定した列は，修飾した列名にする必要があります。

```
    WITH QRY1(CLM1) AS (SELECT SNAME FROM ZAIKO),
         QRY2(CLM1) AS (SELECT SCODE FROM ZAIKO)
         SELECT QRY1.CLM1,QRY2.CLM1 FROM QRY1,QRY2
```

- 副問合せを指定した場合，副問合せ中の WHERE 句，又は HAVING 句中で，その副問合せの外側の FROM 句で指定した表，又は更新する表の列を参照するとき（修飾しないと，外側の表の列を参照できません）

ただし，列名は構文上，修飾できる場合とできない場合があります。各形式中の「列指定」と記述している箇所は，列名に修飾できます。「列名」と記述している箇所は，修飾できません。

列指定：：＝〔表指定.〕 {列名｜繰返し列名〔[添字]〕}

添字：：＝ {整数｜ ANY}

列名を表指定で修飾する場合，次に示す規則があります。

- 列名は，指定する相関名，又は表名の有効範囲内でだけ，その相関名，又は表名を修飾できます。相関名及び表名の有効範囲については，「表参照」，「DELETE 文　形式 1（行削除）」，及び「UPDATE 文　形式 1（データ更新）」を参照してください。
- 副問合せ中に，同じ名前の有効な表指定（相関名，又は表名）を複数指定した場合，その表指定（相関名，又は表名）を参照すると，最も内側にある問合せで指定している表指定（相関名，又は表名）が指定されます。

  同じ名前の有効な表名が複数ある例を次に示します。

  （例）

  副問合せ中では，T1[1] と T1[2] の両方の指定が有効です。T1 を参照すると，最も内側にある問合せの T1[2] が適用されます。

```
SELECT * FROM T1[1]
    WHERE T1[1].C2 >=
      ( SELECT AVG(C2) FROM T1[2]
        WHERE T1[2].C1 = 1       )
```

  T1[1] の有効範囲：SELECT ＊ FROM T1[1] 以降

  T1[2] の有効範囲：(SELECT AVG（C2） FROM T1[2] 以降

  なお，最も局所的な有効範囲を持つ表が複数ある（一つの FROM 句に同一の表名が複数ある）場合，相関名で修飾する必要があります。

  FROM 句に同一の表名が複数ある例を次に示します。

  （誤った例）

  副問合せの範囲は最も局所的な有効範囲を持つ表 T1[1] と T1[2] がある場合，この指定では誤りになるため，相関名を指定してください。

```
SELECT * FROM T2
    WHERE C3 IN
      ( SELECT T1[1].C3 FROM T1[1],T1[2]
        WHERE T1[1].C1 = T1[2].C3     )
```

  最も局所的な有効範囲：(SELECT T1[1].C3 FROM T1[1],T1[2] 以降

  （正しい例）

  指定した列は，修飾子で指定した表になければなりません。

```
SELECT * FROM T2
    WHERE C3 IN
      ( SELECT X.C3 FROM T1[1] X,T1[2] Y
        WHERE X.C1 = Y.C2           )
```

- 列名を表指定で修飾できますが，修飾しない場合，最も局所的な有効範囲を持つ中に，指定した列を含む表を示すものが一つだけ必要です。

1.1.9　スキーマパス

スキーマパスとは，認可識別子で明示的に修飾していないルーチン名，インデクス型名，ユーザ定義型名のスキーマの探索順序を決定するものです。

インデクス型名，ユーザ定義型名の場合は，対象となるスキーマにインデクス型名，ユーザ定義型名がないときに，次のスキーマを探索します。

関数名の場合は，呼び出す関数の決定規則に従って候補となる関数がないときに，次のスキーマを探索します。呼び出す関数の決定規則については，「関数呼出し」を参照してください。

(1)　探索順序

探索するスキーマは，次の順序で仮定されます。

1. 省略時に仮定する認可識別子のユーザのスキーマ
   省略時に仮定する認可識別子については，「名前の修飾」を参照してください。
2. MASTER のスキーマ

(2)　スキーマパスの適用範囲

認可識別子で修飾しない指定で，定義済みのルーチン名，インデクス型名，ユーザ定義型名の参照時に適用します。

1.2　データ型

1.2.1　データ型

データ型は次の二つに分けられます。

- 既定義型
- ユーザ定義型

形式

データ型：：＝　{既定義型｜ユーザ定義型}

説明

- **既定義型**
  HiRDB が提供するデータ型を指定します。
- **ユーザ定義型**
  ユーザ定義型は次の形式で指定します。
  〔認可識別子.〕データ型識別子
  認可識別子
  ユーザ定義型の認可識別子を指定します。
  データ型識別子
  ユーザ定義型のデータ型識別子を指定します。

(1)　既定義型

既定義型を次の表に示します。

表 1-3　既定義型

分　類	データ型※1	データ形式	説　明
数データ	INT〔EGER〕	整数 （4 バイトの 2 進形式）	値の範囲が－2147483648～2147483647 の整数です。
	SMALLINT	整数 （2 バイトの 2 進形式）	値の範囲が－32768～32767 の整数です。
	〔LARGE〕 DEC〔IMAL〕 〔(m〔, n〕)〕 又は	固定小数点数 （↑（m ＋ 1）/2↑バイトのパック 10 進形式）※8	精度（全体のけた数）が m けたで，位取り（小数点以下のけた数）が n けたの固定小数点数です。 m，n は正整数で，1≦m≦38，0≦n≦38，n≦m です。

分　類	データ型※1	データ形式	説　明
	NUMERIC 〔(m〔, n〕)〕		m を省略すると，15 が仮定されます。 n を省略すると，0 が仮定されます。
	FLOAT 又は DOUBLE PRECISION	倍精度浮動小数点数 （8 バイト）	値の範囲が約±4.9×10^{-324}～±1.7×10^{308} の倍精度浮動小数点数です。※4
	SMALLFLT 又は REAL	単精度浮動小数点数 （4 バイト）	値の範囲が約±1.4×10^{-45}～±3.4×10^{38} の単精度浮動小数点数です。※4
文字データ※10	CHAR〔ACTER〕 〔(n)〕　〔CHARACTER SET 文字集合指定〕	固定長文字列 （長さ n バイト）	長さが n バイトの固定長文字列です。 • 文字集合指定なしの場合 1 バイト文字から成る固定長文字列です。 • 文字集合指定ありの場合 文字集合の任意の文字から成る固定長文字列です。 n は正整数で 1≦n≦30,000 です。n を省略すると，1 が仮定されます。 文字集合指定は文字データに対する属性を指定します。詳細は「文字集合」を参照してください。
	〔LONG〕 VARCHAR(n) 〔CHARACTER SET 文字集合指定〕又は CHAR〔ACTER〕 VARYING(n) 〔CHARACTER SET 文字集合指定〕	可変長文字列 （最大長 n バイト）	最大長が n バイトの可変長文字列です。 • 文字集合指定なしの場合 1 バイト文字から成る可変長文字列です。 • 文字集合指定ありの場合 文字集合の任意の文字から成る可変長文字列です。 n は正整数で 1≦n≦32,000 です。実長は 0 以上です。 文字集合指定は文字データに対する属性を指定します。詳細は「文字集合」を参照してください。
各国文字データ※6※9	NCHAR〔(n)〕又は NATIONAL CHAR〔ACTER〕〔(n)〕	固定長各国文字列 （長さ n 文字）	長さ n 文字（2n バイト）の 2 バイト文字から成る固定長各国文字列です。 n は正整数で 1≦n≦15,000 です。n を省略すると，1 が仮定されます。
	〔LONG〕 NVARCHAR(n)又は NATIONAL CHAR〔ACTER〕 VARYING(n)又は NCHAR VARYING(n)	可変長各国文字列 （最大長 n 文字）	最大長が n 文字（2n バイト）の 2 バイト文字から成る可変長各国文字列です。 n は正整数で 1≦n≦16,000 です。実長は 0 以上です。
混在文字データ※6	MCHAR〔(n)〕	固定長混在文字列 （長さ n バイト）	長さ n バイトの半角文字と全角文字の混在する固定長混在文字列です。

分　類	データ型※1	データ形式	説　明
			n は正整数で 1≦n≦30,000 です。n を省略すると，1 が仮定されます。
	〔LONG〕 MVARCHAR(n)	可変長混在文字列 （最大長 n バイト）	最大長 n バイトの半角文字と全角文字の混在する可変長混在文字列です。 n は正整数で 1≦n≦32,000 です。実長は 0 以上です。
日付データ	DATE	日付 （4 バイト符号なしパック形式 YYYYMMDD） YYYY：0001～9999（年） MM：01～12（月） DD：01～該当年月の最終日（日）	年，月，日の三つの領域を持つ日付のデータ型です。
時刻データ	TIME	時刻 （3 バイト符号なしパック形式 hhmmss） hh：00～23（時） mm：00～59（分） ss：00～61（秒）※11	時，分，秒の三つの領域を持つ時刻のデータ型です。
時刻印データ	TIMESTAMP〔(p)〕	時刻印 （7～10 バイトの符号なしパック形式 YYYYMMDDhhmmss〔nn…n〕） YYYY：0001～9999（年） MM：01～12（月） DD：01～該当年月の最終日（日） hh：00～23（時） mm：00～59（分） ss：00～61（秒）※11 nn…n：p けたの小数秒（n：0～9）	年，月，日，時，分，秒の六つの領域を持つ時刻印のデータ型です。 p は整数で，p=0，2，4，又は 6 です。省略した場合は，p=0 が仮定されます。
日間隔データ	INTERVAL YEAR TO DAY	日間隔 （5 バイトパック形式 0YYYYMMDDs） YYYY：0000～9999（か年） MM：00～99※2（か月） DD：00～99（か日） s：符号（正：C，F　負：D）※8	日付データ型が，ある一時点を表すのに対して，日間隔データ型は，日付と日付の間隔を表すデータ型です。 日間隔の範囲は，－9999 か年 11 か月 99 か日間～9999 か年 11 か月 99 か日間です。
時間隔データ	INTERVAL HOUR TO SECOND	時間隔 （4 バイトパック形式 0hhmmsst） hh：00～99（時間） mm：00～99※3（分） ss：00～99※3（秒） t：符号（正：C，F　負：D）※8	時刻データ型が，ある一時点を表すのに対して，時間隔データ型は，時刻と時刻の間隔を表すデータ型です。 時間隔の範囲は－99 時間 59 分 59 秒～99 時間 59 分 59 秒です。

分　類	データ型※1	データ形式	説　明
長大データ	BLOB〔(n〔{K ｜ M ｜ G}〕)〕 又は BINARY LARGE OBJECT〔(n〔{K ｜ M ｜ G}〕)〕	バイナリデータ列 （最大長 n バイト） K：キロバイト単位 M：メガバイト単位 G：ギガバイト単位	最大長 n バイトのバイナリデータ列です。 n を省略すると，2,147,483,647 バイトが仮定されます。実長は 0 以上です。単位として K，M，及び G が指定できます。※5 単位（K，M，又は G）を省略すると，バイト単位となります。
バイナリデータ	BINARY(n)	バイナリデータ列 （最大長 n バイト）	最大長 n バイトのバイナリデータ列です。 n は省略できません。実長は 0 以上です。 n は正整数で，1≦n≦2,147,483,647 バイトです。
論理データ	BOOLEAN※7	論理値（4 バイト）	論理値として真（TRUE），偽（FALSE），否定（UNKNOWN）をとります。

注 1

文字データ，及び混在文字データの比較，及びデータ変換時，埋字の空白として使用する半角文字を次に示します。

データ型	文字集合指定	空白文字コード
文字データ	省略	X’20’
	EBCDIK	X’40’
	UTF16	X’0020’※
混在文字データ		X’20’
各国文字データ		使用している文字コードに依存

注※

？パラメタを介して文字コードが UTF-16 のデータを操作する場合，文字集合名記述領域に文字集合名を指定してください。

また，プリプロセスオプション，及び埋込み変数の定義に，文字コードが UTF-16 のデータを操作できるように指定することで，埋込み変数を介して文字コードが UTF-16 のデータを操作できるようになります。

文字集合名に指定できる値を示します。

- UTF16
- UTF-16BE
- UTF-16LE

文字集合名を指定した場合の空白文字コードを次に示します。

- UTF16，及び UTF-16BE の場合
  X’0020’
- UTF-16LE の場合

X’2000’

UTF-16LE 及び UTF-16BE（文字集合名記述領域の SQLCSN に設定できる文字集合情報）については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注 2

使用できる半角文字，及び全角文字の文字コードについては，表「SQL で使用できる文字」の注を参照してください。

注※1

各データ型はこれ以降，それぞれ代表的なデータ型を使用して示します。

注※2

12 以上を指定した場合，年に繰り上げます。

注※3

mm，ss に 60 以上を指定した場合，それぞれ時，分に繰り上げます。

注※4

浮動小数点数の値の範囲は，ハードウェア表現に従います。

注※5

最大長を単位指定する場合の，指定範囲と実際の最大長を次に示します。

単　位	n の指定範囲	実際の最大長（バイト）
K	1≦n≦2097152	n×1024
M	1≦n≦2048	n×1048576
G	1≦n≦2	n×1073741824

ただし，実際の最大長の計算結果が 2147483648 の場合は，2147483647 になります。

注※6

pdsetup コマンドで文字コード種別に lang-c を指定した場合，各国文字データ，及び混在文字データは定義できません（UNIX 版限定）。

注※7

BOOLEAN は，関数の戻り値のデータ型としてだけ使用できます。列，SQL 変数，又は SQL パラメタのデータ型としては使用できません。

注※8

DECIMAL 型，日間隔型，及び時間隔型の符号部については，「DECIMAL 型使用上の注意事項」を参照してください。

注※9

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，utf-8_ivs，又は chinese-gb18030 を指定した場合，各国文字データは定義できません。

注※10

- pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に sjis 以外を指定した場合，文字集合指定に EBCDIK を指定した文字データは使用できません。
- pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，utf-8_ivs 以外を指定した場合，文字集合指定に UTF16 を指定した文字データは使用できません。

  文字集合指定に UTF16 を指定した場合，n は 2 の倍数にしてください。

注※11

pd_leap_second が N の場合，ss の範囲は 00～59（秒）となります。pd_leap_second オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

(2) ユーザ定義型

ユーザ定義型を次の表に示します。

表 1-4 ユーザ定義型

データ型	データ形式	説明
抽象データ型	－	CREATE TYPE で定義するデータ型を示します。データ型中に属性定義，ルーチンなどを定義できます。

（凡例）－：該当しません。

1.2.2 変換（代入，比較）できるデータ型

変換（代入，比較）できるデータ型を次の表に示します。ただし，CAST 指定，及びスカラ関数を使用して変換できるデータ型については，それぞれ「スカラ関数」，「CAST 指定」を参照してください。

表 1-5 変換（代入，比較）できるデータ型（1/2）

<table>
<tr><th rowspan="3">変換前データ型</th><th colspan="8">変換後データ型</th></tr>
<tr><th>数データ</th><th colspan="3">文字データ</th><th>各国文字データ</th><th>混在文字データ</th><th>日付データ</th><th>時刻データ</th><th>時刻印データ</th></tr>
<tr><td>INTEGER</td><td colspan="3" rowspan="2">CHARACTER</td><td rowspan="2">NCHAR</td><td rowspan="2">MCHAR</td><td rowspan="4">DATE</td><td rowspan="4">TIME</td><td rowspan="4">TIMESTAMP</td></tr>
<tr><td>SMALLINT</td></tr>
<tr><td>DECIMAL</td><td colspan="3" rowspan="2">VARCHAR</td><td rowspan="3">NVARCHAR</td><td rowspan="3">MVARCHAR</td></tr>
<tr><td>FLOAT</td></tr>
<tr><td>SMALLFLT</td><td>DF</td><td>EK</td><td>U16</td></tr>
<tr><td>数データ
• INTEGER
• SMALLINT</td><td>○,○</td><td colspan="3">○,×※1</td><td>×,×</td><td>○,×※1</td><td>×,×</td><td>×,×</td><td>×,×</td></tr>
</table>

変換前データ型		変換後データ型								
		数データ	文字データ			各国文字データ	混在文字データ	日付データ	時刻データ	時刻印データ
		INTEGER	CHARACTER			NCHAR	MCHAR	DATE	TIME	TIMESTAMP
		SMALLINT								
		DECIMAL	VARCHAR			NVARCHAR	MVARCHAR			
		FLOAT								
		SMALLFLT	DF	EK	U16					
• DECIMAL • FLOAT • SMALLFLT										
文字データ • CHARACTER • VARCHAR	DF	△,△※2※3	○,○	△,△※14	△,△※17	△,△※7	○,○	△,△※4	△,△※5	△,△※6
	EK		△,×※15	○,○	×,×	×,×	×,×			
	U16		△,△※16	×,×	○,○	×,×	△,△※16			
各国文字データ • NCHAR • NVARCHAR		×,×	×,×			○,○	×,×	×,×	×,×	×,×
混在文字データ • MCHAR • MVARCHAR		△,△※2※3	○,○	×,×	△,△※17	×,×	○,○	×,×	×,×	×,×
日付データ • DATE		×,×	×,△※8			×,×	×,×	○,○	×,×	×,×
時刻データ • TIME		×,×	×,△※9			×,×	×,×	×,×	○,○	×,×
時刻印データ • TIMESTAMP		×,×	×,△※10			×,×	×,×	×,×	×,×	○,○※11
日間隔データ • INTERVAL YEAR TO DAY		×,△※12	×,×			×,×	×,×	×,×	×,×	×,×
時間隔データ • INTERVAL HOUR TO SECOND		×,△※13	×,×			×,×	×,×	×,×	×,×	×,×

変換前データ型	変換後データ型								
	数データ	文字データ			各国文字データ	混在文字データ	日付データ	時刻データ	時刻印データ
	INTEGER	CHARACTER			NCHAR	MCHAR	DATE	TIME	TIMESTAMP
	SMALLINT								
	DECIMAL	VARCHAR			NVARCHAR	MVARCHAR			
	FLOAT								
	SMALLFLT	DF	EK	U16					
長大データ • BLOB	×,×	×,×			×,×	×,×	×,×	×,×	×,×
バイナリデータ • BINARY(n) 1≦n≦32,000	×,×	×,×			×,×	×,×	×,×	×,×	×,×
バイナリデータ • BINARY(n) n > 32,000	×,×	×,×			×,×	×,×	×,×	×,×	×,×
論理データ • BOOLEAN	×,×	×,×			×,×	×,×	×,×	×,×	×,×
抽象データ型	×,×	×,×			×,×	×,×	×,×	×,×	×,×

（凡例）

○：変換できます。

△：制限付きで変換できます。

×：変換できません。

DF：既定文字集合

EK：EBCDIK

U16：UTF16，UTF-16LE 又は UTF-16BE

文字集合名 UTF-16LE，UTF-16BE は，次の場合に指定してください。

- ?パラメタを介して文字コードが UTF-16 のデータを操作する場合
- プリプロセスオプション，及び埋込み変数の定義に，文字コードが UTF-16 のデータを操作できるように指定し，埋込み変数を介して文字コードが UTF-16 のデータを操作する場合

文字集合 UTF-16LE 及び UTF-16BE（文字集合名記述領域の SQLCSN に設定できる文字集合情報）については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注

表中の変換可否は，「代入可否，比較可否」となっています。例えば，「○，○」の場合は代入及び比較ができて，「○，×」の場合は代入はできるが比較はできないということになります。

注※1

数データを文字データに変換する場合の対応関係を次に示します。なお，変換後のデータ型が混在文字データの場合は，VARCHAR を MVARCHAR に，CHAR を MCHAR に読み替えてください。

- INTEGER 型の数データは，VARCHAR(11)の文字データに変換されます。
- SMALLINT 型の数データは，VARCHAR(6)の文字データに変換されます。
- DECIMAL(p,0)型の数データは，VARCHAR(p ＋ 1)の文字データに変換されます。
- DECIMAL(p,s)型の数データは，VARCHAR(p ＋ 2)の文字データに変換されます。
- FLOAT 型又は SMALLFLT 型の数データは，CHAR(23)の文字データに変換されます。

INTEGER 型，SMALLINT 型，及び DECIMAL 型での数データが正の場合，変換された文字データには正の符号は含みません。また，FLOAT 型及び SMALLFLT 型の数データを文字データに変換する場合，指数部と仮数部に必ず符号を付加します。

注※2

文字データと数データを比較する場合，文字データを数データに変換して比較をします。文字データと数データを比較できるのは，比較述語，限定述語，IN 述語，及び BETWEEN 述語です。各述語ごとの比較可能な条件を次に示します。

述語	比較可能な条件
比較述語	左右のどちらかが数データの場合に比較できます。
IN 述語，限定述語	左側が数データの場合に比較できます。
BETWEEN 述語	行値構成子 1 の行値構成子要素が数データの場合に比較できます。

文字データと数データを比較できるのは，以下に示す指定をした場合です。

＜文字データ＞

- 定数（副問合せの選択式に指定した場合を含む）
- ?パラメタ
- 埋込み変数
- SQL 変数
- SQL パラメタ

ただし，文字データ又は数データを，以下に示す位置に指定した場合は，SQL 変数，SQL パラメタで指定した文字データを，数データに変換できません。

- 行値構成子要素数が 2 以上である行値構成子中の行値構成子要素
- 行副問合せ，又は表副問合せの選択式
- 比較述語の右側，IN 述語の右側，限定述語の右側以外に指定したスカラ副問合せの選択式

＜数データ＞

- 列指定（副問合せの選択式に指定した場合を含む）

注※3

文字データを数データに代入，比較する場合の対応関係を次に示します。

- 整数の文字列表現は，INTEGER 型に変換されます。
- 10 進数の文字列表現は，DECIMAL 型に変換されます。
- 浮動小数点数の文字列表現は，FLOAT 型に変換されます。

数の文字列表現の前後に空白がある場合，空白を無視して変換をします。

注※4

次に示すものは，日付データへの代入，及び日付データとの比較ができます。

- 日付を既定の文字列で表現した定数
- CHAR（10）に対応する埋込み変数（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，又は UTF-16BE の場合は CHAR(20)に対応する埋込み変数）

注※5

次に示すものは，時刻データへの代入，及び時刻データとの比較ができます。

- 時刻を既定の文字列で表現した定数
- CHAR（8）に対応する埋込み変数（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，UTF-16BE の場合は CHAR(16)に対応する埋込み変数）

注※6

次に示すものは，時刻印データとの比較，及び時刻印データへの変換ができます。

- 時刻印を既定の文字列表現で指定した定数
- 長さが 19～26 バイトの CHAR に対応する埋込み変数（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，UTF-16BE の場合は長さが 38～52 バイトの CHAR で，長さが 2 の倍数に対応する埋込み変数）

注※7

定義系 SQL を除く，次に示す項目で文字列定数を記述した場合，各国文字列定数とみなし，文字データの長さだけをチェックし，文字コードはチェックしません。

- INSERT 文
- UPDATE 文
- 探索条件

INSERT 文，及び UPDATE 文については，「操作系 SQL」を，探索条件については，「探索条件」を参照してください。

注※8

次に示すものは，日付データと比較できます。

- 日付を既定の文字列で表現した定数

- CHAR（10）に対応する埋込み変数（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，又は UTF-16BE の場合は CHAR(20)に対応する埋込み変数）

また，次の場合は，文字データに変換できます。

- 日付データを，長さが 10 バイト以上の CHAR，又は VARCHAR の埋込み変数に代入した場合（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，又は UTF-16BE の場合は，長さが 20 バイト以上で長さが 2 の倍数の CHAR，又は VARCHAR の埋込み変数に代入した場合）

注※9

次に示すものは，時刻データと比較できます。

- 時刻を既定の文字列で表現した定数
- CHAR（8）に対応する埋込み変数（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，又は UTF-16BE の場合は CHAR(16)に対応する埋込み変数）

また，次の場合は，文字データに変換できます。

- 時刻データを，長さが 8 バイト以上の CHAR，又は VARCHAR の埋込み変数に代入した場合（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，又は UTF-16BE の場合は，長さが 16 バイト以上で長さが 2 の倍数の CHAR，又は VARCHAR の埋込み変数に代入した場合）

注※10

次に示すものは，時刻印データと比較できます。

- 時刻印を既定の文字列表現で指定した定数
- 長さが 19～26 バイトの CHAR に対応する埋込み変数（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，又は UTF-16BE の場合は長さが 38～52 バイトで長さが 2 の倍数の CHAR に対応する埋込み変数）

また，次の場合は文字データに代入できます。

- 変換前の時刻印データの小数秒精度を p とした場合，p=0 の場合は 19 バイト以上，p>0 の場合は 20+p バイト以上の，CHAR 又は VARCHAR の埋込み変数への代入（ただし，埋込み変数の文字集合が UTF16，UTF-16LE，又は UTF-16BE の場合は，変換前の時刻印データの小数秒精度を p とした場合，p=0 の場合は 38 バイト以上，p>0 の場合は 40+2p バイト以上で長さが 2 の倍数の，CHAR 又は VARCHAR の埋込み変数への代入）

注※11

代入元の小数秒精度が代入先の小数秒精度より高い場合，代入先の精度に合わせて小数行部分を切り捨てます。代入元の小数秒精度が代入先の代入先の小数秒精度より低い場合，代入先の精度に合わせて拡張した小数秒部分に 0 を補って格納します。

注※12

次に示すものは，日間隔データと比較できます。

- 日間隔を 10 進数で表現した定数
- DECIMAL（8，0）に対応する埋込み変数

また，次の場合は，10 進数データに変換できます。

- 日間隔データを，DECIMAL（8，0）の埋込み変数に代入した場合

注※13

次に示すものは，時間隔データと比較できます。

- 時間隔を 10 進数で表現した定数
- DECIMAL（6，0）に対応する埋込み変数

また，次の場合は，10 進数データに代入できます。

- 時間隔データを，DECIMAL（6，0）の埋込み変数に代入した場合

注※14

変換前の項目として次に示すものは，代入対象，比較対象の文字集合に変換するため，代入，比較できます。

- 文字列定数
- 埋込み変数（既定文字集合）

注※15

変換後の項目として次に示すものは，代入対象の文字集合に変換するため，代入できます。

- 埋込み変数（既定文字集合）

注※16

変換後の項目として次に示すものは，代入対象，比較対象の文字集合に変換するため，代入，比較できます。

- 埋込み変数（既定文字集合）

変換前の項目として次に示すものは，代入対象，比較対象の既定文字集合に変換するため，代入，比較できます。

- 埋込み変数（UTF16，UTF-16LE，又は UTF-16BE）

注※17

変換前の項目として次に示すものは，代入対象，比較対象の文字集合に変換するため，代入，比較できます。

- 文字列定数
- 埋込み変数（既定文字集合）

変換後の項目として次に示すものは，代入対象の文字集合に変換するため，代入できます。

- 埋込み変数（UTF16，UTF-16LE，又は UTF-16BE）

表 1-6　変換（代入，比較）できるデータ型（2/2）

変換前データ型		変換後データ型						
		日間隔データ	時間隔データ	長大データ	バイナリデータ		論理データ	抽象データ型
		INTERVAL YEAR TO DAY	INTERVAL HOUR TO SECOND	BLOB	BINARY(n) 1≦n≦32,000	BINARY(n) n > 32,000	BOOLEAN	
数データ • INTEGER • SMALLINT • DECIMAL • FLOAT • SMALLFLT		△,△※1	△,△※2	×,×	×,×	×,×	×,×	×,×
文字データ • CHARACTER • VARCHAR	DF	×,×	×,×	×,×	△,△※3	×,×	×,×	×,×
	EK				×,×			
	U16							
各国文字データ • NCHAR • NVARCHAR		×,×	×,×	×,×	×,×	×,×	×,×	×,×
混在文字データ • MCHAR • MVARCHAR		×,×	×,×	×,×	×,×	×,×	×,×	×,×
日付データ • DATE		×,×	×,×	×,×	×,×	×,×	×,×	×,×
時刻データ • TIME		×,×	×,×	×,×	×,×	×,×	×,×	×,×
時刻印データ • TIMESTAMP		×,×	×,×	×,×	×,×	×,×	×,×	×,×
日間隔データ • INTERVAL YEAR TO DAY		○,○※4	×,×	×,×	×,×	×,×	×,×	×,×
時間隔データ • INTERVAL HOUR TO SECOND		×,×	○,○※5	×,×	×,×	×,×	×,×	×,×
長大データ • BLOB		×,×	×,×	○,×	○,×	○,×	×,×	×,×
バイナリデータ • BINARY(n)		×,×	×,×	○,×	○,○	○,×	×,×	×,×

変換前データ型	変換後データ型						
	日間隔データ	時間隔データ	長大データ	バイナリデータ		論理データ	抽象データ型
	INTERVAL YEAR TO DAY	INTERVAL HOUR TO SECOND	BLOB	BINARY(n) 1≦n≦32,000	BINARY(n) n > 32,000	BOOLEAN	
1≦n≦32,000							
バイナリデータ • BINARY(n) n > 32,000	×,×	×,×	○,×	○,×	○,×	×,×	×,×
論理データ • BOOLEAN	×,×	×,×	×,×	×,×	×,×	×,×	×,×
抽象データ型	×,×	×,×	×,×	×,×	×,×	×,×	△,×※6

（凡例）

○：変換できます。

△：制限付きで変換できます。

×：変換できません。

DF：既定文字集合

EK：EBCDIK

U16：UTF16，UTF-16LE 又は UTF-16BE

注

表中の変換可否は，「代入可否，比較可否」となっています。例えば，「○，○」の場合は代入及び比較ができて，「○，×」の場合は代入はできるが比較はできないということになります。

注※1

次に示すものは，日間隔データへの代入，及び日間隔データとの比較ができます。

- 日間隔を 10 進数で表現した定数
- DECIMAL（8，0）に対応する埋込み変数

注※2

次に示すものは，時間隔データへの代入，及び時間隔データとの比較ができます。

- 時間隔を 10 進数で表現した定数
- DECIMAL（6，0）に対応する埋込み変数

注※3

次に示すものは，BINARY 型への代入，及び BINARY 型との比較ができます。

- 16 進文字列定数

注※4

日間隔データ同士の比較は，年，月，日の順で比較します。

注※5

時間隔データ同士の比較は，時，分，秒の順で比較します。

注※6

抽象データ型 ADT2 で定義された列，変数に対して，抽象データ型 ADT1 が ADT2 の値又は ADT2 のサブタイプの値の場合に，代入できます。

1.2.3 文字データ，各国文字データ，及び混在文字データ使用上の注意事項

文字データ，各国文字データ，及び混在文字データを使用する場合の注意事項を次の表に示します。

表 1-7 文字データ，各国文字データ，及び混在文字データ使用上の注意事項

項 目	CHAR，NCHAR，又は MCHAR		VARCHAR，NVARCHAR， 又は MVARCHAR	
	定義長 1～255 (1～127)	定義長 256～30,000 (128～15,000)	定義長 1～255 (1～127)	定義長 256～32,000 (128～16,000)
インデクスの定義	○	△	○	△
ソート	○	○	○	○
グループ分け	○	○	○	○
集合関数	○	○	○	○
探索条件	○	○	○	○
データの挿入・更新	○	○	○	○
重複排除	○	○	○	○
集合演算	○	○	○	○

(凡例)

○：使用できます。

△：使用できます。

インデクスを構成する列の長さの合計は，次の計算式を満たすようにしてください。

列の長さの合計≦MIN（インデクス格納用 RD エリアのページサイズ÷2－1242，4036）

( )：定義長の括弧内の値は，各国文字データの場合を表しています。

1.2.4 DECIMAL 型使用上の注意事項

DECIMAL 型，日間隔型，及び時間隔型で使用できる符号の種類を次に示します。

符号部の値	符号との関係
X'A'	符号としてみなされません（エラー）。※
X'B'	符号としてみなされません（エラー）。※
X'C'	正の符号としてみなされます。
X'D'	負の符号としてみなされます。
X'E'	符号としてみなされません（エラー）。※
X'F'	正の符号としてみなされます。

注※

システム定義の pd_dec_sign_normalize オペランドが Y の場合，符号としてみなされます。

pd_dec_sign_normalize オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

システム定義の pd_dec_sign_normalize オペランドがN，又は省略している場合，格納データの符号部は正規化されないので，格納データの符号部は同じ正の符号を意味する X'C'，及び X'F'が混在する可能性があります。また，SQL 実行中の型変換，演算などで符号が変換される場合があります。

既に符号が混在しているデータベースの符号を統一する場合には，表のデータをデータロードし直す必要があります。

DECIMAL 型の符号正規化機能については，マニュアル「HiRDB システム運用ガイド」の DECIMAL 型の符号正規化機能を参照してください。

DECIMAL 型の列の格納データが正の場合，データの符号部が X'C'に統一されていないときは，その列を検索すると格納データの符号部と検索結果の符号部が異なることがあります。符号部を意識して UAP を作成する場合は，次の点を考慮してください。

(1) インデクスを定義した DECIMAL 型の列を検索した場合の，検索結果の符号部の扱い

インデクスを定義した DECIMAL 型の列を検索した場合，次に示す検索結果の符号部の扱いとなります。

- 格納データの符号部がすべて X'C'のとき
  検索結果の符号部は，必ず X'C'になります。
- 格納データの符号部がすべて X'F'のとき
  DECIMAL 型の列に定義したインデクスが単一列インデクスの場合，検索結果の符号部は必ず X'F'になります。

DECIMAL 型の列を含む複数列インデクスの場合，検索結果の符号部が X'C'になることがあります。

- 格納データの符号部に X'C'と X'F'とが混在するとき

格納データの符号部と検索結果の符号部は，一致しないことがあります。

(2) GROUP BY 句を指定した検索で，DECIMAL 型のグループ化列を選択式に指定して検索した場合の，検索結果の符号部の扱い

GROUP BY 句を指定した検索で，DECIMAL 型のグループ化列を選択式に指定して検索した場合，次に示す検索結果の符号部の扱いになります。

- 選択式中の格納データの符号部がすべて X'C'のとき
  検索結果の符号部は，必ず X'C'になります。
- 選択式中の格納データの符号部がすべて X'F'のとき
  システム共通定義の pd_optimize_level オペランドでグループ分け高速化処理を選択した場合，検索結果の符号部が X'C'になることがあります。
  グループ分け高速化処理を選択しない場合，検索結果の符号部は必ず X'F'になります。
- 選択式中の格納データの符号部に X'C'と X'F'とが混在するとき
  検索結果の符号部は，X'C'又は X'F'のどちらかになります。

(3) SQL 拡張最適化オプションを指定した場合の検索結果の符号部の扱い

SQL 拡張最適化オプションで「ハッシュジョイン，副問合せのハッシュ実行」を適用した場合，複数の表を＝で結合した問合せ指定又は副問合せを指定した検索では，次に示す検索結果の符号部の扱いとなります。

- 選択式中の格納データの符号部がすべて X'C'のとき
  検索結果の符号部は，必ず X'C'になります。
- 選択式中の格納データの符号部がすべて X'F'のとき
  検索結果の符号部は，X'C'なることがあります。
- 選択式中の格納データの符号部に X'C'と X'F'とが混在するとき
  格納データの符号部が X'F'でも，検索結果の符号部が X'C'になることがあります。

1.2.5 長大データ使用上の注意事項

長大データを使用する場合の注意事項を次に示します。

(1) 長大データを使用できない項目

長大データは，次に示す項目では使用できません。

- インデクスの定義
- ソート
- グループ分け
- 集合演算，四則演算
- 集合関数
- 重複排除
- CASE 式
- CAST 指定（ただし，CAST(NULL AS BLOB 型)は指定できます）
- ユーザ定義関数の引数中の外への参照列

ただし，抽象データ型の属性として定義で指定した長大データの場合，プラグイン機能を使用するとインデクスの定義に指定できます。また，長大データの連結演算は，UPDATE 文の SET 句の更新値以外では使用できません。

(2) スカラ関数

長大データが指定できるスカラ関数は，LENGTH 関数，SUBSTR 関数，及び POSITION 関数だけです。これ以外のスカラ関数では，長大データは使用できません。

(3) データの挿入，更新

長大データは，INSERT 文，又は UPDATE 文で定数を指定して，データを挿入，又は更新できません。

長大データを挿入する場合は，埋込み変数，？パラメタ，SQL 変数，SQL パラメタ，スカラ関数 SUBSTR，関数呼出し，及び NULL を使用します。更新する場合は，列指定，コンポネント指定，埋込み変数，？パラメタ，SQL 変数，SQL パラメタ，連結演算，スカラ関数 SUBSTR，関数呼出し，副問合せ，及び NULL を使用します。

(4) 位置付け子の利用

UAP で長大データを扱う場合には，位置付け子を用いることで，クライアント上にデータの実体を保持しないで，長大データを扱う SQL の処理ができます。位置付け子については「位置付け子（locator）」を参照してください。

1.2.6 BINARY 型使用上の注意事項

(1) BINARY 型を使用できない項目

BINARY 型は，次に示す項目では使用できません。

- インデクスの定義
- 外への参照列

また，BINARY 型の定義長で指定可否が変わる項目を次の表に示します。

表 1-8　BINARY 型の定義長で指定可否が変わる項目

項目	定義長		
	1～255	256～32,000	32,001～2,147,483,647
ソート	○	○	×
グループ分け	○	○	×
集合関数	○	○	×
探索条件	○	○	×
データの挿入，更新※1	○	○	○
重複排除	○	○	×
集合演算	○	○	×
連結演算	○	○	○※2
スカラ関数	○	○	○※3
CASE 式	○	○	×
CAST 指定	○	○	×

（凡例）

○：使用できます。

×：使用できません。

注※1

BINARY 型の列への更新値，挿入値に定数を指定する場合は，16 進文字列定数だけ指定できます。

注※2

最大長が 32,001 バイト以上の BINARY 型の連結演算は，UPDATE 文の SET 句の更新値以外では使用できません。

注※3

定義長が 32,001 バイト以上の BINARY 型を指定できるスカラ関数は，LENGTH 関数，SUBSTR 関数，及び POSITION 関数だけです。これ以外のスカラ関数では，定義長が 32,001 バイト以上の BINARY 型は使用できません。

(2)　位置付け子の利用

UAP で BINARY 型を扱う場合には，位置付け子を用いることで，クライアント上にデータの実体を保持しないで， BINARY 型データを扱う SQL の処理ができます。位置付け子については「位置付け子 (locator)」を参照してください。

1.2.7　論理データ使用上の注意事項

論理データは，関数の戻り値にだけ指定できます。

1.2.8　抽象データ型使用上の注意事項

抽象データ型は，次に示す項目では使用できません。

- インデクスの定義
- ソート
- グループ分け
- 集合演算，四則演算，連結演算
- 集合関数
- 重複排除
- CASE 式
- CAST 指定
- 外への参照列

ただし，プラグインを使用する場合は，インデクスの定義ができます。

1.3　文字集合

1.3.1　文字集合の形式と規則

(1)　説明

文字集合は文字データに対する属性であり，次の三つの属性を持ちます。

- 使用形式
  文字を表現する規約のことです。
- 文字レパートリ
  表現できる文字の集合のことです。
- 既定の照合順
  二つの文字列データを表現する規約のことです。どの文字集合にも，既定の照合順があります。

(2)　形式

文字集合指定：：＝　［MASTER.］文字集合名

文字集合名　：：＝　{EBCDIK ｜ UTF16}

(3)　規則

1. 使用できる文字集合について次の表に示します。

表 1-9　HiRDB で使用できる文字集合

文字集合名	使用形式	文字レパートリ	既定の照合順
EBCDIK	EBCDIK コード。 文字は 8 ビットの単一バイト文字コードで表現されます。	EBCDIK コードのすべての文字	ビット組み合わせによるコード順
UTF16	JIS X 0221(ISO/IEC 10646)で規定され，各文字が 2 又は 4 バイトで符号化される文字符号化形式で表現されます。バイト順序はビッグエンディアンです。	Unicode のすべての文字	ビット組み合わせによるコード順

2. pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に sjis 以外を指定した場合，文字集合指定に EBCDIK を指定できません。
3. pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，utf-8_ivs 以外を指定した場合，文字集合指定に UTF16 を指定できません。

4. 文字集合指定は，文字データ型が指定できる箇所に指定できます。混在文字データ型，各国文字データ型に対しては指定できません。
5. 文字集合指定を省略した場合は，pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で指定した文字コード種別の文字コードを文字集合として仮定します。文字集合指定を省略して仮定された文字集合を，既定文字集合といいます。pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で指定した文字コード種別と文字集合との対応を，次の表に示します。

表 1-10　pdntenv（pdsetup）コマンドで指定した文字コードと仮定する既定文字集合

コマンドで指定した文字コード種別	既定文字集合
sjis	シフト JIS 漢字コード
chinese	EUC 中国語漢字コード
ujis	EUC 日本語漢字コード
utf-8	Unicode（UTF-8）
utf-8_ivs	Unicode（IVS 対応 UTF-8）
lang-c	単一バイト文字コード
chinese-gb18030	中国語漢字コード（GB18030）

(4)　留意事項

？パラメタを介して，文字コードが UTF-16 のデータを操作する場合，文字集合名記述領域に文字集合名を指定します。また，プリプロセスオプション，及び埋込み変数の定義に，文字コードが UTF-16 のデータを操作できる指定をすることで，埋込み変数を介して文字コードが UTF-16 のデータを操作できるようになります。このとき，SQL プリプロセサは，指定されたプリプロセスオプション，及び埋込み変数に基づいて文字集合名を仮定します。

文字集合名には UTF16 に加え，UTF-16LE，及び UTF-16BE が指定できます。

以降の説明では，文字集合名 UTF16 には，UTF-16LE，UTF-16BE が含まれているものとします。

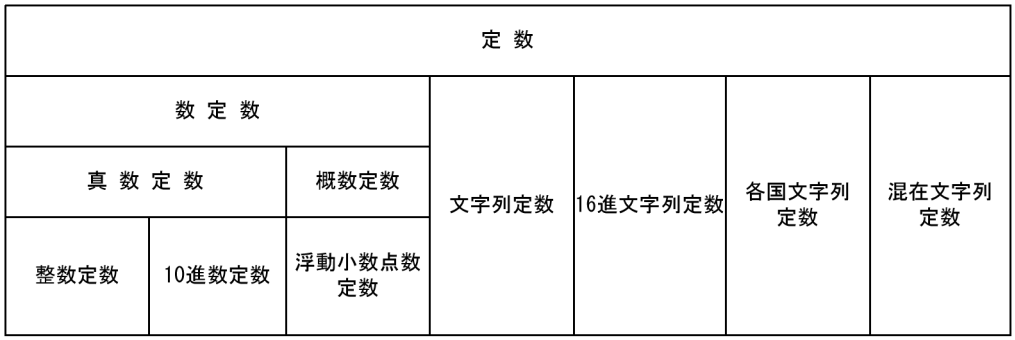
なお，プリプロセスオプション，埋込み変数の定義，及び文字集合名記述領域に指定する文字集合の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

1.4 定数

定数は，プログラム中で値を変更できないデータです。定数には，数値を表す数定数と文字列を表す文字列定数，各国文字列定数，及び混在文字列定数があります。

SQL で指定できる定数を次の図に示します。

図 1-4 SQL で指定できる定数



定数の表記法を次の表に示します。

表 1-11 定数の表記法

定 数	表 記 法		HiRDB で解釈するデータ型
整数定数※1	〔符号〕符号なし整数 （例） －123 45 6789	符号なし整数は，数字の並びで表します。 符号は，＋，又は－で表します。	INTEGER
10 進数定数	〔符号〕整数部.小数部 （例） 12.3 －456. .789	整数部と小数部は符号なし整数で表します。整数と小数のどちらかを指定する必要があります。小数点は必ず付けてください。	DECIMAL（m〔，n〕） m，n は表記したけた数
浮動小数点数定数	仮数 E 指数 （例） 1.0E2 .5E＋67	仮数は整数定数，又は 10 進数定数で表します。指数は 1～3 けたの整数定数で表します。指数は 10 のべき乗を表します。文字「E」は必ず付けてください。	FLOAT
文字列定数※3	'文字列' （例）'HITACHI' '88' '''95.7.30'	文字列は半角文字の列で表します。文字列にアポストロフィを書く場合，1 個のアポストロフィを表すのに，2 個続けて書いてください。文字列の長さは 32,000 バイト以内です。	VARCHAR(n) n は表記した文字列長※2
16 進文字列定数※6	X'16 進文字列'	16 進文字列は，0～9，及び A～F（又は a～f）で表しま	VARCHAR(n)

定　数	表　記　法		HiRDB で解釈するデータ型
	（例）X'82A0' X'82a0'	す。16 進文字列の長さは 64,000 文字以内で，2 の倍数にしてください。16 進文字 2 文字で 1 バイトとなります。	n は表記した文字列長÷2※2
各国文字列定数※3※4※5	N'各国文字列' （例）N'SQL 文法'	文字列は全角文字の列で表します。文字列の長さは 16,000 文字以内です。	NVARCHAR(n) n は表記した文字列長※2
混在文字列定数※3※4	M'文字列' （例）M'1996 年'	文字列は半角文字と全角文字の列で表します。文字列の長さは 32,000 バイト以内です。	MVARCHAR(n) n は表記した文字列長※2

注

日付や時刻は，既定の文字列表現の定数として指定できます。また，日間隔や時間隔は 10 進数表現の定数（10 進数定数）として SQL 中に指定できます。

注※1

整数定数の値の範囲を超える定数を整数定数の表記法で指定すると，HiRDB は，定数の右側に小数点を仮定し，10 進数定数が指定されたものと解釈します。

注※2

長さ 0 の文字列定数（' '，X' '，N' '，M' '）の場合，n は 1 です。

注※3

COMMENT 文，EXECUTE IMMEDIATE 文，及び PREPARE 文に指定する文字列については，各 SQL 文の文法詳細を参照してください。

注※4

pdsetup コマンドで文字コード種別に lang-c を指定した場合，各国文字列定数，及び混在文字列定数は使用できません（UNIX 版限定）。

注※5

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，utf-8_ivs，又は chinese-gb18030 を指定した場合，各国文字列定数は使用できません。

注※6

16 進文字列定数は，文字列定数と記述の形式が異なるだけです。マニュアル中の文字列定数に関する記述は，16 進文字列定数にも適用されます。

数定数の使用上の制限を次の表に示します。

表 1-12　数定数の使用上の制限

数定数	範　囲	指定できるけた数の最大値 （上位の無効数字 0 のけた数を含む）
整数定数	−2147483648～2147483647	10 けた

数定数	範　囲	指定できるけた数の最大値 （上位の無効数字 0 のけた数を含む）
10 進数定数	－（$10^{38}-1$）～-10^{-37}，0， 10^{-37}～$10^{38}-1$	38 けた
浮動小数点数定数※	約-1.7×10^{308}～-4.9×10^{-324}，0， 約4.9×10^{-324}～1.7×10^{308}	仮数部：17 けた 指数部：3 けた

注※

値の範囲は，ハードウェア表現に依存します。

1.4.1　日付データの既定の文字列表現

（形式）'YYYY-MM-DD'

日付を既定の文字列表現の定数にする場合，年（YYYY），月（MM），日（DD）をハイフンで結んで'YYYY-MM-DD'として表現します。そのときに，年（YYYY），月（MM），日（DD）のけた数に満たない場合は，足りないけた数分だけ左側に 0 を補ってください。

また，既定の文字列表現で指定した日付の定数は，スカラ関数 DATE の引数や，日付データが要求される場所で指定すると，日付データ型に変換されます。

日付を既定の文字列表現の定数にする例を次に示します。

（例）

日付：1995 年 7 月 30 日

日付の既定の文字列表現：'1995-07-30'

1.4.2　時刻データの既定の文字列表現

（形式）'hh:mm:ss'

時刻を既定の文字列表現の定数にする場合，時（hh），分（mm），秒（ss）をコロン（：）で結んで'hh:mm:ss'として表現します。そのときに，時（hh），分（mm），秒（ss）のけた数に満たない場合は，足りないけた数分だけ左側に 0 を補ってください。

また，既定の文字列表現で指定した時刻の定数は，スカラ関数 TIME の引数や，時刻データが要求される場所で指定すると，時刻データ型に変換されます。

時刻を既定の文字列表現の定数にする例を次に示します。

（例）

時刻：11 時 3 分 58 秒

時刻の既定の文字列表現：'11:03:58'

1.4.3 時刻印データの既定の文字列表現

（形式）'YYYY-MM-DD hh:mm:ss'

時刻印を既定の文字列表現の定数にする場合，年（YYYY），月（MM），日（DD）をハイフンで結び，空白を含めてから，更に時（hh），分（mm），秒（ss）をコロン（：）で結んで'YYYY-MM-DD hh:mm:ss[nn…n]'として表現します。そのときに，年（YYYY），月（MM），日（DD）のけた数に満たない場合は，足りないけた数分だけ左側に 0 を補ってください。同様に，時（hh），分（mm），秒（ss）のけた数に満たない場合は，足りないけた数分だけ左側に 0 を補ってください。

小数秒精度を表現する場合，秒（ss）と小数秒（nn…n）の間を，ピリオドで結んでください。小数秒精度を省略し，ピリオドだけを指定すると，小数秒精度が 0 のデータとして扱われます。小数秒精度が 7 以上の場合はエラーとなります。

また，既定の文字列表現で指定した時刻印の定数は，スカラ関数 TIMESTAMP の引数や，時刻印データが要求される場所で指定できます。

時刻印を既定の文字列表現の定数にする例を次に示します。

（例）

時刻印：1995 年 7 月 30 日 11 時 3 分 58 秒

時刻印の既定の文字列表現：'1995-07-30 11:03:58'

1.4.4 日間隔データの 10 進数表現

（形式）±YYYYMMDD.

日間隔を 10 進数表現の定数にする場合，年（YYYY），月（MM），日（DD），符号（s）を，精度 8，位取り 0 の固定小数点数で±YYYYMMDD.として表現します。

10 進数表現した日間隔の定数は，日間隔データが要求される場所で指定すると，日間隔データ型に変換されます。

日間隔を 10 進数表現の定数にする例を次に示します。

（例）

1 か年 1 か月 1 日間を 10 進数で表現する場合：00010101.

1.4.5　時間隔データの 10 進数表現

（形式）±hhmmss.

時間隔を 10 進数表現の定数にする場合，時（hh），分（mm），秒（ss），符号（t）を，精度 6，位取り 0 の固定小数点数で±hhmmss.として表現します。

10 進数表現した時間隔の定数は，時間隔データが要求される場所で指定すると，時間隔データ型に変換されます。

時間隔を 10 進数表現の定数にする例を次に示します。

（例）

1 時間 1 分 1 秒を 10 進数表現する場合：010101.

1.4.6　日時間隔データの 10 進数表現

（形式）±YYYYMMDDhhmmss.

日時間隔を 10 進数表現の定数にする場合，年（YYYY），月（MM），日（DD），

時（hh），分（mm），秒（ss），符号（t）を，精度 14，位取り 0 の固定小数点数で
±YYYYMMDDhhmmss.として表現します。

日時間隔を 10 進数表現の定数にする例を次に示します。

（例）

1 か年 2 か月 3 日間 4 時間 5 分 6 秒を 10 進数で表現する場合：00010203040506.

1.5 USER 値関数，CURRENT_DATE 値関数，CURRENT_TIME 値関数，及び CURRENT_TIMESTAMP 値関数

USER 値関数，CURRENT_DATE 値関数，CURRENT_TIME 値関数，及び CURRENT_TIMESTAMP 値関数は，SQL 中で値指定として使用できます。

1.5.1 USER 値関数

(1) 機能

実行ユーザの認可識別子を表します。

(2) 形式

```
USER値関数：：＝ {USER | CURRENT_USER}
```

(3) 規則

1. USER 値関数を指定した場合，HiRDB は既定文字集合の VARCHAR(30)が指定されたものとして解釈します。
2. 表名及びインデクス名の認可識別子としては使用できません。

1.5.2 CURRENT_DATE 値関数

(1) 機能

現在の日付を表します。

(2) 形式

```
CURRENT_DATE値関数：：＝ {CURRENT_DATE | CURRENT DATE}
```

(3) 規則

1. CURRENT_DATE 値関数を指定した場合，HiRDB は日付データ型（DATE）が指定されたものとして解釈します。
2. CURRENT_DATE は，現在の日付を表します。CURRENT_DATE を指定できる項目を次に示します。
   - 選択式及び条件式中

- 日付データ型への更新値，挿入値
- CHAR(10)（文字集合指定に UTF16 を指定している場合は CHAR(20)）の列への更新値，挿入値

  CHAR(10)（文字集合指定に UTF16 を指定している場合は CHAR(20)）の列への更新値，挿入値として指定すると，現在の日付が，日付の既定の文字列表現に変換された後，更新又は挿入をします。

3. 一つの SQL 文中で CURRENT_DATE を複数回指定すると，それらはすべて同じ値になります。また，CURRENT_DATE，CURRENT_TIME，及び CURRENT_TIMESTAMP を一つの SQL 文中で組み合わせて指定すると，日付と時刻は同時点の値になります。

1.5.3 CURRENT_TIME 値関数

(1) 機能

OS の localtime 関数で変換した現在の時刻を表します。タイムゾーンはシステム共通定義の TZ オペランドに従います。

(2) 形式

```
CURRENT_TIME値関数：：＝ {CURRENT_TIME | CURRENT TIME}
```

(3) 規則

1. CURRENT_TIME 値関数を指定した場合，HiRDB は時刻データ型（TIME）が指定されたものとして解釈します。
2. CURRENT_TIME は，現在の時刻を表します。CURRENT_TIME を指定できる項目を次に示します。
   - 選択式及び条件式中
   - 時刻データ型への更新値，挿入値
   - CHAR(8)（文字集合指定に UTF16 を指定している場合は CHAR(16)）の列への更新値，挿入値

     CHAR(8)（文字集合指定に UTF16 を指定している場合は CHAR(16)）の列への更新値，挿入値として指定すると，現在の時刻が，時刻の既定の文字列表現に変換された後，更新又は挿入をします。
3. 一つの SQL 文中で CURRENT_TIME を複数回指定すると，それらはすべて同じ値になります。また，CURRENT_DATE，CURRENT_TIME，及び CURRENT_TIMESTAMP を一つの SQL 文中で組み合わせて指定すると，日付と時刻は同時点の値になります。

1.5.4 CURRENT_TIMESTAMP 値関数

(1) 機能

現在の時刻印を表します。

(2) 形式

```
CURRENT_TIMESTAMP値関数：：＝ {CURRENT_TIMESTAMP〔（小数秒精度）〕
                              ｜CURRENT TIMESTAMP〔（小数秒精度）〕}
```

(3) 規則

1. CURRENT_TIMESTAMP 値関数を指定した場合，HiRDB は時刻印データ型（TIMESTAMP）が指定されたものとして解釈します。
2. CURRENT_TIMESTAMP は，現在の時刻印を表します。CURRENT_TIMESTAMP を指定できる項目を次に示します。
   - 選択式及び条件式中
   - 時刻印データ型の列への更新値，挿入値
   - 長さが 19，22，24，26 バイトの CHAR（文字集合指定に UTF16 を指定している場合は，長さが 38，44，48，52 バイトの CHAR）の列への更新値，挿入値

   小数秒精度 p（p=0，2，4，又は 6）を指定した場合，小数点以下 p けたまで有効な小数秒を含む時刻印を返します。小数秒精度 p を指定しない場合，p=0 が仮定されます。

   データ型が CHAR の列に，更新値，挿入値として指定する場合，現在の時刻印が，時刻印の既定の文字列表現に変換された後，更新又は挿入をします。

   更新値又は挿入値として指定する場合，指定可能な CHAR 列の定義長を次の表に示します。

   表 1-13 更新値又は挿入値として指定する場合の値関数小数秒精度と CHAR 列の定義長の対応

小数秒精度 p の値	CHAR の定義長	
	既定文字集合又は文字集合(UTF16 以外)	文字集合(UTF16)
0	19	38
2，4，又は 6	20 + p	40 + p×2

3. 一つの SQL 文中で CURRENT_TIMESTAMP を複数回指定すると，それらはすべて同じ値になります。また，CURRENT_DATE，CURRENT_TIME，及び CURRENT_TIMESTAMP を一つの SQL 文中で組み合わせて指定すると，日付と時刻は同時点の値になります。

1.6 埋込み変数，標識変数，?パラメタ，SQL パラメタ，及び SQL 変数

この節では，UAP 中，SQL 中，外部ルーチン中，及び SQL ルーチン中で値の受け渡しをする変数，パラメタについて説明します。

- **埋込み変数，標識変数，？パラメタ**

  UAP 又は外部ルーチンの SQL 中に指定できます。指定すると，その SQL と UAP との間で値の受け渡しができます。

- **SQL 変数**

  SQL ルーチンの SQL 中に指定できます。指定すると，その SQL ルーチンとの間で値の受け渡しができます。なお，外部ルーチン中には SQL 変数は指定できません。

- **SQL パラメタ**

  呼び出されるルーチンのルーチン定義時に指定できます。指定すると，SQL の CALL 文や関数呼出しをする UAP 又はルーチンと，呼び出されるルーチンとの間で値の受け渡しができます。

  SQL ルーチン中では，SQL パラメタを直接指定します。外部ルーチン中では，SQL パラメタに対応した外部ルーチン本体のパラメタ変数で指定します。

1.6.1 埋込み変数，標識変数

(1) 形式

：埋込み変数〔：標識変数〕

(2) 機能と用途，及び指定箇所

埋込み変数，及び標識変数の機能とその用途，及び指定箇所を次の表に示します。

表 1-14 埋込み変数，及び標識変数の機能とその用途，及び指定箇所

機　能	用　途		指定箇所
	埋込み変数	標識変数	
検索結果の列の値の受け取り	ナル値以外の値を受け取ります。※1	ナル値も含む列の値を受け取るために，埋込み変数とともに使用して，埋込み変数に読み込んだ値がナル値かどうかを知ることができます。 また，受け取ったデータが文字データ，各国文字データ，混在文字データ，又は長大データの場合に，埋込み変数に正しく受け取れたかどうかを知ることができます。	SELECT 文，FETCH 文，EXECUTE 文，EXECUTE IMMEDIATE 文の INTO 句

機　能	用　途		指定箇所
	埋込み変数	標識変数	
定数値の変更	ナル値以外の値を指定します。※1	ナル値を指定するために，埋込み変数とともに使用して，SQL に渡す埋込み変数の値がナル値かどうかを指示します。	SQL 中に指定する定数の値を変えて実行する場合に，定数の代わりに指定します。
?パラメタに対する値の指定	ナル値以外の値を指定します。※1	ナル値を指定するために，埋込み変数とともに使用して，SQL に渡す埋込み変数の値がナル値かどうかを指示します。	PREPARE 文で前処理する SQL 中，又は EXECUTE IMMEDIATE 文で前処理・実行する SQL 中で指定した?パラメタに対する値を指定するために，EXECUTE 文，OPEN 文，EXECUTE IMMEDIATE 文の USING 句に指定します。※2
埋込み変数の変更	（埋込み変数） カーソル宣言中で指定した SELECT 文中の埋込み変数の代わりに，ほかの埋込み変数を指定します。	－	OPEN 文の USING 句
SQL 文字列の指定	UAP の実行時に生成した SQL を前処理して実行するために，SQL 文字列を指定します。	－	PREPARE 文，EXECUTE IMMEDIATE 文
認可識別子，及びパスワードの指定	UAP の実行ユーザを HiRDB に連絡します。	－	CONNECT 文の認可識別子，及びパスワード

（凡例）－：該当しません。

注※1

ナル値も扱う場合は，標識変数を指定する必要があります。標識変数の値については，「標識変数の値の設定」を参照してください。

注※2

OPEN 文の USING 句には，標識変数は指定できません。

(3)　埋込み変数及び標識変数のデータ型と各言語のデータ記述の関係

埋込み変数及び標識変数を UAP 中に記述するとき，データ変換を起こさないで SQL と UAP との間でデータの受け渡しをするためのデータ型とデータ記述の関係については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4)　埋込み変数及び標識変数の修飾指定

(a)　COBOL 言語の場合

埋込み変数，及び標識変数を集団項目で修飾できます。

集団項目で修飾する場合の形式：

```
：〔変数名1.〕変数名2
```

修飾した結果，埋込み変数，又は標識変数は一意に定まるように指定してください。修飾する必要のない名前を修飾してもかまいません。また，一意に修飾する組み合わせが幾つかある場合，どれを使用してもかまいません。

変数名 1 は，変数名 2 が従属する集団項目です。

(b)　C 言語の場合

構造体のメンバを，構造体又は構造体のポインタで修飾できます。

構造体で修飾する場合の形式：

```
：構造体名.メンバ名
```

ポインタで修飾する場合の形式：

```
：ポインタ名->メンバ名
```

(5)　埋込み変数と SQL のデータ型の関係

UAP 中の埋込み変数のデータ型と SQL のデータ型の代入の関係を次の表に示します。

表 1-15　UAP の埋込み変数と SQL のデータ型の代入の関係

埋込み変数のデータ型	SQL のデータ型		
	文字データ型	各国文字データ型	混在文字データ型
文字データ型	○	△	○
各国文字データ型	△	○	△
混在文字データ型	○	△	○

（凡例）

○：代入できます。

△：代入できるがデータ型に関係なく代入するため，注意が必要です（注 2，注 3）。

注 1

文字データ型，各国文字データ型，混在文字データ型は固定長，又は可変長を含みます。

注 2

データが短くデータの右側に空白を埋める場合は，次のようにします。

- UAP の埋込み変数を SQL のデータ型に代入する場合

  SQL のデータ型の空白を埋めます。文字集合の指定がある場合は，その文字集合の空白です。各国文字データ型の場合は全角空白です。

- SQL のデータ型を UAP の埋込み変数に代入する場合

  UAP の埋込み変数の，データ型の空白を埋めます。文字集合の指定がある場合は，その文字集合の空白です。各国文字データ型の場合は全角空白です。

ただし，各国文字データ型への代入の場合，クライアント環境定義の PDSPACELVL，又はシステム共通定義の pd_space_level オペランドで空白変換レベルが設定されているとき，埋めた各国文字の空白も変換対象となります。そのため，各国文字の空白が 2 バイトの空白に変換される場合があります。

注 3

各国文字の空白と 1 バイトの空白が混在するような使い方をすると，空白を含む比較などが意図したように実行できなくなるので注意してください。

注 4

各国文字データ型の列に 1 バイト文字を代入しないでください。代入した場合，結果は保証できません。

注 5

LIKE 述語を使用した場合，パターン文字の特殊文字は SQL のデータ型の仕様に合わせてください。

注 6

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，utf-8_ivs，又は chinese-gb18030 を指定した場合，SQL の文字データ型又は混在文字データ型から，埋込み変数の各国文字データ型への代入はできません。また，埋込み変数の各国文字データ型から，SQL の文字データ型又は混在文字データ型への代入もできません。

1.6.2 ?パラメタ

(1) 機能

UAP 実行時に，プログラム中で SQL の文字列を組み立て，その SQL 文字列を PREPARE 文で前処理して EXECUTE 文，OPEN 文，FETCH 文，又は CLOSE 文で実行したり，EXECUTE IMMEDIATE 文で前処理して実行したりする機能があります。この機能で実行する SQL に対しても，UAP から値を渡すことができます。この場合に，PREPARE 文で前処理する SQL，又は EXECUTE IMMEDIATE 文で前処理して実行する SQL の文字列の中で，UAP から値を渡される箇所に「?」を指定します。この「?」を?パラメタといいます。

(2) ?パラメタに渡す値の指定

?パラメタに渡す値は，PREPARE 文に対応する EXECUTE 文，OPEN 文，FETCH 文，CLOSE 文，又は EXECUTE IMMEDIATE 文の USING 句の埋込み変数で指定します。標識変数は EXECUTE 文及び EXECUTE IMMEDIATE 文でだけ指定できます。

SQL 記述領域を使用しない?パラメタの例を次に示します。

（例）?パラメタの使用例（SQL 記述領域を使用しない場合）

（C 言語の場合）

```
      :
EXEC SQL BEGIN DECLARE SECTION;
struct{
long xcmnd_len;
char xcmnd_txt[58];
}xcmnd;
char  XSCODE[5];
char  XSNAME[17];
char  XCOL[3];
long  XTANKA;
long  XZSURYO;
EXEC SQL END DECLARE SECTION;
      :
strcpy(xcmnd.xcmnd_txt,"INSERT INTO ZAIKO VALUES(?,?,?,?,?)");
xcmnd.xcmnd_len = strlen(xcmnd.xcmnd_txt);
EXEC SQL PREPARE ST1 FROM :xcmnd;

strcpy(XSCODE,"595M");
strcpy(XSNAME,"ソックス");
strcpy(XCOL,"赤");
XTANKA = 300;
XZSURYO = 200;
EXEC SQL
EXECUTE ST1 USING :XSCODE,:XSNAME,:XCOL,:XTANKA,:XZSURYO;
      :
```

（COBOL 言語の場合）

```
      :
DATA DIVISION.

WORKING-STORAGE SECTION.
EXEC SQL
BEGIN DECLARE SECTION
END-EXEC.
01 XCMND.
  02 XCMND-LEN PIC S9(9) COMP VALUE 58.
  02 XCMND-TXT PIC X(58)   VALUE SPACE.
 77 XSCODE   PIC X(4).
 77 XSNAME   PIC N(8).
 77 XCOL    PIC N(1).
 77 XTANKA   PIC S9(9) COMP.
 77 XZSURYO   PIC S9(9) COMP.
```

```
  EXEC SQL
   END DECLARE SECTION
  END-EXEC.
     :
PROCEDURE DIVISION.
     :
  MOVE 'INSERT INTO ZAIKO VALUES(?,?,?,?,?)' TO XCMND-TXT.
  EXEC SQL
   PREPARE ST1 FROM :XCMND
  END-EXEC.

  MOVE '123S'   TO XSCODE.
  MOVE N'Tシャツ' TO XSNAME.
  MOVE N'赤'    TO XCOL.
  MOVE 200     TO XTANKA.
  MOVE 300     TO XZSURYO.

  EXEC SQL
   EXECUTE ST1 USING :XSCODE,:XSNAME,:XCOL,:XTANKA,:XZSURYO
  END-EXEC.
     :
```

1.6.3 SQL パラメタ，SQL 変数

(1) 形式

- SQL パラメタ
  〔〔認可識別子.〕ルーチン識別子.〕SQL パラメタ名
- SQL 変数
  〔文ラベル.〕SQL 変数名

(2) 機能

(a) SQL パラメタ

SQL パラメタは，手続き定義時や関数定義時に宣言したルーチンのパラメタです。

SQL パラメタは，CALL 文や関数呼出しでルーチンが呼び出されたときに，CALL 文や関数呼出しを記述した UAP 又はルーチンと，呼び出されるルーチンとの間で値の受け渡しをするための変数です。

ルーチン宣言時に指定するパラメタモードによって，SQL パラメタでは次のことができます。

- パラメタモードが IN 又は INOUT の場合
  ルーチン中で SQL パラメタの値を参照できます。値を参照することで，ルーチンを呼び出した UAP 又はルーチンから値を取得できます。
- パラメタモードが OUT 又は INOUT の場合

ルーチン中で SQL パラメタの値を設定できます。値を設定することで，ルーチンを呼び出した UAP 又はルーチンに値を返せます。

SQL パラメタの指定方法は，SQL で記述したルーチンに指定するか，又は SQL 以外で記述した外部ルーチンに指定するかで，次のように異なります。

- **SQL で記述したルーチン中に指定する場合**

  "〔〔認可識別子.〕ルーチン識別子.〕SQL パラメタ名"で指定します。

  ただし，パブリック手続き定義時，又はパブリック関数定義時に認可識別子を指定するときは，PUBLIC を指定してください。

- **SQL 以外で記述した外部ルーチン中に指定する場合**

  SQL パラメタに対応する外部ルーチン本体のパラメタ名で指定します。この場合，外部ルーチンは，指定したパラメタを SQL パラメタとしてではなく，外部ルーチンの実装に用いた言語のパラメタとして参照，更新します。

なお，UAP とルーチンとの間で値の受け渡しをするとき，UAP 側の CALL 文中又は関数呼出し中には，埋込み変数，標識変数，又は?パラメタを指定します。

(b) SQL 変数

SQL 変数は，SQL ルーチン中の SQL 間の値の受け渡しや，表と SQL ルーチンとの間の値の受け渡しに使用できます。SQL 変数は，SQL ルーチン中の複合文で宣言し，宣言した複合文中で参照できます。

SQL パラメタ，及び SQL 変数は，埋込み変数と異なり，ナル値が格納できます。このため，標識変数を使用したり，変数名の前に「：」を付けたりする必要はありません。

(3) データ型

SQL パラメタ，及び SQL 変数のデータ型は，SQL のデータ型を指定します。SQL のデータ型については，「データ型」を参照してください。

1.6.4 指定できる箇所

埋込み変数，標識変数，?パラメタ，SQL 変数，及び SQL パラメタを指定できる箇所を次の表に示します。

表 1-16 変数及びパラメタを指定できる箇所

SQL 文	指定できる箇所	埋込み変数	標識変数	?パラメタ	SQL 変数又は SQL パラメタ
WRITE 指定	第 1 引数	×	×	×	×
	第 2 引数，第 3 引数	○	○	○	×

SQL 文	指定できる箇所	埋込み変数	標識変数	?パラメタ	SQL 変数又は SQL パラメタ
GET_JAVA_STORED_ROUTINE_SOURCE 指定	第 1 引数，第 2 引数	○	○	○	○
	第 3 引数	×	×	×	×
DECLARE CURSOR	探索条件中で定数が指定できる箇所※1	○	○	×	○
ALLOCATE CURSOR 形式 1	拡張カーソル名	○	×	×	×
	拡張文名	○	×	×	×
ALLOCATE CURSOR 形式 2	拡張カーソル名	○	×	×	×
SELECT	探索条件中で定数が指定できる箇所	○	○	○	○
	INTO 句	○	○	×	○
INSERT	VALUES 句で定数が指定できる箇所	○	○	○	○
	探索条件中で定数が指定できる箇所	○	○	○	○
UPDATE	SET 句で定数が指定できる箇所	○	○	○	○
	探索条件中で定数が指定できる箇所	○	○	○	○
準備可能動的 UPDATE 文：位置付け	SET 句で定数が指定できる箇所	×	×	○	×
DELETE	探索条件中で定数が指定できる箇所	○	○	○	○
OPEN	USING 句	○	×	×	×
FETCH	INTO 句	○	○	×	○
PREPARE	SQL 文字列の箇所	○	×	×	×
	拡張文名	○	×	×	×
DEALLOCATE PREPARE	拡張文名	○	×	×	×
DESCRIBE	拡張文名	○	×	×	×
DESCRIBE CURSOR	拡張カーソル名	○	×	×	×
DESCRIBE TYPE	拡張文名	○	×	×	×

SQL 文	指定できる箇所	埋込み変数	標識変数	?パラメタ	SQL 変数又は SQL パラメタ
EXECUTE	INTO 句	○	○	×	×
	USING 句	○	○	×	×
	拡張文名	○	×	×	×
EXECUTE IMMEDIATE	SQL 文字列の箇所	○	×	×	×
	INTO 句	○	○	×	×
	USING 句	○	○	×	×
CALL	引数	○	○	○	○
代入文	代入先及び代入値	○※2	○	○※2	○
FREE LOCATOR	位置付け子参照	○	×	×	×
CONNECT	認可識別子，及びパスワード	○	×	×	×
SET SESSION AUTHORIZATION 文	認可識別子，及びパスワード	○	×	×	×
ALLOCATE CONNECTION HANDLE	:PDCNCTHDL 型変数，:リターンコード受け取り変数，;接続 PDHOST 変数，及び接続 PDNAMEPORT 変数	○	×	×	×
FREE CONNECTION HANDLE	:PDCNCTHDL 型変数，及び:リターンコード受け取り変数	○	×	×	×
DECLARE CONNECTION HANDLE SET	:PDCNCTHDL 型変数	○	×	×	×
GET DIAGNOSTICS	文情報項目名，及び条件情報項目名	○	×	×	×
WRITE LINE 文	値式	×	×	×	○

（凡例）

○：指定できます。

×：指定できません。

注 1

埋込み変数，及び標識変数は，UAP 中で指定します。？パラメタは，PREPARE 文で前処理する SQL 文字列中に指定します。

外部ルーチン中の SQL パラメタは，SQL パラメタに対応した外部ルーチン本体のパラメタ変数指定で指定します。外部ルーチン本体のパラメタを SQL やルーチンに渡す場合，SQL パラメタとしてではなく，埋込み変数又は？パラメタとして指定します。外部ルーチン中に SQL 変数は指定できません。

注 2

選択式中には，埋込み変数，標識変数，及び?パラメタは指定できません。

ただし，次のような場合には，埋込み変数，標識変数，及び？パラメタを指定できます。

- 関数呼出しの引数に指定した場合
- スカラ関数 SUBSTR の引数に指定した場合

指定方法については，「問合せ指定」を参照してください。

注 3

埋込み変数，標識変数，及び?パラメタ間の四則演算及び比較演算は指定できません。

注 4

集合関数の引数中に埋込み変数，標識変数，及び?パラメタは指定できません。

注 5

スカラ関数 HEX の引数中には，埋込み変数，標識変数，及び?パラメタは指定できません。

注 6

VALUE，BIT_AND_TEST，SUBSTR の第 2，第 3 引数，及び POSITION の第 3 引数を除くスカラ関数の引数には，埋込み変数，標識変数，及び?パラメタを単独（単項演算式に指定した場合も含む）で指定できません。ただし，SUBSTR の第 1 引数，LENGTH の引数，POSITION の第 1，第 2 引数には AS データ型を記述すれば BLOB 型，又は BINARY 型の場合だけ指定できます。

注 7

日付演算，時刻演算，及び連結演算中には，埋込み変数，標識変数，及び?パラメタは指定できません。

注 8

スカラ関数 VALUE の最初の値式に，埋込み変数，標識変数，及び?パラメタを単独（単項演算式に指定した場合も含む）で指定できません。

注 9

単純 CASE 式，又は探索 CASE 式の CASE，THEN，及び ELSE の値式に，埋込み変数，標識変数，及び?パラメタを単独（単項演算式に指定した場合も含む）で指定できません。

注 10

単純 CASE 指定中の最初の WHEN の値式，COALESCE の最初の値式，又は NULLIF の両方の値式に，埋込み変数，標識変数，及び?パラメタを単独（単項演算式に指定した場合も含む）で指定できません。

注 11

スカラ関数 BIT_AND_TEST の両方の値式には，単独（単項演算式に指定した場合も含む）で埋込み変数，標識変数，及び?パラメタは指定できません。

注※1

選択式中の CASE 式の探索条件を除きます。

ただし，関数呼出しの引数に指定した場合，その関数呼出しを選択式中の CASE 式の探索条件に指定できます。

注※2

代入文（SET）の代入値に単独で埋込み変数，標識変数，及び？パラメタを指定する場合は必ず AS データ型を指定してください。

1.6.5 標識変数の値の設定

(1) データを受け取る場合（FETCH 文，SELECT 文，EXECUTE 文，及び EXECUTE IMMEDIATE 文の INTO 句）

FETCH 文，SELECT 文，EXECUTE 文，及び EXECUTE IMMEDIATE 文を実行すると，その INTO 句に指定した標識変数には，次の表に示す値が設定されます。埋込み変数にナル値が返された場合，埋込み変数の値は保証しません。この場合，標識変数の指定がなければ，エラーになります。

表 1-17 FETCH 文，SELECT 文，EXECUTE 文，及び EXECUTE IMMEDIATE 文で返される標識変数の値（繰返し列以外の場合，及び繰返し列の各要素値の場合）

標識変数の値	対応する埋込み変数が受け取った値
負	ナル値です。 ただし，標識変数の値が－2 の場合は，繰返し列の添字で指定した要素がないときに設定されたナル値です。また，標識変数の値が-4 の場合は，SQL 実行時に四則演算，集合関数演算，ウィンドウ関数演算，及びオーバフローエラー抑止の対象となるスカラ関数で，オーバフローエラー抑止オプションによって設定されたナル値です。このナル値を受け取った埋込み変数と演算は対応しているので，オーバフローが発生した演算を判別できます。オーバフローエラー抑止の対象となるスカラ関数については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。
0	ナル値ではない値です。
正	ナル値ではない値です。 ただし，受け取ったデータは文字データ，又は長大データで，埋込み変数の長さが不足していたため，右側が切り捨てられた値であることを示します。このときの標識変数には，切り捨てられる前の長さが設定されます。 ただし，クライアント環境定義 PDCLTCNVMODE に UOC を指定して文字コード変換機能を使用している場合は，標識変数には切り捨て後の長さを設定します。

表 1-18 FETCH 文，SELECT 文，EXECUTE 文，及び EXECUTE IMMEDIATE 文で返される標識変数の値（繰返し列全体の情報の場合）

標識変数の値	対応する埋込み変数が受け取った値
負	ナル値です（要素数が 0）。
0	ナル値ではない値です（要素数が 1 以上）。
正	ナル値ではない値です（要素数が 1 以上）。 ただし，埋込み変数の領域の要素数が不足していたため，後の要素が切り捨てられたことを示します。このときの標識変数には，切り捨てられる前の要素数が設定されます。

繰返し列のデータを受け取る場合の標識変数，埋込み変数の構造とその例を次の図に示します。

図 1-5　繰返し列のデータを受け取る場合の標識変数，埋込み変数の構造

＜最大要素数nの列の場合＞

・埋込み変数の構造

現在要素数	データ構造			
	第1要素のデータ	第2要素のデータ	･･･	第n要素のデータ

・標識変数の構造

繰返し列 全体の標識	標識の情報			
	第1要素の標識	第2要素の標識	･･･	第n要素の標識

図 1-6　繰返し列のデータを受け取る例（その 1）

（例1）　最大要素数4，現在要素数3のデータの例を示します。

埋込み変数の構造

現在要素数	第1要素のデータ	第2要素のデータ	第3要素のデータ	第4要素のデータ
3	ABC		XYZ	
ナル値でない	ナル値でない	ナル値	ナル値でない	

(3)　(2)　(3)　(3)

標識変数の構造

全体の標識	第1要素の標識	第2要素の標識	第3要素の標識	第4要素の標識
0	0	負	0	

(1)　有効な要素は1～3

〔説明〕
繰返し列の値の参照は，(1)～(3)の順序で行います。
(1)標識変数の繰返し列全体のNULL判定（0以上又は負）を参照します。
(2)(1)が負でなければ，埋込み変数の現在要素数を参照します。
(3)標識変数の各要素の標識と，埋込み変数の各要素のデータ情報とを，(2)の数分だけ突き合わせて参照します。標識が負の場合，データ情報は参照しません。標識が負でなければ，データ情報を参照します。

図 1-7　繰返し列のデータを受け取る例（その 2）

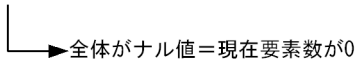
（例2）　最大要素数4，現在要素数0のデータの例を示します。

埋込み変数の構造

現在要素数	第1要素のデータ	第2要素のデータ	第3要素のデータ	第4要素のデータ
0				

標識変数の構造

全体の標識	第1要素の標識	第2要素の標識	第3要素の標識	第4要素の標識
負				



注　現在要素数が0の場合，標識変数がなくてもデータを受け取れます。

(2)　データを渡す場合（FETCH 文，SELECT 文，EXECUTE 文，及び EXECUTE IMMEDIATE 文の INTO 句以外）

FETCH 文，SELECT 文，EXECUTE 文，及び EXECUTE IMMEDIATE 文以外の SQL を実行する場合，その SQL の実行前の標識変数には，次の表に示す値を UAP で設定してください。標識変数の値によって，SQL の実行時に対応する埋込み変数の値を使用してください。

表 1-19　SQL 実行前に設定する標識変数の値（繰返し列以外の場合，及び繰返し列の各要素値の場合）

標識変数の値	対応する埋込み変数が SQL に渡す値
負	ナル値です。 埋込み変数が持っている値は無視されます。
0 又は正	ナル値ではない値です。 埋込み変数が持っている値です。

表 1-20　SQL 実行前に設定する標識変数の値（繰返し列全体の情報の場合）

標識変数の値	対応する埋込み変数が SQL に渡す値
負	ナル値です（要素数が 0）。 埋込み変数の要素の値は無視されます。
0 又は正	要素数が示す要素の値です。 ただし，要素数に 0 は指定できません。

繰返し列のデータを渡す場合の標識変数，埋込み変数の構造とその例を次の図に示します。

図 1-8　繰返し列のデータを渡す場合の標識変数，埋込み変数の構造

＜最大要素数nの列の場合＞

・埋込み変数の構造

現在要素数	データ構造			
	第1要素のデータ	第2要素のデータ	･･･	第n要素のデータ

・標識変数の構造

繰返し列全体の標識	標識の情報			
	第1要素の標識	第2要素の標識	･･･	第n要素の標識

図 1-9　繰返し列のデータを渡す例（その 1）

（例1）　最大要素数4，現在要素数3のデータの例を示します。

埋込み変数の構造

現在要素数	第1要素のデータ	第2要素のデータ	第3要素のデータ	第4要素のデータ
3	ABC		XYZ	

ナル値でない　ナル値でない　ナル値　ナル値でない

標識変数の構造

全体の標識	第1要素の標識	第2要素の標識	第3要素の標識	第4要素の標識
0	0	負	0	

有効な要素は1～3

図 1-10　繰返し列のデータを渡す例（その 2）

（例2）　最大要素数4，現在要素数0のデータの例を示します。

埋込み変数の構造

現在要素数	第1要素のデータ	第2要素のデータ	第3要素のデータ	第4要素のデータ
0				

標識変数の構造

全体の標識	第1要素の標識	第2要素の標識	第3要素の標識	第4要素の標識
負				

全体がナル値＝現在要素数が0

注　現在要素数が0の場合，標識変数がなくてもデータを渡せます。

1.6.6　埋込み変数へのナル値の既定値設定

FETCH 文，SELECT 文，EXECUTE 文，及び EXECUTE IMMEDIATE 文で埋込み変数にデータを取り出す場合，取り出した値がナル値のとき，標識変数には負の値が設定されます。このときの埋込み変数の値は保証しません。ただし，埋込み変数へのナル値の既定値設定機能を使用すると，埋込み変数に既定値が設定されます。また，ナル値のデータに対しても標識変数を指定する必要はありません。ただし，この機能はデータを取り出す場合にだけ適用されます。

既定値設定機能を使用するには，UAP の実行時に，クライアント環境定義で PDDFLNVAL を設定しておく必要があります。

クライアント環境定義については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

埋込み変数に設定されるナル値の既定値を次の表に示します。

表 1-21　埋込み変数に設定されるナル値の既定値

分　類	データ型		既定値
数データ	INTEGER		0
	SMALLINT		
	DECIMAL		
	LARGE DECIMAL		
	FLOAT		
	SMALLFLT		
文字データ	CHARACTER(n)	既定文字集合又は文字集合（UTF16 以外）	n バイトの空白
		文字集合（UTF16）	n/2 文字の空白
	VARCHAR(n)	既定文字集合又は文字集合（UTF16 以外）	1 バイトの空白※1
		文字集合（UTF16）	1 文字の空白
各国文字データ	NCHAR(n)		n 文字の空白※2
	NVARCHAR(n)		1 文字の空白※2
混在文字データ	MCHAR(n)		n バイトの空白
	MVARCHAR(n)		1 バイトの空白
日付データ	DATE		1 年 1 月 1 日
時刻データ	TIME		0 時 0 分 0 秒
時刻印データ	TIMESTAMP		1 年 1 月 1 日 0 時 0 分 0 秒※3

分　類	データ型	既定値
日間隔データ	INTERVAL YEAR TO DAY	0 か年 0 か月 0 か日間
時間隔データ	INTERVAL HOUR TO SECOND	0 時間 0 分 0 秒間
長大データ※4	BLOB(n)	長さ 0 バイトのデータ
バイナリデータ※4	BINARY(n)	長さ 0 バイトのデータ

注

単純構造，繰返し構造のデータの各要素についてもこの表に従います。繰返し構造のデータ全体がナル値の場合，現在の要素数として 0 が設定されます。

注※1

WRITE 指定の結果の場合は，IP アドレスとなります。

注※2

HiRDB で扱う文字コードに依存します。

（例）

シフト JIS の場合：X'8140'

また，クライアント環境定義の PDSPACELVL，又はシステム共通定義の pd_space_level オペランドで空白変換レベルを設定している場合，埋込み変数に設定した空白も変換対象となります。そのため，NCHAR の空白が n×2 バイト，又は NVARCHAR の空白が 2 バイトに変換される場合があります。

注※3

小数秒精度を指定した場合は，指定したけた数分 0 が設定されます。

注※4

位置付け子を用いた場合，位置付け子の埋込み変数には，サーバ上の長さ 0 バイトのデータを識別する値が設定されます。

1.6.7　代入規則

代入の種類を次の表に示します。

表 1-22　代入の種類

代入の種類	内　容	代入元	代入先
取り出し代入	列の値の受け取り	列	埋込み変数，SQL 変数，又は SQL パラメタ
	手続きの出力パラメタ値の，CALL 文の引数への受け取り	SQL パラメタ	埋込み変数，SQL 変数，又は SQL パラメタ
	関数の RETURN 文に指定した値式から，戻り値への受け取り	値式	関数の戻り値

代入の種類	内　容	代入元	代入先
格納代入	列への値の代入	埋込み変数，SQL 変数，又は SQL パラメタ	列
	CALL 文の引数から手続きの入力パラメタへの代入	値式	SQL パラメタ
	関数呼出しの引数から関数のパラメタへの代入	値式	SQL パラメタ
	代入文（SET）による値の代入	SQL 変数， SQL パラメタ，又は埋込み変数	SQL 変数， SQL パラメタ，又は埋込み変数

(1)　代入先が固定長データの場合

代入先が固定長の文字列データ，混在文字列データ，又は各国文字列データの場合，代入元のデータの長さによって代入規則が異なります。代入先が固定長データの場合の代入規則を次の表に示します。

表 1-23　代入先が固定長データの場合の代入規則

代入の種類	データ長		
	代入元※1 ＞代入先	代入元※1 ＝代入先	代入元※1 ＜代入先
取り出し代入	△※2	○	△※3
格納代入	×	○	△※3

（凡例）

○：そのまま代入されます。

△：代入先のデータ長に合わせて，左寄せで代入されます。

×：エラーになります。

注※1

代入元が可変長のデータの場合，代入元のデータ長は実長となります。

注※2

データがあふれた右側部分は切り捨てられます。このとき，SQL 連絡領域の SQLWARN1 に警告情報が設定されます。また，標識変数の指定があると，切り捨てられる前のデータの長さが設定されます。

ただし，クライアント環境定義 PDCLTCNVMODE に UOC を指定して文字コード変換機能を使用している場合は，標識変数には切り捨て後の長さを設定します。

注※3

データが格納されていない右側部分に空白が埋められます。

(2) 代入先が可変長データ，長大データ，BINARY の場合

代入先が可変長の文字列データ，混在文字列データ，各国文字列データ，長大データ，又は BINARY の場合，代入元のデータの長さによって代入規則が異なります。代入先が可変長データの場合の代入規則を次の表に示します。

表 1-24 代入先が可変長データの場合の代入規則

代入の種類	データ長	
	代入元※1 ＞代入先※2	代入元※1 ＜代入先※2
取り出し代入	△※3	△※4
格納代入	×	△※4

（凡例）

△：代入先のデータ長※2 に合わせて，左寄せで代入されます。

×：エラーになります。

注※1

代入元が可変長のデータの場合，代入元のデータ長は実長となります。

代入元が位置付け子の埋込み変数の場合は，代入元のデータ長は位置付け子に割り当てられたデータの実長となります。

注※2

代入先のデータ長は，可変長データの最大長です。

代入先が位置付け子の埋込み変数の場合は，代入先のデータ長は次に示す長さとなります。

BLOB 位置付け子　：2147483647

BINARY 位置付け子 ：2147483647

注※3

データがあふれた右側部分は切り捨てられ，実長は代入先の最大長となります。このとき，SQL 連絡領域の SQLWARN1 に警告情報が設定されます。また，標識変数の指定があると，切り捨てられる前のデータの長さが設定されます。

ただし，クライアント環境定義 PDCLTCNVMODE に UOC を指定して文字コード変換機能を使用している場合は，標識変数には切り捨て後の長さを設定します。

注※4

代入元が固定長のデータの場合，そのデータ長が実長となります。

(3) 異なる文字集合間の代入規則

代入元と代入先の文字集合が異なる場合の代入はできません。ただし，次の表に示す場合は代入できます。

表 1-25　異なる文字集合間の代入ができる場合

代入元の文字集合	代入先の文字集合		
	既定文字集合	EBCDIK	UTF16
既定文字集合	○	○※1	○※2
EBCDIK	○※3	○	×
UTF16	○※4	×	○

（凡例）

○：代入できます。

×：代入できません。

注※1

代入元の文字集合が既定文字集合（SJIS）で代入先の文字集合が EBCDIK の場合に代入できる組み合わせは次のとおりです。

代入元の値式	代入先の値式
文字列定数又は埋込み変数	任意の項目

注※2

代入元の文字集合が既定文字集合（UTF-8）で代入先の文字集合が UTF16 の場合に代入できる組み合わせは次のとおりです。

代入元の値式	代入先の値式
文字列定数又は埋込み変数	任意の項目
上記以外の値式	埋込み変数

注※3

代入元の文字集合が EBCDIK で代入先の文字集合が既定文字集合（SJIS）の場合に代入できる組み合わせは次のとおりです。

代入元の値式	代入先の値式
値式	埋込み変数

注※4

代入元の文字集合が UTF16 で代入先の文字集合が既定文字集合（UTF-8）の場合に代入できる組み合わせは次のとおりです。

代入元の値式	代入先の値式
埋込み変数	任意の項目
値式	埋込み変数

代入元と代入先の文字集合が異なる場合の代入は次のようになります。

1. 取り出し代入の場合

- 代入先の文字集合に変換してから代入します。このとき，変換後の文字データ長が代入先のデータ長よりも長い場合は，データのあふれた右側部分は切り捨てられ，SQL 連絡領域の SQLWARN1 に警告情報が設定されます。また，標識変数の指定があると，切り捨てられる前のデータの長さ（バイト数）が設定されます。
- 代入先が固定長文字データで，代入する文字データ長が代入先のデータ長より短い場合は，代入先の文字集合で使用する空白を補って代入します。

2. 格納代入の場合

- 代入先の文字集合に変換してから代入します。このとき，変換後の文字データ長が代入先のデータ長よりも長い場合はエラーになります。
- 代入先が固定長文字データで，代入する文字データ長が代入先のデータ長より短い場合は，代入先の文字集合で使用する空白を補って代入します。

(4) 代入先，代入元の構造規則

代入先の構造と代入元の構造が異なると，代入できない場合があります。代入先，代入元の構造規則を次の表に示します。

表 1-26 代入先，代入元の構造規則

代入元の構造	代入先の構造	
	単純構造	繰返し構造
単純構造	○	×
繰返し構造	×	○

（凡例）

○：代入できます。

×：エラーになります。

注

添字付き繰返し列は，単純構造として扱います。

1.7　ナル値

ナル値とは，値がないこと，又は値が設定されていないことを示すために使用する特殊な値です。領域に値がないか，又は設定されていなければ，ナル値が設定されます。また，抽象データ型の値がナル値かどうかは，コンストラクタ関数によって値が生成されているかどうかを示すために使用します。

1.7.1　ナル値の扱い

ナル値の扱いについて次に示します。

(1)　検索結果の列の値の受け取り

標識変数の値でナル値を受け取ったかどうかが分かります。埋込み変数でナル値を受け取ることはできません。詳細は，「標識変数の値の設定」を参照してください。

(2)　表への値の格納

標識変数の値でナル値を格納したかどうかが分かります。埋込み変数ではナル値を表へ格納できません。詳細は，「標識変数の値の設定」を参照してください。

(3)　比較

NULL 述語以外の述語で，指定した値式，列，埋込み変数の値がナル値である行に対して，その述語は不定になります。ただし，埋込み変数でナル値を指定するには，標識変数が必要です。

(4)　結合

結合列にナル値を持つ行は，どのような行に対しても結合条件を満たしません。

(5)　ソート

昇順の場合は最後に出力します。降順の場合は，最初に出力します。

(6)　グループ分け

グループ分けの条件となる列にナル値を持つ行がある場合は，ナル値同士を同じ値として扱い，グループ分けします。

(7)　重複排除

ナル値同士は，重複するものとして扱います。

(8) 集合関数

集合関数では，基本的にナル値を無視します。ただし，COUNT（＊）の場合は，ナル値に関係なく条件を満たすすべての行を計算します。

(9) ウィンドウ関数

COUNT（＊） OVER（）の場合は，ナル値に関係なく条件を満たすすべての行を計算します。

(10) インデクス

ナル値を含む列に対してもインデクスを定義できます。

(11) 四則演算，日付演算，時刻演算，連結演算

データがナル値の場合，四則演算，日付演算，時刻演算，及び連結演算の結果もナル値になります。

(12) スカラ関数

VALUE 又は STRTONUM 以外のスカラ関数では，引数の値式のどちらかがナル値の場合，結果もナル値になります。スカラ関数 VALUE では，引数の値式がすべてナル値であれば，結果もナル値になります。スカラ関数 STRTONUM では，引数 1 の値式がナル値であれば，結果もナル値になります。

(13) CASE 式

CASE 略式の COALESCE の場合，引数の値式がすべてナル値であれば，結果もナル値になります。

(14) 抽象データ型

抽象データ型のナル値の扱いについて，抽象データ型にコンストラクタ関数を指定して値を生成している場合とそうでない場合に分けて，説明します。

- コンストラクタ関数を指定して値を生成していない場合

  抽象データ型全体の値はナル値になります。
- コンストラクタ関数を指定して値を生成している場合

  抽象データ型を構成する各属性の値に関係なく，抽象データ型全体としてはナル値でない値になります。抽象データ型を構成する各属性の値がナル値であっても，抽象データ型全体としての値はナル値にはなりません。

(15) 論理述語

論理値が不定であることと，論理値がナル値であることは等価です。

(16) 繰返し列

要素にはナル値を持つ場合もあります。また，列の要素が 0 の場合，列全体をナル値とします。

(17) WRITE 指定

WRITE 指定の場合，引数のどれかがナル値のときは，結果もナル値になります。

(18) GET_JAVA_STORED_ROUTINE_SOURCE 指定

次のどれかの条件を満たす場合，GET_JAVA_STORED_ROUTINE_SOURCE 指定の結果はナル値になります。

- 引数のうちで，どれかがナル値の場合
- 指定した JAR ファイルがインストールされていない場合
- JAR ファイル中に指定したクラスに対応するソースファイルがない場合

(19) CAST 指定

値式に NULL を指定するか，又は値式の結果がナル値の場合は，結果の値はナル値になります。

(20) 参照制約

外部キー構成列中にナル値が含まれる場合，その列は参照制約動作の対象になりません。

1.8 コンポネント指定

抽象データ型で定義された，抽象データ型の構成要素である属性を SQL 文中で指定します。

コンポネント指定：：＝項目指定..属性名〔..属性名〕…

コンポネント指定をする場合，次の規則があります。

- 属性に値を代入する場合，その属性を含む抽象データ型の値がコンストラクタ関数で生成されていなければなりません（その抽象データ型の値がナル値であってはなりません）。
- コンポネント指定によって属性を参照する場合，その属性を含む抽象データ型がナル値のときは（コンストラクタ関数でその値が生成されていないときは），参照値もナル値となります。
- 項目指定には，添字付きの列指定は指定できません。

1.9 ルーチン

ルーチンは，実装に使用する言語ごとに次の種別に分けられます。

- SQL ルーチン
  SQL によって実装したルーチン
- 外部ルーチン
  SQL 以外の言語によって実装したルーチン

また，ルーチンは定義 SQL ごとに次の二つに分けられます。

- 手続き
- 関数

1.9.1 手続き

CREATE PROCEDURE，又は CREATE TYPE の手続き本体で定義した手続きを示します。

ルーチンの実装に使用する言語ごとに次の種別に分けられます。

- SQL 手続き
  SQL で実装した手続き
- 外部手続き
  SQL 以外の言語で実装した手続き

なお，トリガ定義で作成されるトリガ動作手続きも SQL 手続きの一つですが，ルーチンとは別のスキーマ要素（トリガ）として CREATE TRIGGER で定義します。また，トリガ動作手続きは，指定したトリガ契機で自動的に実行され，CALL 文での呼び出しや，SQL パラメタでの値の受け渡しはできません。

1.9.2 関数

(1) ユーザ定義関数

CREATE FUNCTION，又は CREATE TYPE の関数本体で定義した関数，及びプラグインが提供する関数を示します。プラグインが提供する関数をプラグイン関数といいます。

関数には次の二つがあります。

- SQL 関数
  SQL で実装した関数

- 外部関数
  SQL 以外の言語で実装した関数

(2) システム定義関数

CREATE TYPE でシステムが生成するコンストラクタ関数を示します。

1.9.3 結果集合返却機能

結果集合返却機能を用いた場合，手続き中でカーソルを使用して検索した結果を，コール元で参照できます。

ここでは，手続きで結果集合を返却する方法，及び UAP で手続きが返却した結果集合を受け取る方法について説明します。

(1) SQL 手続き定義の場合

CREATE PROCEDURE の DYNAMIC RESULT SETS 句に，結果集合の最大数（コール元に返すカーソルの最大数）を指定します。また，手続きから結果集合として返却するカーソルのカーソル宣言では，WITH RETURN を指定します。

WITH RETURN を指定して宣言したカーソルを開いた状態のまま手続きを終了すると，カーソルの結果集合をコール元に返却できます。返却する結果集合が二つ以上ある場合は，カーソルを開いた順番で返却します。

SQL 手続きの定義例を次に示します。

```
CREATE PROCEDURE ORDERED_EMPS(IN REGION INTEGER)
    DYNAMIC RESULT SETS 2
    BEGIN
        DECLARE CUR1 CURSOR WITH RETURN
            FOR SELECT id_no, name FROM emps_1
                WHERE id_no < REGION ORDER BY id_no;
        DECLARE CUR2 CURSOR WITH RETURN
            FOR SELECT id_no, name FROM emps_2
                WHERE id_no < REGION ORDER BY id_no;
        OPEN CUR1;
        OPEN CUR2;
    END;
```

(2) 外部 Java 手続き定義の場合

CREATE PROCEDURE の DYNAMIC RESULT SETS 句に，結果集合の最大数（コール元に返すカーソルの最大数）を指定します。

外部 Java 手続きの定義例を次に示します。

```
CREATE PROCEDURE ORDERD_EMPS(IN REGION INTEGER)
    DYNAMIC RESULT SETS 2 LANGUAGE JAVA
    EXTERNAL NAME
      'jfile.jar:Routines3.orderedEmps
       (int,java.sql.ResultSet[],java.sql.ResultSet[])
       returns void'
    PARAMETER STYLE JAVA;
```

(3) 外部 Java 手続きの実体である Java メソッドの作成の場合

外部 Java 手続きの実体である Java メソッドの最後の引数に，java.sql.ResultSet[]型のパラメタを指定します。また，Java メソッドは SQL を実行して結果集合を受け取り，java.sql.ResultSet[]型のパラメタである変数を設定します。

Java メソッドの作成例を次に示します。

```
import java.sql.*;

public class Routines3 {
    public static void orderedEmps(int region,
        java.sql.ResultSet[] rs1, java.sql.ResultSet[] rs2)
    throws SQLException {
        java.sql.Connection conn=DriverManager.getConnection(
            "jdbc:hitachi:PrdbDrive","USER1","USER1");
        java.sql.PreparedStatement stmt1=
          conn.prepareStatement(
              "SELECT id_no,name FROM emps_1 WHERE id_no < ? ORDER BY id_no");
        stmt1.setInt(1, region);
        rs1[0]=stmt1.executeQuery();
        java.sql.PreparedStatement stmt2=
          conn.prepareStatement(
              "SELECT id_no,name FROM emps_2 WHERE id_no < ? ORDER BY id_no");
        stmt2.setInt(1, region);
        rs2[0]=stmt2.executeQuery();
        return;
    }
}
```

(4) 埋込み型 UAP の作成の場合

埋込み型 UAP から手続きを実行し，ALLOCATE CURSOR 文によって手続きから返却された結果集合の組がカーソルに割り当てられます。カーソルはその先頭の結果集合と関連づけられ，FETCH 文によって結果集合からデータを取り出すことができます。

二つ目以降の結果集合は結果集合の組に割り当てられます。前の結果集合と関連づけられているカーソルに対して CLOSE 文を実行すると，カーソルと関連づけられ，FETCH 文によって結果集合からデータを取り出すことができます。

C 言語による埋込み型 UAP の作成例を次に示します。

```
EXEC SQL WHENEVER SQLERROR GOTO error_end;
EXEC SQL CALL ORDERED_EMP(1000);
if (SQLCODE==120) {                    /* 結果集合の組が返却された */
    EXEC SQL ALLOCATE GLOBAL :cur1 FOR PROCEDURE ORDERED_EMP;
                                       /* カーソルを割り当てる */
                                       /* cur1にはカーソル名を指定する */
    while (1) {
        while (1) {
            EXEC SQL WHENEVER NOT FOUND DO break;
            EXEC SQL FETCH GLOBAL :cur1 INTO :emp_id, :emp_name;
            printf("ID No.=%s¥n", emp_id);
            printf("Name=%s¥n", emp_name);
        }
        EXEC SQL WHENEVER NOT FOUND DO break;
        EXEC SQL CLOSE GLOBAL :cur1;
    }
}
error_end:
```

(5) Java を用いた UAP の作成の場合

Java 言語で記述した UAP から手続きを実行し，手続きから送られてくる結果集合を受け取ります。

java.sql.PreparedStatement.execute()を使用して結果集合を受け取る場合の，Java を用いた UAP の作成例を次に示します。

```
java.sql.CallableStatement stmt=conn.prepareCall(
    "{call ordered_emps(?)}");
stmt.setInt(1,3);
stmt.execute();                              //結果集合を
java.sql.ResultSet rs=stmt.getResultSet();  //受け取る
while(rs.next()) {
    int id_no=rs.getInt(1);
    java.lang.String name=rs.getString(2);
    System.out.println("ID No.="+id_no);
    System.out.println("Name="+name);
    System.out.println();
}
rs.close();
while (stmt.getMoreResults()){
    rs = stmt.getResultSet();
    while(rs.next()) {
        int id_no=rs.getInt(1);
        java.lang.String name=rs.getString(2);
        System.out.println("ID No.="+id_no);
        System.out.println("Name="+name);
        System.out.println();
    }
    rs.close();
}
```

(6) 留意事項

外部 Java 手続き中で取得した DatabaseMetaData クラスの，メソッドが返却する結果集合（ResultSet）は，動的結果集合として返却できません。外部 Java 手続きのコール元のコネクションのメタデータを使用して，情報を取得してください。

1.10　外部ルーチン

この節では，外部ルーチンについて説明します。外部ルーチンは，SQL 以外の言語で記述したルーチンです。HiRDB では，次の外部ルーチンを使用できます。

- 外部 Java ルーチン
  Java 言語で記述した外部ルーチンです。外部 Java ルーチンは，次のように分類することができます。
  - 外部 Java 手続き
    Java 言語で実装した手続きです。
  - 外部 Java 関数
    Java 言語で実装した関数です。
- 外部 C ストアドルーチン
  C 言語で記述した外部ルーチンです。外部 C ストアドルーチンは，次のように分類することができます。
  - 外部 C 手続き
    C 言語で実装した手続きです。
  - 外部 C 関数
    C 言語で実装した関数です。

1.10.1　外部 Java ルーチン

この項では，外部 Java ルーチンに関連する次の項目について説明します。

- 外部 Java ルーチン名
- 型マッピング
- 結果集合返却機能

なお，AIX の場合は，POSIX ライブラリ版の HiRDB をセットアップ（pdsetup コマンドを実行）していないとき，又は POSIX ライブラリ版の HiRDB から非 POSIX ライブラリ版の HiRDB に再セットアップしたときは，外部 Java ルーチンを使用できません。pdsetup コマンドについては，マニュアル「HiRDB コマンドリファレンス」を参照してください。

(1)　外部 Java ルーチン名

外部 Java ルーチン名は，CREATE PROCEDURE 及び CREATE FUNCTION で指定できます。外部 Java 手続き及び外部 Java 関数を，外部ルーチンとして定義するためです。

(a)　形式

```
外部Javaルーチン名：：＝'JARファイル名:Javaメソッド名〔Javaシグネチャ〕'
```

```
Javaメソッド名：：＝Javaクラス名.メソッド識別子
Javaクラス名：：＝〔パッケージ名.〕クラス識別子
Javaシグネチャ：：＝（〔Javaパラメタ〕）〔returns 型名〕
Javaパラメタ：：＝型名〔，型名〕…
```

(b) 説明

JAR ファイル名

Java で定義したクラス又はパッケージの集合である，アーカイブファイルの名称を指定します。

JAR ファイル名は，パス名で指定しないでください。JAR ファイル名は，最大 255 バイトで指定できます。

メソッド識別子

実際に処理が記述されている Java メソッドの識別子を指定します。

メソッド識別子は，最大 255 バイトで指定できます。

パッケージ名

Java で定義したクラスの集合である，パッケージの名称を指定します。

クラス識別子

Java メソッドが定義されているクラスの識別子を指定します。

パッケージ名と合わせて，最大 255 バイトで指定できます。

returns 型名

Java メソッドの引数及び戻り値に対応する Java の型名を指定します。

外部 Java 手続きを定義する場合は，戻り値の型名を void（戻り値なし）にするか，"returns 型名"を省略してください。

外部 Java 関数を定義する場合は，"returns 型名"を必ず指定してください。ただし，Java シグネチャを省略する場合は"returns 型名"を指定する必要はありません。なお，外部 Java 関数では戻り値の型名に void（戻り値なし）は指定できません。

型名〔，型名〕…

Java メソッドの引数及び戻り値に対応する Java の型名を指定します。

規則を次に示します。

1. Java パラメタで記述する Java メソッドのパラメタの型名は，CREATE PROCEDURE 及び CREATE FUNCTION で指定した SQL パラメタ名の順番に記述してください。
2. SQL パラメタ名に対応する Java メソッドのパラメタの型名をすべて記述してください。
3. CREATE PROCEDURE の DYNAMIC RESULT SETS 句で結果集合の最大数を 1 以上にした場合，指定した結果集合の最大個数以下の型名"java.sql.ResultSet[]"を，Java パラメタの最後に指定してください。
4. CREATE FUNCTION で関数を定義する場合は，HiRDB のデータ型とマッピングできる Java データ型を指定します。

CREATE PROCEDURE で定義する手続きで SQL パラメタの入出力モードが IN の場合は，HiRDB のデータ型とマッピングできる Java データ型を指定します。

CREATE PROCEDURE で定義する手続きで SQL パラメタの入出力モードが OUT 又は INOUT の場合は，HiRDB のデータ型とマッピングできる Java データ型の一次元の配列型を指定します。

例えば，INTEGER や BLOB の出力パラメタに対しては，java.lang.Integer[]や byte[] []で指定します。なお，HiRDB のデータ型とマッピングできる Java データ型については，「型マッピング」を参照してください。

(c) 規則

1. Java シグネチャを省略すると，外部 Java 手続き又は外部 Java 関数定義で指定している HiRDB のデータ型と，SQL パラメタの入出力モードを基に，次に示す規則で Java データ型が決定されます。
   - CREATE FUNCTION で関数を定義する場合，Java メソッドのデータ型はマッピング規則に従って決定されます。
   - CREATE PROCEDURE で定義する手続きの SQL パラメタの入出力モードが IN の場合，Java メソッドのデータ型はマッピング規則に従って決定されます。
   - CREATE PROCEDURE で定義する手続きの SQL パラメタの入出力モードが OUT 又は INOUT の場合，Java メソッドのデータ型はマッピング規則の一次元の配列型で決定されます。
   - CREATE PROCEDURE の DYNAMIC RESULT SETS 句で結果集合の最大数を 1 以上指定して Java シグネチャを省略した場合は，マッピング規則で決定したデータ型から更に，java.sql.ResultSet[]型のパラメタが，Java メソッドのパラメタとして結果集合の最大個数分追加されます。

   例えば，INTEGER や BLOB の出力パラメタに対しては，java.lang.Integer[]や byte[] []が決定されます。なお，マッピング規則については，「型マッピング」を参照してください。
2. 外部ルーチン名は，必ずアポストロフィ（’）で囲んでください。
3. 外部ルーチン名で指定できる Java メソッドは，クラス定義の static で宣言したクラスメソッドだけです。
4. 各項目に指定できる文字を次に示します。
   - JAR ファイル名
     アルファベットの大文字と小文字，数字，'_'（アンダースコア），'$'，'.'，'-'
   - クラス識別子，メソッド識別子，型名
     アルファベットの大文字と小文字，数字，'_'（アンダースコア），'$'
   - パッケージ名
     アルファベットの大文字と小文字，数字，'_'（アンダースコア），'$'，'.'
5. パッケージ名，クラス識別子，メソッド識別子，及び型名の先頭一文字には数字を指定できません。
6. パッケージ名を省略した型名を記述する場合，HiRDB は次のようにパッケージ名を含む型名として解釈します。

省略した場合の記述形式	HiRDB が解釈するパッケージ名を含む型名
Integer	java.lang.Integer
Short	java.lang.Short
BigDecimal	java.math.BigDecimal
Double	java.lang.Double
Float	java.lang.Float
String	java.lang.String
Date	java.sql.Date
Time	java.sql.Time
Timestamp	java.sql.Timestamp
ResultSet	java.sql.ResultSet

(2) 型マッピング

ここでは，Java 言語で扱われるデータ型と，HiRDB で扱うデータ型のマッピングについて説明します。

外部ルーチン指定の Java シグネチャを省略した場合の暗黙的マッピングを次の表に示します。

表 1-27 外部ルーチン指定の Java シグネチャを省略した場合の暗黙的マッピング

HiRDB のデータ型	Java データ型（NULL 値可）
INT〔EGER〕	java.lang.Integer
SMALLINT	java.lang.Short
〔LARGE〕 DEC〔IMAL〕	java.math.BigDecimal
FLOAT, DOUBLE PRECISION	java.lang.Double
SMALLFLT, REAL	java.lang.Float
CHAR〔ACTER〕	java.lang.String
VARCHAR(n), CHAR〔ACTER〕 VARYING	
NCHAR, NATIONAL CHAR〔ACTER〕	
NVARCHAR, NATIONAL CHAR〔ACTER〕, NCHAR VARYING	
MCHAR	

HiRDB のデータ型	Java データ型（NULL 値可）
MVARCHAR	
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp
INTERVAL YEAR TO DAY	java.math.BigDecimal
INTERVAL HOUR TO SECOND	
BLOB, BINARY LARGE OBJECT	byte[]
BINARY	

外部ルーチン指定の Java シグネチャで指定できる Java データ型と HiRDB のデータ型のマッピングを次の表に示します。

表 1-28　外部ルーチン指定の Java シグネチャで指定できる Java データ型と HiRDB のデータ型のマッピング

HiRDB のデータ型	Java データ型 （NULL 値可）	Java データ型 （NULL 値不可）※1
INT〔EGER〕	java.lang.Integer	int
SMALLINT	java.lang.Short	short
〔LARGE〕 DEC〔IMAL〕	java.math.BigDecimal	－
FLOAT，又は DOUBLE PRECISION	java.lang.Double	double
SMALLFLT，又は REAL	java.lang.Float	float
CHAR〔ACTER〕	java.lang.String，又は byte[]※2	－※3
VARCHAR(n)，又は CHAR〔ACTER〕 VARYING		
NCHAR，又は NATIONAL CHAR〔ACTER〕		
NVARCHAR, NATIONAL CHAR〔ACTER〕，又は NCHAR VARYING		
MCHAR		
MVARCHAR		

HiRDB のデータ型	Java データ型 （NULL 値可）	Java データ型 （NULL 値不可）※1
DATE	java.sql.Date	－
TIME	java.sql.Time	
TIMESTAMP	java.sql.Timestamp	
INTERVAL YEAR TO DAY	java.math.BigDecimal	
INTERVAL HOUR TO SECOND		
BLOB，又は BINARY LARGE OBJECT	byte[]	
BINARY		
抽象データ型	－	

（凡例）

－：Java 言語に対応するデータ型がないことを示します。

注※1

NULL 値を設定すると，実行時にエラーになります。

注※2

HiRDB の文字列データ型に対応する Java データ型として，java.lang.String 又は byte[]のどちらかを指定できます。byte[]を指定した場合，文字コード変換はされません。

注※3

HiRDB と外部 Java ルーチンの間で，日本語データを String クラス及びそれを継承したクラスを介して受け渡す場合，Java の文字コードのマッピング規則（該当する文字コードと Unicode）によって，外字コードは正しく変換されないので注意してください。

(3) 非 POSIX ライブラリ版の HiRDB での SQL 実行可否（AIX 版限定）

非 POSIX ライブラリ版の HiRDB での SQL 実行可否を次の表に示します。非 POSIX ライブラリ版の HiRDB とは，AIX の場合に，pdsetup コマンドで POSIX ライブラリ版のロードモジュールをセットアップしていない HiRDB のことをいいます。

表 1-29 非 POSIX ライブラリ版の HiRDB での SQL 実行可否

分類	SQL 文	実行可否	実行できない条件
定義系 SQL	ALTER PROCEDURE	△	外部 Java 手続きの指定
	ALTER ROUTINE	△	外部 Java 手続き及び外部 Java 関数の指定
	CREATE FUNCTION	△	外部 Java ルーチンの使用
	CREATE PROCEDURE	△	外部 Java ルーチンの使用

分類	SQL 文	実行可否	実行できない条件
	CREATE TYPE	△	外部 Java ルーチンの使用
	CREATE VIEW	△	外部 Java 関数の指定
	DROP FUNCTION	○	−
	DROP PROCEDURE	○	−
	DROP SCHEMA	○	−
操作系 SQL	ASSIGN LIST 文	○	−
	CALL 文	△	• 外部 Java 手続き • 外部 Java ルーチンを使う手続き
	DELETE 文	○	−
	DROP LIST 文	○	−
	FREE LOCATOR 文	○	−
	INSERT 文	△	外部 Java 関数の使用
	PURGE TABLE 文	○	−
	1 行 SELECT 文	△	外部 Java 関数の使用
	動的 SELECT 文	△	外部 Java 関数の使用
	UPDATE 文	△	外部 Java 関数の使用
	代入文	△	外部 Java 関数の使用
制御系 SQL	LOCK 文	○	−
埋込み SQL 言語	INSTALL JAR	×	−
	REPLACE JAR	×	−
	REMOVE JAR	○	−
	INSTALL CLIB	○	−
	REPLACE CLIB	○	−
	REMOVE CLIB	○	−
その他	関数呼出し	△	• 外部 Java 関数の呼び出し • 外部 Java 関数を使う関数

(凡例)

○：実行できます。

△：条件によってはエラーとなります。

×：実行できません。

－：該当しません。

1.10.2　外部Ｃストアドルーチン

この項では，外部Ｃストアドルーチン名について説明します。なお，SQL を発行する外部Ｃストアドルーチンは実行できません。

(1)　外部Ｃストアドルーチン名

外部Ｃストアドルーチン名は，CREATE PROCEDURE 及び CREATE FUNCTION で指定できます。外部Ｃ手続き及び外部Ｃ関数を，外部ルーチンとして定義するために指定します。

(a)　形式

```
外部Cストアドルーチン名：：＝'Cライブラリファイル名!外部関数識別子'
```

(b)　説明

Ｃライブラリファイル名

Ｃ言語で実装した関数を含む，ライブラリファイルの名称を指定します。

Ｃライブラリファイル名は，パス名で指定しないで，ファイル名だけを指定してください。

外部関数識別子

外部Ｃストアドルーチンの処理が記述されているＣ関数の識別子を指定します。

(c)　規則

1. 各項目に指定できる文字を次に示します。
   - Ｃライブラリファイル名
     次に示す半角文字が使用できます。
     アルファベットの大文字と小文字，数字，'_'（アンダースコア），'$'，'.'，'-'
   - 外部関数識別子
     次に示す半角文字が使用できます。
     アルファベットの大文字と小文字，数字，'_'（アンダースコア）
2. 外部関数識別子の先頭一文字には数字を指定できません。
3. Ｃライブラリファイル名の長さと，外部関数識別子の長さの合計は，最大で 254 バイトです。

1.11　日時書式の指定

1.11.1　日時書式の概要と規則

(1)　概要

次の場合に日時書式を指定します。

- スカラ関数 VARCHAR_FORMAT で，日付データ，時刻データ，又は時刻印データを，既定以外の文字列表現に変換する場合
- スカラ関数 DATE，TIME，及び TIMESTAMP_FORMAT で，日付，時刻，又は時刻印の既定以外の文字列表現を，日付データ，時刻データ，又は時刻印データに変換する場合

(2)　日時書式の規則

1. 日時書式として指定できるものを次に示します。
   - 文字列定数，及び混在文字列定数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 連結演算
   - 集合関数（MAX，及び MIN）
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 日時書式のデータ型は，文字データ型（CHAR，VARCHAR），又は混在文字データ型（MCHAR，MVARCHAR）にしてください。
3. 日時書式の長さは，最大 240 バイトにしてください。ただし，変換対象の値式の文字集合が UTF16 の場合は，最大 480 バイトにしてください。

(3)　日時書式の要素

1. 日時書式で指定できる日時書式の要素とその意味を次の表に示します。

表 1-30　日時書式の要素とその意味

日時項目	日時書式の要素※1	意味
年	YYYY	4 けたで表した年（0001～9999）
	YY	2 けたで表した年（00～99）※4
月	MM	月（01～12）
	MON	月の省略名※2※3
	MONTH	月の名前※2※3
日	DD	日（01～該当する月の最終日）
時	HH	時（00～23）
分	MI	分（00～59）
秒	SS	秒（00～61）※8
小数秒	FF	小数秒※4※5
	NN…N	p けたの小数秒（p=N の数(1～6)）※6
その他	空白（ ）	区切り文字として使用できる要素
	ハイフン（-）	
	斜線（/）	
	コンマ（,）	
	ピリオド（.）	
	セミコロン（;）	
	コロン（:）	
	"文字列"	文字列自身を表す引用符で囲まれた文字列※7

注※1

日時書式の要素は，引用符（"）で囲まれた文字列以外は半角文字で指定してください。また，MON と MONTH の 1，2 文字目，及び引用符で囲まれた文字列中の文字以外の文字は，大文字，小文字の区別はされません。

注※2

MON，MONTH では，月の省略名，及び月の名前の大文字，小文字の指定ができます。大文字と小文字は，指定した日時書式の要素の 1，2 文字目に従います。

```
(例)
MONTH→JUNE
Month→June
month→june
```

日時書式の要素 MON，MONTH の 1，2 文字目と，月の名前，省略名の形式の関係を次の表に示します。

表 1-31　日時書式の要素 MON，MONTH の 1，2 文字目と，月の名前，省略名の形式の関係

2 文字目	1 文字目	
	大文字	小文字
大文字	すべて大文字	すべて小文字
小文字	1 文字目だけ大文字	すべて小文字

注※3

日時書式の要素を MON，MONTH で指定した場合の，各月の省略名及び名前を次の表に示します。

表 1-32　各月の省略名及び名前（日時書式の要素を MON，MONTH で指定した場合）

月	省略名	名前
1	JAN	JANUARY
2	FEB	FEBRUARY
3	MAR	MARCH
4	APR	APRIL
5	MAY	MAY
6	JUN	JUNE
7	JUL	JULY
8	AUG	AUGUST
9	SEP	SEPTEMBER
10	OCT	OCTOBER
11	NOV	NOVEMBER
12	DEC	DECEMBER

注※4

YY，FF は，スカラ関数 VARCHAR_FORMAT でだけ有効で，ほかのスカラ関数で指定した場合はエラーとなります。

注※5

FF では，結果の文字列表現の小数秒部分のけた数は，スカラ関数 VARCHAR_FORMAT の引数で指定した時刻印データの型に従います。小数秒精度が 0 の場合は，長さ 0 の文字列になります。

注※6

NN…N は，次の形式で変換されます。

- 日時を文字列表現に変換する場合

時刻印データの小数秒精度より p が小さければ切り捨て，大きければ拡張した小数秒部分を 0 で補います。

- 文字列表現を日時に変換する場合

  日時の文字列表現の小数秒部分のけた数は，p と一致している必要があります。

注※7

引用符で囲まれた文字列中に更に引用符を指定する場合，連続する二つの引用符（""）で表します。

注※8

システム共通定義の pd_leap_second が N の場合，秒の範囲は次のようになります。システム共通定義 pd_leap_second オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

日時項目	日時書式の要素	意味
秒	SS	秒（00～59）

(4) 日時書式の要素についての規則

1. 日時書式の文字列中には，区切り文字，引用符で囲まれた文字列以外は同じ日時項目の日時書式の要素を 2 回以上指定できません。
2. 日時を文字列表現に変換する場合に，変換対象の日時のデータと関係のない日時書式の要素が日時書式にあるとき，その部分は次に示す表にある文字列が補われます。

```
(例)
VARCHAR_FORMAT(DATE('2002-01-01'),'YYYY-MM-DD HH:MI')
→'2002-01-01 00:00'
```

3. 日時書式の要素に補う文字列を次の表に示します。

表 1-33 日時書式の要素に補う文字列

日時書式の要素	補う文字列
YYYY	現在の年（例：'2002'）
YY	現在の年の下 2 けた（例：'02'）
MM	現在の月（例：'08'）
MON	現在の月の省略名（例：'AUG'）
MONTH	現在の月の名前（例：'AUGUST'）
DD	現在の日（例：'05'）
HH	'00'
MI	'00'
SS	'00'
FF	'00'
NN…N	'00…0'（p けたの 0，p=N の数）

4. 文字列表現を日時に変換するスカラ関数で必要となる日時項目を次の表に示します。必要な日時項目がない場合はエラーとなります。

表 1-34　文字列表現を日時に変換するスカラ関数で必要となる日時項目

スカラ関数	必要な日時項目
DATE	年，月，日
TIME	時，分，秒
TIMESTAMP_FORMAT	年，月，日，時，分，秒

5. 文字列表現を日時に変換する場合，変換後のデータに関係ない日時書式の要素は，結果には反映されません。

6. 文字列表現を日時に変換する場合，文字列中の書式要素間に，指定した日時書式にはない余分な空白が含まれているときは，その空白は無視されます。

7. 日時書式を指定できるスカラ関数と日時書式の要素の関係を次の表に示します。

表 1-35　日時書式を指定できるスカラ関数と日時書式の要素の関係

日時書式の要素	VARCHAR_FORMAT			DATE	TIME	TIMESTAMP_FORMAT
	DATE 型※1	TIME 型※1	TIMESTAMP 型※1			
YYYY	○※2	○※2 (現在の年)	○※2	◎	○※5	◎
YY	○※2	○※2 (現在の年の下 2 けた)	○※2	×	×	×
MM	○※3	○※3 (現在の月)	○※3	◎※3	○※3※5	◎※3
MON	○※3	○※3 (現在の月の省略名)	○※3	◎※3	○※3※5	◎※3
MONTH	○※3	○※3 (現在の月の名前)	○※3	◎※3	○※3※5	◎※3
DD	○	○ (現在の日)	○	◎	○※5	◎
HH	○ ('00')	○	○	○※5	◎	◎
MI	○ ('00')	○	○	○※5	◎	◎
SS	○ ('00')	○	○	○※5	◎	◎

日時書式の要素	VARCHAR_FORMAT			DATE	TIME	TIMESTAMP_FORMAT
	DATE 型※1	TIME 型※1	TIMESTAMP 型※1			
FF	○※4 ('00')	○※4 ('00')	○※4	×	×	×
NN…N	○※4 ('00…0')	○※4 ('00…0')	○※4	○※5	○※5	○

(凡例)

◎：必ず指定する必要があり，1 回だけ指定できます。指定しないとエラーとなります。

○：1 回だけ指定できます。

×：指定できません。指定するとエラーとなります。

( )：括弧内は，変換される文字列を表します。

注※1

スカラ関数 VARCHAR_FORMAT で指定した型のデータを，文字列表現に変換する場合について表します。

注※2

YYYY，及び YY は，どちらか一つを 1 回だけ指定できます。

注※3

MM，MON，及び MONTH は，どれか一つを 1 回だけ指定できます。

注※4

FF，及び NN…N は，どちらか一つを 1 回だけ指定できます。

注※5

結果のデータ型と関係ないため，結果には反映されません。

1.12　インナレプリカ機能使用時の制限

HiRDB Staticizer Option を組み込んでインナレプリカ機能を使用している場合，及び更新可能なオンライン再編成を使用している場合は，次の制限があります。

- インナレプリカ機能使用時
  インナレプリカ機能を使用している RD エリアに対して，実行できる SQL に制限があります。
- 更新可能なオンライン再編成使用時
  オンライン再編成閉塞の RD エリアに対して，実行できる SQL に制限があります。

インナレプリカ機能使用時の SQL の実行可否を次の表に示します。

表 1-36　インナレプリカ機能使用時の SQL の実行可否

SQL		RD エリアの状態	
		オンライン再編成閉塞以外	オンライン再編成閉塞
定義系 SQL	ALTER TABLE	△※1	×
	CREATE INDEX	△※1	×
	CREATE TABLE	△※1※2	×
	DROP INDEX	△※1	×
	DROP SCHEMA	△※1	×
	DROP TABLE	△※1	×
	GRANT	△※1	△※1
操作系 SQL	ASSIGN LIST 文	○	△※1
	CALL 文	○	△※1
	DELETE 文	○	△※1
	FETCH 文	○	△※1
	INSERT 文	○	△※1
	PURGE TABLE 文	○	×
	1 行 SELECT 文	○	△※1
	動的 SELECT 文	○	△※1
	UPDATE 文	○	△※1
	代入文	○	△※1
制御系 SQL	LOCK 文	○	△※1

（凡例）

○：実行できます。

△：条件によってエラーになることがあります。

×：実行できません。

注※1

条件の詳細については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。

注※2

インナレプリカ機能を使用している RD エリアには，改竄防止表は作成できません。

1.13　位置付け子（locator）

1.13.1　位置付け子（locator）の概要と規則

(1)　概要

位置付け子はサーバ上にある特定のデータ値を識別する４バイトの値を持つデータです。位置付け子を用いることで，クライアント上にデータの実体を保持しないで，そのデータを扱う SQL の処理ができます。

(2)　規則

1. 位置付け子の種類を次の表に示します。

表 1-37　位置付け子の種類

種類	内容
BLOB 位置付け子	サーバ上の BLOB 型データ値を識別する値を持つ
BINARY 位置付け子	サーバ上の BINARY 型データ値を識別する値を持つ

2. 位置付け子の指定方法と指定できる箇所を次の表に示します。

表 1-38　位置付け子の指定方法と指定できる箇所

種類	指定方法	指定できる箇所
BLOB 位置付け子	埋込み変数	BLOB 型の埋込み変数が指定できる箇所
BINARY 位置付け子	埋込み変数	BINARY 型の埋込み変数が指定できる箇所 ただし，割り当てられたサーバ上のデータのデータ型が，最大長が 32,001 バイト以上の BINARY 型である位置付け子の埋込み変数の場合は，最大長が 32,001 バイト以上の BINARY 型の埋込み変数を指定できる箇所だけとなります。

3. サーバ上のデータ値を割り当てられていない位置付け子は無効です。
4. 位置付け子を次に示す箇所に指定することで，データ値が割り当てられ，その位置付け子は有効となります。有効な位置付け子は同じトランザクション中で，対応するデータ型のデータと同様に SQL 文中に指定できます。
   - 1 行 SELECT 文の INTO 句
   - FETCH 文の INTO 句
   - 代入文の代入先
   - 手続きの OUT パラメタ及び INOUT パラメタに対する CALL 文の引数
   - EXECUTE 文の INTO 句
5. 位置付け子は次の場合に無効となります。

- FREE LOCATOR 文に指定
- COMMIT 文の実行
- ROLLBACK 文の実行
- DISCONNECT 文の実行
- PURGE TABLE 文の実行時の自動的な COMMIT
- クライアント環境変数の PDCMMTBFDDL に YES を設定している場合の定義系 SQL 実行時の自動的な COMMIT
- 暗黙的ロールバックによるトランザクションの終了

6. 無効な位置付け子を，割り当てられたデータを扱う SQL 文中，又は FREE LOCATOR 文に指定した場合はエラーとなります。
7. 代入文形式 2 の代入先，及び代入値に位置付け子の埋込み変数を指定することで，サーバ上の同じデータを識別する位置付け子を複数作成した場合は，どれかを無効にしても，ほかの位置付け子は有効のままとなります。
8. 既にデータを割り当てられている有効な位置付け子に対して，再びほかのデータを割り当てた場合でも，元の位置付け子の値は有効のままとなります。
9. 位置付け子の値を UAP の記述言語によって書き換えた場合は，位置付け子が無効となります。又は書き換える前のデータと異なるデータを識別する場合があります。
10. 無効な位置付け子に対して，UAP の記述言語によって有効な位置付け子の値を代入した場合，その位置付け子は有効となり，代入元の位置付け子と同じデータを識別します。また，この場合にどれかを無効にすると，どちらの位置付け子も無効となります。

1.14　XML 型

XML 型は XML 文書を格納するための抽象データ型で，HiRDB XML Extension と連携することで使用できます。

XML 型の値は XQuery シーケンスとなります。XQuery シーケンスについては「XQuery」を参照してください。

XML 型に対しては，XML 型の値を処理する SQL（SQL/XML）を使用して，特定の条件に合致する XML 文書の絞込みや，XML 文書中の，特定の部分構造の値の取り出しなどの操作ができます。

1.14.1　XML 型の記述形式

(1)　形式

XML 型はデータ型を指定する箇所に，次に示す形式で指定します。

```
XML
```

(2)　データ形式

XML 型は XML 変換コマンド（phdxmlcnv）又は XML 変換ライブラリを使用して，XML 文書を解析した形式で処理します。この形式を ESIS-B 形式と呼びます。

(3)　最大長

XML 型の値は，ESIS-B 形式で 2,147,483,647 バイト以下である必要があります。ESIS-B 形式の長さは，XML 文書の長さだけではなく，XML 要素数などの構造にも依存します。

(4)　値の生成

XML 型の値は XML コンストラクタ関数又は XMLPARSE 関数で生成できます。

XML コンストラクタ関数については，「XML コンストラクタ関数」を参照してください。XMLPARSE 関数については，「XMLPARSE」を参照してください。

(5)　値の取り出し

UAP で XML 型の値を直接取り出すことはできません。XML 型の値は，次の方法で VARCHAR 型又は BINARY 型の値として取り出すことができます。

- XMLSERIALIZE 関数で，VARCHAR 型又は BINARY 型の値に変換する。

(6) 値式の指定

XML 型の値式として指定できるものを次に示します。

- 列指定
- XML コンストラクタ関数
- XMLQUERY 関数
- XMLAGG 集合関数
- XMLPARSE 関数

(7) 変換（代入，比較）できるデータ型

XML 型の値は，XML 型で定義された列，SQL 変数，SQL パラメタにだけ代入できます。ほかのデータ型の値は XML 型に代入できません。また，ほかのデータ型及び XML 型との比較はできません。

(8) ナル値

XML コンストラクタ関数を指定して値を生成していない場合，及び XML コンストラクタ関数の引数にナル値を指定した場合は，XML 型の値はナル値になります。また，XMLPARSE 関数の引数にナル値を指定した場合，XML 型の値はナル値になります。

(9) 使用上の注意事項

1. XML 型及び次の機能を使用するには，所有者を MASTER として HiRDB XML Extension と連携する必要があります。
   - XML コンストラクタ関数
   - XMLQUERY 関数
   - XMLSERIALIZE 関数
   - XMLEXISTS 述語
   - XMLAGG 集合関数
   - XMLPARSE 関数
   - CREATE INDEX 形式 3 （部分構造インデクス定義）
2. XML 型は，次に示す機能では使用できません。
   - CREATE INDEX 形式 1 によるインデクス定義
   - ソート
   - グループ分け
   - 集合演算，四則演算，連結演算
   - XMLAGG を除く集合関数

- 重複排除
- CASE 式
- CAST 指定
- 外への参照列

3. INSERT 文及び UPDATE 文で列に格納できる XML 型の値は，XML コンストラクタ関数で生成した値を，挿入値又は更新値として指定した場合だけです。

1.14.2 XML コンストラクタ関数

(1) 機能

XML 型の値を生成します。

(2) 形式

```
XML（ 値式 〔 AS データ型 〕 ）
```

(3) 規則

1. 値式に指定できるものを次に示します。
   - SQL 変数又は SQL パラメタ
   - 埋込み変数又は？パラメタ
2. 値式には XML 文書に対して XML 変換コマンド又は XML 変換ライブラリを使用し，出力された ESIS-B 形式の値を BINARY 型で指定してください。
3. 値式に埋込み変数又は？パラメタを指定する場合は，必ず値式のデータ型を AS データ型で指定してください。指定できるデータ型は BINARY 型だけです。AS データ型を指定した場合，埋込み変数及び？パラメタ以外は指定できません。
4. 結果のデータ型は XML 型になります。
5. 値式がナル値の場合は，結果もナル値になります。
6. XML コンストラクタ関数は，次の箇所に単独で指定できます。
   - INSERT 文の挿入値
   - UPDATE 文の更新値

(4) 使用例

書籍管理表に，埋込み変数（bookinfo）に格納された XML 文書を挿入します。

```
INSERT INTO 書籍管理表
     VALUES (310494321, XML(:bookinfo AS BINARY(102400)))
```

1.14.3 SQL/XML スカラ関数

(1) XMLQUERY

(a) 機能

XQuery を評価し，その結果の XML 型の値を生成します。

(b) 形式

```
XMLQUERY ( XQuery問合せ
           PASSING BY VALUE XML問合せ引数 〔, XML問合せ引数 〕…
            〔 RETURNING SEQUENCE 〔 BY VALUE 〕〕
           EMPTY ON EMPTY )

XQuery問合せ ::= 文字列定数
XML問合せ引数 ::= {XML問合せ文脈項目 | XML問合せ変数}
XML問合せ文脈項目 ::= 値式 〔 BY VALUE 〕
XML問合せ変数 ::= 値式 AS XQuery変数識別子 〔 BY VALUE 〕
```

(c) オペランド

- XQuery 問合せ ::= 文字列定数

評価する XQuery を文字列定数で指定します。XQuery の指定方法については，「XQuery」を参照してください。

- XML 問合せ引数 ::= {XML 問合せ文脈項目 | XML 問合せ変数}

XQuery に渡す引数を指定します。

XML 問合せ引数として XQuery に渡す，XQuery シーケンスに含まれる XQuery 項目の親プロパティは空になります。また，異なる XML 問合せ引数に指定した XQuery シーケンスに，同じ XML 型の値の同じ部分を表すノードが含まれていた場合でも，XQuery の評価ではそれらを異なるノードとして扱います。

- XML 問合せ文脈項目 ::= 値式 〔 BY VALUE 〕

XQuery の評価対象である文脈項目を指定します。

指定しない場合は，XQuery の評価対象の文脈項目は設定しないで，XQuery 問合せに指定した XQuery を一度評価します。

なお，この項目は PASSING 句中に一つしか指定できません。

値式

結果が XML 型の値となる値式を指定します。

この値式の結果である，XML 型の値の XQuery シーケンスを構成する各 XQuery 項目が，XQuery の評価対象の文脈項目となります。

値式の結果がナル値の場合は，XMLQUERY 関数の結果はナル値になります。

指定できるものを次に示します。

- 列指定
- 列指定によって導出した名前付き導出表の列

BY VALUE

指定の有無によって，XML 問合せ文脈項目に指定した値式の結果の値の渡し方は変わりません。ISO 規格との互換のためにサポートしています。

- **XML 問合せ変数：：＝ 値式 AS XQuery 変数識別子〔 BY VALUE 〕**

XQuery 問合せに指定された，XQuery 中の XQuery 変数に値を渡す場合に指定します。

値式

XQuery 問合せに指定された，XQuery 中の XQuery 変数に渡す値式を指定します。

値式の結果が XML 型でない場合は，XML 型の値である XQuery シーケンスに変換して渡します。

指定できるデータ型と，変換後の形式を次の表に示します。

表 1-39　XQuery 変数に渡す値のデータ型と，変換後の XML 型の値の形式

データ型	変換後の形式
INTEGER	xs:int 型の値※
SMALLINT	xs:int 型の値※
DECIMAL	xs:decimal 型の値※
FLOAT	xs:double 型の値※
SMALLFLT	xs:double 型の値※
CHAR	xs:string 型の値※
VARCHAR	xs:string 型の値※
MCHAR	xs:string 型の値※
MVARCHAR	xs:string 型の値※
DATE	xs:date 型の値※
TIME	xs:time 型の値※
TIMESTAMP	xs:dateTime 型の値※
XML	変換しません

データ型	変換後の形式
	（値式に指定した XML 型の値と同じ形式になります）

注※

一つの基本単位値から構成される XQuery シーケンスになります。

XQuery シーケンスに変換できない場合は，エラーとなります。

また，値式の結果がナル値の場合は，空の XQuery シーケンスである値を XQuery 変数に渡します。

値式に指定できるものを次に示します。

- 定数
- USER 値関数，CURRENT_DATE 値関数，CURRENT_TIME 値関数，CURRENT_TIMESTAMP 値関数
- 列指定
- SQL 変数又は SQL パラメタ
- 四則演算，日付演算，時刻演算又は連結演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ
- 上記のどれかの値式によって導出した名前付き導出表の列

AS XQuery 変数識別子

値を渡す XQuery 変数の識別子を指定します。

XQuery 変数識別子には，XQuery 問合せ中に指定した XQuery 変数名を指定します。指定方法については，「名前の指定」を参照してください。

ここで指定した XQuery 変数識別子の XML 名前空間は，XQuery 問合せでの既定の XML 名前空間となります。

ここで指定した XQuery 変数識別子に対応する XQuery 変数は，XQuery 問合せで指定した XQuery 全体が有効範囲になります。

同じ PASSING 句中のほかの XML 問合せ変数で指定した XQuery 変数識別子は指定できません。

BY VALUE

指定の有無によって，XML 問合せ変数に指定した値式の結果の値の渡し方は変わりません。ISO 規格との互換のためにサポートしています。

値式の結果が XML 型となる場合だけ指定してください。値式の結果が XML 型でない場合に指定すると，エラーとなります。

RETURNING SEQUENCE 〔 BY VALUE 〕

指定の有無によって，XQuery の評価結果の値の形式及び返却方法は変わりません。ISO 規格との互換のためにサポートしています。

(d) 規則

1. 結果は XML 型の値で XQuery シーケンスとなります。また，結果の値である XQuery シーケンスを構成する XQuery 項目の親プロパティは，空となります。
2. この関数は，関数呼出しが指定できる箇所にだけ指定できます。
3. INSERT 文で，この関数の結果又は結果を基にした XML 型の値を，列に格納することはできません。
4. UPDATE 文で，この関数の結果を列に格納する場合，次の条件をすべて満たす必要があります。
   - XML 問合せ文脈項目を指定している。
   - XQuery 問合せ中の XQuery 式が XQuery 変換式である。

(e) 使用例

書籍管理表の書籍情報列から，埋込み変数（category）で指定されたカテゴリの書籍のタイトルを取得します。

```
SELECT XMLSERIALIZE(
         XMLQUERY( '/書籍情報[カテゴリ=$CATEGORY]/タイトル'
                   PASSING BY VALUE 書籍情報,
                   :category AS CATEGORY
                   RETURNING SEQUENCE BY VALUE
                   EMPTY ON EMPTY)
         AS VARCHAR(32000))
       FROM 書籍管理表
```

(2) XMLSERIALIZE

(a) 機能

XML 型の値を直列化（文字列に変換）した，VARCHAR 型又は BINARY 型の値を生成します。

(b) 形式

```
XMLSERIALIZE (  〔 CONTENT 〕
                値式 AS データ型
                 〔 VERSION  '1.0'  〕
                 〔 { INCLUDING XMLDECLARATION
                    | EXCLUDING XMLDECLARATION } 〕 )
```

(c) オペランド

- CONTENT

指定の有無によって，結果の形式は変わりません。ISO 規格との互換のためにサポートしています。

結果の形式は，整形式 XML 文書の形式である必要はありません。

- **値式**

VARCHAR 型又は BINARY 型の値を生成する XML 型の値を指定します。

指定できるものを次に示します。

- 列指定
- XMLQUERY 関数
- XMLAGG 集合関数
- 列指定によって導出した名前付き導出表の列

- **AS データ型**

結果のデータ型を指定します。

指定できるデータ型を次に示します。

- VARCHAR 型（文字集合指定は指定できません）
- BINARY 型

結果が指定したデータ型の最大長に格納できない場合は，エラーとなります。

- 〔 VERSION ‘1.0’ 〕

結果の XML のバージョンを指定します。省略した場合は，’1.0’ を仮定します。

- 〔{ INCLUDING XMLDECLARATION ｜ EXCLUDING XMLDECLARATION }〕

結果の XML に XML 宣言（例：’<?xml version=”1.0” encoding=”UTF-8”?>’）を含めるかどうかを指定します。省略した場合は， EXCLUDING XMLDECLARATION を仮定します。

INCLUDING XMLDECLARATION
　結果の XML に XML 宣言を含める場合に指定します。

EXCLUDING XMLDECLARATION
　結果の XML に XML 宣言を含めない場合に指定します。

(d) 規則

1. 値式がナル値の場合は，結果もナル値になります。
2. 結果の文字コードは，pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で指定した文字コードになります。

3. 直列化した結果は，値式が示す XQuery シーケンス中の各 XQuery 項目を，次に示す規則に従って変換した結果を連結した文字列になります。ただし，隣接する基本単位値は，空白 1 文字を入れて連結します。

   - 文書ノードを直列化した結果は，その子ノードを直列化した結果を連結した文字列になります。
   - 要素ノードを直列化した結果は，次に示す要素ノードの文字列表現になります。子ノードを持つ要素ノードの場合は，開始タグから始まる文字列が直列化した結果になります。子ノードを持たない要素ノードの場合は，空要素タグが直列化した結果になります。

```
要素ノードの文字列表現 ：：= ｛ 開始タグ　子ノードの文字列表現　終了タグ ｜ 空要素タグ ｝
子ノードの文字列表現 ：：= ｛ 要素ノードの文字列表現 ｜ 処理命令ノードの文字列表現 ｜
                           コメントノードの文字列表現 ｜ テキストノードの文字列表現 ｝

開始タグ ：：= ＜ 修飾名 〔 空白 XML名前空間の文字列表現 〕…
                 〔 空白 属性ノードの文字列表現 〕… ＞
終了タグ ：：= ＜/ 修飾名 ＞
空要素タグ ：：= ＜ 修飾名 〔 空白 XML名前空間の文字列表現 〕…
                   〔 空白 属性ノードの文字列表現 〕… /＞
```

   XML 名前空間の文字列表現

   要素ノード以下で有効な XML 名前空間のうち，祖先の要素ノードで指定されていない XML 名前空間を表す XML 属性を，属性ノードの文字列表現で表した文字列になります。

   - 属性ノードを直列化した結果は，次に示す属性ノードの文字列表現になります。

```
属性ノードの文字列表現 ：：= 修飾名 等号演算子 “属性値“
```

   属性値

   属性ノードの文字列値に対して，次の文字を対応する文字列に置き換えた文字列になります。

置き換える文字	対応する文字列
&（アンパーサンド）	&amp;
＜（小なり演算子）	&lt;
＞（大なり演算子）	&gt;
“(引用符)	&quot;
‘（アポストロフィ）	&apos;

   - 処理命令ノードを直列化した結果は，次に示す処理命令ノードの文字列表現になります。

```
処理命令ノードの文字列表現：：= ＜? 処理命令ターゲット 空白 〔 処理命令ノードの内容 〕 ?＞
```

   - コメントノードを直列化した結果は，次に示すコメントノードの文字列表現になります。

```
コメントノードの文字列表現：：= ＜!-- コメントノードの内容 --＞
```

   - テキストノードを直列化した結果は，次に示すテキストノードの文字列表現になります。

```
テキストノードの文字列表現：：= テキスト値
```

テキスト値

テキストノードの内容に対して，次の文字を対応する文字列に置き換えた文字列になります。

置き換える文字	対応する文字列
&（アンパーサンド）	&amp;
＜（小なり演算子）	&lt;
＞（大なり演算子）	&gt;

- 基本単位値を直列化した結果は，xs:string 型の基本単位値に変換した文字列に対して，次の文字を対応する文字列に置き換えた文字列になります。

置き換える文字	対応する文字列
&（アンパーサンド）	&amp;
＜（小なり演算子）	&lt;
＞（大なり演算子）	&gt;

4. この関数は，関数呼出しが指定可能な箇所にだけ指定できます。

(e) 使用例

書籍管理表の書籍情報列の値を，VARCHAR 型の値として取得します。

```
SELECT XMLSERIALIZE(書籍情報 AS VARCHAR(32000)
                    INCLUDING XMLDECLARATION) FROM 書籍管理表
```

(3) XMLPARSE

(a) 機能

XML 文書から XML 型の値を生成します。

(b) 形式

```
XMLPARSE(  DOCUMENT 値式
             〔 AS データ型 〕
             〔 WHITESPACE指定 〕  )

WHITESPACE指定 ：：＝  { PRESERVE | STRIP } WHITESPACE
```

(c) オペランド

- DOCUMENT

値式に指定した XML 文書が，整形式 XML 文書であることを表します。このオペランドを省略することはできません。

- **値式**

XML 型の値を生成する整形式 XML 文書を指定します。指定できるデータ型を次に示します。

- CHAR 型（文字集合指定は指定できません）
- VARCHAR 型（文字集合指定は指定できません）
- MCHAR 型
- MVARCHAR 型
- BINARY 型

- **AS データ型**

値式のデータ型を指定します。値式に埋込み変数又は？パラメタを指定する場合は，AS データ型を必ず指定してください。AS データ型を指定した場合は，値式に埋込み変数及び？パラメタ以外を指定できません。

- WHITESPACE 指定 ：：＝ ｛PRESERVE ｜ STRIP｝ WHITESPACE

XML 文書に含まれる空白類の取り扱い方を指定します。空白類は，空白(X’20’)，水平タブ(X’09’)，改行(X’0A’)，及び復帰(X’0D’)です。省略時は，STRIP WHITESPACE を仮定します。

なお，WHITESPACE 指定に関係なく，復帰(X’0D’)及び復帰改行(X’0D0A’)は，改行(X’0A’)に置き換えます。続いて，WHITESPACE 指定に従った処理を行います。

PRESERVE WHITESPACE

すべての空白類を保持します。

STRIP WHITESPACE

テキストノードに含まれる空白類に対して，次の正規化処理を行います。ただし，xml:space=”preserve”属性を持つ要素の子孫のテキストノードは除きます。

- 先頭及び末尾の空白類を除去
- 連続する空白類を一つの空白(X’20’)に置き換え

(d) 規則

1. 値式がナル値の場合は，結果もナル値になります。
2. この関数は，関数呼出しが指定できる箇所にだけ指定できます。
3. DTD，XML スキーマ，コメント，及び処理命令が記述されていても無視し，記述されていないものとして処理します。値式に指定した XML 文書の妥当性は，検証しません。
4. 使用できる実体は，定義済み実体（< > & ' &quat;）だけです。これ以外の内部実体及び外部実体は使用できません。使用した場合は文字として扱います。
5. XML 宣言の encoding 属性に指定できる文字コード種別を，次の表に示します。省略時は，UTF-8 が仮定されます。

XML 文書の encoding 属性	HiRDB の文字コード		
	sjis （シフト jis 漢字）	ujis （EUC 日本語漢字）	utf-8 （Unicode（UTF-8））
Shift_JIS	○	×	×
EUC-JP	×	○	×
UTF-8	×	×	○
US-ASCII	○	○	○
上記以外	×	×	×

（凡例）

○：指定できます。

×：指定できません。

6. 値式には 5 メガバイト以内の XML 文書を指定してください。
7. 要素名及び属性名は，4,096 バイト以内で記述してください。
8. 要素のネスト数は，100 以内で記述してください。
9. 接頭辞が記述されている場合，HiRDB XML Extension のバージョンによって規則が異なります。
   - HiRDB XML Extension のバージョンが 08-05 より前の場合

     接頭辞が記述されていても，接頭辞が指定されていないものとして処理し，すべての要素及び属性は既定の XML 名前空間として扱います。
   - HiRDB XML Extension のバージョンが 08-05 以降の場合

     XML Extension の XML データ型プラグインの環境設定（XMLPARSE 名前空間処理指定）に従います。XML データ型プラグインの環境設定については，マニュアル「HiRDB XML 拡張機能 HiRDB XML Extension」を参照してください。

(e) 使用例

書籍管理表に BINARY 型の埋込み変数（bookxml）に格納された XML 文書を挿入します。

```
INSERT INTO 書籍管理表
  VALUES(310494321, XMLPARSE(DOCUMENT :bookxml AS BINARY(32000)))
```

(f) 留意事項

1. この関数を使用するには，バージョンが 08-04 以降の HiRDB XML Extension が必要です。
2. HiRDB サーバと HiRDB クライアントで文字コードが異なる場合は，値式のデータ型を BINARY 型にしてください。

1.14.4 SQL/XML 述語

(1) XMLEXISTS 述語

(a) 機能

XQuery を評価した結果の XML 型の値が，一つ以上の XQuery 項目から構成される XQuery シーケンスであるかどうかを判別します。

(b) 形式

```
XMLEXISTS ( XQuery問合せ
            PASSING BY VALUE XML問合せ引数 〔, XML問合せ引数 〕… )

XQuery問合せ ::= 文字列定数
XML問合せ引数 ::= {XML問合せ文脈項目 | XML問合せ変数}
XML問合せ文脈項目 ::= 値式 〔 BY VALUE 〕
XML問合せ変数 ::= 値式 AS XQuery変数識別子 〔 BY VALUE 〕
```

(c) オペランド

- XQuery 問合せ ::= 文字列定数

評価する XQuery を文字列定数で指定します。XQuery の指定方法については，「XQuery」を参照してください。

- XML 問合せ引数 ::= {XML 問合せ文脈項目 | XML 問合せ変数}

XQuery に渡す引数を指定します。

XML 問合せ引数として XQuery に渡す，XQuery シーケンスに含まれる XQuery 項目の親プロパティは空になります。また，異なる XML 問合せ引数に指定した XQuery シーケンスに，同じ XML 型の値の同じ部分を表すノードが含まれていた場合でも，XQuery の評価ではそれらを異なるノードとして扱います。

- XML 問合せ文脈項目 ::= 値式 〔 BY VALUE 〕

XQuery の評価対象である文脈項目を指定します。

指定しない場合は，XQuery の評価対象の文脈項目は設定しないで，XQuery 問合せに指定した XQuery を一度評価します。

なお，この項目は PASSING 句中に一つしか指定できません。

値式

結果が XML 型の値となる値式を指定します。

この値式の結果である，XML 型の値の XQuery シーケンスを構成する各 XQuery 項目が，XQuery の評価対象の文脈項目となります。

値式の結果がナル値の場合は，XMLEXISTS 述語は不定となります。
指定できるものを次に示します。

- 列指定
- 列指定によって導出した名前付き導出表の列

BY VALUE

指定の有無によって，XML 問合せ文脈項目に指定した値式の結果の値の渡し方は変わりません。ISO 規格との互換のためにサポートしています。

- **XML 問合せ変数 ：：= 値式 AS XQuery 変数識別子 〔 BY VALUE 〕**

XQuery 問合せに指定された，XQuery 中の XQuery 変数に値を渡す場合に指定します。

値式

XQuery 問合せに指定された，XQuery 中の XQuery 変数に渡す値式を指定します。
値式の結果が XML 型でない場合は，XML 型の値である XQuery シーケンスに変換して渡します。
指定できるデータ型と，変換後の形式を次の表に示します。

表 1-40　XQuery 変数に渡す値のデータ型と変換後の XML 型の値の形式

データ型	変換後の形式
INTEGER	xs:int 型の値※
SMALLINT	xs:int 型の値※
DECIMAL	xs:decimal 型の値※
FLOAT	xs:double 型の値※
SMALLFLT	xs:double 型の値※
CHAR	xs:string 型の値※
VARCHAR	xs:string 型の値※
MCHAR	xs:string 型の値※
MVARCHAR	xs:string 型の値※
DATE	xs:date 型の値※
TIME	xs:time 型の値※
TIMESTAMP	xs:dateTime 型の値※
XML	変換しません （値式に指定した XML 型の値と同じ形式になります）

注※

一つの基本単位値から構成される XQuery シーケンスになります。

XQuery シーケンスに変換できない場合は，エラーとなります。

また，値式の結果がナル値の場合は，空の XQuery シーケンスである値を XQuery 変数に渡します。

値式に指定できるものを次に示します。

- 定数
- USER 値関数，CURRENT_DATE 値関数，CURRENT_TIME 値関数，CURRENT_TIMESTAMP 値関数
- 列指定
- SQL 変数又は SQL パラメタ
- 四則演算，日付演算，時刻演算，又は連結演算
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ
- 上記のどれかの値式から導出した名前付き導出表の列

AS XQuery 変数識別子

値を渡す XQuery 変数の識別子を指定します。

XQuery 変数識別子には XQuery 問合せ中に指定した XQuery 変数名を指定します。指定方法については，「名前の指定」を参照してください。

ここで指定した XQuery 変数識別子の XML 名前空間は XQuery 問合せでの既定の XML 名前空間となります。

ここで指定した XQuery 変数識別子に対応する XQuery 変数は，XQuery 問合せで指定した XQuery 全体が有効範囲になります。

同じ PASSING 句中のほかの XML 問合せ変数で指定した XQuery 変数識別子は指定できません。

BY VALUE

指定の有無によって，XML 問合せ変数に指定した値式の結果の値の渡し方は変わりません。ISO 規格との互換のためにサポートしています。

値式の結果が，XML 型となる場合だけ指定してください。値式の結果が，XML 型でない場合に指定すると，エラーとなります。

(d) 述語が真となる場合

XQuery を評価した結果の XML 型の値が一つ以上の XQuery 項目から構成される XQuery シーケンスである場合に，XMLEXISTS 述語は真になります。

(e) 規則

1. この述語は，WHERE 句にだけ指定できます。

2. 副問合せ中には指定できません。

(f) 使用例

書籍情報列に，書籍管理表から埋込み変数（price）で指定された値以上の価格である書籍情報が格納された行の書籍 ID 列の値を取得します。

```
SELECT 書籍ID FROM 書籍管理表
  WHERE XMLEXISTS( '/書籍情報[価格>=$PRICE]'
                  PASSING BY VALUE 書籍情報,
                  :price AS PRICE)
```

1.14.5 SQL/XML 集合関数

(1) XMLAGG

(a) 機能

引数に指定した XML 型の各行の値を連結した XML 型の値を，行の集まりから生成します。

(b) 形式

```
XMLAGG ( 値式
          〔 RETURNING SEQUENCE 〕 )
```

(c) オペランド

- 値式

連結する XML 型の値式を指定します。

指定できるものを次に示します。

- 列指定
- XMLQUERY 関数
- 列指定によって導出した名前付き導出表の列

- RETURNING SEQUENCE

指定の有無によって，連結した結果の値の形式は変わりません。ISO 規格との互換のためにサポートしています。

(d) 規則

1. 結果は XML 型の値で XQuery シーケンスとなります。

2. 連結する順序は保証しません。
3. 値式の結果がナル値の行は無視します。
4. 適用対象が 0 件，又はナル値だけの場合は，結果はナル値となります。
5. INSERT 文及び UPDATE 文で，この関数の結果又は結果を基にした XML 型の値を，列に格納することはできません。

その他の規則については，「集合関数」の規則に従います。

(e) 使用例

書籍管理表のすべての行の書籍情報列から，タイトルが”SQL 徹底解説”の書籍情報と同じカテゴリの書籍情報を取得します。

```
SELECT XMLSERIALIZE(
  XMLQUERY(
     '$BOOKS/書籍情報[カテゴリ=$BOOKS/書籍情報[タイトル=”SQL徹底解説”]/カテゴリ]'
    PASSING BY VALUE XMLAGG(書籍情報 ) AS BOOKS
    RETURNING SEQUENCE BY VALUE EMPTY ON EMPTY)
AS VARCHAR(32000))
  FROM 書籍管理表
```

1.14.6 XML 型用定義系 SQL

(1) CREATE INDEX 形式 3 （部分構造インデクス定義）

(a) 機能

XML 型列の，値の特定の部分構造をキーとするインデクスを定義します。

(b) 使用権限

表の所有者

公用 RD エリアに，自分が所有する表のインデクスを定義できます。

私用 RD エリアの利用権限を持つ表の所有者

利用権限を持つ私用 RD エリアに，自分が所有する表のインデクスを定義できます。

(c) 形式＜部分構造インデクスの定義＞

```
CREATE 〔UNIQUE〕 INDEX 〔認可識別子.〕インデクス識別子
      ON 〔認可識別子.〕表識別子（列名〔 {ASC｜DESC} 〕）
       〔IN {RDエリア名
            ｜（RDエリア名）
            ｜（（RDエリア名）〔,（RDエリア名）〕…）
            ｜マトリクス分割インデクス格納用RDエリア指定} 〕
```

```
      KEY 〔 USING UNIQUE TAG 〕FROM 部分構造指定 AS データ型
       〔インデクスオプション〕…

マトリクス分割インデクス格納用RDエリア指定 ：：＝ 2次元格納用RDエリア指定
2次元格納用RDエリア指定：：＝
            （マトリクス分割用RDエリアリスト〔，マトリクス分割用RDエリアリスト〕…）
マトリクス分割用RDエリアリスト ：：＝ （RDエリア名〔，RDエリア名〕…）
部分構造指定 ：：＝ 文字列定数
インデクスオプション ：：＝ {PCTFREE=未使用領域の比率
                           |UNBALANCED SPLIT
                           |EMPTY}
```

(d) オペランド

KEY 句以外のオペランド及びオペランド規則については「CREATE INDEX　形式 1（インデクス定義）」を参照してください。

- KEY 〔USING UNIQUE TAG〕 FROM 部分構造指定 AS データ型

USING UNIQUE TAG

一つの XML 型列の値の中で，部分構造指定に該当する部分構造が一意に決まる場合に指定します。ただし，部分構造指定に該当する部分構造が XML 属性の場合は，その XML 属性を持つ XML 要素が一意に決まる必要があります。

CREATE の直後の UNIQUE オペランドを指定した場合は，この指定が仮定されます。

部分構造指定 ：：＝ 文字列定数

キーとする部分構造を表す文字列定数を，部分構造パスの形式の値で指定します。

部分構造パスの形式を次に示します。なお部分構造パス中のキーワードは，すべて小文字で指定します。

```
部分構造パス ：：＝ 〔XML名前空間宣言〕… 部分構造パス式
XML名前空間宣言 ：：＝ { declare namespace 接頭辞 = XML名前空間URI;
                       | declare default element namespace XML名前空間URI;}

部分構造パス式 ：：＝ /ステップ式〔/ステップ式〕…
ステップ式 ：：＝ 〔 @ 〕 修飾名
修飾名 ：：＝ 〔接頭辞:〕局所名
```

- declare namespace 接頭辞 = XML 名前空間 URI;

  部分構造パス式中に接頭辞付きの修飾名を含む場合，その修飾名の接頭辞の XML 名前空間を宣言します。同じ接頭辞の XML 名前空間は複数宣言できません。

  接頭辞

  部分構造パス式中の修飾名の接頭辞を指定します。

  大文字小文字を区別します。

  XML 名前空間 URI

  接頭辞に関連付ける XML 名前空間の URI を引用符（”）又はアポストロフィ（’）で囲んで指定します。

  大文字小文字を区別します。

- declare default element namespace XML 名前空間 URI;

  部分構造パス式の既定の XML 名前空間を宣言します。部分構造パス式中の接頭辞のない修飾名の XML 名前空間は，ここで宣言した XML 名前空間となります。

  部分構造パス中には，既定の XML 名前空間を複数宣言できません。

  この指定を省略した場合，既定の XML 名前空間の XML 名前空間 URI は，’ http://www.w3.org/XML/1998/namespace’ を仮定します。

  XML 名前空間 URI

  接頭辞を省略した修飾名に対する XML 名前空間の URI を引用符（”）又はアポストロフィ（’）で囲んで指定します。

  大文字小文字を区別します。

- **部分構造パス式**

  キーとする部分構造を表すパスを指定します。

  各指定項目の意味を次の表に示します。

指定項目	意味
先頭の斜線(/)	斜線の後のステップ式が示す要素が，XML 型の値の中の最上位の XML 要素であることを意味します。
途中の斜線(/)	斜線の後のステップ式が示す XML 要素又は XML 属性が，斜線の前のステップ式が示す XML 要素の，子の XML 要素又は XML 属性であることを意味します。
ステップ式	XML 要素又は XML 属性を意味します。 ただし，XML 属性を意味するステップ式は最後にだけ指定できます。
@	@の後の修飾名が XML 属性の名前であることを意味します。 @がない場合は，修飾名は XML 要素の名前となります。
修飾名	XML 要素又は XML 属性の名前を意味します。
接頭辞:	後に続く局所名が，接頭辞が示す XML 名前空間中の局所名であることを意味します。 XML 名前空間宣言で XML 名前空間に対して割り当てた接頭辞を指定します。 「接頭辞:」を省略した場合は，局所名は既定の XML 名前空間中の名前となります。
局所名	XML 名前空間中での XML 要素又は XML 属性の名前を意味します。

部分構造指定の指定例を次の表に示します。

項番	指定例	意味
1	‘/書籍情報/カテゴリ’	最上位の XML 要素である「書籍情報」の子の XML 要素「カテゴリ」をキーとします。
2	‘/書籍情報/@書籍 ID’	最上位の XML 要素である「書籍情報」の属性「書籍 ID」をキーとします。
3	‘declare namespace b=”http://www.foo.co.jp/bookinfo”; /b:書籍情報/b:カテゴリ’	最上位の XML 要素である「書籍情報」の子の XML 要素「カテゴリ」をキーとします。ただし，「書籍情報」及び「カテゴリ」の XML 要素の名前はの XML 名前空間 URI が”

項番	指定例	意味
		http://www.foo.co.jp/bookinfo”である XML 名前空間中の名前となります。
4	‘declare default element namespace ”http://www.foo.co.jp/bookinfo”; /書籍情報/カテゴリ’	項番 3 と同じ意味となります。

データ型

キー値のデータ型を指定します。

指定できるデータ型を次に示します。

・INTEGER

・DECIMAL

・FLOAT

・VARCHAR（文字集合指定は指定できません）

キー値は，キーとする部分構造に対応する XQuery データモデルでのノードの型付き値プロパティの値を，ここで指定した SQL のデータ型に変換した値になります。

インデクス定義時に，既に列に格納されている値のキーとする部分構造の値が指定した SQL のデータ型に変換できない場合は，インデクスは定義できません。また，インデクスを定義した列に対する値の挿入又は値の更新時に，その挿入値又は更新値のキーとする部分構造の値が指定した SQL のデータ型に変換できない場合は，挿入又は更新ができません。

各ノードが持つ型付き値プロパティの値については，「XQuery データモデル」を参照してください。型付き値プロパティの値の XQuery データ型と SQL のデータ型の対応関係を，次の表に示します。

表 1-41　XQuery データ型と SQL のデータ型の対応関係

部分構造に対応する XQuery データモデルのノードの型付き値プロパティの XQuery データ型	SQL のデータ型			
	INTEGER	DECIMAL	FLOAT	VARCHAR
xs:untypedAtomic	△	△	△	△
xs:int	○	×	×	×
xs:decimal	×	○	×	×
xs:double	×	×	○	×
xs:string	×	×	×	○
その他	×	×	×	×

（凡例）

○：等価であるため，そのままキー値になります。

△：型付き値プロパティの値を文字列として見た場合に，SQL のデータ型に対応した文字列表現であれば，変換されキー値になります。この場合，インデクスが定義されていない場合と結果が異なるおそれがあります。それ以外は変換できません。

×：変換できません。

(e) 共通規則

1. インデクスは，一つの表に最大 255 個定義できます。
2. インデクスは，ナル値を含む列，指定した部分構造がない XML 型の値を含む列，又は行のない列に対しても定義できます。
3. 部分構造インデクスは XML 型の列に対してだけ指定できます。
4. キー値の長さは，以下の計算式を満たすようにしてください。

   キー値の長さ≦

   MIN((インデクス格納用 RD エリアのページサイズ÷2)－1242，4036)

   各キー値の長さを次の表に示します。

表 1-42　キー値の長さ

データ型		キー値の長さ
INTEGER		4
DECIMAL〔(m〔，n〕)〕		↓m÷2↓＋1
FLOAT		8
VARCHAR	実際のデータ長が 255 バイト以下	n1+1
	実際のデータ長が 256 バイト以上	n1+2

（凡例）

m，n：正の整数

n1：実際のデータ長

5. 同じ列の同じ部分構造をキーとするインデクスは，一つだけ定義できます。
6. インデクスを定義した部分構造を XQuery 中で評価する場合は，「AS データ型」で指定した SQL のデータ型に対応する XQuery データ型の値として評価します。各 SQL のデータ型に対応する XQuery データ型を次の表に示します。

表 1-43　XQuery を評価する時の XQuery データ型

SQL のデータ型	XQuery データ型
INTEGER	xs:int
DECIMAL	xs:decimal
FLOAT	xs:double
VARCHAR	xs:string

インデクスを定義した部分構造に対応する，XQuery データモデルのノードの型付き値プロパティの XQuery データ型が xs:untypedAtomic 型であった場合，その部分構造に対してインデクスを定義した場合と，定義していない場合とで，XQuery 中で評価する XQuery データ型が異なります。このため，インデクスを削除すると，結果が異なる場合があります。

7. インデクスを定義しようとしている表に対して手続き及びトリガが定義されている場合，その SQL オブジェクト中のインデクス情報は無効になります。この場合，トリガは実行できなくなります。また，手続きからこの手続き又はトリガは実行できなくなるため，SQL オブジェクトを再作成する必要があります。
8. 同一のインデクスオプションは，繰り返して指定できません。
9. 実行中の SQL オブジェクト中のインデクス情報が無効になる場合，Java 手続き中から CREATE INDEX は実行できません。
10. インデクスを格納する RD エリアに，インナレプリカ機能を適用している RD エリアと適用していない RD エリアは混在して指定できません。インナレプリカ機能を適用している RD エリアを指定する場合，RD エリア名にはオリジナル RD エリア名を指定します。
11. インナレプリカ機能を使用している場合の CREATE INDEX の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
12. 一つの RD エリアに格納できるインデクスは，最大 500 個です。
13. インデクスを格納する RD エリアに一時表用 RD エリアは指定できません。

(f) 留意事項

1. インデクスが付いている列の値をユーザが更新すると，インデクスも更新されます。
2. CREATE INDEX は，OLTP 下の X/Open に従った UAP からは指定できません。
3. EMPTY を指定してインデクスを定義した場合，データベース再編成ユティリティでインデクスを再作成する必要があります。インデクスの再作成については，マニュアル「HiRDB コマンドリファレンス」を参照してください。
4. 大量のデータが登録されている表に対して CREATE INDEX 文でインデクスを作成する場合，処理時間が長くなるおそれがあります。CREATE INDEX 文を実行する前に，タイマ監視に関する指定を次のように設定してください。
   - クライアント環境定義 PDCWAITTIME

     実績やデータ量から CREATE INDEX の実行時間を推測できる場合は，0 以外の余裕を持たせた値を指定してください。

     実行時間を推測できない場合は，0 又は指定を省略してください。
5. HiRDB/パラレルサーバでサーバ間横分割した表にインデクスを定義する場合で，表格納用 RD エリア，又は指定したインデクス格納用 RD エリアが閉塞状態のときは，RD エリアの排他待ちとなります。このとき，排他待ちの時間がシステム定義 pd_lck_wait_timeout オペランドの指定値に達してタイムアウトしても，すぐにエラーリターンされないことがあります。これを避けるため，CREATE INDEX 文を実行する前に，表格納用 RD エリア，又は指定したインデクス格納用 RD エリアの閉塞状態を解除してください。

(g) 使用例

1. 書籍管理表の書籍情報列に対して，「書籍情報」XML 要素の子の「カテゴリ」XML 要素の値を VARCHAR 型に変換した値をキー値とするインデクス（INDX1）を定義します。

```
CREATE INDEX INDX1 ON 書籍管理表(書籍情報)
  KEY FROM ‘/書籍情報/カテゴリ’ AS VARCHAR (100)
```

2. 書籍管理表の書籍情報列に対して，「書籍情報」XML 要素の子の「価格」XML 要素の値を INTEGER 型に変換した値をキー値とするインデクス（INDX2）を定義します。ただし，各行の値に「書籍情報」XML 要素の子の「価格」XML 要素が一意に決まる必要があります。

```
CREATE INDEX INDX2 ON 書籍管理表(書籍情報)
  KEY USING UNIQUE TAG FROM ‘/書籍情報/価格’ AS INTEGER
```

3. 書籍管理表の書籍情報列に対して，「書籍情報」XML 要素の「書籍 ID」XML 属性の値を INTEGER 型に変換した値をキー値とするインデクス（INDX3）を定義します。ただし，「書籍情報」XML 要素は，各行の値中でに一意に決まり，かつキー値はすべての行で異なっている必要があります。

```
CREATE UNIQUE INDEX INDX3 ON 書籍管理表(書籍情報)
  KEY FROM ‘/書籍情報/@書籍ID’ AS INTEGER
```

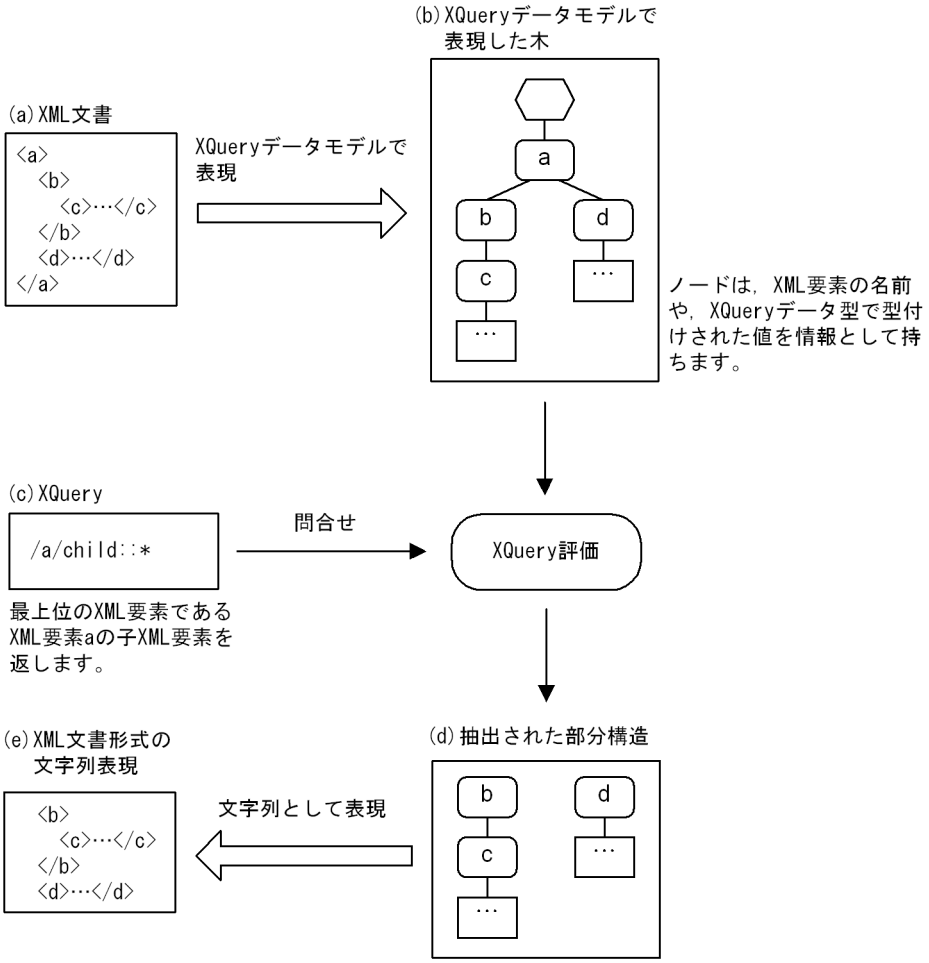
1.15 XQuery

XQuery は，XQuery データモデルの値として表現した XML 文書の内容に対して，問合せを実行する言語です。HiRDB では，XML 文書の内容を保持した XML 型の値に対して，XML 文書中の特定の部分構造を抽出するために，XMLQUERY 関数や XMLEXISTS 述語中で XQuery を指定します。

XQuery データモデルは，XML 文書の情報を木構造として表現するデータモデルです。XQuery データモデルは，6 種類のノードを構成要素として持ちます。これらのノードは，XML 要素や XML 属性を識別するための名前（修飾名）や，XQuery データ型によって型付けられた基本単位値を情報として保持します。このような情報を含んだノードや基本単位値で表現される XQuery データモデルの値に対して，XQuery による問合せを実行します。

XQuery の概念図を，次の図に示します。XML 文書（図中の(a)）は，XQuery データモデルの木（図中の(b)）として表現されます。XQuery データモデルの木に対して，入力された XQuery（図中の(c)）が評価され，木から特定の部分構造（図中の(d)）が抽出されます。XQuery によって抽出された部分構造も，XQuery データモデルで表現される値となります。抽出された部分構造は，XML 文書の形式で文字列表現することができます（図中の(e)）。

図 1-11 XQuery の概念図



1.15.1 XQuery データモデル

XQuery データモデルは，XQuery が処理対象とする XML 文書を表現するためのモデルです。XQuery データモデルは，木構造であり，ノードを持ちます。

ノードには，次に示すものがあります。また，各ノードが持つ情報をプロパティといいます。

- 文書ノード
- 要素ノード
- 属性ノード
- 処理命令ノード
- コメントノード
- テキストノード

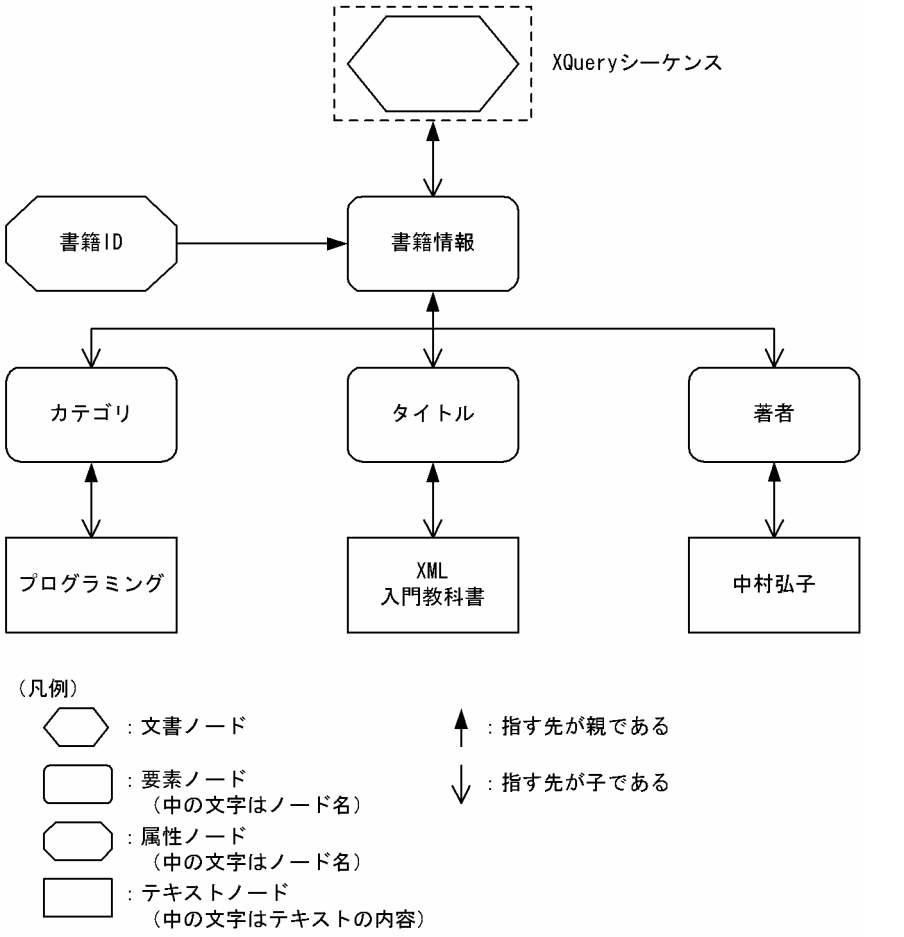
ノード又は基本単位値を XQuery 項目といいます。0 個以上の XQuery 項目の順序付けられた集まりを，XQuery シーケンスといいます。XQuery データモデルで表現される値は，XQuery シーケンスとなります。

例として，XML 文書を XQuery データモデルで表現した木を次の図に示します。例となる XML 文書に対応する XQuery シーケンスには，図中の点線内に示す文書ノードだけが含まれます。文書ノードは XML 文書の内容を保持する子孫ノードを指すため，文書ノードだけで XML 文書全体を扱うことができます。

＜ XML 文書＞

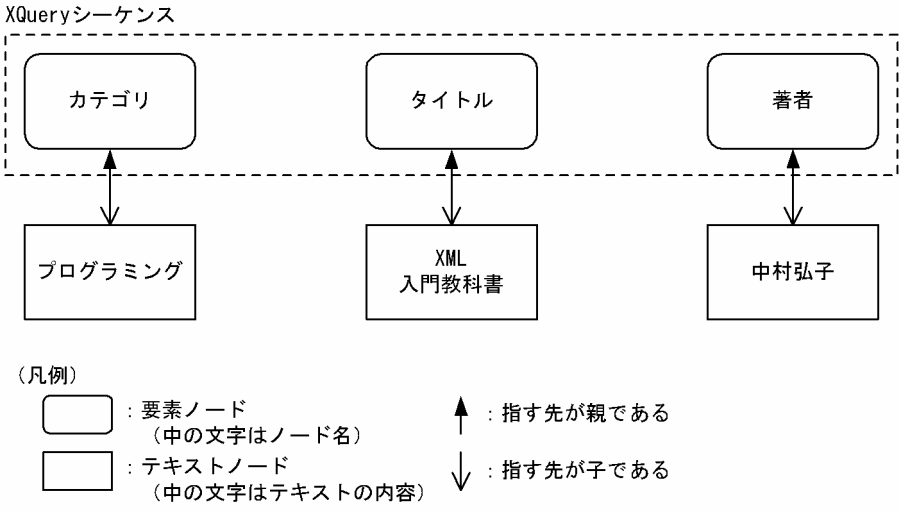
```
<書籍情報 書籍ID=”310494321”>
  <カテゴリ> プログラミング</カテゴリ>
  <タイトル> XML入門教科書</タイトル>
  <著者>中村弘子</著者>
</書籍情報>
```

図 1-12　XQuery データモデルの例



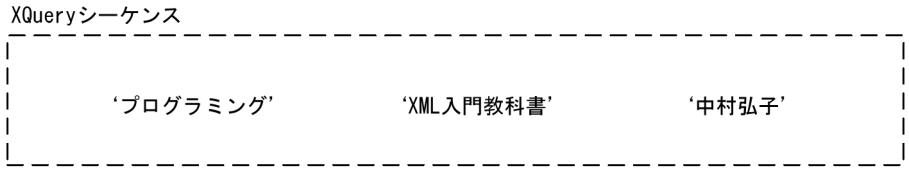
図「XQuery データモデルの例」で示した XML 文書に対する，’/書籍情報/child::element()’（「書籍情報」要素ノードの子である要素ノードを抽出します）という XQuery の問い合わせ結果は，次の図の点線内に示す，三つのノードから構成される XQuery シーケンスとなります。

図 1-13　ノードから構成される XQuery シーケンスの例



また，図「XQuery データモデルの例」で示した XML 文書に対する，’fn:data(/書籍情報/child::element()/text())’（「書籍情報」要素ノードの子である要素ノードが持つテキストノードの基本単位値を抽出します）という XQuery の問い合わせ結果は，次の図の点線内に示す，’プログラミング’，’XML 入門教科書’，’中村弘子’という三つの基本単位値から構成される XQuery シーケンスになります。

図 1-14　基本単位値から構成される XQuery シーケンスの例



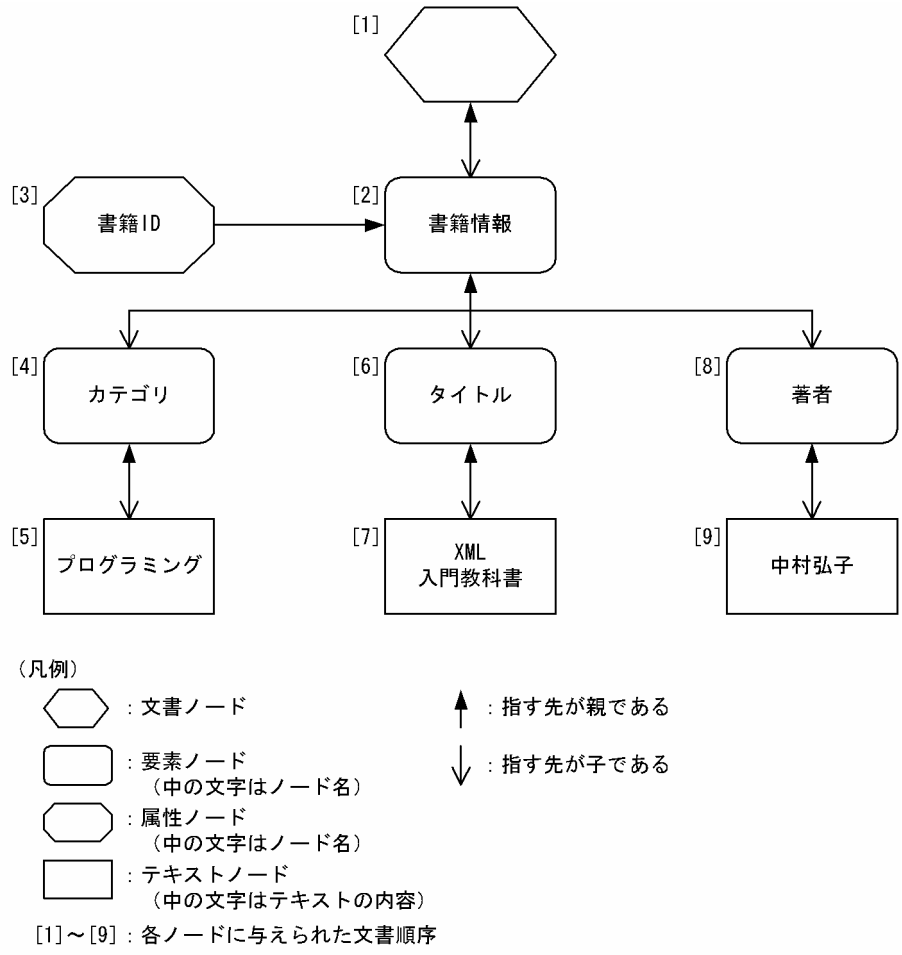
(1)　ノード間の順序

ノードには順番が与えられています。この順序を文書順序といいます。文書順序は，各ノードに対応する XML 要素や XML 属性が XML 文書で出現した順序と考えられます。文書順序の規則を次に示します。

1. すべてのノードは，自身の子孫ノードよりも前に出現します。
2. 属性ノードは，関連づけられた要素ノードの直後に出現します。
3. 兄弟の順序は，親ノードの子プロパティで出現する順序になります。
4. 子孫ノードは，兄弟ノードよりも前に出現します。

図「XQuery データモデルの例」の木のノードに与えられた文書順序を，次の図に示します。

図 1-15　文書順序の例



(2)　プロパティの種類

ノードが持つプロパティには，次の表に示すものがあります。

表 1-44　プロパティの種類と内容

項番	プロパティ名	説明
1	ノード名	XML 要素，XML 属性の修飾名です。
2	名前空間	ノードが属する XML 名前空間です。
3	親	親ノードです。
4	子	子ノードの XQuery シーケンスです。
5	属性	属性ノードの XQuery シーケンスです。
6	型名	XML 要素や XML 属性が持つ XQuery データ型の修飾名です。
7	文字列値	ノードの内容として指定されたテキストが示す文字列の xs:string 型の値です。fn:string 関数で参照できます。
8	型付き値	文字列値プロパティから得られる XQuery データ型の値です。fn:data 関数で参照できます。

項番	プロパティ名	説明
9	nilled	XML 要素の内容が空であるかどうかを示すプロパティです。内容が空であれば，子プロパティは，要素ノードもテキストノードも含みません。
10	処理命令ターゲット	XML 処理命令の対象となるアプリケーションです。
11	内容	XML コメントの内容，XML 要素の内容，XML 処理命令の内容です。

ノードが持つプロパティの種類は，ノードの種類によって異なります。各ノードが持つプロパティを次の表に示します。各ノードが持つプロパティの内容については，「ノードの詳細」を参照してください。

表 1-45　各ノードが持つプロパティ

項番	プロパティ名	文書ノード	要素ノード	属性ノード	処理命令ノード	コメントノード	テキストノード
1	ノード名		○	○			
2	名前空間		○				
3	親		○	○	○	○	○
4	子	○	○				
5	属性		○				
6	型名		○	○			
7	文字列値	○	○	○			
8	型付き値	○	○	○			
9	nilled		○				
10	処理命令ターゲット				○		
11	内容				○	○	○

（凡例）

○：プロパティを持ちます。

空白：プロパティを持ちません。

(3)　ノードの詳細

XQuery データモデルが持つ 6 種類のノードについて説明します。

(a)　文書ノード

文書ノードは，XML 文書の情報を保持します。

- **プロパティ**

  文書ノードは，次のプロパティを持ちます。

プロパティ名	説明
子	文書ノードの子ノードの XQuery シーケンスです。
文字列値	子孫となるテキストノードの内容プロパティの値を文書順序順に連結した結果の値です。子孫となるテキストノードが存在しない場合は，長さ 0 の文字列となります。
型付き値	xs:untypedAtomic 型で，値は文字列値プロパティの値となります。

- 制約

  文書ノードは，次の制約に従います。

  1. 子ノードとなり得るノードは，要素ノード，処理命令ノード，コメントノード，テキストノードのどれかです。子ノードは，空となることがあります。属性ノード，文書ノードは，子ノードとなることができません。
  2. ノード N が文書ノード D の子プロパティに含まれる場合，ノード N の親プロパティはノード D です。
  3. ノード N が文書ノード D を親プロパティとして持つ場合，ノード N はノード D の子プロパティです。

(b) 要素ノード

要素ノードは，XML 要素の情報を保持します。

- プロパティ

  要素ノードは，次のプロパティを持ちます。

プロパティ名	説明
ノード名	XML 要素の修飾名を示します。
親	要素ノードの親ノードを示します。親ノードは，文書ノードか要素ノードのどちらかです。
型名	要素ノードの XQuery データ型を示します。XML スキーマによる検査が行われていない場合，又は XML スキーマによる検査の結果，要素ノードの型名が HiRDB で提供している XQuery データ型に該当しない場合，型名プロパティの値は不定になります。
子	要素ノードの子ノードの XQuery シーケンスを示します。
属性	要素ノードが持つ属性ノードの XQuery シーケンスを示します。
名前空間	要素ノードが属する XML 名前空間を示します。
nilled	要素ノードの内容が空であるかどうかを示します。内容が空であれば，子プロパティは，要素ノードもテキストノードも含みません。
文字列値	子孫であるテキストノードの内容プロパティの値を，文書順序順に連結した結果の値を示します。要素の内容が空の場合は，そのプロパティ値は長さ 0 の文字列となります。
型付き値	型名プロパティの値が不定の場合，xs:untypedAtomic 型の実現値としての文字列値プロパティの値となります。XML 要素が空の場合は，プロパティの値は空の XQuery シーケンスとなります。それ以外の場合は，型名プロパティの値の XQuery データ型に従って，文字列値プロパティの値から得られる基本単位値の XQuery シーケンスとなります。

- 制約

  要素ノードは，次の制約に従います。

  1. 子ノードとなり得るノードは，要素ノード，処理命令ノード，コメントノード，テキストノードのどれかです。子ノードは，空となることがあります。属性ノード，文書ノードは子ノードとなることができません。
  2. 同じ要素ノードを親プロパティに持つ属性ノードは，それぞれが異なる XML 属性の修飾名を持たなければなりません。
  3. ノード N が要素ノード E の子プロパティに含まれれば，ノード N の親プロパティはノード E です。
  4. 属性ノードでないノード N が要素ノード E を親プロパティとして持てば，ノード N はノード E の子プロパティに含まれます。
  5. 属性ノード A が要素ノード E の属性プロパティに含まれれば，ノード A の親プロパティはノード E です。
  6. 属性ノード A が要素ノード E を親プロパティとして持てば，ノード A はノード E の属性プロパティに含まれます。
  7. 要素ノードの型名プロパティが不定であれば，子孫である要素ノードの型名プロパティはすべて不定となり，属性ノードの型名プロパティは xs:untypedAtomic 型となります。
  8. 要素ノードの型名プロパティが不定であれば，nilled プロパティは偽となります。
  9. nilled プロパティが真であれば，子プロパティに，要素ノードとテキストノードを含みません。
  10. 要素ノード E のノード名プロパティが持つ修飾名に対して，ノード E が持つ属性ノード A のノード名プロパティは，ノード E の XML 名前空間 URI に関連づけられた XML 名前空間接頭辞を持たなければなりません。

(c) 属性ノード

属性ノードは，XML 属性の情報を保持します。

- プロパティ

  属性ノードは，次のプロパティを持ちます。

プロパティ名	説明
ノード名	XML 属性の修飾名を示します。
親	属性ノードの親ノードを示します。親ノードは，XML 属性が属する要素のノードです。
型名	属性ノードの XQuery データ型名を示します。
文字列値	属性ノードが持つ値を示します。このプロパティが空となることはありません。
型付き値	型名が xs:untypedAtomic の場合，xs:untypedAtomic 型の実現値としての文字列値プロパティの値となります。 それ以外の場合は，XML スキーマで検査された結果として得られる XQuery データ型に従って，文字列値プロパティから得られる基本単位値の XQuery シーケンスとなります。

- 制約

属性ノードは，次の制約に従います。

1. 属性ノードＡが要素ノードＥの属性プロパティに含まれる場合，ノードＡの親プロパティはノードＥです。
2. 属性ノードＡが要素ノードＥを親プロパティとして持つ場合，ノードＡはノードＥの属性プロパティに含まれます。

(d) 処理命令ノード

処理命令ノードは，XML 処理命令の情報を保持します。

- プロパティ

  処理命令ノードは，次のプロパティを持ちます。

プロパティ名	説明
処理命令ターゲット	処理命令の対象となるアプリケーションを示します。
内容	処理命令の内容を示します。
親	処理命令ノードの親ノードを示します。

- 制約

  処理命令ノードは，次の制約を満たします。

  1. 処理命令ターゲットは局所名です。

(e) コメントノード

コメントノードは，XML コメントの情報を保持します。

- プロパティ

  コメントノードは，次のプロパティを持ちます。

プロパティ名	説明
内容	XML コメントの内容を示します。
親	コメントノードの親ノードを示します。

(f) テキストノード

テキストノードは，XML 要素の内容の情報を保持します。

- プロパティ

  テキストノードは，次のプロパティを持ちます。

プロパティ名	説明
内容	XML 要素の内容の文字列を示します。
親	テキストノードの親ノードを示します。

- 制約

  テキストノードは，次の制約に従います。

  1. テキストノードが親ノードを持てば，内容プロパティの値が長さ 0 の文字列になることはありません。テキストノードが親ノードを持たない場合だけ，長さ 0 の文字列となることがあります。

(4) ノード間の親子関係

各ノード種別に対して，親ノード，子ノードとなり得るノード種別を，次の表に示します。

表 1-46 各ノード種別に対して親ノードになり得るノード種別

自ノードの種別	親ノードの種別					
	文書ノード	要素ノード	属性ノード	処理命令ノード	コメントノード	テキストノード
文書ノード	×	×	×	×	×	×
要素ノード	○	○	×	×	×	×
属性ノード	×	○	×	×	×	×
処理命令ノード	○	○	×	×	×	×
コメントノード	○	○	×	×	×	×
テキストノード	○	○	×	×	×	×

（凡例）

○：親ノードとなることができます。

×：親ノードとなることができません。

表 1-47 各ノード種別に対して子ノードになりうるノード種別

自ノードの種別	子ノードの種別					
	文書ノード	要素ノード	属性ノード	処理命令ノード	コメントノード	テキストノード
文書ノード	×	○	×	○	○	○
要素ノード	×	○	×	○	○	○
属性ノード	×	×	×	×	×	×
処理命令ノード	×	×	×	×	×	×
コメントノード	×	×	×	×	×	×
テキストノード	×	×	×	×	×	×

（凡例）

○：子ノードとなることができます。

×：子ノードとなることができません。

1.15.2　基本項目

この節では，次の表に示す XQuery の基本項目について説明します。

表 1-48　XQuery の基本項目

項番	基本項目	概要
1	焦点	XQuery 式を評価している，ある時点での評価対象である XQuery 項目の情報です。
2	修飾名	XML 要素や XML 属性，XQuery 関数，XQuery 変数を識別するために，XQuery で使用する名前です。
3	XQuery データ型	XQuery データモデルで，ノードや基本単位値を型付ける型です。
4	基本単位化	XQuery シーケンスを，基本単位値から構成される XQuery シーケンスに変換する処理です。
5	ブーリアン値への暗黙的な型変換	ブーリアン値以外の値が XQuery 式の評価対象である場合，暗黙的にブーリアン値に変換する処理です。

(1)　焦点

焦点は，XQuery 式を評価しているある時点での評価対象である XQuery 項目の情報です。

焦点には，次の表に示す要素があります。

表 1-49　焦点

項番	情報	概要
1	文脈項目	評価対象である XQuery 項目
2	文脈位置	文脈項目の，文脈項目から構成される XQuery シーケンス中での位置
3	文脈サイズ	文脈項目の数

次に，それぞれの要素を詳細に説明します。

(a)　文脈項目

XQuery 式を評価しているある時点で，評価の対象となっている XQuery 項目を持ちます。

(b)　文脈位置

XQuery 式を評価しているある時点での，各文脈項目の文脈項目から構成される XQuery シーケンス中での位置を持ちます。文脈位置は，1 から始まる整数値をとります。

(c)　文脈サイズ

XQuery 式を評価しているある時点での文脈項目の数を持ちます。

焦点は，XQuery 式を順次評価していくことで移動します。これを次の例を用いて説明します。

（例）

＜評価対象の XML 文書＞

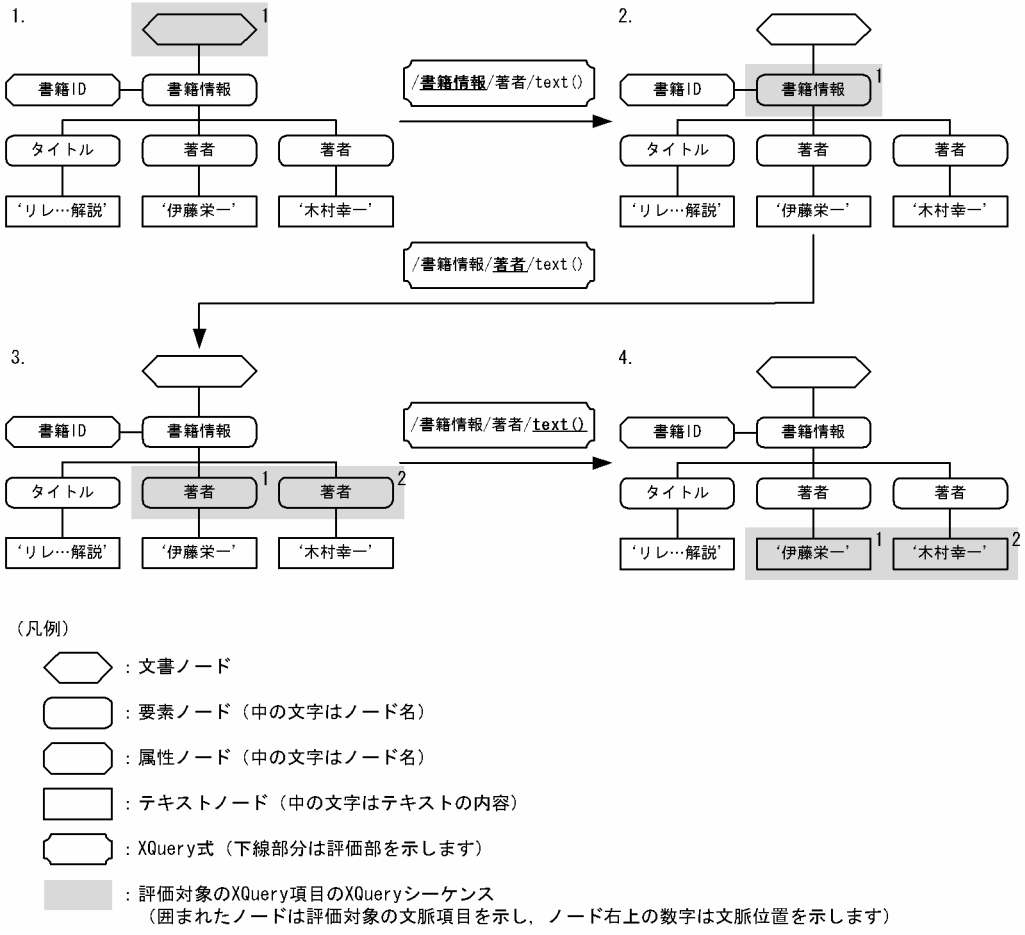
```
<書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
</書籍情報>
```

＜ XQuery 式＞

```
/書籍情報/著者/text()
```

この例の XQuery 式は，「書籍情報」「著者」「text()」という順に評価されます。このときの焦点の移動の流れを，次の図に示します。

図 1-16　焦点の移動の流れ



〔説明〕

1. XQuery 式の評価を開始する前に，文脈項目を文書ノードに初期化します。文書ノードの文脈位置は 1，文脈サイズは 1 になります。
2. XQuery 式の’書籍情報’部分を評価すると，文脈項目は，現在の文脈項目である文書ノードの子ノードの「書籍情報」要素ノードに移動します。文脈項目は一つであるため，「書籍情報」要素ノードの文脈位置は 1，文脈サイズは 1 になります。
3. XQuery 式の'著者'部分を評価すると，文脈項目は，現在の文脈項目である「書籍情報」要素ノードの子ノード中の「著者」要素ノードに移動します。「書籍情報」要素ノードの子ノードの「著者」要素ノードは二つあるため，文書順序に従って，一つ目の「著者」要素ノードの文脈位置は 1，二つ目の「著者」要素ノードの文脈位置は 2 になり，文脈サイズは 2 になります。
4. XQuery 式の’text()’部分を評価すると，文脈項目は，現在の文脈項目である「著者」要素ノードの子ノードのテキストノードに移動します。現在の文脈項目である「著者」要素ノードの子ノードのテキストノードは二つあるため，一つ目の「著者」要素ノードの子ノードのテキストノードの文脈位置は 1，二つ目の「著者」要素ノードの子ノードのテキストノードの文脈位置は 2 になり，文脈サイズは 2 になります。

(2) 修飾名

XML 要素や XML 属性，XQuery 関数，XQuery 変数を識別するために，XQuery で使用する名前を修飾名といいます。修飾名は，接頭辞とコロン文字と局所名から構成されます。ユーザは，XML 名前空間宣言（XML 名前空間宣言については，「XQuery 宣言部」を参照してください）を使って，接頭辞と XML 名前空間 URI を関連づけることができます。XML 名前空間 URI は，XML 名前空間を識別するための URI です。局所名は，個々の XML 名前空間内で使用する名前です。

HiRDB では，次の表に示す接頭辞があらかじめ定義されています。

表 1-50 HiRDB で定義されている接頭辞

項番	接頭辞	XML 名前空間 URI
1	xml	http://www.w3.org/XML/1998/namespace
2	xs	http://www.w3.org/2001/XMLSchema
3	xsi	http://www.w3.org/2001/XMLSchema-instance
4	fn	http://www.w3.org/2006/xpath-functions
5	err	http://www.w3.org/2005/xqt-errors
6	hi-fn	http://www.hitachi.co.jp/Prod/comp/soft1/hirdb/xquery-functions

接頭辞とコロン文字を省略し，局所名だけを指定して，修飾名として使用することもできます。接頭辞とコロン文字が省略された修飾名は，既定の XML 名前空間に属することになります。既定の XML 名前空間については，「XQuery 宣言部」を参照してください。修飾名の例を，次の表に示します。

表 1-51　修飾名の例

項番	修飾名	接頭辞	局所名
1	xs:string	xs	string
2	fn:data	fn	data
3	element_name	（省略）	element_name

接頭辞に対応する XML 名前空間 URI と，局所名から構成される名前を展開後修飾名といいます。二つの修飾名は，それらの展開後修飾名の XML 名前空間 URI と局所名が共に等しい場合に，等価になります。この場合，二つの修飾名の接頭辞が異なっていても，二つの修飾名は等価とみなされます。例えば，次の表に示す二つの修飾名は，接頭辞が異なります。しかし，展開後修飾名の XML 名前空間 URI と局所名が共に等しいため，二つの修飾名は等価とみなされます。

表 1-52　接頭辞は異なっていても，等価とみなされる二つの修飾名

項番	修飾名	接頭辞	局所名	関連づけされた XML 名前空間 URI
1	xxx:element_name	xxx	element_name	http://www.hitachi.com
2	yyy:element_name	yyy	element_name	http://www.hitachi.com

接頭辞及び局所名に指定できる文字を次の表に示します。

表 1-53　接頭辞及び局所名に使用できる文字

項番	分類	接頭辞及び局所名に使用できる文字	先頭指定可否
1	半角文字コード文字	英小文字（A～Z）	○
2		英小文字（a～z）	○
3		数字（0-9）	×
4		片仮名文字	○
5	全角文字コード	全角文字コードの文字すべて	○
6	特殊記号（半角文字コード）	ピリオド（ . ）	×
7		ハイフン又は負符号（ - ）	×
8		下線文字（ _ ）	○

（凡例）

○：接頭辞及び局所名の先頭に指定できます。

×：接頭辞及び局所名の先頭に指定できません。

XQuery の接頭辞及び局所名として使用できる文字は，pdsetup コマンドで指定した文字コード種別によって異なります。pdsetup コマンドについては，マニュアル「HiRDB コマンドリファレンス」を参照

してください。指定した文字コード種別と使用できる文字の関係は，「SQL で使用できる文字」を参照してください。

(3) XQuery データ型

(a) HiRDB が定義する XQuery データ型

XQuery で扱う XQuery 項目は，XQuery データ型で型付けられています。HiRDB では，次の表に示す XQuery データ型を提供します。文字列データ型，数値データ型，時間データ型，その他のデータ型，xs:untypedAtomic 型の単一の値を持つ XQuery データ型を基本単位型といいます。基本単位型の値を，基本単位値といいます。

表 1-54 HiRDB が定義する XQuery データ型

項番	分類	XQuery データ型	説明
1	文字列データ型	xs:string	文字列を表す型です。
2	数値データ型	xs:decimal	最大 38 けたの固定小数点数を表す型です。
3		xs:int	値の範囲が−2147483648～2147483647 の整数を表す型です。
4		xs:double	値の範囲が約±4.9×10^{-324}～±1.7×10^{308} の倍精度浮動小数点数を表す型です。※
5	時間データ型	xs:dateTime	日付と時刻の組み合わせを値とし，ある日付のある時刻を表す型です。「YYYY-MM-DDThh:mm:ss[.nn…n]」の形式で表現します。YYYY は年，MM は月，DD は日，hh は時，mm は分，ss は秒，nn…n は小数秒（1～6 けた）を表します。T は日付と時刻を分離するために表記します。形式の各要素が取り得る値の範囲は，YYYY：0001～9999（年），MM：01～12（月），DD：01～該当月最終日（日），hh：00～23（時），mm：00～59（分），ss：00～59（秒），nnnnnn：000000～999999（小数秒）です。
6		xs:date	日付を表す値をとる型です。「YYYY-MM-DD」の形式で表現します。YYYY は年，MM は月，DD は日を表します。形式の各要素が取り得る値の範囲は，YYYY：0001～9999（年），MM：01～12（月），DD：01～該当月最終日（日）です。
7		xs:time	時刻を表す値をとる型です。「hh:mm:ss」の形式で表現します。hh は時，mm は分，ss は秒を表します。形式の各要素が取り得る値の範囲は，hh：00～23（時），mm：00～59（分），ss：00～59（秒）です。
8	その他のデータ型	xs:hexBinary	バイナリデータを 16 進数の文字の並び（「0～9」「a～f」「A～F」の文字の並び）として表現する型です。
9		xs:boolean	論理値として真と偽をとる型です。
10	不定データ型	xs:untypedAtomic	型が不定である基本単位値の型を表します。

注※

浮動小数点数の値の範囲は，ハードウェア表現に従います。

(b)　コンストラクタ関数

(a)で示したそれぞれの基本単位型に対して，ある基本単位値から，基本単位値を生成するコンストラクタ関数が暗黙的に定義されます。コンストラクタ関数は，次の形式をとります。

- 形式

```
XQueryデータ型名（ 引数 ）
```

- 規則

1. XQuery データ型名には，生成する基本単位値の XQuery データ型の修飾名を指定します。
2. 引数には，基本単位値又は空の XQuery シーケンスを指定します。空の XQuery シーケンスを指定した場合，空の XQuery シーケンスを返却します。
3. 引数に指定された基本単位値が，生成する基本単位値の XQuery データ型に変換できない場合エラーとなります。変換可能な XQuery データ型の組み合わせは「(c)　変換可能な XQuery データ型」を参照してください。

(c)　変換可能な XQuery データ型

変換可能な XQuery データ型を次の表に示します。

表 1-55　変換可能な XQuery データ型

変換前の XQuery データ型	変換後の XQuery データ型									
	数値データ型			文字列データ型	時間データ型			その他のデータ型		
	xs:double	xs:decimal	xs:int	xs:string	xs:dateTime	xs:date	xs:time	xs:hexBinary	xs:boolean	xs:untypedAtomic
xs:double	○	△	△	○	×	×	×	×	○	○
xs:decimal	○	○	△	○	×	×	×	×	○	○
xs:int	○	○	○	○	×	×	×	×	○	○
xs:string	△	△	△	○	△	△	△	△	△	○
xs:dateTime	×	×	×	○	○	○	○	×	×	○
xs:date	×	×	×	○	○	○	×	×	×	○
xs:time	×	×	×	○	×	×	○	×	×	○
xs:hexBinary	×	×	×	○	×	×	×	○	×	○

変換前の XQuery データ型	変換後の XQuery データ型									
	数値データ型			文字列データ型	時間データ型			その他のデータ型		
	xs:double	xs:decimal	xs:int	xs:string	xs:dateTime	xs:date	xs:time	xs:hexBinary	xs:boolean	xs:untypedAtomic
xs:boolean	○	○	○	○	×	×	×	×	○	○
xs:untypedAtomic	△	△	△	○	△	△	△	△	△	○

（凡例）

○：変換できます。

△：値によっては変換できます。

×：変換できません。

- xs:string 型及び xs:untypedAtomic 型への変換

  次の表に示す規則に従って変換します。

表 1-56　xs:string 型及び xs:untypedAtomic 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:string xs:untypedAtomic	すべて	変換しません。
2	xs:int	すべて	XQuery 整数定数の形式に変換します。 ただし，先頭の 0 は取り除きます。 また，値が 0 未満の場合は負符号（-）を付加します。
3	xs:decimal	小数部分が 0	
4		小数部分が 0 でない	XQuery10 進数定数の形式に変換します。ただし，10 の位以上の先頭の 0 及び末尾の 0 は取り除きます。また，値が 0 未満の場合は負符号（-）を付加します。
5	xs:double	絶対値が 0.000001 以上 1000000 未満	xs:decimal 型の変換規則（項番 3 及び 4）に従います。
6		正の 0	“0”に変換します。
7		負の 0	“-0”に変換します。
8		正の最大値	“INF”に変換します。
9		負の最大値	“-INF”に変換します。
10		NaN（非数）	“NaN”に変換します。

項番	変換前		変換結果
	XQuery データ型	値の条件	
11		上記以外	仮数部を XQuery10 進数定数の形式に変換した文字列，その後に”E”，その後に指数部を XQuery 整数定数の形式に変換した文字列の形式に変換します。 ただし，先頭 0 は取り除きます。仮数部の整数部分は 1 けたであり，0 にはなりません。 仮数部の小数第 1 位より下位の末尾の 0 は取り除きます。 また，値が 0 未満の場合は負符号（-）を付加します。
12	xs:boolean	真	“true”に変換します。
13		偽	“false”に変換します。
14	xs:dateTime	すべて	“YYYY-MM-DDThh:mm:ss.s…s”の形式に変換します。YYYY は年，MM は月，DD は日，hh は時，mm は分，ss は秒，s…s は小数秒を表します。 ただし，小数秒が 0 でない場合も小数秒の末尾の 0 は取り除きます。 さらに，小数秒が 0 の場合は，小数秒部分の直前の小数点は取り除きます。
15	xs:date	すべて	“YYYY-MM-DD”の形式に変換します。YYYY は年，MM は月，DD は日を表します。
16	xs:time	すべて	“hh:mm:ss”の形式に変換します。hh は時，mm は分，ss は秒を表します。
17	xs:hexBinary	すべて	16 進数の文字の並びがそのまま結果になります。 ただし，A～F は大文字になります。

- xs:double 型への変換

  次の表に示す規則に従って変換します。

表 1-57　xs:double 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:double	すべて	変換しません。
2	xs:int xs:decimal	すべて	対応する xs:double 型の値に変換します。
3	xs:boolean	真	1.0E0 に変換します。
4		偽	0.0E0 に変換します。
5	xs:string	“INF”	正の無限大に変換します。
6		“-INF”	負の無限大に変換します。
7		“NaN”	NaN（非数）に変換します。

項番	変換前		変換結果
	XQuery データ型	値の条件	
8		XQuery 数値定数の形式の文字列※	対応する数値データ型に変換した後，xs:double 型に変換します。
9		上記以外	変換できません。
10	xs:untypedAtomic	すべて	xs:string 型に変換した後，xs:double 型に変換します。

注※

先頭及び末尾のホワイトスペース（半角空白(X’20’)，タブ(X’09’)，NL(X’0A’)，又は CR(X’0D’)）を取り除いた結果が XQuery 数値定数の形式であれば，該当します。

- xs:decimal 型への変換

次の表に示す規則に従って変換します。

表 1-58　xs:decimal 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:decimal	すべて	変換しません。
2	xs:double	正の 0	0.に変換します。
3		負の 0	0.に変換します。
4		正の無限大	変換できません。
		負の無限大	
		NaN（非数）	
5		整数部分が 38 けた以下である値	変換前の値に数値上最も近い，38 けた以下の xs:decimal 型の値に変換します。該当する値が二つ存在する場合は，0 に近い方の値に変換します。
6		上記以外	変換できません。
7	xs:int	すべて	対応する xs: decimal 型の値に変換します。
8	xs:boolean	真	1.に変換します。
9		偽	0.に変換します。
10	xs:string	XQuery 数値定数の形式の文字列※	対応する数値データ型に変換した後，xs:decimal 型に変換します。
11		上記以外	変換できません。
12	xs:untypedAtomic	すべて	xs:string 型に変換した後，xs:decimal 型に変換します。

注※

先頭及び末尾のホワイトスペース（半角空白(X’20’)，タブ(X’09’)，NL(X’0A’)，又は CR(X’0D’)）を取り除いた結果が XQuery 数値定数の形式であれば，該当します。

- xs:int 型への変換

次の表に示す規則に従って変換します。

表 1-59　xs:int 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:int	すべて	変換しません。
2	xs:decimal	小数点以下を切り捨てた値が xs:int 型の最小値以上かつ最大値以下	小数点以下のけたを切り捨てた値に変換します。
3		上記以外	変換できません。
4	xs:double	正の 0	0 に変換します。
5		負の 0	0 に変換します。
6		正の無限大	変換できません。
		負の無限大	
		NaN（非数）	
7		小数点以下のけたを切り捨てた値が xs:int 型の最小値以上かつ最大値以下	小数点以下のけたを切り捨てた値に変換します。
8		上記以外	変換できません。
9	xs:boolean	真	1 に変換します。
10		偽	0 に変換します。
11	xs:string	XQuery 数値定数の形式の文字列※	対応する数値データ型に変換した後，xs:int 型に変換します。
12		上記以外	変換できません。
13	xs:untypedAtomic	すべて	xs:string 型に変換した後，xs:int 型に変換します。

注※

先頭及び末尾のホワイトスペース（半角空白(X’20’)，タブ(X’09’)，NL(X’0A’)，又は CR(X’0D’)）を取り除いた結果が XQuery 数値定数の形式であれば，該当します。

- xs:dateTime 型への変換

次の表に示す規則に従って変換します。

表 1-60　xs:dateTime 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:dateTime	すべて	変換しません。
2	xs:date	すべて	日付部分が変換前の値と等しく，時刻部分が 00:00:00 である値に変換します。
3	xs:string	xs:dateTime 型の文字列表現の形式の値	文字列表現に対応した値に変換します。
4		上記以外	変換できません。
5	xs:untypedAtomic	すべて	xs:string 型に変換した後，xs:dateTime 型に変換します。

- xs:date 型への変換

  次の表に示す規則に従って変換します。

表 1-61　xs:date 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:date	すべて	変換しません。
2	xs:dateTime	すべて	変換前の値の日付部分と等しい値に変換します。
3	xs:string	xs:date 型の文字列表現の形式の値	文字列表現に対応した値に変換します。
4		上記以外	変換できません。
5	xs:untypedAtomic	すべて	xs:string 型に変換した後，xs:date 型に変換します。

- xs:time 型への変換

  次の表に示す規則に従って変換します。

表 1-62　xs:time 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:time	すべて	変換しません。
2	xs:dateTime	すべて	変換前の値の時刻部分（小数秒部分を除く）と等しい値に変換します。
3	xs:string	xs:time 型の文字列表現の形式の値	文字列表現に対応した値に変換します。
4		上記以外	変換できません。
5	xs:untypedAtomic	すべて	xs:string 型に変換した後，xs:time 型に変換します。

- xs:boolean 型への変換

  次の表に示す規則に従って変換します。

表 1-63　xs:boolean 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:boolean	すべて	変換しません。
2	数値データ型	0	偽に変換します。
		正の 0	
		負の 0	
		0.0	
		0.0E0	
		NaN（非数）	
3		上記以外	真に変換します。
4	xs:string	“true”※	真に変換します。
5		“false”※	偽に変換します。
6		上記以外	変換できません。
7	xs:untypedAtomic	すべて	xs:string 型に変換した後，xs:boolean 型に変換します。

注※

先頭及び末尾のホワイトスペース（半角空白(X’20’)，タブ(X’09’)，NL(X’0A’)，又は CR(X’0D’)）を取り除いた結果が等しければ，該当します。また，大文字小文字を区別しません。

- xs:hexBinary 型への変換

  次の表に示す規則に従って変換します。

表 1-64　xs:hexBinary 型への変換規則

項番	変換前		変換結果
	XQuery データ型	値の条件	
1	xs:hexBinary	すべて	変換しません。
2	xs:string	16 進数の文字の並び	対応した xs:hexBinary 型の値に変換します。
3		上記以外	変換できません。
4	xs:untypedAtomic	すべて	xs:string 型に変換した後，xs:hexBinary 型に変換します。

(4) 基本単位化

基本単位化とは，XQuery シーケンスを，基本単位値だけから構成される XQuery シーケンスに変換する処理です。XQuery 中に基本単位値だけから構成される XQuery シーケンスが必要となる，次に示す式の評価時に基本単位化を行います。

- XQuery 算術式
- XQuery 比較式
- XQuery 関数呼出しの引数と戻り値
- XQuery 挿入式
- XQuery 名前変更式
- XQuery 置換式

基本単位化の結果は，XQuery シーケンスに対して，XQuery 関数の fn:data 関数を呼び出した結果になります。fn:data 関数の詳細については「XQuery 関数」を参照してください。

(5) ブーリアン値への暗黙的な型変換

XQuery 中のブーリアン値（xs:boolean 型の値）として評価する必要がある箇所で，ブーリアン値以外の値が評価対象である場合，暗黙的にブーリアン値に変換されます。ブーリアン値への変換結果は XQuery シーケンスに対して，XQuery 関数の fn:boolean 関数を呼び出した結果になります。fn:boolean 関数の詳細については「XQuery 関数」を参照してください。

1.15.3 XQuery の指定

(1) 機能

XML 型の値への問合せを指定します。

次に示す箇所に文字列定数として指定できます。

- XMLQUERY 関数
- XMLEXISTS 述語

(2) 形式

```
XQuery : := XQuery宣言部　XQuery問合せ本体
```

(3) 使用例

XQuery 宣言部で既定の XML 名前空間を宣言します。XQuery 問合せ本体中で，XQuery 問合せの結果を定義します。

```
declare default element namespace ‘http://www.aaa.com’ ;      … XQuery宣言部
/書籍情報/書籍/価格/child::text()                              … XQuery問い合わせ本体
```

1.15.4 XQuery の記述形式

(1) XQuery でのキーワードの指定

XQuery の機能を使用するために指定する語（for や if など）をキーワードといいます。XQuery でのキーワードは，すべて小文字で指定します。XQuery は，大文字と小文字を区別します。

(2) XQuery での区切り

XQuery の区切りには，次の文字又は構成要素を指定できます。

- 空白（X'20'）
- TAB（X'09'）
- NL（X'0a'）
- CR（X'0d'）
- XQuery コメント

(a) 区切りを挿入する位置

区切りを挿入する位置を次に示します。

- キーワードとキーワードの間
- キーワードと修飾名の間
- キーワードと数値定数の間
- 修飾名の後の減算演算子（ - ）の前

(b) 区切りを挿入してはいけない位置

区切りを挿入してはいけない位置を次に示します。

- キーワードの中
- 修飾名の中

- 数値定数の中
- 文字列定数の中
- 演算子の中
- XQuery パス式中で指定する’//’の中
- XQuery パス式中で指定する’..’の中
- XQuery パス式中で指定する’::’の中
- XQuery 変数参照中の’$’と XQuery 変数名の間
- XQuery コメントの’(‘と’:’の間
- XQuery コメントの’:‘と’)’の間
- FLWOR 式中で指定する’:=’の中
- XQuery 変換式中で指定する’:=’の中

(c) 区切りを挿入してもよい位置

区切りを挿入してもよい位置を次に示します。

- 次に示す特殊記号の前後で，かつ「区切りを挿入してはいけない位置」で挿入を禁止されていない位置
「,」「.」「-」「＋」「*」「 ‘」「 “」「(」「)」「＜」「＞」「＝」「!」「/」「:」「;」「|」「[」「]」「{」「}」「空白（X'20')」「TAB（X'09')」「NL（X'0a')」「CR（X'0d')」

1.15.5 XQuery 宣言部

(1) 機能

XQuery 問合せ中で使用する修飾名の XML 名前空間を宣言します。

(2) 形式

```
XQuery宣言部：：=〔 {XML名前空間宣言｜既定のXML名前空間宣言} ; 〕…

XML名前空間宣言：：= declare namespace 接頭辞 = XML名前空間URI
既定のXML名前空間宣言：：= declare default {element | function}
                                                namespace XML名前空間URI
```

(3) オペランド

- XML 名前空間宣言：：= declare namespace 接頭辞 = XML 名前空間 URI

XQuery 問合せ本体中に，接頭辞付きの修飾名を含む場合，その修飾名の接頭辞の XML 名前空間を宣言します。

同じ接頭辞の XML 名前空間は複数宣言できません。

接頭辞

XQuery 問合せ本体中の修飾名の接頭辞を指定します。

XML 名前空間 URI

接頭辞に対応する XML 名前空間の URI を XQuery 文字列定数の形式で指定します。

長さ 0 の文字列を指定した場合，「HiRDB で定義されている接頭辞」で定義されている，XML 名前空間の接頭辞と XML 名前空間の対応付けを削除できます。

- **既定の XML 名前空間宣言：：= declare default {element | function} namespace XML 名前空間 URI**

XQuery 問合せ本体中の既定の XML 名前空間を宣言します。XQuery 問合せ本体中の接頭辞のない修飾名の XML 名前空間は，ここで宣言した XML 名前空間となります。

XQuery 中に，’element’ を指定した既定の XML 名前空間宣言，及び’function’ を指定した既定の XML 名前空間宣言は，それぞれ一つしか指定できません。

この指定を省略した場合，XML 要素名及び XQuery データ型名に対する既定の XML 名前空間の XML 名前空間 URI は，’http://www.w3.org/XML/1998/namespaces’ を，XQuery 関数に対する既定の XML 名前空間の XML 名前空間 URI は，’http://www.w3.org/2006/xpath-functions’ を仮定します。

element

XML 要素名と XQuery データ型名に対する既定の XML 名前空間を宣言する場合に指定します。

function

XQuery 関数名に対する既定の XML 名前空間を宣言する場合に指定します。

XML 名前空間 URI

接頭辞を省略した修飾名に対する XML 名前空間 URI を XQuery 文字列定数の形式で指定します。

長さ 0 の文字列を指定した場合，接頭辞を省略した修飾名は，どの XML 名前空間にも属さない名前となります。

1.15.6 XQuery 問合せ本体

(1) 機能

XQuery 問合せ本体には，XQuery 問合せの結果を定義する XQuery 式を指定します。

(2) 形式

```
XQuery問合せ本体：：= XQuery式

XQuery式 ：：= XQueryシーケンス連結式
```

(a) XQuery 式を構成する式

XQuery 式は，次の表に示す式を組み合わせて指定します。

表 1-65 XQuery 式を構成する式

項番	式	説明
1	XQuery シーケンス連結式	XQuery シーケンスを連結する式です。
2	FLWOR 式	• 指定した XQuery シーケンスを構成するすべての XQuery 項目ごとに，XQuery 式を評価し，その結果の XQuery シーケンスを返却します。 • 指定した XQuery シーケンスを XQuery 変数に割り当て，その XQuery 変数を指定した XQuery 式を評価し，その結果の XQuery シーケンスを返却します。
3	XQuery 限定式	指定した XQuery シーケンスを構成する XQuery 項目のどれか（some），又はすべて（every）が指定した条件を満たすかどうかを判定する式です。
4	XQuery 条件式	指定した条件の判定結果に応じて XQuery 式を選択し，選択した XQuery 式を評価する式です。
5	XQuery 論理式	論理演算（OR，AND）を行う式です。
6	XQuery 比較式	比較を行う式です。比較には値比較，汎用比較，ノード比較があります。
7	XQuery 範囲式	連続する整数（xs:int 型）の基本単位値の XQuery シーケンスを生成する式です。
8	XQuery 算術式	加算，減算，乗算，除算，剰余算を行う式です。
9	XQuery シーケンス集合演算式	ノードから構成される XQuery シーケンスに対して，和，共通部分，差を求める式です。
10	XQuery 単項式	単項演算を行う式です。
11	XQuery 値式	値を指定する式です。
12	XQuery パス式	XQuery データモデルの木をたどって，特定のノードを選択する式です。
13	XQuery 基本式	XQuery パス式を構成する基本的な構成要素です。 次に示すものがあります。 • XQuery 定数 • XQuery 変数参照 • 括弧付き XQuery 式 • 文脈項目式 • XQuery 関数呼出し
14	XQuery 変換式	XQuery データモデルの木に，ノードの挿入，削除，名前変更，又は置換をして，その結果の XQuery データモデルの木を返却する式です。

項番	式	説明
15	XQuery 挿入式	XQuery データモデルの木の，指定した位置にノードを挿入する式です。
16	XQuery 削除式	XQuery データモデルの木の，指定した位置のノードを削除する式です。
17	XQuery 名前変更式	XQuery データモデルの木の，指定した位置のノードの修飾名又は処理命令ターゲットを変更する式です。
18	XQuery 置換式	XQuery データモデルの木の，指定した位置のノード又はノードの値を置換する式です。

(3) XQuery シーケンス連結式

(a) 機能

XQuery シーケンスを連結します。

複数の XQuery シーケンスを連結すると，連結対象の XQuery シーケンスが左から順に評価され，その XQuery シーケンスに含まれる XQuery 項目が結果の XQuery シーケンスの末尾に追加されていきます。連結結果の XQuery シーケンスは，連結対象の XQuery シーケンスのすべての XQuery 項目を含みます。また，連結結果の XQuery シーケンスに含まれる XQuery 項目の数は，連結対象の XQuery シーケンスに含まれる XQuery 項目の数の合計となります。

(b) 形式

```
XQueryシーケンス連結式 ： ：= XQuery式単独 〔, XQuery式単独 〕…

XQuery式単独 ： ：= { FLWOR式
                    |XQuery限定式
                    |XQuery条件式
                    |XQuery論理式
                    |XQuery変換式
                   }
```

(c) 使用例

XQuery シーケンス連結式の使用例を次に示します。

項番	例	結果	説明
1	1, 2, 3, 4, 5	(1, 2, 3, 4, 5)	1 から 5 までの XQuery 整数定数の評価結果である xs:int 型の基本単位値一つから構成される，XQuery シーケンス五つを連結した XQuery シーケンスを返します。
2	(1, 2, 3), (), (4, 5)	(1, 2, 3, 4, 5)	xs:int 型の基本単位値 1,2,3 から構成される XQuery シーケンス，空の XQuery シーケンス，及び xs:int 型の基本単位値 4,5 から構成される XQuery シーケンスを連結した XQuery シーケンスを返します。 「1, 2, 3, 4, 5」で表現される XQuery 式と同じ評価結果になります。

(4) FLWOR 式

(a) 機能

XQuery シーケンスを XQuery 変数に割り当て，XQuery 変数を含む XQuery 式を評価し，その結果の XQuery シーケンスを返却します。

XQuery 変数の割り当て方法及び XQuery 式の評価方法を次に示します。

- 指定した XQuery シーケンスを構成するすべての XQuery 項目ごとに，単一 XQuery シーケンスとして XQuery 変数に割り当て，それぞれの XQuery 項目に対して，XQuery 式を評価します（FOR 句）。
- 指定した XQuery シーケンスを XQuery 変数に割り当て，XQuery 式を評価します（LET 句）。

W3C の規格では FLWOR 式は，FOR 句，LET 句，WHERE 句，ORDER BY 句，及び RETURN 句から構成されますが，HiRDB では，FOR 句，LET 句及び RETURN 句だけサポートしています。

(b) 形式

```
FLWOR式 ： ：= ｛ FOR句 ｜ LET句 ｝ 〔 ｛ FOR句 ｜ LET句 ｝ 〕…
               RETURN句
FOR句 ： ：＝ for $ XQuery変数名 in  XQuery式単独
               〔, $ XQuery変数名 in XQuery式単独 〕…
LET句 ： ：＝ let $ XQuery変数名 ： 等号演算子 XQuery式単独
               〔, $ XQuery変数名 ： 等号演算子 XQuery式単独〕…
RETURN句 ： ：＝ return  XQuery式単独
```

(c) オペランド

- ｛FOR 句 ｜ LET 句｝〔 ｛FOR 句 ｜ LET 句｝〕…

FOR 句及び LET 句を一つ以上の任意の組み合わせで指定することができます。

複数の FOR 句や LET 句を指定した場合，各句がネストして指定されたものとして扱います。

（例）

次に示す XQuery 式 1 と XQuery 式 2 の内容は同じです。

＜ XQuery 式 1 ＞

```
for $XQuery変数名1 in XQuery式単独1
let $XQuery変数名2 := XQuery式単独2
for $XQuery変数名3  in XQuery式単独3
let $XQuery変数名4 := XQuery式単独4
return XQuery式単独5
```

＜ XQuery 式 2 ＞

```
for $XQuery変数名1 in XQuery式単独1
return
  let $XQuery変数名2 := XQuery式単独2
  return
```

```
for $XQuery変数名3 in XQuery式単独3
return
  let $XQuery変数名4 := XQuery式単独4
    return XQuery式単独5
```

- FOR 句 ：：＝ for $ XQuery 変数名 in XQuery 式単独
  〔, $ XQuery 変数名 in XQuery 式単独〕…

次に示す方法で RETURN 句の評価を繰り返します。

1. 単一の XQuery 変数を使用する場合は，XQuery 式単独の評価結果である XQuery シーケンスを構成するすべての XQuery 項目に対して，RETURN 句の評価を繰り返します。
2. 複数の XQuery 変数を使用する場合は，FOR 句がネストして指定されたものとして扱います。

   （例）

   次に示す XQuery 式 1 と XQuery 式 2 の内容は同じです。

   ＜ XQuery 式 1 ＞

```
for $XQuery変数名1 in XQuery式単独1 ,
    $XQuery変数名2 in XQuery式単独2
return XQuery式単独3
```

   ＜ XQuery 式 2 ＞

```
for $XQuery変数名1 in XQuery式単独1
return for $XQuery変数名2 in XQuery式単独2
        return XQuery式単独3
```

XQuery 変数名

XQuery 式単独を評価した XQuery シーケンス中の各 XQuery 項目に関連づけられる XQuery 変数の名前を指定します。

ここで指定した XQuery 変数は，該当する FLWOR 式の中で，該当する XQuery 変数名より後に指定した XQuery 変数名の入力として指定した XQuery 式単独内，この FOR 句より後に指定した FOR 句及び LET 句中の XQuery 式単独内，及び RETURN 句に指定した XQuery 式単独内で有効となります。

XQuery 式単独

入力とする XQuery シーケンスを返す XQuery 式単独を指定します。

- LET 句：：＝ let $ XQuery 変数名：等号演算子 XQuery 式単独
  〔, $ XQuery 変数名：等号演算子 XQuery 式単独〕…

XQuery 式単独を評価した結果を XQuery 変数に関連づけ，RETURN 句を評価します。

XQuery 変数名

XQuery 式単独を評価した XQuery シーケンスに関連づけられる XQuery 変数の名前を指定します。

ここで指定した XQuery 変数は，該当する FLWOR 式の中で，該当する XQuery 変数名より後に指定した XQuery 変数名の入力として指定した XQuery 式単独内，この LET 句より後に指定した FOR 句及び LET 句中の XQuery 式単独内，及び RETURN 句に指定した XQuery 式単独内で有効となります。

XQuery 式単独

入力とする XQuery シーケンスを返す XQuery 式単独を指定します。

- RETURN 句 ：：＝ return XQuery 式単独

XQuery 式単独

返却する XQuery シーケンスを表す XQuery 式単独を指定します。

FOR 句で得られた XQuery 項目の組み合わせ一つにつき，1 回評価します。

FOR 句を指定せずに，LET 句を指定した場合は，1 回だけ評価します。

FLWOR 式の評価結果は，ここで指定した XQuery 式単独を繰り返して評価した結果の XQuery シーケンスを連結した XQuery シーケンスとなります。

(d) 使用例

FLWOR 式の例を次に示します。

＜ XQuery 変数 book が表す要素＞

```
<書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <カテゴリ>データベース </カテゴリ>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
  <価格>3000</価格>
</書籍情報>
```

＜ FLWOR 式の例と結果＞

項番	例	結果	説明
1	for $a in $book/著者 return fn:string($a)	(“伊藤栄一”,”木村幸一”)	「書籍情報」要素ノードの子であるすべての「著者」要素ノードに対して，fn:string 関数を適用し，文字列として表現した結果を返します。
2	for $i in (10, 20), $j in (1,2) return ($i + $j)	(11, 12, 21, 22)	for $i in (10, 20) return for $j in (1, 2) return ($i + $j) と同じ結果を返します。
3	let $a := $book/著者 return $a/fn:string()	(“伊藤栄一”,”木村幸一”)	「書籍情報」要素ノードの子であるすべての「著者」要素ノードに対して，fn:string 関数を適用し，文字列として表現した結果を返します。
4	for $i in (1,2) let $a := $book/著者[$i] return fn:string($a)	(“伊藤栄一”,”木村幸一”)	「書籍情報」要素ノードの子である「著者」要素ノードのうち，1 番目の要素ノード及び 2 番目の要素ノードに対して，fn:string 関数を適用し，文字列として表現した結果を返します。
5	let $a := $book/著者 for $i in (1,2)	(“伊藤栄一”,”木村幸一”)	「書籍情報」要素ノードの子である「著者」要素ノードのうち，1 番目の要素ノード及び 2 番目

項番	例	結果	説明
	return fn:string($a[$i])		の要素ノードに対して，fn:string 関数を適用し，文字列として表現した結果を返します。
6	for $a in $book/著者 return fn:count($a)	(1,1)	「書籍情報」要素ノードの子であるそれぞれの「著者」要素ノードに対して，fn:count 関数を適用した結果 1 を連結した XQuery シーケンスを返します。
7	let $a := $book/著者 return fn:count($a)	(2)	「書籍情報」要素ノードの子である「著者」要素ノードから成る XQuery シーケンスに対して，fn:count 関数を適用した結果である 2 を返します。

(5) XQuery 限定式

(a) 機能

指定した XQuery シーケンスを構成する XQuery 項目のどれか（some），又はすべて（every）が指定した条件を満たすかどうかを判定し，その結果を返却します。

(b) 形式

```
XQuery限定式：：= { some | every }  $ XQuery変数名  in  XQuery式単独
                [ , $ 変数名 in  XQuery式単独 ]… satisfies  XQuery式単独
```

(c) オペランド

- { some | every } $ XQuery 変数名 in XQuery 式単独
  [, $ 変数名 in XQuery 式単独]… satisfies XQuery 式単独

some

‘some’を指定すると，in 句により生成された XQuery シーケンスの XQuery 項目に対して，satisfies 句に指定した条件が一つでも真であれば，結果は真となります。そうでなければ，結果は偽となります。in 句により生成された XQuery シーケンスが空の XQuery シーケンスである場合，XQuery 限定式の結果は偽となります。

every

‘every’を指定すると，in 句により生成された XQuery シーケンスの XQuery 項目に対して，satisfies 句に指定した条件がすべて真であれば，結果は真となります。そうでなければ，結果は偽となります。in 句で生成された XQuery シーケンスが空の XQuery シーケンスである場合，XQuery 限定式の結果は真となります。

XQuery 変数名

XQuery 式単独を評価した XQuery シーケンス中の XQuery 項目に関連づけられる XQuery 変数の名前を指定します。

ここで指定した XQuery 変数は，該当する XQuery 限定式の中で，該当する XQuery 変数名より後に指定した XQuery 変数名の入力として指定した XQuery 式単独内，及び satisfies 句に指定した XQuery 式単独内で有効となります。

in XQuery 式単独

入力となる XQuery シーケンスを意味する XQuery 式単独を指定します。

satisfies XQuery 式単独

XQuery 変数に対して，条件を評価する XQuery 式単独を指定します。複数の in 句を指定した場合，複数の XQuery 変数の組み合わせに対して，XQuery 式単独を評価します。

(d) 使用例

XQuery 限定式の例を次に示します。

＜ XQuery 変数 book が表す要素＞

```
<書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <カテゴリ>データベース </カテゴリ>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
  <価格>3000</価格>
</書籍情報>
```

＜ XQuery 限定式の例と結果＞

項番	例	結果	説明
1	some $text in $book /著者/text() satisfies ($text eq “木村幸一”)	真	「書籍情報」要素ノードの子である「著者」要素ノードが持つテキストノードの内容の一つが”木村幸一”に等しいため，真となります。
2	some $i in (1, 2, 3), $j in (4, 5, 6) satisfies $i + $j >= 6	真	‘some’ を指定した場合，$i+$j>=6 となる XQuery 項目の組み合わせ（例：$i=1，$j=5）を含むため，XQuery 限定式の結果は真となります。
3	every $i in (1, 2, 3), $j in (4, 5, 6) satisfies $i + $j >= 6	偽	‘every’ を指定した場合，すべての XQuery 項目の組み合わせが$i+$j>=6 となるわけではないため（例：$i=1，$j=4），XQuery 限定式の結果は偽となります。

(6) XQuery 条件式

(a) 機能

指定した条件の判定結果に応じて XQuery 式を選択し，選択した XQuery 式を評価した結果を返却します。

(b) 形式

```
XQuery条件式 ::= if （ XQuery式 ） then XQuery式単独 else XQuery式単独
```

(c) オペランド

- if (XQuery 式)

XQuery 式

条件として評価する XQuery 式を指定します。ここで指定した XQuery 式の評価結果が真ならば then 句の直後に指定した XQuery 式単独を評価し，偽ならば else 句の直後に指定した XQuery 式単独を評価します。ただし，ここで指定した XQuery 式の評価結果が xs:boolean 型の基本単位値の単一 XQuery シーケンスでない場合は，xs:boolean 型の基本単位値の単一 XQuery シーケンスに変換して判定します。

- then XQuery 式単独

XQuery 式単独

if 句に指定した条件が真となった場合に評価する XQuery 式単独を指定します。

- else XQuery 式単独

XQuery 式単独

if 句に指定した条件が偽となった場合に評価する XQuery 式単独を指定します。

(d) 使用例

XQuery 条件式の例を次に示します。

＜ XQuery 変数 book1 が表す XML 要素＞

```
<書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <カテゴリ>データベース </カテゴリ>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
  <価格>3000</価格>
</書籍情報>
```

＜ XQuery 変数 book2 が表す XML 要素＞

```
<書籍情報 書籍ID=”452469631”>
  <タイトル> XML入門教科書</タイトル>
  <著者>中村弘子</著者>
  <価格>2500</価格>
</書籍情報>
```

＜ XQuery 条件式の例と結果＞

項番	例	結果	説明
1	if ($book1/価格 > $book2/価格) then $book1 else $book2	then 句で指定された，XQuery 変数 book1 が示す要素ノード	$book1 が示す要素ノードの子である「価格」要素ノードの型付き

項番	例	結果	説明
			値(=3000)は，$book2 が示す要素ノードの子である「価格」要素ノードの型付き値(=2500)より大きいため，$book1 が示す要素ノードが結果となります。

(7) XQuery 論理式

(a) 機能

XQuery 比較式に対して論理演算（OR，AND）を行った結果を返却します。

又は，XQuery 比較式の評価結果をそのまま返却します。

(b) 形式

```
XQuery論理式 ：：= XQueryOR式
XQueryOR式 ：：= XQueryAND式 〔or XQueryAND式 〕…
XQueryAND式 ：：= XQuery比較式〔and XQuery比較式〕…
```

(c) オペランド

- XQueryOR 式 ：：= XQueryAND 式 〔or XQueryAND 式 〕…

or 演算子を指定した場合は，XQueryAND 式の評価結果に対して OR 論理演算をします。

XQueryAND 式を単独で指定した場合は，その XQueryAND 式の評価結果をそのまま返します。

OR 論理演算をする場合に，各 XQueryAND 式の評価結果がブーリアン値（xs:boolean 型の基本単位値）の単一 XQuery シーケンスでない場合は，暗黙的にブーリアン値（xs:boolean 型の基本単位値）の単一 XQuery シーケンスに変換してから演算します。

第 1 演算項及び第 2 演算項のブーリアン値による OR 論理演算の結果を，次の表に示します。

表 1-66 XQuery 論理式での OR 論理演算の結果

項番	第 1 演算項	第 2 演算項	
		真	偽
1	真	真	真
2	偽	真	偽

- XQueryAND 式 ：：= XQuery 比較式〔and XQuery 比較式〕…

and 演算子を指定した場合は，XQuery 比較式の評価結果に対して AND 論理演算をします。

XQuery 比較式を単独で指定した場合は，その XQuery 比較式の評価結果をそのまま返します。

AND 論理演算をする場合に，各 XQuery 比較式の評価結果がブーリアン値（xs:boolean 型の基本単位値）の単一 XQuery シーケンスでない場合は，ブーリアン値（xs:boolean 型の基本単位値）の単一 XQuery シーケンスに変換してから演算します。

第 1 演算項及び第 2 演算項のブーリアン値による AND 論理演算の結果を，次の表に示します。

表 1-67　XQuery 論理式での AND 論理演算の結果

項番	第 1 演算項	第 2 演算項	
		真	偽
1	真	真	偽
2	偽	偽	偽

(8)　XQuery 比較式

(a)　機能

XQuery 範囲式に対して比較を行った結果を返却します。

又は，XQuery 範囲式の評価結果をそのまま返却します。

(b)　形式

```
XQuery比較式 ::= XQuery範囲式 〔{値比較|汎用比較|ノード比較} XQuery範囲式〕

値比較 ::= {eq|ne|lt|le|gt|ge}
汎用比較 ::= {=|!=|<|<=|>|>=|<>}
ノード比較 ::= {is|<<|>>}
```

(c)　オペランド

- XQuery 範囲式 〔{値比較|汎用比較|ノード比較} XQuery 範囲式〕

値比較，汎用比較，又はノード比較を指定した場合は，XQuery 範囲式の評価結果を比較します。

XQuery 範囲式を単独で指定した場合は，その XQuery 範囲式の評価結果をそのまま返却します。

- 値比較 ::= {eq | ne | lt | le | gt | ge}

単一の基本単位値同士を比較します。

各指定での比較方法を次の表に示します。

表 1-68　値比較の比較方法

項番	値比較演算子	真となる条件
1	eq	第 1 比較項と第 2 比較項の値が等しい
2	ne	第 1 比較項と第 2 比較項の値が等しくない
3	lt	第 1 比較項の値が第 2 比較項の値より小さい
4	le	第 1 比較項の値が第 2 比較項の値以下である
5	gt	第 1 比較項の値が第 2 比較項の値より大きい
6	ge	第 1 比較項の値が第 2 比較項の値以上である

次に示す手順で比較を行います。

1. 各比較項に基本単位化を適用します。
2. 比較項のどれかを基本単位化した結果が空の XQuery シーケンスであれば，比較の結果は空の XQuery シーケンスになります。
3. 比較項のどれかを基本単位化した結果が二つ以上の XQuery 項目を持つ XQuery シーケンスであれば，エラーとなります。
4. 基本単位化した結果が xs:untypedAtomic 型であれば，xs:stirng 型に変換し比較します。
5. xs:string 型の値同士の比較で，比較するデータの長さが異なる場合は，短い方のデータの長さ分だけ左側から比較します。このとき，結果が等しければ更に文字列長を比較します。比較した結果，文字列長が大きい値を持つ xs:string 型の値が大きいと定まります。
6. 数値データ同士の比較で，評価対象となる XQuery データ型が異なる場合は，範囲の広い方の XQuery データ型で比較します。範囲の広さを次に示します。

   xs:double ＞ xs:decimal ＞ xs:int
7. xs:hexBinary 型同士の比較で，比較するデータの長さが異なる場合は，短い方のデータの長さ分だけを左側から比較します。このとき，結果が等しければ更にデータ長を比較します。比較した結果，データ長が大きい値を持つ xs:hexBinary 型の値が大きいと定まります。

比較項の XQuery データ型が比較できない組み合わせの場合はエラーになります。比較できる XQuery データ型の組み合わせを次の表に示します。

表 1-69　比較可能な XQuery データ型の組み合わせ

第 1 演算項の XQuery データ型	第 2 演算項の XQuery データ型						
	xs:stirng	数値データ型	xs:dateTime	xs:date	xs:time	xs:hexBinary	s:boolean
xs:stirng	○	×	×	×	×	×	×
数値データ型	×	○	×	×	×	×	×
xs:dateTime	×	×	○	×	×	×	×

第 1 演算項の XQuery データ型	第 2 演算項の XQuery データ型						
	xs:stirng	数値データ型	xs:dateTime	xs:date	xs:time	xs:hexBinary	s:boolean
xs:date	×	×	×	○	×	×	×
xs:time	×	×	×	×	○	×	×
xs:hexBinary	×	×	×	×	×	○※	×
xs:boolean	×	×	×	×	×	×	○※

（凡例）

○：比較できます。

×：比較できません。

注※

比較演算子 eq 又は ne を用いて比較できます。その他の比較演算子で比較することはできません。

xs:double 型で比較する場合の比較結果を，演算子ごとに次の表に示します。

表 1-70　xs:double 型で比較する場合の比較結果（eq 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	真	真	偽	偽	偽	偽	偽
2	-0	真	真	偽	偽	偽	偽	偽
3	+INF	偽	偽	真	偽	偽	偽	偽
4	-INF	偽	偽	偽	真	偽	偽	偽
5	NaN	偽	偽	偽	偽	偽	偽	偽
6	上記以外の負数	偽	偽	偽	偽	偽	比較結果	比較結果
7	上記以外	偽	偽	偽	偽	偽	比較結果	比較結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

比較結果：表「値比較の比較方法」の比較方法に従った比較結果

表 1-71　xs:double 型で比較する場合の比較結果（ne 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	偽	偽	真	真	真	真	真
2	-0	偽	偽	真	真	真	真	真
3	+INF	真	真	偽	真	真	真	真
4	-INF	真	真	真	偽	真	真	真
5	NaN	真	真	真	真	真	真	真
6	上記以外の負数	真	真	真	真	真	比較結果	比較結果
7	上記以外	真	真	真	真	真	比較結果	比較結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

比較結果：表「値比較の比較方法」の比較方法に従った比較結果

表 1-72　xs:double 型で比較する場合の比較結果（lt 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	偽	偽	真	偽	偽	偽	真
2	-0	偽	偽	真	偽	偽	偽	真
3	+INF	偽	偽	偽	偽	偽	偽	偽
4	-INF	真	真	真	偽	偽	真	真
5	NaN	偽	偽	偽	偽	偽	偽	偽
6	上記以外の負数	真	真	真	偽	偽	比較結果	比較結果
7	上記以外	偽	偽	真	偽	偽	比較結果	比較結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

比較結果：表「値比較の比較方法」の比較方法に従った比較結果

表 1-73　xs:double 型で比較する場合の比較結果（le 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	真	真	真	偽	偽	偽	真
2	-0	真	真	真	偽	偽	偽	真
3	+INF	偽	偽	真	偽	偽	偽	偽
4	-INF	真	真	真	真	偽	真	真
5	NaN	偽	偽	偽	偽	偽	偽	偽
6	上記以外の負数	真	真	真	偽	偽	比較結果	比較結果
7	上記以外	偽	偽	真	偽	偽	比較結果	比較結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

比較結果：表「値比較の比較方法」の比較方法に従った比較結果

表 1-74　xs:double 型で比較する場合の比較結果（gt 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	偽	偽	偽	真	偽	真	偽
2	-0	偽	偽	偽	真	偽	真	偽
3	+INF	真	真	偽	真	偽	真	真
4	-INF	偽	偽	偽	偽	偽	偽	偽
5	NaN	偽	偽	偽	偽	偽	偽	偽

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
6	上記以外の負数	偽	偽	偽	真	偽	比較結果	比較結果
7	上記以外	真	真	偽	真	偽	比較結果	比較結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

比較結果：表「値比較の比較方法」の比較方法に従った比較結果

表 1-75　xs:double 型で比較する場合の比較結果（ge 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	真	真	偽	真	偽	真	偽
2	-0	真	真	偽	真	偽	真	偽
3	+INF	真	真	真	真	偽	真	真
4	-INF	偽	偽	偽	真	偽	偽	偽
5	NaN	偽	偽	偽	偽	偽	偽	偽
6	上記以外の負数	偽	偽	偽	真	偽	比較結果	比較結果
7	上記以外	真	真	偽	真	偽	比較結果	比較結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

比較結果：表「値比較の比較方法」の比較方法に従った比較結果

- **汎用比較** ：：= {= | != | < | <= | > | >= | <>}

任意の長さの XQuery シーケンス同士を比較します。

次に示す手順で比較を行います。

1. 各比較項に基本単位化を適用します。
2. 基本単位化した各比較項の XQuery シーケンス間での基本単位値の組み合わせの中で，次に示す方法で値比較した結果が真となる基本単位値の組が一組でも存在すれば，結果は真となります。一組も存在しない場合は，結果が偽となります。
   - 二つの基本単位値に対して，汎用比較の指定に対応する表「汎用比較と対応する値比較」に示す値比較を行います。ただし，xs:untypedAtomic 型である基本単位値の場合は，他方の基本単位値の XQuery データ型に従って表「xs:untypedAtomic 型の基本単位値の型変換」に示す型変換を行ってから値比較を行います。

表 1-76　汎用比較と対応する値比較

項番	汎用比較演算子	対応する値比較
1	=	eq
2	!=，<>	ne
3	<	lt
4	<=	le
5	>	gt
6	>=	ge

表 1-77　xs:untypedAtomic 型の基本単位値の型変換

項番	他方の基本単位値 XQuery データ型	型変換後の XQuery データ型
1	数値データ型	xs:double
2	xs:untypedAtomic	xs:string
3	xs:string	xs:string
4	上記以外	他方の基本単位値と同じ XQuery データ型

- **ノード比較 ：：= {is | << | >>}**

ノード ID と，文書順序に基づいてノード同士を比較します。

ノード ID とは，XQuery データモデルで表現した木の中のすべてのノードに割り当てられている，木で一意な識別子です。

各指定での比較方法を次の表に示します。

表 1-78　ノード比較の比較方法

項番	ノード比較演算子	真となる条件
1	is	比較する二つのノードのノード ID が等しい。
2	<<	第 1 比較項のノードが第 2 比較項のノードよりも文書順序で前に出現する。
3	>>	第 1 比較項のノードが第 2 比較項のノードよりも文書順序で後に出現する。

次に示す手順で比較を行います。

1. 各比較項は，単一のノードの XQuery シーケンス，又は空の XQuery シーケンスでなければなりません。
2. どちらか一方でも空の XQuery シーケンスである場合，結果は空の XQuery シーケンスとなります。
3. どちらも単一ノードの XQuery シーケンスである場合，各比較項の XQuery シーケンスを構成するノードを比較します。
4. 各比較項で指定されるノードが，XQuery データモデルでの異なる木に属する場合は，第 1 比較項として指定されたノードの順序が前になります。

(d)　使用例

XQuery 比較式の例を次に示します。

＜ XQuery 変数 book が表す要素＞

```
<書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <カテゴリ>データベース </カテゴリ>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
  <価格>3000</価格>
</書籍情報>
```

＜ XQuery 比較式の例と結果＞

項番	例	結果	説明
1	$book[@書籍 ID eq ”452469630”]	「書籍情報」要素ノード	XQuery 変数 book が表す「書籍情報」要素ノードに属する「書籍 ID」属性ノードの型付き値が”452469630”と等しいため，結果は，XQuery 変数 book が表す「書籍情報」要素ノードとなります。
2	(1, 2) = (2, 3)	真	第 1 比較項の 2 番目の XQuery 項目と，第 2 比較項の 1 番目の XQuery 項目の間で eq の関係が成り立つため，結果は真となります。
3	(1, 2) != (2, 3)	真	第 1 比較項の 1 番目の XQuery 項目と，第 2 比較項の 1 番目の XQuery

項番	例	結果	説明
			項目の間で ne の関係が成り立つため，結果は真となります。

(9) XQuery 範囲式

(a) 機能

連続する整数値（xs:int 型）の基本単位値から構成される XQuery シーケンスを返却します。

又は，XQuery 算術式の評価結果をそのまま返却します。

(b) 形式

```
XQuery範囲式 ::= XQuery算術式 〔to XQuery算術式〕
```

(c) オペランド

- XQuery 算術式 〔to XQuery 算術式〕

to を指定した場合は，XQuery 算術式の評価結果を比較します。

XQuery 算術式を単独で指定した場合は，その XQuery 算術式の評価結果をそのまま返却します。

to を指定した場合，次の規則に従って生成した XQuery シーケンスを返却します。

1. 各 XQuery 算術式の評価結果に基本単位化を適用します。
2. 基本単位化した結果は，空の XQuery シーケンス，又は xs:int 型に変換可能な基本単位値一つから構成される単一 XQuery シーケンスである必要があります。
3. XQuery 算術式のどれかの評価結果が空の XQuery シーケンスである場合は，結果は空の XQuery シーケンスになります。
4. 一つ目の XQuery 算術式の評価結果である基本単位値を xs:int 型に変換した整数値が，二つ目の XQuery 算術式の評価結果である基本単位値を xs:int 型に変換した整数値よりも大きい場合，結果は空の XQuery シーケンスになります。
5. 上記 2.又は 3.に該当しない場合，結果は，XQuery 算術式の評価結果である基本単位値を xs:int 型に変換した二つの整数値の間（一つ目の整数値以上，二つ目の整数値以下）に含まれるすべての整数値から構成される XQuery シーケンスになります。

(d) 使用例

XQuery 範囲式の例を次に示します。

項番	例	結果	説明
1	(1 to 10)	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)	1 から 10 までの整数値から成る XQuery シーケンスを生成します。

(10) XQuery 算術式

(a) 機能

加算，減算，乗算，除算，剰余算を行った結果を返却します。

又は，XQuery シーケンス集合演算式の評価結果をそのまま返却します。

(b) 形式

```
XQuery算術式 ::= XQuery加減算式
XQuery加減算式 ::= XQuery乗除算式 〔{+|-} XQuery乗除算式〕…
XQuery乗除算式 ::= XQueryシーケンス集合演算式
                   〔{*|div|idiv|mod} XQueryシーケンス集合演算式〕…
```

(c) オペランド

- XQuery 加減算式 ::= XQuery 乗除算式 〔{+|-} XQuery 乗除算式〕…

演算子（+，−）を指定した場合は，XQuery 乗除算式の評価結果に対して，加算又は減算を行います。

XQuery 乗除算式を単独で指定した場合は，その XQuery 乗除算式の評価結果をそのまま返却します。

各演算子の意味と機能を次の表に示します。

表 1-79 XQuery 加減算式での演算子の機能

項番	演算子	意味	機能
1	+	加算	第 1 演算項に第 2 演算項を加えます。
2	-	減算	第 1 演算項から第 2 演算項を減らします。

- XQuery 乗除算式 ::= XQuery シーケンス集合演算式
  〔{* | div | idiv | mod} XQuery シーケンス集合演算式〕…

演算子（*，div，idiv，mod）を指定した場合は，XQuery シーケンス集合演算式の評価結果に対して，乗算，除算又は剰余算を行います。

XQuery シーケンス集合演算式を単独で指定した場合は，その XQuery シーケンス集合演算式の評価結果をそのまま返却します。

各演算子の意味と機能を次の表に示します。

表 1-80　XQuery 乗除算式での演算子の機能

項番	演算子	意味	機能
1	*	乗算	第 1 演算項に第 2 演算項を掛けます。
2	div	除算	第 1 演算項を第 2 演算項で割ります。
3	idiv	整数除算	第 1 演算項を第 2 演算項で割った商を切り捨てた整数値を求めます。
4	mod	剰余算	第 1 演算項（仮に a とします）を第 2 演算項（仮に b とします）で割ったときの剰余を求めます。演算結果は，a - (a idiv b) * b となります。

(d)　XQuery 算術式の評価

XQuery 算術式では，次に示す手順で，加算，減算，乗算，除算，剰余算を行います。

1. 各演算項に基本単位化を適用します。
2. 基本単位化した演算項のどれかが空の XQuery シーケンスである場合は，結果は空の XQuery シーケンスになります。
3. 基本単位化した演算項のどれかが二つ以上の基本単位値を持つ XQuery シーケンスである場合は，エラーになります。
4. 基本単位化した演算項の XQuery データ型が xs:untypedAtomic 型であった場合は，xs:double 型に変換します。変換できない場合はエラーになります。
5. 上記手順によって得られた二つの XQuery シーケンスに対して演算を行います。各演算子での，演算項の XQuery データ型と，結果の XQuery データ型を次の表に示します。

表 1-81　演算項の XQuery データ型と演算結果の XQuery データ型の関係（+演算子）

項番	第 1 演算項	第 2 演算項		
		xs:int	xs:decimal	xs:double
1	xs:int	xs:int	xs:decimal	xs:double
2	xs:decimal	xs:decimal	xs:decimal	xs:double
3	xs:double	xs:double	xs:double	xs:double

表 1-82　演算項の XQuery データ型と演算結果の XQuery データ型の関係（-演算子）

項番	第 1 演算項	第 2 演算項		
		xs:int	xs:decimal	xs:double
1	xs:int	xs:int	xs:decimal	xs:double
2	xs:decimal	xs:decimal	xs:decimal	xs:double
3	xs:double	xs:double	xs:double	xs:double

表 1-83　演算項の XQuery データ型と演算結果の XQuery データ型の関係（*演算子）

項番	第 1 演算項	第 2 演算項		
		xs:int	xs:decimal	xs:double
1	xs:int	xs:int	xs:decimal	xs:double
2	xs:decimal	xs:decimal	xs:decimal	xs:double
3	xs:double	xs:double	xs:double	xs:double

表 1-84　演算項の XQuery データ型と演算結果の XQuery データ型の関係（div 演算子）

項番	第 1 演算項	第 2 演算項		
		xs:int	xs:decimal	xs:double
1	xs:int	xs:decimal	xs:decimal	xs:double
2	xs:decimal	xs:decimal	xs:decimal	xs:double
3	xs:double	xs:double	xs:double	xs:double

表 1-85　演算項の XQuery データ型と演算結果の XQuery データ型の関係（idiv 演算子）

項番	第 1 演算項	第 2 演算項		
		xs:int	xs:decimal	xs:double
1	xs:int	xs:int	xs:int	xs:int
2	xs:decimal	xs:int	xs:int	xs:int
3	xs:double	xs:int	xs:int	xs:int

表 1-86　演算項の XQuery データ型と演算結果の XQuery データ型の関係（mod 演算子）

項番	第 1 演算項	第 2 演算項		
		xs:int	xs:decimal	xs:double
1	xs:int	xs:int	xs:decimal	xs:double
2	xs:decimal	xs:decimal	xs:decimal	xs:double
3	xs:double	xs:double	xs:double	xs:double

6. 演算項が次のどちらかの条件を満たす場合，エラーとなります。
   - 演算項がどちらも xs:double 型ではなく，かつ div 演算又は mod 演算の第 2 演算項に 0 を指定した場合
   - idiv 演算の第 2 演算項に 0 を指定した場合
7. 演算項の少なくとも一方が xs:double 型である場合の演算結果を，演算子ごとに次の表に示します。

表 1-87　演算項の少なくとも一方が xs:double 型である場合の演算結果（＋演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	+0	+0	+INF	-INF	NaN	加算結果	加算結果
2	-0	+0	-0	+INF	-INF	NaN	加算結果	加算結果
3	+INF	+INF	+INF	+INF	NaN	NaN	+INF	+INF
4	-INF	-INF	-INF	NaN	-INF	NaN	-INF	-INF
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	上記以外の負数	加算結果	加算結果	+INF	-INF	NaN	加算結果	加算結果
7	上記以外	加算結果	加算結果	+INF	-INF	NaN	加算結果	加算結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

加算結果：表「XQuery 加減算式での演算子の機能」に示す機能に従った演算結果

表 1-88　演算項の少なくとも一方が xs:double 型である場合の演算結果（-演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	+0	+0	-INF	+INF	NaN	減算結果	減算結果
2	-0	-0	+0	-INF	+INF	NaN	減算結果	減算結果
3	+INF	+INF	+INF	NaN	+INF	NaN	+INF	+INF
4	-INF	-INF	-INF	-INF	NaN	NaN	-INF	-INF
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	上記以外の負数	減算結果	減算結果	-INF	+INF	NaN	減算結果	減算結果
7	上記以外	減算結果	減算結果	-INF	+INF	NaN	減算結果	減算結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

減算結果：表「XQuery 加減算式での演算子の機能」に示す機能に従った演算結果

表 1-89　演算項の少なくとも一方が xs:double 型である場合の演算結果（*演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	+0	-0	NaN	NaN	NaN	乗算結果	乗算結果
2	-0	-0	+0	NaN	NaN	NaN	乗算結果	乗算結果
3	+INF	NaN	NaN	+INF	-INF	NaN	-INF	+INF
4	-INF	NaN	NaN	-INF	+INF	NaN	+INF	-INF
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	上記以外の負数	乗算結果	乗算結果	-INF	+INF	NaN	乗算結果	乗算結果
7	上記以外	乗算結果	乗算結果	+INF	-INF	NaN	乗算結果	乗算結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

乗算結果：表「XQuery 乗除算式での演算子の機能」に示す機能に従った演算結果

表 1-90　演算項の少なくとも一方が xs:double 型である場合の演算結果（div 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	NaN	NaN	+0	-0	NaN	除算結果	除算結果
2	-0	NaN	NaN	-0	+0	NaN	除算結果	除算結果
3	+INF	+INF	-INF	NaN	NaN	NaN	-INF	+INF
4	-INF	-INF	+INF	NaN	NaN	NaN	+INF	-INF
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
6	上記以外の負数	-INF	INF	-0	+0	NaN	除算結果	除算結果
7	上記以外	INF	-INF	+0	-0	NaN	除算結果	除算結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

除算結果：表「XQuery 乗除算式での演算子の機能」に示す機能に従った演算結果

表 1-91　演算項の少なくとも一方が xs:double 型である場合の演算結果（idiv 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	0 除算エラー	0 除算エラー	0	0	型変換エラー	整数除算結果	整数除算結果
2	-0	0 除算エラー	0 除算エラー	0	0	型変換エラー	整数除算結果	整数除算結果
3	+INF	0 除算エラー	0 除算エラー	型変換エラー	型変換エラー	型変換エラー	型変換エラー	型変換エラー
4	-INF	0 除算エラー	0 除算エラー	型変換エラー	型変換エラー	型変換エラー	型変換エラー	型変換エラー
5	NaN	0 除算エラー	0 除算エラー	型変換エラー	型変換エラー	型変換エラー	型変換エラー	型変換エラー
6	上記以外の負数	0 除算エラー	0 除算エラー	0	0	型変換エラー	整数除算結果	整数除算結果
7	上記以外	0 除算エラー	0 除算エラー	0	0	型変換エラー	整数除算結果	整数除算結果

（凡例）

0：xs:int 型の 0

整数除算結果：表「XQuery 乗除算式での演算子の機能」に示す機能に従った演算結果

0 除算エラー：除数が 0 であるため，エラーとなります

変換エラー：div 演算の結果を xs:int 型で表現できないため，エラーとなります

表 1-92　演算項の少なくとも一方が xs:double 型である場合の演算結果（mod 演算子）

項番	第 1 演算項（xs:double 型に変換した値）	第 2 演算項（xs:double 型に変換した値）						
		+0	-0	+INF	-INF	NaN	左記以外の負数	左記以外
1	+0	NaN	NaN	+0	+0	NaN	剰余算結果	剰余算結果
2	-0	NaN	NaN	-0	-0	NaN	剰余算結果	剰余算結果
3	+INF	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	-INF	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	上記以外の負数	NaN	NaN	第 1 演算項（xs:double 型に変換した値）	第 1 演算項（xs:double 型に変換した値）	NaN	剰余算結果	剰余算結果
7	上記以外	NaN	NaN	第 1 演算項（xs:double 型に変換した値）	第 1 演算項（xs:double 型に変換した値）	NaN	剰余算結果	剰余算結果

（凡例）

+0：正の 0

-0：負の 0

+INF：正の無限大

-INF：負の無限大

NaN：非数

第 1 演算項：第 1 演算項の値

剰余算結果：表「XQuery 乗除算式での演算子の機能」に示す機能に従った演算結果

(e)　留意事項

減算演算子（-）の前には，空白（X’20’），TAB（X'09'），NL（X'0a'）又は CR（X'0d'）のどれかを指定してください。空白（X’20’），AB（X'09'），NL（X'0a'）又は CR（X'0d'）を指定しない場合，減算演算子の直前に指定した名前の一部と見なされます。例えば,’a-b’と指定した場合は名前として解釈しますが,’a - b'や’a -b’と指定した場合は，XQuery 算術式として解釈します。

(f)　使用例

XQuery 算術式の例を次に示します。

項番	例	結果（XQuery データ型）
1	4 - 2	2 (xs:int)

項番	例	結果（XQuery データ型）
2	5 div 2	2.5 (xs:decimal)
3	5 idiv 2	2 (xs:int)
4	5 mod 2	1 (xs:int)

(11) XQuery シーケンス集合演算式

(a) 機能

ノードから構成される XQuery シーケンスを順序付けられた集合として，和集合演算，共通集合演算又は差集合演算を行った結果の XQuery シーケンスを返却します。

又は，XQuery 単項式の評価結果をそのまま返却します。

(b) 形式

```
XQueryシーケンス集合演算式 ::= XQuery和集合式
XQuery和集合式 ::= {XQuery共通集合式|XQuery差集合式}
                   〔{union|垂直棒} {XQuery共通集合式|XQuery差集合式}〕…
XQuery共通集合式 ::= XQuery単項式 〔intersect XQuery単項式〕…
XQuery差集合式 ::= XQuery単項式 〔except XQuery単項式〕…
```

(c) オペランド

- XQuery 和集合式 ::= {XQuery 共通集合式 | XQuery 差集合式}
  〔{union |垂直棒} {XQuery 共通集合式 | XQuery 差集合式}〕…

union 又は垂直棒（|）を指定した場合は，XQuery 共通集合式又は XQuery 差集合式の評価結果である二つの XQuery シーケンス中に含まれる異なるすべてのノードから構成される XQuery シーケンスを求めます。

XQuery 共通集合式又は XQuery 差集合式を単独で指定した場合は，その XQuery 共通集合式又は XQuery 差集合式の評価結果をそのまま返却します。

- XQuery 共通集合式 ::= XQuery 単項式 〔intersect XQuery 単項式〕…

intersect を指定した場合は，XQuery 単項式の評価結果である二つの XQuery シーケンスの両方に含まれる異なるすべてのノードから構成される XQuery シーケンスを求めます。

XQuery 単項式を単独で指定した場合は，その XQuery 単項式の評価結果をそのまま返却します。

- XQuery 差集合式 ::= XQuery 単項式 〔except XQuery 単項式〕…

except を指定した場合は，一つ目の XQuery 単項式の評価結果である XQuery シーケンスに含まれるが，二つ目の XQuery 単項式の評価結果である XQuery シーケンスに含まれない異なるすべてのノードから構成される XQuery シーケンスを求めます。

XQuery 単項式を単独で指定した場合は，その XQuery 単項式の評価結果をそのまま返却します。

(d) 規則

1. 結果の XQuery シーケンスからは，重複したノードは除かれます。また，結果の XQuery シーケンス中のノードの順序は文書順序になります。
2. 結果の XQuery シーケンス内に，異なる木に属するノードが混在する場合，次の規則に従って木の順序を定めます。順序が定まった木の中では，文書順序に従って結果のノードの順序を決定します。ここで，第 1 演算項の XQuery シーケンス内の各ノードが属する木を(T_{1-1},T_{1-2},…T_{1-n})，第 2 演算項の XQuery シーケンス内の各ノードが属する木を(T_{2-1},T_{2-2},…T_{2-m})とします。
   - T_{1-1}～T_{1-n} の木の間での順序は，木が第 1 演算項の XQuery シーケンス内で初めて出現した位置の順序になります。
   - T_{2-1}～T_{2-m} のうち，T_{1-1}～T_{1-n} のどの木とも異なる木の間での順序は，木が第 2 演算項の XQuery シーケンス内で初めて出現した位置の順序になります。また，T_{1-1}～T_{1-n} のどの木よりも後の順序になります。

(e) 留意事項

XQuery 中の XQuery シーケンス集合演算式（union，intersect，及び except）と SQL の問合せ本体中の集合演算（UNION，及び EXCEPT）では，評価順序が異なりますので注意してください。

(f) 使用例

1. XQuery シーケンス集合演算式の例を次に示します。

＜ XQuery シーケンス集合演算式の例と結果＞

項番	例	結果
1	(A,B) union (A,B)	(A, B)
2	(A,B) union (B,C)	(A, B, C)
3	(A,B) intersect (A,B)	(A, B)
4	(A,B) intersect (B,C)	(B)
5	(A,B) except (A,B)	()
6	(A,B) except (B,C)	(A)

注

A，B 及び C は，同じ XML 文書内の要素ノードを表します。

文書順序は A，B，C の順に割り当てられています。

(12) XQuery 単項式

(a) 機能

XQuery 値式に対して，単項演算を行った結果を返却します。

又は，XQuery 値式の評価結果をそのまま返却します。

(b) 形式

```
XQuery単項式 ： := 〔 {-｜+} 〕… XQuery値式
```

(c) オペランド

- 〔{-｜+}〕… XQuery 値式

単項演算子（-, +）を指定した場合は，XQuery 値式に対して，単項演算を行った結果を返却します。

XQuery 値式を単独で指定した場合は，XQuery 値式の評価結果をそのまま返却します。

各演算子の意味と機能を次の表に示します。

表 1-93 XQuery 単項式での演算子の機能

項番	演算子	意味	機能
1	-	負符号	符号を反転します。
2	+	正符号	符号を反転しません。

(d) XQuery 単項式の評価

XQuery 単項式では，次に示す手順で単項演算を行います。

1. 演算項に基本単位化を適用します。
2. 基本単位化した演算項が空の XQuery シーケンスである場合は，結果は空の XQuery シーケンスになります。
3. 基本単位化した演算項が二つ以上の基本単位値を持つ XQuery シーケンスである場合は，エラーになります。
4. 基本単位化した演算項の XQuery データ型が xs:untypedAtomic 型であった場合は，xs:double 型に変換します。変換できない場合はエラーになります。
5. 上記手順によって得られた XQuery シーケンスに対して演算を行います。結果の XQuery データ型は，手順 4 を終えた時点での XQuery データ型と同じになります。

(13) XQuery 値式

(a) 機能

XQuery パス式を評価した結果を返却します。

(b) 形式

```
XQuery値式 ::= XQueryパス式
```

(c) オペランド

- XQuery パス式

評価する値を XQuery パス式で指定します。

(14) XQuery パス式

(a) 機能

XQuery パス式は，XQuery データモデルでのノードを特定し，その評価結果を返却します。

(b) 形式

```
XQueryパス式 ::= {/〔相対パス式〕 | //相対パス式 | 相対パス式}
相対パス式 ::= ステップ式 〔 {/ | // } ステップ式 〕…
ステップ式 ::= {軸ステップ式 | フィルタ式}
軸ステップ式 ::= 軸ステップ〔 XQuery述語 〕…
軸ステップ ::= {軸 {名前テスト |種別テスト} | 省略軸ステップ}
名前テスト ::= {修飾名 | * | 接頭辞:* | *:局所名}
省略軸ステップ ::= { 〔@〕 {名前テスト | 種別テスト} |.. }
フィルタ式 ::= XQuery基本式〔 XQuery述語 〕…
XQuery述語 ::= 左角括弧 XQuery式 右角括弧
```

(c) オペランド

- XQuery パス式の先頭の斜線（/）

文脈項目のノードが属する木の根となるノードを特定します。

文脈項目がノードでない場合はエラーになります。

- XQuery パス式の先頭の二つの斜線（//）

文脈項目のノードが属する木の根となるノード及びそのノードが持つすべての子孫ノードを特定します。

文脈項目がノードでない場合はエラーになります。

- XQuery パス式の先頭以外の斜線（/）

XQuery パス式の先頭以外で指定する’/’は，ステップ式の区切りを意味します。ステップ式を評価して得られた XQuery シーケンスは，左側から右側に向かって評価されます。

例えば,’E1/E2’という相対パス式は，次のように評価します。

1. ステップ式 E1 を評価します。
2. ステップ式 E1 を評価して得られた XQuery シーケンスの各 XQuery 項目をステップ式 E2 の評価での文脈項目とし，その各文脈項目に対してステップ式 E2 を評価します。ステップ式 E1 を評価して得られた XQuery シーケンスが空の場合は，E2 は評価しないで,’E1/E2’の評価結果は空の XQuery シーケンスになります。
3. 2.で各文脈項目に対してステップ式 E2 を評価して得られた XQuery シーケンスを連結します。連結した結果の XQuery シーケンスが’E1/E2’を評価した結果となります。

斜線の左側のステップ式を評価した結果の XQuery シーケンスは，基本単位値を含むことはできません。

- **XQuery パス式の先頭以外の二つの斜線（//）**

’/descendant-or-self::node()/’の省略記法です。

左側に指定したステップ式を評価結果した後の各文脈項目での，子孫ノード及び文脈項目自身から構成される XQuery シーケンスに対して右側に指定したステップ式を評価します。

二つの斜線の左側のステップ式を評価した結果の XQuery シーケンスは，基本単位値を含むことはできません。

- **軸ステップ式 ：：＝ 軸ステップ〔 XQuery 述語 〕…**

軸ステップ式は，文脈項目を基点として，指定された木のたどり方で到達可能なノードに対して，指定された名前，ノードの種別及び条件に従って絞り込んだ結果のノードの XQuery シーケンスを返します。この XQuery シーケンス中のノードが次の文脈項目になります。

文脈項目がノードでない場合はエラーになります。

- **軸ステップ ：：＝ {軸 {名前テスト ｜種別テスト} ｜ 省略軸ステップ}**

軸ステップは，文脈項目を基点として，指定された軸を経由して到達可能なノード中の，名前テスト又は種別テストに合致するノードの XQuery シーケンスを返します。この XQuery シーケンス中のノードが次の文脈項目になります。

軸

文脈項目からの木のたどり方を指定します。

軸として指定できる項目，及び返却する文脈項目の文脈位置の割り当て方法を，次の表と図で示します。

表 1-94　軸の一覧

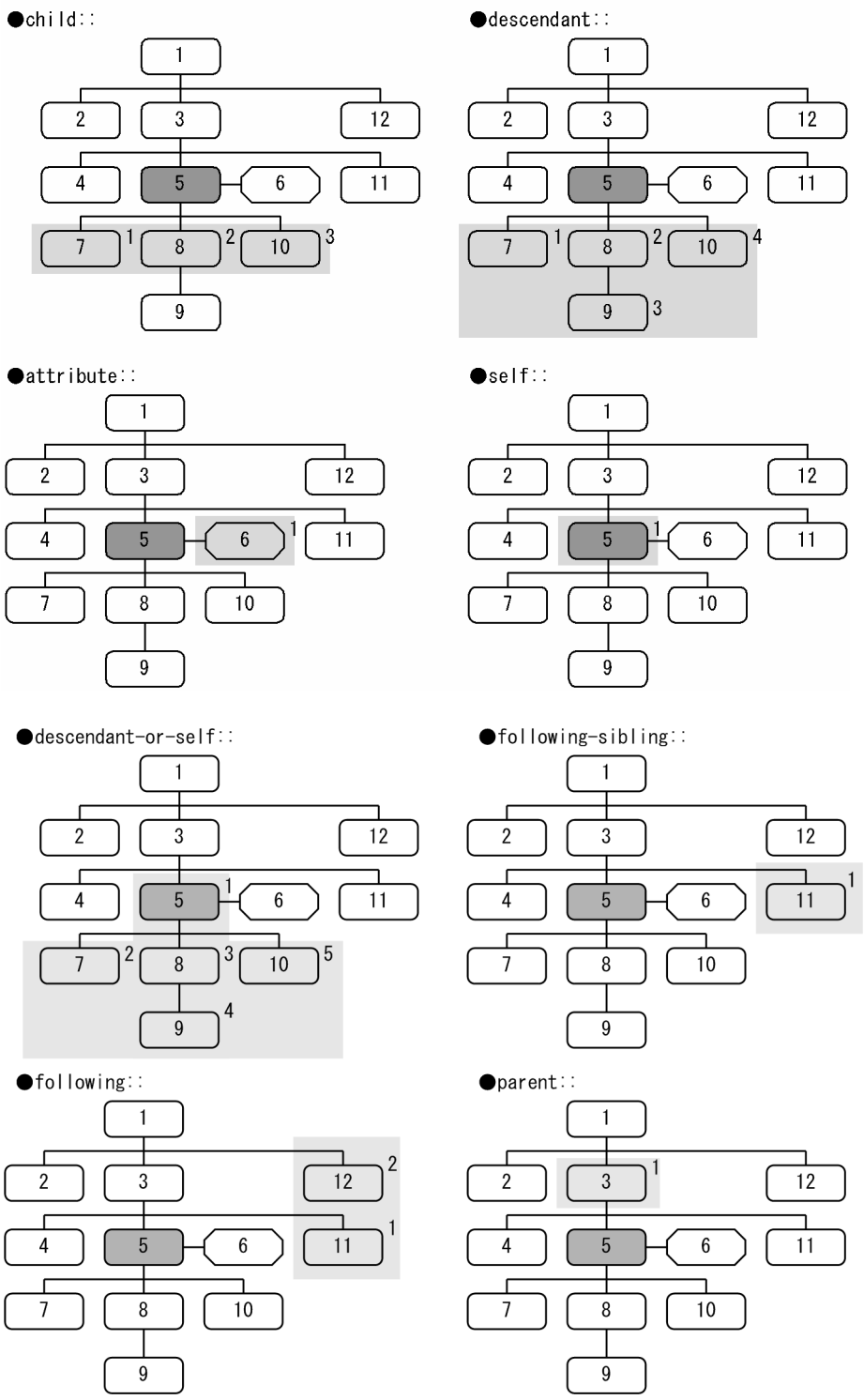
項番	軸	説明	文脈位置の割り当て
1	child::	子ノードを示します。	文書順
2	descendant::	子孫ノードを示します。	
3	attribute::	文脈項目であるノードに属する属性ノードを示します。	
4	self::	文脈項目であるノード自身を示します。	
5	descendant-or-self	子孫ノード及び文脈項目であるノード自身を示します。	
6	following-sibling::	文脈項目であるノードの兄弟となるノードの中で，後の文書順序にあるノードを示します。文脈項目であるノードが属性ノードである場合，結果は空の XQuery シーケンスです。	
7	following::	文脈項目であるノードの子孫ではなく，後の文書順序にあるノードを示します。なお，属性ノードは選択されません。	
8	parent::	親ノードを示します。文脈項目であるノードが親ノードを持たない場合，結果は空の XQuery シーケンスです。	
9	ancestor::	祖先ノードを示します。	逆文書順
10	preceding-sibling::	文脈項目であるノードの兄弟となるノードで，前の文書順序にあるノードを示します。文脈項目であるノードが属性ノードの場合，結果は空の XQuery シーケンスです。	
11	preceding::	文脈項目であるノードの祖先でなく，前の文書順序にあるノードを示します。なお，属性ノードは含まれません。	
12	ancestor-or-self::	祖先ノードと，文脈項目であるノード自身を示します。	

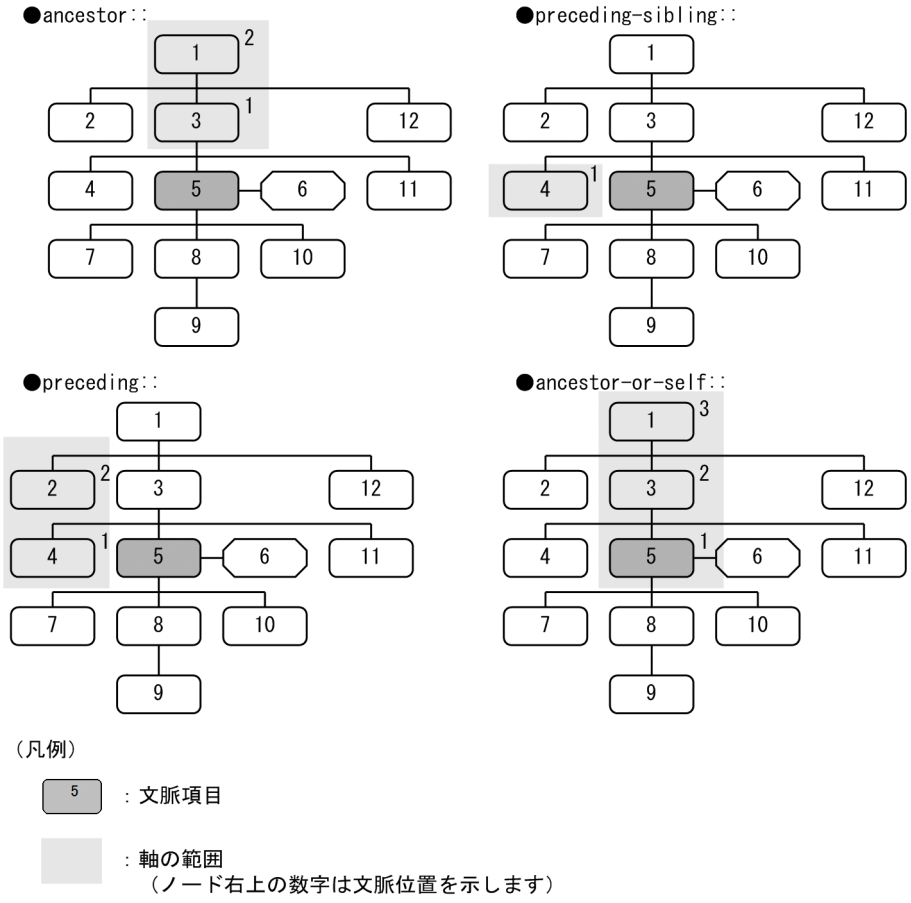
（凡例）

文書順：文書順序に沿った順序で，文脈位置が割り当てられます。

逆文書順：文書順序とは逆の順序で，文脈位置が割り当てられます。

図 1-17　軸


●child::
●descendant::
●attribute::
●self::
●descendant-or-self::
●following-sibling::
●following::
●parent::



- **名前テスト ：：＝ {修飾名｜*｜接頭辞:*｜*:局所名}**

軸で指定されたたどり方で到達可能なノード中から，名前が合致するノードを絞り込む場合に指定します。

修飾名

指定した修飾名の展開後修飾名と，名前テストの対象であるノードの展開後修飾名が等価であるノードを絞り込みます。

*****

名前テストの対象であるノードを絞り込まない場合に指定します。

接頭辞:*

名前テストの対象であるノードのうち，その展開後修飾名の XML 名前空間 URI がここで指定された接頭辞に対応する URI と等しいノードを絞り込みます。局所名には依存しません。

***:局所名**

名前テストの対象であるノードのうち，局所名がここで指定された局所名に等しいノードを絞り込みます。XML 名前空間 URI には依存しません。

- **種別テスト**

軸で指定されたたどり方でたどって得られたノード中から，種別が合致するノードを絞り込む場合に指定します。

種別テストには，次の表に示すものがあります。

表 1-95　種別テスト一覧

項番	種類	指定方法	説明
1	文書テスト	document-node()	文書ノードに合致します。
2		document-node(要素テスト)	要素ノードを子ノードとして必ず一つだけ持ち，その他の子ノードがコメントノード又は処理命令ノードだけである文書ノードのうち，子ノードの要素ノードが要素テストに合致する文書ノードに合致します。
3	要素テスト	element()	任意の要素ノードに合致します。
4		element(*)	
5		element(修飾名)	XML 要素のうち，指定した修飾名の展開後修飾名と等価な展開後修飾名を持つ要素ノードに合致します。
6	属性テスト	attribute()	任意の属性ノードに合致します。
7		attribute(*)	
8		attribute(修飾名)	XML 属性のうち，指定した修飾名の展開後修飾名と等価な展開後修飾名を持つ属性ノードに合致します。
9	処理命令テスト	processing-instruction()	任意の処理命令ノードに合致します。
10		processing-instruction(処理命令ターゲット)	処理命令ノードのうち，指定した処理命令ターゲットと一致する処理命令ターゲットプロパティの値を持つ処理命令ノードに合致します。
11	コメントテスト	comment()	任意のコメントノードに合致します。
12	テキストテスト	text()	任意のテキストノードに合致します。
13	任意種別テスト	node()	任意の種別のノードに合致します。

- **省略軸ステップ ：：＝ ｛〔@〕｛名前テスト ｜ 種別テスト｝ ｜.. ｝**

軸ステップを簡略化して記述する場合に指定します。

〔@〕｛名前テスト ｜ 種別テスト｝

‘@’は’，attribute::’に置き換わります。

‘@’を省略した場合は，種別テストが属性テストの場合は’attribute::’が，それ以外の場合は’child::’が名前テストや種別テストの前に指定されたものと仮定します。

..

‘..’は,’parent::node()’に置き換わります。

省略軸ステップの指定例を次の表に示します。

表 1-96　省略軸ステップの指定例

項番	省略軸ステップ	同じ意味を持つ省略しない記法	意味
1	/elem/ElemName	/elem/child::ElemName	「elem」評価後での文脈項目の子ノード中の，「ElemName」という名前の子ノードを意味します。
2	/elem/@AttrName	/elem/attribute::AttrName	「elem」評価後での文脈項目の属性ノード中の，「AttrName」という名前の属性ノードを意味します。
3	/elem/attribute()	/elem/attribute::attribute()	「elem」評価後での文脈項目のすべての属性ノードを意味します。
4	/elem/..	/elem/parent::node()	「elem」評価後での文脈項目の親ノードを意味します。

- フィルタ式 ：：= XQuery 基本式〔 XQuery 述語 〕…

XQuery 基本式を評価します。XQuery 述語を指定した場合は，XQuery 基本式の評価結果である XQuery シーケンスの XQuery 項目を XQuery 述語で指定した条件で絞り込みます。

- XQuery 述語：：= 左角括弧 XQuery 式 右角括弧

文脈項目を絞り込む条件を XQuery 式で指定します。

各文脈項目に対して XQuery 述語に指定した XQuery 式を評価します。

XQuery 述語の評価結果は，次のようになります。

- XQuery 述語に指定した XQuery 式の結果が，数値データ型の基本単位値一つの XQuery シーケンスである場合，その値と等しい文脈位置である文脈項目が XQuery 述語の評価結果となります。
- XQuery 述語に指定した XQuery 式の結果が，数値データ型の基本単位値一つの XQuery シーケンスでない場合，XQuery 式の結果をブーリアン値に変換して真となる文脈項目が XQuery 述語の評価結果となります。

(d)　規則

1. 一つ以上の斜線（/）を含む XQuery パス式の評価結果である XQuery シーケンスは次のようになります。
   - XQuery パス式の最後のステップ式の結果がノードだけを含む XQuery シーケンスである場合，重複されるノードは除かれ，ノードの順序は文書順序に並び替えた XQuery シーケンスが結果になります。
   - XQuery パス式の最後のステップ式の結果が基本単位値だけを含む XQuery シーケンスである場合は，そのまま結果になります。
   - XQuery パス式の最後のステップ式の結果がノード及び基本単位値を含む XQuery シーケンスの場合は，エラーになります。

(e) 使用例

XQuery パス式の例を次に示します。

項番	例	説明
1	child::書籍情報	文脈項目の子ノードである「書籍情報」要素ノードを選択します。
2	書籍情報	‘child::書籍情報’ の省略記法です。
3	attribute::書籍 ID	文脈項目の書籍 ID 属性ノードを選択します。
4	@書籍 ID	‘attribute::書籍 ID’ の省略記法です。
5	attribute::*	文脈項目に属するすべての属性ノードを選択します。
6	parent::element()	文脈項目の親ノードである要素ノードを選択します。
7	書籍情報[カテゴリ=”プログラミング”]	子ノードとして，「カテゴリ」要素ノードを持ち，その型付き値プロパティの値が”プログラミング”である「書籍情報」要素ノードを選択します。
8	書籍情報[@書籍 ID = “12345”]	文脈項目の子ノードであり，書籍 ID 属性の値が”12345”である「書籍情報」要素ノードを選択します。
9	著者[2]	文脈項目の子ノードである，2 番目の「著者」要素ノードを選択します。

(15) XQuery 基本式

(a) 機能

XQuery パス式を構成する基本的な構成要素です。

XQuery 基本式には，次に示すものがあります。

- XQuery 定数
- XQuery 変数参照
- 括弧付き XQuery 式
- 文脈項目式
- XQuery 関数呼出し

(b) 形式

```
XQuery基本式 ： := {XQuery定数 | XQuery変数参照 |
                    括弧付きXQuery式 | 文脈項目式 | XQuery関数呼出し}
```

各 XQuery 基本式について説明します。

(c) XQuery 定数

- 機能

  XQuery 定数は，基本単位値を直接表現するための式です。解析時に値が決定します。

  XQuery 中で指定できる XQuery 定数を次の表に示します。

表 1-97 XQuery 定数

項番	XQuery 定数		結果		解釈する XQuery データ型
			形式	説明	
1	XQuery 数値定数	XQuery 整数定数※	〔符号〕符号なし整数 (例) -123 45 6789	符号なし整数は，数字の並びで表します。 符号は，＋又は－で表します。	xs:int
2		XQuery10 進数定数	〔符号〕〔整数部〕.〔小数部〕 (例) 12.3 -456. .789	整数部と小数部は符号なし整数で表します。整数部か小数部のどちらか一方は必ず指定する必要があります。	xs:decimal
3		XQuery 浮動小数点定数	仮数〔E ｜ e〕指数 (例) 1.0E2 .5E+67	仮数は,XQuery 整数定数又は XQuery10 進数定数で表します。指数は 1～3 けたの XQuery 整数定数で表します。指数は 10 のべき乗を表します。	xs:double
4	XQuery 文字列定数		{“文字列” ｜ ‘文字列’} (例) “HiRDB” ‘HITACHI’ “’ 06.9.13’	文字列は文字の並びで表します。 引用符で囲んだ文字列中に引用符そのものを指定する場合は，引用符を 2 個続けて書いてください。 アポストロフィで囲んだ文字列中にアポストロフィそのものを指定する場合は，アポストロフィを 2 個続けて書いてください。 次に示す文字列は，対応する文字を意味します。 •「&」：& (アンパーサンド) •「<」：< (小なり演算子) •「>」：> (大なり演算子) •「"」：“ (引用符) •「'」：’ (アポストロフィ) ’&’ は単独で指定できません。	xs:string

注※

XQuery 整数定数の値の範囲を超える定数を整数定数の表記法で指定すると，HiRDB は，定数の右側に小数点を仮定し，10 進数定数が指定されたものと解釈します。

(d) XQuery 変数参照

- 機能

  XQuery 変数参照は，XQuery 式中で XQuery 変数の値を返却します。

- 形式

```
XQuery変数参照 ： := $ XQuery変数名
```

- オペランド

  XQuery 変数名

  参照したい XQuery 変数の修飾名を指定します。

  指定した修飾名を持つ有効な XQuery 変数の値を返却します。該当する XQuery 変数が複数該当する場合は，最も内側の FLWOR 式や XQuery 限定式で指定した XQuery 変数の値を返却します。

- 使用例

  変数名が PRICE である XQuery 変数の値より大きい型付き値プロパティを持つ「価格」XML 要素を，子の XML 要素として持つ「書籍情報」XML 要素に表します。

```
/書籍情報[価格> $PRICE]
```

(e) 括弧付き XQuery 式

- 機能

  XQuery 式を評価した結果を返却します。括弧で囲まれた XQuery 式の評価を先に行う場合に使用します。

  また，括弧内に XQuery 式を指定しない場合は，空の XQuery シーケンスを表します。

- 形式

```
括弧付きXQuery式 ： := （ 〔 XQuery式 〕 ）
```

- 使用例

  括弧付き XQuery 式の例を次に示します。

項番	例	結果	説明
1	(2 + 4) * 5	(30)	括弧で囲まれた’2+4’が先に評価されます。’2+4’を括弧で囲わない場合は，XQuery 乗除算式である’4*5’を先に評価するため，結果は 22 になります。

(f) 文脈項目式

- 機能

文脈項目式は，文脈項目式を評価する直前の文脈項目を構成する XQuery シーケンスを返却します。

- 形式

```
文脈項目式 ：：= ピリオド
ピリオド ：：= .
```

- 使用例

文脈項目式の例を次に示します。

項番	例	結果	説明
1	(1 to 20)[. mod 5 eq 0]	(5, 10, 15, 20)	まず，XQuery 範囲式により，1 から 20 までの整数値から構成される XQuery シーケンスが生成されます。生成された XQuery シーケンスの各 XQuery 項目が文脈項目となり，範囲式に続く XQuery 述語が評価されます。この評価の結果が真となる XQuery 項目から構成される XQuery シーケンスが結果となります。

(g) XQuery 関数呼出し

- 機能

XQuery 関数呼出しは，XQuery 関数を呼び出し，評価した結果を返却します。

- 形式

```
関数呼出し ：：= XQuery関数名（ 〔 引数 〔 , 引数〕… 〕 ）
引数 ：：= XQuery式単独
```

- オペランド

- XQuery 関数名

呼び出す XQuery 関数の修飾名を指定します。

- 引数：：= XQuery 式単独

XQuery 式単独

呼び出す XQuery 関数のパラメタに対する値を表す XQuery 式単独を指定します。

引数にコンマ演算子を含む場合，最上位のコンマ演算子を含む XQuery 式を括弧で囲んでください。

- 規則

1. 引数は，指定した順序でパラメタと対応します。
2. XQuery 関数の引数の XQuery データ型が基本単位型である場合，次の手順で引数を評価して関数に渡します。

   (a) XQuery 関数の引数に指定した XQuery 単項式を評価し，評価結果である XQuery シーケンスを求めます。

(b) 引数の評価結果に基本単位化を適用し，基本単位値から構成された XQuery シーケンスに変換します。

(c) XQuery シーケンスに含まれる XQuery 項目それぞれに対して，次の変換をします。

・XQuery 項目の XQuery データ型が xs:untypedAtomic 型，又は xs:string 型であり，かつ関数のパラメタの XQuery データ型が数値データ型の場合

xs:double 型に変換します。

・上記以外の場合

関数のパラメタの XQuery データ型に変換します。

3. XQuery 関数の引数の XQuery データ型が基本単位型以外である場合，XQuery 関数の引数に指定した XQuery 単項式を評価し，評価結果である XQuery シーケンスを関数に渡します。
4. 規則 2 及び 3 の各手順の XQuery データ型の変換で，互換性がなく変換できない場合はエラーとなります。互換性があるデータ型については，「XQuery データ型」を参照してください。

- **関数の呼出し規則**

1. XQuery 関数の修飾名と引数の数が一致すれば，その XQuery 関数を呼び出します。それ以外の場合は，エラーとなります。

(16) XQuery 変換式

(a) 機能

XQuery データモデルの木に，ノードの挿入，削除，名前変更，又は置換をして，その結果の XQuery データモデルの木を返却する式です。

XQuery 変換式では，テキストノードに含まれる空白類に対して，次の正規化処理をしてからノードの挿入又は置換をします。

- 先頭及び末尾の空白類を除去
- 連続する空白類を一つの空白（X'20'）に置き換え

空白類とは，空白（X'20'），水平タブ（X'09'），改行（X'0A'），及び復帰（X'0D'）を指します。

XQuery 変換式は，UPDATE 文の更新値に指定した XMLQUERY の XQuery 式の最上位にだけ指定できます。

XQuery 変換式では，DTD，XML スキーマがないものとして処理されます。

XQuery 変換式を使用する場合は，マニュアル「HiRDB XML Extension」の「XQuery 変換式使用時の運用」を参照してください。

(b) 形式

```
XQuery変換式 ::= copy $ XQuery変数名 :=  文脈項目式
                 modify  { XQuery変更式
```

1. 基本項目

```
                              |( XQuery変更式 〔, XQuery変更式 〕… ) }
                    return XQuery変数参照
XQuery変更式 ::= { XQuery挿入式
                     | XQuery削除式
                     | XQuery名前変更式
                     | XQuery置換式
                   }
```

(c) オペランド

- XQuery 変換式 ::= copy $ XQuery 変数名 := 文脈項目式
  modify {XQuery 変更式
  |(XQuery 変更式 〔,XQuery 変更式〕…)}
  return XQuery 変数参照

XQuery 変換式は，最初に copy 句で XQuery データモデルの木をコピーします。次に，modify 句で指定した XQuery 変更式で，XQuery データモデルの木に対して，ノードの挿入，削除，名前変更，又は置換をします。最後に，return 句で変換後の XQuery データモデルの木を返却する式です。

XQuery 変数名

文脈項目式をコピーする XQuery 変数の名前を指定します。ここで指定した XQuery 変数は，該当する XQuery 変換式の中の modify 句に指定する XQuery 変更式中，及び return 句に指定する XQuery 変数参照で有効となります。

XQuery 変更式

copy 句でコピーした XQuery データモデルの木に，ノードの挿入，削除，名前変更，又は置換をする XQuery 変更式を指定します。同じ XQuery 変換式中に XQuery 変更式を複数指定する場合は，次の順序で XQuery データモデルの木を変更します。

1. as first into，as last into，before，若しくは after の指定がない XQuery 挿入式，value of 指定がある XQuery ターゲット式が要素ノード以外の XQuery 置換式，又は XQuery 名前変更式
2. as first into，as last into，before，又は after の指定がある XQuery 挿入式
3. value of の指定がない XQuery 置換式
4. value of の指定がある XQuery ターゲット式が要素ノードの XQuery 置換式
5. XQuery 削除式

変更処理中の XQuery データモデルの木が，次の条件のどれかを満たす場合はエラーになります。

- 属性ノードの修飾名の接頭辞が，属性ノードの親ノードの接頭辞又は同じ親を持つ属性ノードの接頭辞と同じで，各接頭辞の XML 名前空間 URI が異なる。
- 一つの要素ノードが重複する属性ノードの修飾名を持っている。
- ドキュメントノードの子ノードに要素ノードが存在しない。
- ドキュメントノードの子ノードに要素ノードが複数存在する。
- ドキュメントノードの子ノードにテキストノードが存在する。

- 接頭辞及び局所名の長さが 4,096 バイトを超えている。
- 要素のネスト数が 100 を超えている。

また，変更処理後の XML のサイズが 5 メガバイトを超える場合はエラーになります。

XQuery 変数参照

XQuery 変数参照には，copy 句で指定した XQuery 変数名だけ指定できます。

- **XQuery 変更式 ：：＝ { XQuery 挿入式**
  | XQuery 削除式
  | XQuery 名前変更式
  | XQuery 置換式}

XQuery 変更式は，XQuery 変換式の copy 句で指定した XQuery データモデルの木に存在するノードだけ変更できます。例えば，XQuery 削除式は，同じ XQuery 変換式中の XQuery 挿入式で追加したノードは削除できませんが，XQuery 変換式の copy 句に指定した XQuery シーケンス中のノードは削除できます。また，XQuery 名前変更式で名前変更したノードの変更はできますが，XQuery 置換式で置換したノードの変更はできません。

(d) 使用例

表の定義と，表に挿入されているデータを次に示します。

＜表定義 SQL ＞

```
CREATE TABLE 書籍管理表（書籍ID INTEGER, 書籍情報 XML）
```

＜書籍管理表に挿入されているデータ＞

書籍 ID	書籍情報
452469631	<書籍情報> 　<タイトル>XML 入門教科書</タイトル> 　<著者>中村弘子</著者> 　<著者>日立太郎</著者> 　<価格>2500</価格> </書籍情報>

XQuery 変換式の例を次に示します。

（例 1）

書籍管理表の書籍情報の価格を"2500"から"2000"に置換します。

＜実行 SQL ＞

```
UPDATE 書籍管理表 SET 書籍情報 = XMLQUERY(
'copy $new := .
modify replace node $new/書籍情報/価格/text() with "2000"
```

```
return $new'
PASSING BY VALUE 書籍情報 EMPTY ON EMPTY)
```

＜実行結果＞

書籍 ID	書籍情報
452469631	<書籍情報> 　<タイトル>XML 入門教科書</タイトル> 　<著者>中村弘子</著者> 　<著者>日立太郎</著者> 　<価格>2000</価格> </書籍情報>

(例 2)

書籍管理表の書籍情報の書籍情報要素に，書籍管理表の書籍 ID の値を属性として挿入します。

＜実行 SQL ＞

```
UPDATE 書籍管理表 SET 書籍情報 = XMLQUERY(
'copy $new := .
modify insert node attribute 書籍ID { $value } into $new/書籍情報
return $new'
PASSING BY VALUE 書籍情報, 書籍ID AS "value" EMPTY ON EMPTY)
```

＜実行結果＞

書籍 ID	書籍情報
452469631	<書籍情報 書籍 ID="452469631"> 　<タイトル>XML 入門教科書</タイトル> 　<著者>中村弘子</著者> 　<著者>日立太郎</著者> 　<価格>2500</価格> </書籍情報>

(17)　XQuery 挿入式

(a)　機能

XQuery データモデルの木の指定した位置にノードを挿入する式です。

(b)　形式

```
XQuery挿入式 ::= insert { node | nodes } XQueryソース式
                  { [ as { first | last } ] into
                  | after
                  | before
                  }
                 XQueryターゲット式
XQueryソース式 ::= { XQuery式単独 | XQuery構成子 }
XQuery構成子 ::= { XQuery要素構成子 | XQuery属性構成子 | XQueryテキスト構成子
```

```
                      | XQueryコメント構成子 | XQuery処理命令構成子 }
XQuery要素構成子 ::= element 修飾名 左波括弧 〔
                       { XQuery式単独 | XQuery構成子 }
                         〔, { XQuery式単独 | XQuery構成子 } 〕…
                       〕 右波括弧
XQuery属性構成子 ::= attribute 修飾名 左波括弧 〔 XQuery式 〕 右波括弧
XQueryテキスト構成子 ::= text 左波括弧 〔 XQuery式 〕 右波括弧
XQueryコメント構成子 ::= comment 左波括弧 〔 XQuery式 〕 右波括弧
XQuery処理命令構成子 ::= processing-instruction 処理命令ターゲット
                          左波括弧 〔 XQuery式 〕 右波括弧
XQueryターゲット式 ::= XQuery式単独
```

(c) オペランド

- XQuery 挿入式 ::= insert {node | nodes} XQuery ソース式
  {〔as {first | last}〕 into
  | after
  | before}
  XQuery ターゲット式

{node | nodes}

XQuery ソース式に指定しているノードの数に関係なく，node 又は nodes を指定できます。

as first into

XQuery ターゲット式で指定したノードの，最初の子としてノードを挿入します。XQuery ソース式の XQuery シーケンスに属性ノードを含む場合は，XQuery ターゲット式に指定したノードに属性ノードを挿入します。

as last into

XQuery ターゲット式で指定したノードの，最後の子としてノードを挿入します。XQuery ソース式の XQuery シーケンスに属性ノードを含む場合は，XQuery ターゲット式に指定したノードに属性ノードを挿入します。

into

XQuery ターゲット式で指定したノードの子としてノードを挿入します。XQuery ソース式の XQuery シーケンスに属性ノードを含む場合は，XQuery ターゲット式に指定したノードに属性ノードを挿入します。

after

XQuery ターゲット式で指定したノードの直後に弟となるノードを挿入します。XQuery ソース式の XQuery シーケンスに属性ノードを含む場合は，XQuery ターゲット式に指定したノードの親ノードに属性ノードを挿入します。

before

XQuery ターゲット式で指定したノードの直前に兄となるノードを挿入します。XQuery ソース式の XQuery シーケンスに属性ノードを含む場合は，XQuery ターゲット式に指定したノードの親ノードに属性ノードを挿入します。

- XQuery ソース式 ：：＝ ｛XQuery 式単独 ｜ XQuery 構成子｝

XQuery ソース式には，挿入するノードの XQuery シーケンスを指定します。

XQuery ソース式の XQuery シーケンスの構成要素として指定する XQuery 項目と，挿入されるノードとの関係を次に示します。

項番	XQuery ソース式の XQuery シーケンスの構成要素として指定する XQuery 項目		挿入されるノード
1	文書ノード		指定した文書ノードの子ノード
2	要素ノード		指定した要素ノード
3	属性ノード※1，※2，※3，※4		指定した属性ノード
4	テキストノード	テキストノードを連続して二つ以上指定した場合	指定した各テキストノードの内容を連結した文字列を内容に持つ，単一のテキストノード※5
		その他の場合	指定したテキストノード※5
5	コメントノード		指定したコメントノード
6	処理命令ノード		指定した処理命令ノード
7	基本単位値	基本単位値を連続して二つ以上指定した場合	指定した基本単位値を xs:string 型に変換し，各値の間に半角空白（X'20'）を追加して連結した文字列を内容に持つ，単一のテキストノード
		その他の場合	指定した基本単位値を xs:string 型に変換した文字列を内容に持つ，テキストノード※5

注※1

一つの要素ノード内で重複しない修飾名を指定してください。

注※2

一つの要素ノード内で，属性ノードは属性ノード以外の前に指定してください。

注※3

親ノードと同じ接頭辞の属性ノードを挿入する場合，それらの接頭辞の XML 名前空間 URI が異なるものは指定できません。

注※4

同じ接頭辞を持つ複数の属性ノードを挿入する場合，それらの接頭辞の XML 名前空間 URI が異なるものは指定できません。

注※5

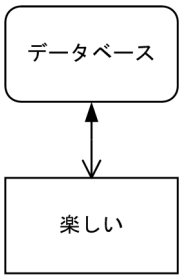
テキストノードの内容が空の場合，そのテキストノードは省略されます。

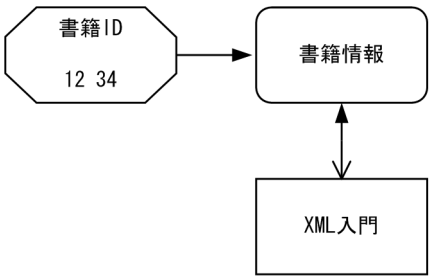
- XQuery 構成子 ：：＝ ｛XQuery 要素構成子 ｜ XQuery 属性構成子
  ｜ XQuery テキスト構成子 ｜ XQuery コメント構成子
  ｜ XQuery 処理命令構成子｝

XQuery 構成子は，要素ノード，属性ノード，テキストノード，コメントノード，又は処理命令ノードから構成される XQuery シーケンスを返します。

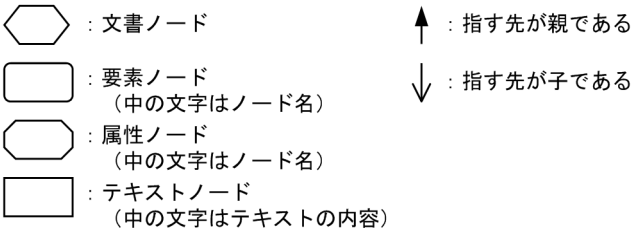
XQuery 構成子の例を次に示します。

＜XQuery 構成子の例＞

項番	例	作成される XQuery データモデル	説明
1	element タイトル{}		ノード名が「タイトル」の要素ノードを返却します。
2	element データベース{"楽しい"}		ノード名が「データベース」の要素ノードを返却します。 要素ノードは，子ノードに内容が「楽しい」のテキストノードを持ちます。
3	attribute 書籍 ID {"1234"}		文字列値に「1234」の値を持つ，ノード名が「書籍 ID」の属性ノードを返却します。
4	text{"日立", "太郎"}		内容が「日立△太郎」のテキストノードを返却します。

5	element 書籍情報{
	attribute 書籍 ID {"12","34"},
	text {"XML"},text{"入門"}
	}
			ノード名が「書籍情報」の要素ノードを返却します。
			要素ノードは，子ノードに内容が「XML 入門」のテキストノードと，文字列値が「12△34」でノード名が「書籍 ID」の属性ノードを持ちます。

（凡例）



＜指定できない XQuery 構成子の例＞

項番	例	説明

1	element 書籍情報{
	attribute 書籍 ID{”123456”},
	attribute 書籍 ID{”A-1”}
	}
		同じ名前の属性ノードは複数指定できません。例の下線部が該当します。

項番	例	説明

2	element 書籍情報{
	text {”XML 入門”} ,
	attribute 書籍 ID{”123456”}
	}
		属性ノード以外の後に属性ノードを指定することはできません。例の下線部が該当します。

- XQuery 要素構成子 ：：= element 修飾名 左波括弧 〔
  { XQuery 式単独 ｜ XQuery 構成子 }
  〔, { XQuery 式単独 ｜ XQuery 構成子 } 〕…
  〕右波括弧

XQuery 要素構成子は，要素ノードの XQuery シーケンスを返します。

修飾名

作成する要素ノードの修飾名を指定します。

修飾名の接頭辞を指定する場合，XML 名前空間宣言で宣言した接頭辞を指定してください。XML 名前空間宣言については，「XQuery 宣言部」を参照してください。

修飾名の接頭辞及び局所名に使用できる文字については，「接頭辞及び局所名に使用できる文字」を参照してください。

なお，次の要素ノードは指定できません。

1. 要素ノードの局所名が「xml」で始まる。
2. 要素ノードの修飾名の接頭辞が「xmlns」である。
3. 要素ノードの修飾名の接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/2000/xmlns/」である。
4. 要素ノードの修飾名の接頭辞が「xml」で，かつその接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/XML/1998/namespace」以外である。
5. 要素ノードの修飾名の接頭辞が「xml」以外で，かつその接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/XML/1998/namespace」である。
6. 接頭辞が「xmlns」の XML 名前空間を含む要素ノードである。

XQuery 式

XQuery 式には，作成する要素ノードが持つ，0 個以上の子ノード又は属性ノードから成る XQuery シーケンスを指定します。

XQuery 式の，XQuery シーケンスの構成要素として指定する XQuery 項目と，作成する要素ノードが持つことになる子ノード又は属性ノードを次に示します。

項番	XQuery 式の XQuery シーケンスの構成要素として指定する XQuery 項目	作成する要素ノードが持つことになる子ノード又は属性ノード
1	文書ノード	指定した文書ノードの子ノード
2	要素ノード	指定した要素ノード

項番	XQuery 式の XQuery シーケンスの構成要素として指定する XQuery 項目		作成する要素ノードが持つことになる子ノード又は属性ノード
3	属性ノード※1，※2		指定した属性ノード
4	テキストノード	テキストノードを連続して二つ以上指定した場合	指定した各テキストノードの内容を連結した文字列を内容に持つ，単一のテキストノード※3
		その他の場合	指定したテキストノード※3
5	コメントノード		指定したコメントノード
6	処理命令ノード		指定した処理命令ノード
7	基本単位値	基本単位値を連続して二つ以上指定した場合	指定した基本単位値を xs:string 型に変換し，各値の間に半角空白（X'20'）を追加して連結した文字列を内容に持つ，単一のテキストノード
		その他の場合	指定した基本単位値を xs:string 型に変換した文字列を内容に持つ，テキストノード※3

注※1

一つの要素ノード内で重複しない修飾名を指定してください。

注※2

一つの要素ノード内で，属性ノードは属性ノード以外の前に指定してください。

注※3

テキストノードの内容が空の場合，そのテキストノードは省略されます。

- **XQuery 属性構成子 ：：= attribute 修飾名**
  左波括弧 〔 XQuery 式 〕 右波括弧

XQuery 属性構成子は属性ノードを返します。

修飾名

作成する属性ノードの修飾名を指定します。

修飾名の接頭辞を指定する場合，XML 名前空間宣言で宣言した接頭辞を指定します。XML 名前空間宣言については，「XQuery 宣言部」を参照してください。

修飾名の接頭辞及び局所名に使用できる文字については，「接頭辞及び局所名に使用できる文字」を参照してください。

なお，次の属性ノードは指定できません。

1. 属性ノードの修飾名の接頭辞が「xmlns」である。
2. 属性ノードの修飾名の接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/2000/xmlns/」である。
3. 属性ノードの修飾名の接頭辞が「xml」で，かつその接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/XML/1998/namespace」以外である。
4. 属性ノードの接頭辞が「xml」以外で，かつその接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/XML/1998/namespace」である。

XQuery 式

属性ノードの文字列値プロパティの値を指定します。

XQuery 式に基本単位化を適用した結果の基本単位値を，xs:string 型に変換した値が文字列値プロパティの値となります。基本単位化で二つ以上の基本単位値が返却される場合，各基本単位値の間に半角空白（X'20'）を追加して，連結した一つの xs:string 型の値になります。

- XQuery テキスト構成子 ：：= text 左波括弧 〔 XQuery 式 〕 右波括弧

XQuery テキスト構成子はテキストノードを返します。

XQuery 式

内容プロパティの値を指定します。内容プロパティの値が空の場合，テキストノードは省略されます。

基本単位化を適用した結果の基本単位値を，xs:string 型に変換した値が内容プロパティの値となります。基本単位化で二つ以上の基本単位値が返却される場合，各基本単位値の間に半角空白（X'20'）を追加して，連結した一つの xs:string 型の値になります。

- XQuery コメント構成子 ：：= comment 左波括弧 〔 XQuery 式 〕 右波括弧

XQuery コメント構成子はコメントノードを返します。

XQuery 式

内容プロパティの値を指定します。

基本単位化を適用した結果の基本単位値を，xs:string 型に変換した値が内容プロパティの値となります。基本単位化で二つ以上の基本単位値が返却される場合，各基本単位値の間に半角空白（X'20'）を追加して，連結した一つの xs:string 型の値になります。

内容プロパティの値には，連続するハイフンを含まない，かつ最後がハイフンで終わらない値を指定してください。

- XQuery 処理命令構成子 ：：= processing-instruction 処理命令ターゲット
  左波括弧 〔 XQuery 式 〕 右波括弧

XQuery 処理命令構成子は処理命令ノードを返します。

処理命令ターゲット

作成する処理命令ノードの処理命令ターゲットを指定します。

「XML」（大文字，小文字を問わない）でない値を指定してください。

処理命令ターゲットに使用できる文字は，「接頭辞及び局所名に使用できる文字」と同じです。

XQuery 式

処理命令ノードの内容プロパティの値を指定します。

XQuery 式に，基本単位化を適用した結果の基本単位値を，xs:string 型に変換した値が内容プロパティの値となります。基本単位化で二つ以上の基本単位値が返却される場合，各基本単位値の間に半角空白（X'20'）を追加して，連結した一つの xs:string 型の値になります。

内容プロパティの値の先頭及び末尾がホワイトスペース（半角空白（X'20'），タブ（X'09'），NL（X'0A'），又は CR（X'0D'））の場合，ホワイトスペースを取り除きます。

内容プロパティの値には，「?>」を含まない値を指定してください。

- XQuery ターゲット式 ：：= XQuery 式単独

ノードを挿入する位置を指定します。XQuery 変換式の copy 句に指定する，XQuery データモデルの木に存在するノードを指定してください。

XQuery 挿入式に as first into，as last into，又は into を指定する場合，XQuery ターゲット式には単一の要素ノードを指定してください。

XQuery 挿入式に after 又は before を指定する場合，XQuery ターゲット式には単一の要素ノード，テキストノード，コメントノード，又は処理命令ノードを指定してください。また，指定するノードは要素ノードである親ノードを持っている必要があります。

(d) 使用例

XQuery 挿入式の例を次に示します。

＜文脈項目である文書ノード以下の XML 要素＞

```
<書籍情報 書籍ID="452469631">
  <タイトル>XML入門教科書</タイトル>
  <著者>中村弘子</著者>
  <著者>日立太郎</著者>
  <価格>2500</価格>
</書籍情報>
```

＜ XQuery 挿入式の例と結果＞

項番	例	結果（下線部が変更箇所）
1	copy $new := . modify insert node element カテゴリ{"XML"} after $new/書籍情報/タイトル return $new	<書籍情報 書籍 ID="452469631"> 　<タイトル>XML 入門教科書</タイトル> 　<u><カテゴリ>XML</カテゴリ></u> 　<著者>中村弘子</著者> 　<著者>日立太郎</著者> 　<価格>2500</価格> </書籍情報>
2	copy $new := . modify insert node attribute 保管場所{"倉庫"} before $new/書籍情報/価格 return $new	<書籍情報 書籍 ID="452469631" <u>保管場所="倉庫"</u> > 　<タイトル>XML 入門教科書</タイトル> 　<著者>中村弘子</著者> 　<著者>日立太郎</著者> 　<価格>2500</価格> </書籍情報>
3	copy $new := .	<書籍情報 書籍 ID="452469631" <u>カテゴリ="XML"</u>> 　<タイトル>XML 入門教科書</タイトル>

項番	例	結果（下線部が変更箇所）
	modify (insert node attribute カテゴリ {"XML"} into $new/書籍情報, insert node element 保管場所{"倉庫"} into $new/書籍情報) return $new	<著者>中村弘子</著者> <著者>日立太郎</著者> <価格>2500</価格> <保管場所>倉庫</保管場所> </書籍情報>
4	declare namespace hirdb = "http://www.hitachi.co.jp/hirdb/"; copy $new := . modify insert node attribute hirdb:管理番号 {"ABC"} into $new/書籍情報 return $new	<書籍情報 xmlns:hirdb="http://www.hitachi.co.jp/hirdb/" 書籍 ID="452469631" hirdb:管理番号="ABC"> <タイトル>XML 入門教科書</タイトル> <著者>中村弘子</著者> <著者>日立太郎</著者> <価格>2500</価格> </書籍情報>

＜指定できない XQuery 挿入式の例と結果＞

項番	例	説明
1	copy $new := . modify insert node element カテゴリ{"XML"} before $new/書籍情報/著者 return $new	XQuery ターゲット式には，単一の要素ノードを指定する必要があります。例の下線部が該当します。
2	copy $new := . modify insert node element カテゴリ {"XML"}into $new/書籍情報/@書籍 ID return $new	into 指定した場合，XQuery ターゲット式には要素ノードを指定する必要があります。例の下線部が該当します。

(18) XQuery 削除式

(a) 機能

XQuery データモデルの木の指定した位置のノードを削除する式です。ノードが属性ノード又は子孫ノードを持つ場合は，その属性ノード又は子孫ノードも削除します。

(b) 形式

```
XQuery削除式 ::= delete { node | nodes } XQueryターゲット式
XQueryターゲット式 ::= XQuery式単独
```

(c) オペランド

- XQuery 削除式 ::= delete {node | nodes} XQuery ターゲット式

XQuery ターゲット式で指定しているノードの数に関係なく，node 又は nodes を指定できます。

- XQuery ターゲット式 ::= XQuery 式単独

削除するノードを指定します。XQuery 変換式の copy 句に指定する，XQuery データモデルの木のノード，又は空の XQuery シーケンスを指定してください。

XQuery ターゲット式で指定したノードが親を持たない場合は，省略されます。

(d) 使用例

XQuery 削除式の例を次に示します。

＜文脈項目である文書ノード以下の XML 要素＞

```
<書籍情報 書籍ID=”452469631”>
  <タイトル>XML入門教科書</タイトル>
  <著者>中村弘子</著者>
  <著者>日立太郎</著者>
  <価格>2500</価格>
</書籍情報>
```

＜ XQuery 削除式の例と結果＞

項番	例	結果
1	copy $new := . modify delete node $new/書籍情報/著者 return $new	<書籍情報 書籍 ID="452469631"> <タイトル>XML 入門教科書</タイトル> <価格>2500</価格> </書籍情報>
2	copy $new := . modify delete node $new/書籍情報/著者[. = "中村弘子"] return $new	<書籍情報 書籍 ID="452469631"> <タイトル>XML 入門教科書</タイトル> <著者>日立太郎</著者> <価格>2500</価格> </書籍情報>
3	copy $new := . modify delete node $new/書籍情報/@書籍 ID return $new	<書籍情報> <タイトル>XML 入門教科書</タイトル> <著者>中村弘子</著者> <著者>日立太郎</著者> <価格>2500</価格> </書籍情報>

(19) XQuery 名前変更式

(a) 機能

XQuery データモデルの木の，指定した位置のノードの修飾名又は処理命令ターゲットを変更する式です。

(b) 形式

```
XQuery名前変更式 ： ：= rename node XQueryターゲット式 as
                        XQuery式単独
XQueryターゲット式 ： ：= XQuery式単独
```

(c) オペランド

- XQuery 名前変更式 ： ：= rename node XQuery ターゲット式 as
  XQuery 式単独

XQuery 式単独

名前変更後の修飾名又は処理命令ターゲットを指定します。

修飾名の接頭辞を指定する場合，XML 名前空間宣言で宣言した接頭辞を指定します。XML 名前空間宣言については，「XQuery 宣言部」を参照してください。

修飾名の接頭辞及び局所名に使用できる文字については，「接頭辞及び局所名に使用できる文字」を参照してください。処理命令ターゲットで使用できる文字は，修飾名の接頭辞及び局所名と同じです。

XQuery 式単独の規則を次に示します。

1. XQuery ターゲット式が要素ノード又は属性ノードの場合は，修飾名を指定します。

   XQuery 式単独に，基本単位化を適用した結果が単一の基本単位値になり，その基本単位値の XQuery データ型が xs:string 又は xs:untypedAtomic になる XQuery 式単独を指定してください。その基本単位化を適用した結果の基本単位値が修飾名になります。

   属性ノードの修飾名に親ノードと同じ接頭辞の指定する場合，それらの接頭辞の XML 名前空間 URI は同じものを指定してください。

   なお，次の修飾名は指定できません。

   ・修飾名の接頭辞が「xmlns」である。

   ・修飾名の接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/2000/xmlns/」である。

   ・修飾名の接頭辞が「xml」で，かつその接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/XML/1998/namespace」以外である。

   ・修飾名の接頭辞が「xml」以外で，かつその接頭辞に対応する XML 名前空間 URI が「http://www.w3.org/XML/1998/namespace」である。

2. XQuery ターゲット式が要素ノードの場合，次の修飾名は指定できません。

   ・変更後の要素ノードの局所名が「xml」で始まる。

3. XQuery ターゲット式が属性ノードの場合，次の修飾名は指定できません。

   ・修飾名の接頭辞を指定しないかつ局所名が「xmlns」である。

4. XQuery ターゲット式が処理命令ノードの場合は，処理命令ターゲットを指定します。

   XQuery 式単独に基本単位化を適用した結果が単一の基本単位値になり，その基本単位値の XQuery データ型が xs:string 又は xs:untypedAtomic になる XQuery 式単独を指定してください。その基本単位化を適用した結果の基本単位値が処理命令ターゲットになります。

処理命令ターゲットが「XML」（大文字，小文字を問わない）になる XQuery 式単独は指定できません。

- XQuery ターゲット式 ：：= XQuery 式単独

名前を変更するノードを指定します。

XQuery 変換式の，copy 句に指定する XQuery データモデルの木に存在する，単一の要素ノード，属性ノード，又は処理命令ノードを指定してください。

同じ XQuery 変換式中で，複数の XQuery 名前変更式に同じノードを指定することはできません。

(d) 使用例

XQuery 名前変更式の例を次に示します。

＜文脈項目である文書ノード以下の XML 要素＞

```
<書籍情報 書籍ID=”452469631”>
 <タイトル>XML入門教科書</タイトル>
  <著者>中村弘子</著者>
  <著者>日立太郎</著者>
  <価格>2500</価格>
</書籍情報>
```

＜ XQuery 名前変更式の例と結果＞

項番	例	結果（下線部が変更箇所）
1	copy $new := . modify rename node $new/書籍情報 as "情報" return $new	<情報 書籍 ID="452469631"> 　<タイトル>XML 入門教科書</タイトル> 　<著者>中村弘子</著者> 　<著者>日立太郎</著者> 　<価格>2500</価格> </情報>
2	copy $new := . modify rename node $new/書籍情報/@書籍 ID as "管理 ID" return $new	<書籍情報 管理 ID="452469631"> 　<タイトル>XML 入門教科書</タイトル> 　<著者>中村弘子</著者> 　<著者>日立太郎</著者> 　<価格>2500</価格> </書籍情報>
3	declare namespace hirdb = "http://www.hitachi.co.jp/hirdb/"; copy $new := . modify rename node $new/書籍情報 as "hirdb:書籍情報" return $new	<hirdb:書籍情報 xmlns:hirdb="http://www.hitachi.co.jp/hirdb/" 書籍 ID="452469631"> 　<タイトル>XML 入門教科書</タイトル> 　<著者>中村弘子</著者> 　<著者>日立太郎</著者> 　<価格>2500</価格> </hirdb:書籍情報>

＜指定できない XQuery 名前変更式の例＞

項番	例	説明
1	copy $new := . modify rename node $new/書籍情報/著者 as "作者" return $new	XQuery ターゲット式には単一の要素ノードを指定する必要があります。例の下線部が該当します。

(20) XQuery 置換式

(a) 機能

XQuery データモデルの木の，指定した位置のノード又はノードの値を置換する式です。

(b) 形式

```
XQuery置換式 ::= replace 〔 value of 〕 node XQueryターゲット式
                 with XQueryソース式
XQueryターゲット式 ::= XQuery式単独
XQueryソース式 ::= { XQuery式単独 | XQuery構成子 }
XQuery構成子 ::= { XQuery要素構成子 | XQuery属性構成子 | XQueryテキスト構成子
                 | XQueryコメント構成子 | XQuery処理命令構成子 }
XQuery要素構成子 ::= element 修飾名 左波括弧 〔
                       { XQuery式単独 | XQuery構成子 }
                         〔, { XQuery式単独 | XQuery構成子 } 〕…
                       〕 右波括弧
XQuery属性構成子 ::= attribute 修飾名 左波括弧 〔 XQuery式 〕 右波括弧
XQueryテキスト構成子 ::= text 左波括弧 〔 XQuery式 〕 右波括弧
XQueryコメント構成子 ::= comment 左波括弧 〔 XQuery式 〕 右波括弧
XQuery処理命令構成子 ::= processing-instruction 処理命令ターゲット
                          左波括弧 〔 XQuery式 〕 右波括弧
```

(c) オペランド

- XQuery 置換式 ::= replace 〔 value of 〕 node XQuery ターゲット式
  with XQuery ソース式

〔 value of 〕

ノードの値を変更する場合は，指定します。

ノードを変更する場合は，省略します。

- XQuery ターゲット式 ::= XQuery 式単独

置換するノードを指定します。

XQuery 変換式の copy 句に指定する，XQuery データモデルの木に存在する単一の要素ノード，属性ノード，テキストノード，コメントノード，又は処理命令ノードを指定してください。

同じ XQuery 変換式中で，value of の指定がない複数の XQuery 置換式に同じノードを指定することはできません。

同じ XQuery 変換式中で，value of の指定がある複数の XQuery 置換式に同じノードを指定することはできません。

- **XQuery ソース式 ：：＝ ｛XQuery 式単独 ｜ XQuery 構成子｝**

置換後のノードの，XQuery シーケンス又はノードの値を指定します。

value of の指定がある場合，XQuery 式単独だけ指定できます。

1. value of を指定しない場合

   置換後のノードの XQuery シーケンスを指定します。

   XQuery ソース式の，XQuery シーケンスの構成要素として指定する XQuery 項目と，置換後のノードの関係を次に示します。

項番	XQuery ソース式の XQuery シーケンスの構成要素として指定する XQuery 項目		置換後のノード
1	文書ノード		指定した文書ノードの子ノード
2	要素ノード		指定した要素ノード
3	属性ノード※1，※2，※3		指定した属性ノード
4	テキストノード	テキストノードを連続して二つ以上指定した場合	指定した各テキストノードの内容を連結した文字列を内容に持つ，単一のテキストノード※4
		その他の場合	指定したテキストノード※4
5	コメントノード		指定したコメントノード
6	処理命令ノード		指定した処理命令ノード
7	基本単位値	基本単位値を連続して二つ以上指定した場合	指定した基本単位値を xs:string 型に変換し，各値の間に半角空白（X'20'）を追加し連結した文字列を内容に持つ，単一のテキストノード
		その他の場合	指定した基本単位値を xs:string 型に変換した文字列を内容に持つ，テキストノード※4

注※1

一つの要素ノード内で重複しない修飾名を指定してください。

注※2

親ノードと同じ接頭辞の属性ノードに置換する場合，それらの接頭辞の XML 名前空間 URI が異なるものは指定できません。

注※3

同じ接頭辞を持つ複数の属性ノードに置換する場合，それらの接頭辞の XML 名前空間 URI が異なるものは指定できません。

注※4

テキストノードの内容が空の場合，そのテキストノードは省略されます。

XQuery ターゲット式が要素ノード，テキストノード，コメントノード，又は処理命令ノードの場合，0 個以上の要素ノード，テキストノード，コメントノード，又は処理命令ノードから構成される XQuery シーケンスを指定してください。

XQuery ターゲット式が属性ノードの場合，0 個以上の属性ノードから構成される XQuery シーケンスを指定してください。

2. value of を指定する場合

置換後のノードの値を指定します。

XQuery ソース式に基本単位化を適用した結果の基本単位値を，xs:string 型に変換した値となります。基本単位化で二つ以上の基本単位値が返却される場合，各基本単位値の間に半角空白（X'20'）を追加して，連結した一つの xs:string 型の値になります。

XQuery ターゲット式がコメントノードの場合，XQuery ソース式は連続するハイフンを含まない，かつ最後がハイフンで終わらない値を指定してください。

XQuery ターゲット式が処理命令ノードの場合，XQeury ソース式は「?>」を含まない値を指定してください。

- **XQuery 構成子 ：：= ｛XQuery 要素構成子 ｜ XQuery 属性構成子**
  ｜ XQuery テキスト構成子 ｜ XQuery コメント構成子
  ｜ XQuery 処理命令構成子｝

XQuery 構成子については，「オペランド」を参照してください。

- **XQuery 要素構成子 ：：= element 修飾名 左波括弧 〔**
  ｛XQuery 式単独 ｜ XQuery 構成子｝
  〔, ｛XQuery 式単独 ｜ XQuery 構成子｝ 〕…
  〕 右波括弧

XQuery 要素構成子については，「オペランド」を参照してください。

- **XQuery 属性構成子 ：：= attribute 修飾名**
  左波括弧 〔XQuery 式〕 右波括弧

XQuery 属性構成子については，「オペランド」を参照してください。

- **XQuery テキスト構成子 ：：= text 左波括弧 〔XQuery 式〕 右波括弧**

XQuery テキスト構成子については，「オペランド」を参照してください。

- **XQuery コメント構成子 ：：= comment 左波括弧 〔XQuery 式〕 右波括弧**

XQuery コメント構成子については，「オペランド」を参照してください。

- **XQuery 処理命令構成子 ：：= processing-instruction 処理命令ターゲット**
  左波括弧 〔XQuery 式〕 右波括弧

XQuery 処理命令構成子については，「オペランド」を参照してください。

(d) 使用例

XQuery 置換式の入力例を次に示します。

＜文脈項目である文書ノード以下の XML 要素＞

```
<書籍情報 書籍ID="452469631">
  <タイトル>XML入門教科書</タイトル>
  <著者>中村弘子</著者>
  <著者>日立太郎</著者>
  <価格>2500</価格>
</書籍情報>
```

＜ XQuery 置換式の例と結果＞

項番	例	結果（下線部が変更箇所）
1	copy $new := . modify replace value of node $new/書籍情報/著者 [.="日立太郎"] with "日立花子" return $new	<書籍情報 書籍 ID="452469631"> <タイトル>XML 入門教科書</タイトル> <著者>中村弘子</著者> <著者>日立花子</著者> <価格>2500</価格> </書籍情報>
2	copy $new := . modify replace node $new/書籍情報 with element 書籍情報 {element タイトル{"データベース入門"} , element 著者{"日立太郎"}} return $new	<書籍情報> <タイトル>データベース入門</タイトル> <著者>日立太郎</著者> </書籍情報>
3	copy $new := . modify replace value of node $new/書籍情報/@書籍 ID with "10000" return $new	<書籍情報 書籍 ID="10000"> <タイトル>XML 入門教科書</タイトル> <著者>中村弘子</著者> <著者>日立太郎</著者> <価格>2500</価格> </書籍情報>

＜指定できない XQuery 置換式の例＞

項番	例	説明
1	copy $new := . modify replace node $new/書籍情報/著者 with element 著者 {"日立花子"} return $new	XQuery ターゲット式には単一の要素ノードを指定する必要があります。例の下線部が該当します。
2	copy $new := . modify (replace node $new/書籍情報/タイトル with element 書籍名 {"データベース"},	複数の XQuery 置換式に同じ XQuery ターゲット式は指定できません。例の下線部が該当します。

項番	例	説明
	replace node $new/書籍情報/タイトル with element タイトル {"XML"}) return $new	

1.15.7　XQuery コメント

XQuery コメントは，XQuery の評価結果に影響を及ぼさない注釈です。

次の形式で指定します。

```
XQueryコメント　：：=（:〔 {XQueryコメント内容｜XQueryコメント} 〕… :）
XQueryコメント内容　：：= 文字
```

1.15.8　XQuery 関数

ここでは，HiRDB が提供する XQuery 関数について説明します。

各 XQuery 関数の形式で記述した引数は，XQuery 関数呼出しの規則に従って XQuery 関数にパラメタとして渡されます。XQuery 関数呼出しの規則については，「XQuery 関数呼出し」を参照してください。

XQuery 関数には，XQuery で定義された関数と，HiRDB で定義された関数があります。

XQuery で定義された関数は，接頭辞 xs に割り当てられた XML 名前空間（http://www.w3.org/2001/XMLSchema），又は fn に割り当てられた XML 名前空間（http://www.w3.org/2006/xpath-functions）中にあります。

HiRDB で定義された関数は，接頭辞 hi-fn に割り当てられた XML 名前空間（http://www.hitachi.co.jp/Prod/comp/soft1/hirdb/xquery-functions）中にあります。HiRDB で定義された関数を呼び出す場合，XQuery 関数名に対する既定の XML 名前空間を"http://www.hitachi.co.jp/Prod/comp/soft1/hirdb/xquery-functions"で宣言している場合を除き，接頭辞を指定する必要があります。

コンストラクタ関数については，「コンストラクタ関数」を参照してください。

HiRDB で使用できる XQuery 関数を，次の表に示します。

表 1-98　HiRDB が提供する XQuery 関数

分類	関数	機能	XQuery 関数の種別
コンストラクタ関数	xs:string	xs:string 型の値を生成します。	XQuery
	xs:decimal	xs:decimal 型の値を生成します。	XQuery

分類	関数	機能	XQuery 関数の種別
	xs:int	xs:int 型の値を生成します。	XQuery
	xs:double	xs:double 型の値を生成します。	XQuery
	xs:dateTime	xs:dateTime 型の値を生成します。	XQuery
	xs:date	xs:date 型の値を生成します。	XQuery
	xs:time	xs:time 型の値を生成します。	XQuery
	xs:hexBinary	xs:hexBinary 型の値を生成します。	XQuery
	xs:boolean	xs:boolean 型の値を生成します。	XQuery
	xs:untypedAtomic	xs:untypedAtomic 型の値を生成します。	XQuery
変換関数	fn:boolean	XQuery シーケンスを xs:boolean 型の値として評価した結果を返します。	XQuery
	fn:data	XQuery シーケンスを基本単位化し，基本単位値から成る XQuery シーケンスを返します。	XQuery
	fn:number	引数に指定した値を xs:double 型に変換した値を返します。	XQuery
	fn:string	XQuery 項目の値を文字列表現した値を xs:string 型の値として返します。	XQuery
文字列操作関数	fn:compare	二つの文字列を比較します。	XQuery
	fn:concat	二つ以上の基本単位値を xs:string 型の値に変換して連結した値を返します。	XQuery
	fn:contains	引数 1 で指定した文字列中に引数 2 で指定した文字列が含まれているかどうかを返します。	XQuery
	hi-fn:contains	引数 1 で指定したノードの文字列値が，引数 2 で指定した全文検索のテキスト検索条件を満たすかどうかを返します。	HiRDB
	fn:ends-with	引数 1 で指定した文字列が引数 2 で指定した文字列で終了するかどうかを返します。	XQuery
	fn:normalize-space	文字列に対してホワイトスペースの正規化を行った結果を返します。	XQuery
	fn:starts-with	引数 1 で指定した文字列が引数 2 で指定した文字列で開始するかどうかを返します。	XQuery
	fn:string-length	文字列の長さ（文字数）を返します。	XQuery
	fn:substring	文字列の部分文字列を返します。	XQuery
	fn:substring-after	引数 1 で指定した文字列中の，引数 2 で指定した文字列が最初に出現する位置より後の部分文字列を返します。	XQuery

分類	関数	機能	XQuery 関数の種別
	fn:substring-before	引数 1 で指定した文字列中の，引数 2 で指定した文字列が最初に出現する位置より前の部分文字列を返します。	XQuery
	fn:translate	引数 1 で指定した文字列中の引数 2 で指定した文字を，引数 3 で指定した文字に置換した結果を返します。	XQuery
数学関数	fn:abs	数値データ型の値の絶対値を返します。	XQuery
	fn:ceiling	引数の値以上の，最小の整数値を返します。	XQuery
	fn:floor	引数の値以下の，最大の整数値を返します。	XQuery
	fn:round	引数の値に最も近い整数値を返します。	XQuery
日時抽出関数	fn:year-from-dateTime	時刻印から年の部分だけを抽出して返します。	XQuery
	fn:month-from-dateTime	時刻印から月の部分だけを抽出して返します。	XQuery
	fn:day-from-dateTime	時刻印から日の部分だけを抽出して返します。	XQuery
	fn:hours-from-dateTime	時刻印から時間の部分だけを抽出して返します。	XQuery
	fn:minutes-from-dateTime	時刻印から分の部分だけを抽出して返します。	XQuery
	fn:seconds-from-dateTime	時刻印から秒の部分だけを抽出して返します。	XQuery
	fn:year-from-date	日付から年の部分だけを抽出して返します。	XQuery
	fn:month-from-date	日付から月の部分だけを抽出して返します。	XQuery
	fn:day-from-date	日付から日の部分だけを抽出して返します。	XQuery
	fn:hours-from-time	時刻から時間の部分だけを抽出して返します。	XQuery
	fn:minutes-from-time	時刻から分の部分だけを抽出して返します。	XQuery
	fn:seconds-from-time	時刻から秒の部分だけを抽出して返します。	XQuery
ブーリアン関数	fn:false	偽を返します。	XQuery
	fn:not	引数を xs:boolean 型の値に変換し，その値を否定した値を返します。	XQuery
	fn:true	真を返します。	XQuery
XQuery シーケンス集約関数	fn:count	XQuery シーケンス中の XQuery 項目数を返します。	XQuery
	fn:distinct-values	XQuery シーケンスから重複する基本単位値を取り除いた XQuery シーケンスを返します。	XQuery
	fn:max	XQuery シーケンスを構成する基本単位値の最大値を返します。	XQuery

分類	関数	機能	XQuery 関数の種別
	fn:min	XQuery シーケンスを構成する基本単位値の最小値を返します。	XQuery
	fn:sum	XQuery シーケンスを構成する基本単位値の合計を返します。	XQuery
XQuery シーケンス操作関数	fn:deep-equal	二つの XQuery シーケンスが深等であるかを判別します。	XQuery
	fn:index-of	指定した XQuery シーケンス中に，指定した基本単位値と一致する XQuery 項目が現れる位置を返します。	XQuery
	fn:insert-before	XQuery シーケンスの指定した位置の前に，別の XQuery シーケンス中の XQuery 項目を挿入した結果を返します。	XQuery
	fn:remove	XQuery シーケンスから指定した位置の XQuery 項目を削除した結果を返します。	XQuery
	fn:reverse	XQuery シーケンス中の XQuery 項目を逆順に並べ替えた XQuery シーケンスを返します。	XQuery
	fn:subsequence	XQuery シーケンスの部分 XQuery シーケンスを返却します。	XQuery
文脈項目関数	fn:last	文脈サイズを返します。	XQuery
	fn:position	文脈位置を返します。	XQuery
ノード関数	fn:local-name	ノードの局所名を返します。	XQuery
	fn:name	ノードの修飾名を返します。	XQuery
	fn:namespace-uri	ノードの修飾名の XML 名前空間 URI を返します。	XQuery

（凡例）

XQuery：XQuery で定義された関数

HiRDB：HiRDB で定義された関数

(1)　abs

(a)　機能

数値データ型の値の絶対値を返します。

(b)　形式

```
fn:abs（ 引数 ）
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-99 fn:abs 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	num	数値データ型 (空の XQuery シーケンスを含む)	入力となる数値

2. 結果は，空の XQuery シーケンス，又は次に示す数値データ型の値になります。

表 1-100 fn:abs 関数の結果の XQuery データ型

項番	パラメタ num の値の XQuery データ型	結果の XQuery データ型
1	xs:double	xs:double
2	xs:decimal	xs:decimal
3	xs:int	xs:int

絶対値が結果の XQuery データ型の範囲外となった場合はエラーになります。例えば，”fn:abs(-2147483648)”を指定した場合，絶対値は 2147483648 で xs:int の範囲外の値になるのでエラーになります。

3. パラメタ num の値が空の XQuery シーケンスである場合，空の XQuery シーケンスを返します。
4. パラメタ num の値が数値データ型の値である場合，パラメタ num の値の絶対値を返します。
5. パラメタ num の値が xs:double 型の NaN（非数）の場合， NaN（非数）を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(括弧内は XQuery データ型)	説明
1	`fn:abs(10.5)`	10.5 (xs:decimal)	10.5 の絶対値は 10.5 であるため，その値を返します。
2	`fn:abs(-10.5)`	10.5 (xs:decimal)	-10.5 の絶対値は 10.5 であるため，その値を返します。

(2) boolean

(a) 機能

XQuery シーケンスを xs:boolean 型の値として評価した結果を返します。

(b) 形式

```
fn:boolean（ 引数 ）
```

(c) 規則

1. この関数は次の表に示すパラメタを持ちます。

表 1-101 fn:boolean 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の任意の XQuery 項目を持つ XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は xs:boolean 型の値になります。

3. パラメタ seq の値である XQuery シーケンスによって，次の表に示す結果を返します。

表 1-102 fn:boolean 関数の結果

項番	パラメタ seq の値	結果（xs:boolean 型）
1	空の XQuery シーケンス	偽
2	最初の XQuery 項目がノードである XQuery シーケンス	真
3	xs:boolean 型の基本単位値一つから構成される XQuery シーケンス	引数の XQuery シーケンスの基本単位値
4	xs:string 型又は xs:untypedAtomic 型の基本単位値一つから構成される XQuery シーケンス	引数の XQuery シーケンスの基本単位値の長さが 0 ならば偽，そうでなければ真
5	数値データ型の基本単位値一つから構成される XQuery シーケンス	引数の XQuery シーケンスの基本単位値が NaN（非数）又は 0 ならば偽，そうでなければ真
6	上記以外	エラー

(d) 使用例

XQuery 変数 book が，次に示す要素を表したノードを示している場合の関数呼出しの指定と，結果の例を次に示します。

＜ XQuery 変数 book が表す要素＞

```
<書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
</書籍情報>
```

＜関数呼出し指定例と結果＞

項番	関数呼出し	結果 (xs:boolean 型)	説明
1	$book/fn:boolean(著者[text() = ”佐藤優”])	偽	パラメタ seq の値が空の XQuery シーケンスになるため，偽を返します。
2	fn:boolean($book)	真	パラメタ seq の値が「書籍情報」要素ノード一つから構成される XQuery シーケンスになるため，真を返します。
3	$book/fn:boolean(@書籍ID = 452469630)	真	パラメタ seq の値が xs:boolean 型の値一つから構成される XQuery シーケンスになるため，真を返します。
4	$book/fn:boolean(タイトル/fn:string(text()))	真	パラメタ seq の値が長さ 1 以上の xs:string 型の値一つから構成される XQuery シーケンスになるため，真を返します。
5	$book/fn:boolean(@書籍ID)	真	パラメタ seq の値が数値データ一つから構成される XQuery シーケンスになるため，真を返します。
6	$book/fn:boolean(著者/fn:string(text()))	エラー	パラメタ seq の値が xs:string 型の値二つから構成される XQuery シーケンスになるため，エラーになります。

(3) ceiling

(a) 機能

引数の値以上の，最小の整数値を返します。

(b) 形式

```
fn:ceiling ( 引数 )
```

(c) 規則

1. この関数は次の表に示すパラメタを持ちます。

表 1-103 fn:ceiling 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	num	数値データ型 (空の XQuery シーケンスを含む)	入力となる数値

2. 結果は，空の XQuery シーケンス，又は次の表に示す数値データ型の値になります。

表 1-104　fn:ceiling 関数の結果の XQuery データ型

項番	パラメタ num の値の XQuery データ型	結果の XQuery データ型
1	xs:double	xs:double
2	xs:decimal	xs:decimal
3	xs:int	xs:int

3. パラメタ num の値が空の XQuery シーケンスの場合，空の XQuery シーケンスを返します。

4. パラメタ num の値が xs:double 型で，結果の値が特殊な値になる場合を次の表に示します。

表 1-105　fn:ceiling 関数の結果が特殊な値になる場合

項番	パラメタ num の値	結果の値
1	正の 0	正の 0
2	負の 0	負の 0
3	-1 より大きいかつ 0 未満	負の 0
4	正の無限大	正の無限大
5	負の無限大	負の無限大
6	NaN（非数）	NaN（非数）

(d)　使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果（括弧内は XQuery データ型）	説明
1	`fn:ceiling(10.5)`	11. (xs:decimal)	10.5 以上の最小の整数値である 11 を返します。
2	`fn:ceiling(-10.5)`	-10. (xs:decimal)	-10.5 以上の最小の整数値である-10 を返します。

(4)　compare

(a)　機能

二つの文字列を比較した結果を返します。

(b)　形式

```
fn:compare ( 引数1 , 引数2 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-106 fn:compare 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	str1	xs:string 型 (空の XQuery シーケンスを含む)	比較する文字列
2	引数 2	str2	xs:string 型 (空の XQuery シーケンスを含む)	比較する文字列

2. 結果は，空の XQuery シーケンス又は xs:int 型の値になります。
3. パラメタ str1 の値，パラメタ str2 の値のどちらかが空の XQuery シーケンスである場合，空の XQuery シーケンスを返します。
4. パラメタ str1 及びパラメタ str2 が xs:string 型の値である場合，パラメタ str1 の値とパラメタ str2 の値の関係により，次に示す結果を返します。

表 1-107 fn:compare 関数の結果

項番	パラメタ str1 と str2 の値の関係	結果(xs:int 型)
1	パラメタ str1 の値がパラメタ str2 の値未満（str1 lt str2 が真になる）	-1
2	パラメタ str1 の値がパラメタ str2 の値と等しい（str1 eq str2 が真になる）	0
3	パラメタ str1 の値がパラメタ str2 の値より大きい（str1 gt str2 が真になる）	1

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果 (xs:int 型)	説明
1	`fn:compare(“abcde”, “abcdef”)`	-1	“abcde”は”abcdef”未満であるため，-1 を返します。
2	`fn:compare(“abcde”, “abcde”)`	0	パラメタ str1 及び str2 の値が両方とも”abcde”であるため，0 を返します。
3	`fn:compare(“abcde”, “abade”)`	1	“abcde”は”abade”より大きいため，1 を返します。

(5) concat

(a) 機能

二つ以上の基本単位値を xs:string 型の値に変換して連結した値を返します。

(b) 形式

```
fn:concat ( 引数 , 引数 〔, 引数〕… )
```

(c) 規則

1. この関数は，次の表に示すパラメタを任意の個数（二つ以上）持ちます。

表 1-108 fn:concat 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	val	任意の基本単位型 (空の XQuery シーケンスを含む)	連結する基本単位値

2. 結果は，空の XQuery シーケンス又は xs:string 型の値になります。
3. すべてのパラメタの値が空の XQuery シーケンスの場合，空の XQuery シーケンスを返します。
4. 基本単位値であるパラメタを含む場合，その値を xs:string 型の値に変換し，指定した順に連結した xs:string 型の値を返します。このとき，空の XQuery シーケンスであるパラメタの値は，長さ 0 の文字列として扱います。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果 (xs:string 型)	説明
1	fn:concat(“abc” ,” defg”)	“abcdefg”	“abc” と” defg” を連結した文字列” abcdefg” を返します。
2	fn:concat(“hijk” , (), “lmn”)	“hijklmn”	空の XQuery シーケンスは長さ 0 の文字列として扱うため,” hijk”,””,” lmn” を連結した文字列” hijklmn” を返します。
3	fn:concat((),(),())	空の XQuery シーケンス	すべてのパラメタの値が空の XQuery シーケンスであるため，空の XQuery シーケンスを返します。

(6) contains（XQuery で定義された関数）

(a) 機能

引数 1 で指定した文字列中に引数 2 で指定した文字列が含まれているかどうかを返します。

(b) 形式

```
fn:contains ( 引数1, 引数2 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-109 fn:contains 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	str1	xs:string 型 (空の XQuery シーケンスを含む)	部分文字列を検索する文字列
2	引数 2	str2	xs:string 型 (空の XQuery シーケンスを含む)	パラメタ str1 の文字列中を検索する部分文字列

2. 結果は xs:boolean 型の値になります。
3. パラメタ str1 の値が空の XQuery シーケンス又は長さ 0 の文字列であり，かつパラメタ str2 の値が長さ 1 以上の文字列である場合は偽を返します。
4. パラメタ str2 の値が空の XQuery シーケンス又は長さ 0 の文字列である場合，パラメタ str1 の値に関係なく真を返します。
5. パラメタ str1 の値及びパラメタ str2 の値が，長さ 1 以上の文字列である場合，パラメタ str1 の文字列中にパラメタ str2 の文字列が含まれていれば，真を返します。含まれていなければ偽を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果 (xs:boolean 型)	説明
1	fn:contains(“abcdefg”, “cde”)	真	“abcdefg” に” cde” が含まれているため，真を返します。
2	fn:contains(“abcdefg”, “cdf”)	偽	“abcdefg” に” cdf” は含まれていないため，偽を返します。
3	fn:contains((), “”)	真	パラメタ str2 の値が長さ 0 の文字列であるため，真を返します。
4	fn:contains((), “a”)	偽	パラメタ str1 の値が空の XQuery シーケンスで，パラメタ str2 の値が長さ 1 以上の文字列であるため，偽を返します。
5	fn:contains(“abcdefg”, ())	真	パラメタ str2 の値が空の XQuery シーケンスであるため，真を返します。

(7) contains（HiRDB で定義された関数）

(a) 機能

引数 1 で指定したノードの文字列値が，引数 2 で指定した全文検索のテキスト検索条件を満たすかどうかを返します。

(b) 形式

```
hi-fn:contains ( 引数1, 引数2 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-110 hi-fn:contains 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	seq	0 個以上のノードを持つ XQuery シーケンス	検索対象のノード
2	引数 2	str	xs:string 型	全文検索のテキスト検索条件の文字列

2. 結果は xs:boolean 型の値になります。
3. この関数は XMLEXISTS 述語中でだけ使用できます。
4. この関数を使用する場合は，XML 問合せ文脈項目に指定した列に XML 型全文検索インデクスが定義されている必要があります。また，使用している HiRDB XML Extension のバージョンが 08-04 以降である必要があります。
5. パラメタ seq には，XML 型全文検索用インデクスの使用条件に合致する部分構造パスが指定されている必要があります。合致しない場合，又はインデクスだけで条件評価できない場合は，エラーになります。XML 型全文検索用インデクスの使用条件については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。
6. パラメタ str には全文検索のテキスト検索条件を XQuery 文字列定数で指定します。テキスト検索条件には，次に示す検索条件が指定できます。なお，パラメタ str の値が長さ 0 の文字列，又はテキスト検索条件の形式に従っていない場合はエラーになります。テキスト検索条件の詳細な指定方法については，マニュアル「HiRDB XML Extension」を参照してください。

表 1-111 テキスト検索条件に指定できる検索条件

項番	検索条件	概要	指定例
1	単純文字列条件	次に示すワイルドカード，又は特殊文字を含む検索文字列に一致する文字列を検索します。 • *	• 「ネットワーク」と「コンピュータ」の間に 0 文字以上の任意の文字列を含む文字列 "ネットワーク*コンピュータ"

項番	検索条件	概要	指定例
		0 文字以上の任意の文字列に相当します。 • ? 任意の 1 文字に相当します。 • | 構造の先頭又は末尾に一致します。 • ¥ ワイルドカード又は特殊文字の意味を消します。 検索したい文字列を引用符（"）で囲んで指定します。ただし，XQuery 文字列定数を引用符（"）で囲んでいる場合は，引用符を二つ続けて記述するか，「"」で囲んでください。	• 「ネットワーク」と「コンピュータ」の間に任意の 1 文字を含む文字列 "ネットワーク?コンピュータ" • 「ネットワーク」で始まる文字列 "|ネットワーク"
2	除外文字検索条件	単純文字列条件から，検索文字列の途中又は前後に特定の文字がある場合を除外して検索します。除外する文字の前に「^」を指定します。	• 「ローマ」の後に任意の 1 文字を含む文字列から，「ローマ字」を含む文字列を除外 "ローマ^字" • 「文書」と「管理」の間に任意の 1 文字を含む文字列から「文書の管理」を含む文字列を除外 "文書^の管理" • 「文書」の前に任意の 1 文字を含む文字列から「誤文書」を含む文字列を除外 "^誤文書"
3	NOT 条件	検索文字列を含まない文字列を検索できます。	• 「ネットワーク」を含まない文字列 NOT ("ネットワーク")
4	AND/OR 条件	検索条件を AND 又は OR で連結し，複数個指定できます。	• 「ネットワーク」と「コンピュータ」の両方を含む文字列 "ネットワーク" AND "コンピュータ" • 「ネットワーク」と「コンピュータ」のどちらかを含む文字列 "ネットワーク" OR "コンピュータ"
5	近傍条件	二つの検索条件に一致する文字列間の文字数（距離）の条件を指定します。	• 「最新」と「技術」（順不同）の間の文字数が 20 文字以下である文字列 PROXIMITY("最新", <=20※1, CHARACTERS※2,ANY_ORDER※3,"技術※4")
6	同義語展開条件	同義語辞書の定義に基づいて，検索文字列の同義語が自動的に展開されます。同義語辞書の作成及び登録方法については，マニュアル「HiRDB XML Extension」を参照してください。	• 「COMPUTER」の同義語（アルファベット異表記及び全角半角異表記を含む）を含む文字列 SYNONYM(USR01※4, "COMPUTER", "AE※5")

項番	検索条件	概要	指定例
7	異表記展開条件	ルールに基づいて，検索文字列の異表記が自動的に展開されます。	• 「COMPUTER」のアルファベット異表記，及び全角半角異表記した文字列を含む文字列 SOUNDEX_EXP("COMPUTER", "AE※5") • 「sing」の英単語派生表記した文字列を含む文字列 SOUNDEX_EXP("sing", "S※6")

注※1

距離が 20 以下で検索することを意味します。

注※2

距離単位を文字とすることを意味します。

注※3

順不同を意味します。

注※4

同義語辞書名を意味します。

注※5

"A"がアルファベット異表記，"E"が全角半角異表記を意味します。

注※6

英単語派生表記を意味します。

7. パラメタ seq の値が空の XQuery シーケンスである場合は偽を返します。
8. パラメタ seq の XQuery シーケンスが持つノードの文字列値のうち，一つ以上の文字列値が，パラメタ str のテキスト検索条件が表す文字列を含んでいる場合，真を返します。テキスト検索条件が表す文字列を含んでいる文字列値が一つもない場合は，偽を返します。

(d) 使用例

次に示す XML 文書を表す値を格納している列（XML 型全文検索インデクスを定義している）を XML 問合せ文脈項目に指定している場合の関数呼出しの指定と，結果の例を次に示します。

＜ XML 問合せ文脈項目に指定した列に格納されている値が表す XML 文書＞

```
<書籍情報 書籍ID=”452469630”>
  <カテゴリ>データベース</カテゴリ>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
  <説明>リレーショナルモデルの概念から，それに基づくRDBMS（Relational Database  Management System）の仕組みを解説している。</説明>
</書籍情報>
```

＜関数呼出し指定例と結果＞

項番	関数呼出し	結果(xs:boolean型)	説明
1	/書籍情報/タイトル[hi-fn:contains(text(), “SYNONYM(USR01, "DB")”)]	真	同義語辞書 USR01 に"データベース"と"DB"が同義語として登録されている場合です。パラメタ seq の値中に"DB"の同義語である"データベース"を含むノードがあるため，真を返します。
2	/書籍情報/説明[hi-fn:contains(text(), “SOUNDEX_EXP("DATABASE", "AE") AND "解説"”)]	真	パラメタ seq の値中に"DATABASE"のアルファベット異表記である"Database"を含み，かつ"解説"を含むノードがあるため，真を返します。
3	/書籍情報/著者[hi-fn:contains(text(), “"佐藤"”)]	偽	パラメタ seq の値中に"佐藤"を含むノードが一つもないため，偽を返します。
4	/書籍情報/目次[hi-fn:contains(text(), “SYNONYM(USR01, "DB")”)]	偽	パラメタ seq が空の XQuery シーケンスであるため，偽を返します。

(8) count

(a) 機能

XQuery シーケンス中の XQuery 項目数を返します。

(b) 形式

```
fn:count ( 引数 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-112 fn:count 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の任意の XQuery 項目を持つ XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は，xs:int 型の値になります。

(d) 使用例

XQuery 変数 book が次に示す要素を表したノードを示している場合の関数呼出しの指定と，結果の例を次に示します。

＜ XQuery 変数 book が表す要素＞

```
<書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
</書籍情報>
```

＜関数呼出し指定例と結果＞

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:count($book)	1	パラメタ seq の値は「書籍情報」要素ノード一つから構成される XQuery シーケンスになるため，1 を返します。
2	fn:count($book/著者)	2	パラメタ seq の値は「書籍情報」要素ノードの子である「著者」要素ノード二つから構成される XQuery シーケンスになるため，2 を返します。
3	fn:count($book/著者[.=”佐藤優”])	0	パラメタ seq の値は空の XQuery シーケンスになるため，0 を返します。

(9) data

(a) 機能

XQuery シーケンスを基本単位化し，基本単位値から成る XQuery シーケンスを返します。

(b) 形式

```
fn:data ( 引数 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-113 fn:data 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の任意の XQuery 項目を持つ XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は，空の XQuery シーケンス又は基本単位値から成る XQuery シーケンスになります。
3. パラメタ seq の値が空の XQuery シーケンスである場合，空の XQuery シーケンスを返します。
4. パラメタ seq の値が空の XQuery シーケンスでない場合，その XQuery シーケンス中の各 XQuery 項目に，次の規則を適用した結果から構成される XQuery シーケンスを返します。
   - XQuery 項目が基本単位値である場合，その値が結果になります。
   - XQuery 項目がノードであれば，ノード種別に従って，次の表に示す値が結果になります。

表 1-114　XQuery 項目がノードである場合，得られる結果

項番	ノード種別	結果
1	文書ノード	型付き値プロパティの値
2	要素ノード	型付き値プロパティの値
3	属性ノード	型付き値プロパティの値
4	処理命令ノード	内容プロパティの値を xs:string 型に変換した値
5	コメントノード	内容プロパティの値を xs:string 型に変換した値
6	テキストノード	内容プロパティの値を xs:untypedAtomic 型に変換した値

(d)　使用例

XQuery 変数 book が次に示す要素を表したノードを示している場合の関数呼出しの指定と，結果の例を次に示します。

＜ XQuery 変数 book が表す要素＞

```
<書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
</書籍情報>
```

＜関数呼出し指定例と結果＞

項番	関数呼出し	結果	説明
1	fn:data($book/価格)	空の XQuery シーケンス	パラメタ seq の値が空の XQuery シーケンスになるため，空の XQuery シーケンスを返します。
2	fn:data($book/著者)	(”伊藤栄一”,”木村幸一”)	パラメタ seq の値が「著者」要素ノード二つから構成される XQuery シーケンスになるため，それらの型付き値プロパティの値二つから成る XQuery シーケンスを返します。

(10) day-from-date

(a) 機能

日付から日の部分だけを抽出して返します。

(b) 形式

```
fn:day-from-date ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-115 fn:day-from-date 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dat	xs:date 型 (空の XQuery シーケンスを含む)	抽出元の日付

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。
3. パラメタ dat の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。
4. パラメタ dat の値が xs:date 型の値である場合は，その日の部分の値を xs:int 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:day-from-date (xs:date(“2006-09-26”))	26	パラメタ dat の値の日の部分である 26 を返します。

(11) day-from-dateTime

(a) 機能

時刻印から日の部分だけを抽出して返します。

(b) 形式

```
fn:day-from-dateTime ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-116　fn:day-from-dateTime 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dtm	xs:dateTime 型 （空の XQuery シーケンスを含む）	抽出元の時刻印

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。

3. パラメタ dtm の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。

4. パラメタ dtm の値が xs:dateTime 型の値である場合は，その日の部分の値を xs:int 型で返します。

(d)　使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:day-from-dateTime(xs:dateTime(“2006-09-26T18:44:58.153”))	26	パラメタ dtm の値の日の部分である 26 を返します。

(12)　deep-equal

(a)　機能

二つの XQuery シーケンスが深等（同じ構造と同じ値を持っている）かどうかを判別します。

(b)　形式

```
fn:deep-equal ( 引数1, 引数2 )
```

(c)　規則

1. この関数は次に示すパラメタを持ちます。

表 1-117　fn:deep-equal 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	seq1	0 個以上の任意の XQuery 項目を持つ XQuery シーケンス	比較する XQuery シーケンス
2	引数 2	seq2	0 個以上の任意の XQuery 項目を持つ XQuery シーケンス	比較する XQuery シーケンス

2. 結果は xs:boolean 型の値になります。

3. パラメタ seq1 及びパラメタ seq2 の両方が空の XQuery シーケンスである場合，結果は真となります。

4. パラメタ seq1 を構成する XQuery 項目の数とパラメタ seq2 を構成する XQuery 項目の数が異なる場合は偽となります。
5. パラメタ seq1 を構成するすべての XQuery 項目が，パラメタ seq2 を構成する同じ位置の XQuery 項目と深等である場合結果は真となり，そうでない XQuery 項目が一つでもある場合結果は偽となります。

   二つの XQuery 項目が深等であるかどうかは，次に示す規則によって決定します。

   二つの XQuery 項目を item1，item2 とします。

表 1-118　XQuery 項目の深等の規則

項番	item1 の種類	item2 の種類	規則
1	基本単位値	基本単位値	次に示すすべての条件を満たす場合だけ真となります。 1. item1 と item2 が eq 演算子で比較できる XQuery データ型である。 2. (item1 eq item2)が真となる，又は item1 及び item2 の両方が xs:double 型で値が NaN（非数）である。
2		上記以外	偽となります。
3	文書ノード	文書ノード	item1 の子である要素ノード及びテキストノードの XQuery シーケンスと，item2 の子である要素ノード及びテキストノードの XQuery シーケンスに対する fn:deep-equal 関数が真となる場合だけ真となります。
4		上記以外	偽となります。
5	要素ノード	要素ノード	次に示すすべての条件を満たす場合だけ真となります。 1. 二つの要素ノードの名前空間及びノード名が一致する。 2. item1 が持つ属性ノードの数と，item2 が持つ属性ノードの数が等しく，item1 が持つ各属性ノードがすべて item2 の対応する属性ノードと深等である。 このとき，item1 が持つ属性ノードの順序と，item2 が持つ属性ノードの順序が同じである必要はない。 3. item1 の子である要素ノード及びテキストノードの XQuery シーケンスと，item2 の子である要素ノード及びテキストノードの XQuery シーケンスに対する fn:deep-equal 関数が真となる。
6		上記以外	偽となります。
7	属性ノード	属性ノード	次に示すすべての条件を満たす場合だけ真となります。 1. 二つの属性ノードの名前空間及びノード名が一致する。 2. item1 の型付き値と item2 の型付き値が深等である。
8		上記以外	偽となります。
9	処理命令ノード	処理命令ノード	次に示すすべての条件を満たす場合だけ真となります。 1. 二つの処理命令ノードの処理命令ターゲットが一致する。 2. item1 の内容と item2 の内容が深等である。
10		上記以外	偽となります。
11	テキストノード	テキストノード	二つのテキストノードの文字列値が深等である場合だけ真となります。

項番	item1 の種類	item2 の種類	規則
12		上記以外	偽となります。
13	コメントノード	コメントノード	二つのコメントノードの文字列値が深等である場合だけ真となります。
14		上記以外	偽となります。

(d) 使用例

XQuery 変数 books が次に示す要素を表したノードを示している場合の，関数呼出しの指定と，結果の例を次に示します。

＜ XQuery 変数 books が表す要素＞

```
<書籍一覧>
  <書籍情報 書籍ID=”452469630”>
    <タイトル>リレーショナルデータベース解説</タイトル>
    <著者>木村幸一</著者>
    <著者>伊藤栄一</著者>
  </書籍情報>
  <書籍情報 書籍ID=”45241350”>
    <タイトル>続・リレーショナルデータベース解説</タイトル>
    <著者>木村幸一</著者>
    <著者>伊藤栄一</著者>
  </書籍情報>
</書籍一覧>
```

＜関数呼出し指定例と結果＞

項番	関数呼出し	結果 (xs:boolean 型)	説明
1	fn:deep-equal($books/書籍情報[1]/タイトル, “リレーショナルデータベース解説”)	偽	パラメタ seq1 は要素ノード一つの XQuery シーケンスであり，パラメタ seq2 は基本単位値一つの XQuery シーケンスであるため，偽となります。
2	fn:deep-equal($books/書籍情報[1], $books/書籍情報[2])	偽	パラメタ seq1 とパラメタ seq2 は，両方とも「書籍情報」要素ノード一つの XQuery シーケンスですが，「書籍 ID」属性ノードが深等でないため，又は子要素の「タイトル」要素が深等でないため，偽となります。
3	fn:deep-equal($books/書籍情報[1]/著者, $books/書籍情報[2]/著者)	真	パラメタ seq1 とパラメタ seq2 は，両方とも「著者」要素ノード二つの XQuery シーケンスであり，かつそれぞれの「著者」要素ノードが深等であるため，真となります。

(13) distinct-values

(a) 機能

XQuery シーケンスから重複する基本単位値を取り除いた XQuery シーケンスを返します。

(b) 形式

```
fn:distinct-values ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-119 fn:distinct-values 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の任意の基本単位型の値を持つ XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は，空の XQuery シーケンス又は基本単位値だけから構成される XQuery シーケンスになります。
3. パラメタ seq の値が空の XQuery シーケンスである場合，空の XQuery シーケンスを返します。
4. 基本単位値が重複しているかどうかの判定は，値比較の eq 演算子を使用して行います。eq 演算子で比較できない値同士又は比較結果が偽になる場合は，重複していないとみなします。比較結果が真になる場合は，重複しているとみなし，一方が取り除かれます。
5. パラメタ seq の値である XQuery シーケンスに xs:double 型の NaN（非数）が複数含まれる場合，それらのうち一つ以外は取り除かれます。
6. 結果の XQuery シーケンス内の XQuery 項目の順序は，パラメタ seq の値である XQuery シーケンス内の XQuery 項目の順序を保証しません。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果	説明
1	fn:distinct-values((“a”, 1, “A”, 0, 1))	(“a”, 1, “A”, 0) 順序は異なる場合があります。	パラメタ seq の値である XQuery シーケンス中に 1 が重複しているため，重複を排除します。

(14) ends-with

(a) 機能

引数 1 で指定した文字列が引数 2 で指定した文字列で終了するかどうかを返します。

(b) 形式

```
fn:ends-with ( 引数1, 引数2 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-120 fn:ends-with 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	str1	xs:string 型 (空の XQuery シーケンスを含む)	部分文字列で終了するか検査する文字列
2	引数 2	str2	xs:string 型 (空の XQuery シーケンスを含む)	パラメタ str1 の文字列の最後にあるか検査する部分文字列

2. 結果は xs:boolean 型の値になります。
3. パラメタ str1 の値が空の XQuery シーケンス又は長さ 0 の文字列であり，かつパラメタ str2 の値が長さ 1 以上の文字列である場合は偽を返します。
4. パラメタ str2 の値が空の XQuery シーケンス又は長さ 0 の文字列である場合，パラメタ str1 の値に関係なく真を返します。
5. パラメタ str1 の値及びパラメタ str2 の値が，長さが 1 以上の文字列である場合，パラメタ str1 の文字列がパラメタ str2 の文字列で終了していれば，真を返します。それ以外の場合は偽を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:boolean 型)	説明
1	fn:ends-with(“abcdefg”, “efg”)	真	“abcdefg”は”efg”で終了しているため，真を返します。
2	fn:ends-with(“abcdefg”, “abc”)	偽	“abcdefg”は”abc”で終了していないため，偽を返します。
3	fn:ends-with((), “”)	真	パラメタ str2 の値が長さ 0 の文字列であるため，真を返します。
4	fn:ends-with((), “a”)	偽	パラメタ str1 の値が空の XQuery シーケンスあり，パラメタ str2 の値が長さ 1 以上の文字列であるため，偽を返します。
5	fn:ends-with(“abcdefg”, ())	真	パラメタ str2 の値が空の XQuery シーケンスであるため，真を返します。

(15) false

(a) 機能

偽を返します。

(b) 形式

```
fn:false ( )
```

(c) 規則

1. この関数はパラメタを持ちません。
2. 結果は，xs:boolean 型の値の偽になります。

(16) floor

(a) 機能

引数の値以下の，最大の整数値を返します。

(b) 形式

```
fn:floor ( 引数 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-121 fn:floor 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	num	数値データ型 (空の XQuery シーケンスを含む)	入力となる数値

2. 結果は，空の XQuery シーケンス，又は次の表に示す数値データ型の値になります。

表 1-122 fn:floor 関数の結果の XQuery データ型

項番	パラメタ num の値の XQuery データ型	結果の XQuery データ型
1	xs:double	xs:double
2	xs:decimal	xs:decimal
3	xs:int	xs:int

3. パラメタ num の値が空の XQuery シーケンスである場合，空の XQuery シーケンスを返します。

4. パラメタ num の値が xs:double 型で，結果が特殊な値になる場合を次の表に示します。

表 1-123　fn:floor 関数の結果が特殊な値になる場合

項番	パラメタ num の値	結果の値
1	正の 0	正の 0
2	負の 0	負の 0
3	正の無限大	正の無限大
4	負の無限大	負の無限大
5	NaN（非数）	NaN（非数）

(d)　使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果（括弧内は XQuery データ型）	説明
1	`fn:floor(10.5)`	10. (xs:decimal)	10.5 以下の最大の整数値である 10 を返します。
2	`fn:floor(-10.5)`	-11. (xs:decimal)	-10.5 以下の最大の整数値である-11 を返します。

(17)　hours-from-dateTime

(a)　機能

時刻印から時の部分だけを抽出して返します。

(b)　形式

```
fn:hours-from-dateTime ( 引数 )
```

(c)　規則

1. この関数は次に示すパラメタを持ちます。

表 1-124　fn:hours-from-dateTime 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dtm	xs:dateTime 型 （空の XQuery シーケンスを含む）	抽出元の時刻印

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。

3. パラメタ dtm の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。

4. パラメタ dtm の値が xs:dateTime 型の値である場合は，その時の部分の値を xs:int 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:hours-from-dateTime(xs:dateTime(“2006-09-26T18:44:58.153”))	18	パラメタ dtm の値の時の部分である 18 を返します。

(18) hours-from-time

(a) 機能

時刻から時の部分だけを抽出して返します。

(b) 形式

```
fn:hours-from-time ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-125 fn:hours-from-time 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	tim	xs:time 型 (空の XQuery シーケンスを含む)	抽出元の時刻

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。
3. パラメタ tim の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。
4. パラメタ tim の値が xs:time 型の値である場合は，その時の部分の値を xs:int 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:hours-from-time(xs:time(“18:44:58”))	18	パラメタ dat 値の時の部分である 18 を返します。

(19) index-of

(a) 機能

指定した XQuery シーケンス中に，指定した基本単位値と一致する XQuery 項目が現れる位置を返します。

(b) 形式

```
fn:index-of ( 引数1 , 引数2 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-126 fn:index-of 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	seq	0 個以上の任意の基本単位型の値を持つ XQuery シーケンス	検索対象の XQuery シーケンス
2	引数 2	val	任意の基本単位型	検索する基本単位値

2. 結果は，0 個以上の xs:int 型の値から構成される XQuery シーケンスになります。
3. パラメタ seq が空の XQuery シーケンスの場合，空の XQuery シーケンスを返します。
4. パラメタ val の値と一致する XQuery 項目がパラメタ seq 中に存在する場合，その位置を xs:int 型の値で返します。先頭位置は 1 となります。
5. 基本単位値が一致しているかどうかの判定は，値比較の eq 演算子を使用して行います。eq 演算子で比較できない値同士又は比較結果が偽になる場合は，一致していないとみなします。
6. 該当する XQuery 項目が，パラメタ seq の XQuery シーケンス中に複数存在する場合は，位置を示す xs:int 型の値を，昇順に並べた XQuery シーケンスを返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果	説明
1	`fn:index-of((10, 20, 30, 40), 35)`	空の XQuery シーケンス	パラメタ seq の XQuery シーケンス中に一致する XQuery 項目がないので，空の XQuery シーケンスを返します。
2	`fn:index-of((10, 20, 30, 40), 30)`	(3)	パラメタ seq の XQuery シーケンス中の 3 番目の XQuery 項目が一致するので，xs:int 型の値 3 から成る XQuery シーケンスを返します。

項番	関数呼出し	結果	説明
3	fn:index-of((10, 20, 30, 40,30,50), 30)	(3,5)	パラメタ seq の XQuery シーケンス中の 3 番目と 5 番目の XQuery 項目が一致するので，xs:int 型の値 3 と 5 から成る XQuery シーケンスを返します。

(20) insert-before

(a) 機能

XQuery シーケンスの指定した位置の前に，別の XQuery シーケンス中の XQuery 項目を挿入した結果を返します。

(b) 形式

```
fn:insert-before ( 引数1 , 引数2 , 引数3 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-127 fn:insert-before 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	seq1	0 個以上の任意の XQuery 項目を持つ XQuery シーケンス	挿入される XQuery シーケンス
2	引数 2	pos	xs:int	挿入位置
3	引数 3	seq2	0 個以上の任意の XQuery 項目を持つ XQuery シーケンス	挿入する XQuery 項目が含まれる XQuery シーケンス

2. 結果は 0 個以上の任意の XQuery 項目から構成される XQuery シーケンスになります。
3. パラメタ seq1 の値が空の XQuery シーケンスである場合，パラメタ seq2 の値も空の XQuery シーケンスであれば空の XQuery シーケンスを，そうでなければパラメタ seq2 の値である XQuery シーケンスを返します。
4. パラメタ seq2 の値が空の XQuery シーケンスである場合，パラメタ seq1 の値である XQuery シーケンスを返します。
5. パラメタ pos には挿入位置を指定します。先頭に挿入する場合は 1 を指定します。パラメタ pos の値が 1 より小さい場合は，挿入位置は 1 として扱います。パラメタ pos の値が，パラメタ seq1 の値である XQuery シーケンスの XQuery 項目数より大きい場合は，挿入位置は，(パラメタ seq1 の値である XQuery シーケンスの XQuery 項目数+1)として扱います。

6. 結果の XQuery シーケンスは，パラメタ seq1 の値である XQuery シーケンス中の挿入位置より一つ前までの XQuery 項目，パラメタ seq3 の値である XQuery シーケンス中の XQuery 項目，パラメタ seq1 の値である XQuery シーケンス中の挿入位置以降の XQuery 項目，の順で構成されます。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果	説明
1	fn:insert-before((“a”, “b”, “c”), 0, “z”)	(“z”, “a”, “b”, “c”)	パラメタ pos の値が 0 であるため，先頭に”z”を挿入します。
2	fn:insert-before((“a”, “b”, “c”), 1, “z”)	(“z”, “a”, “b”, “c”)	パラメタ pos の値が 1 であるため，先頭に”z”を挿入します。
3	fn:insert-before((“a”, “b”, “c”), 3, “z”)	(“a”, “b”, “z”, “c”)	パラメタ pos の値が 3 であるため，三つ目の XQuery 項目”c”の前に”z”を挿入します。
4	fn:insert-before((“a”, “b”, “c”), 4, “z”)	(“a”, “b”, “c”, “z”)	パラメタ pos の値が 4 であるため，最後に”z”を挿入します。

(21) last

(a) 機能

文脈サイズを返します。

(b) 形式

```
fn:last ( )
```

(c) 規則

1. この関数はパラメタを持ちません。
2. 結果は，xs:int 型の値になります。
3. この関数の関数呼出しを評価する時点での文脈項目の数を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果	説明
1	(“a”, “b”, “c”) [fn:last()]	“c” (xs:string) ※fn:last 関数の結果は 3	文脈項目の数は 3 であるため，fn:last 関数は 3 を返します。よって，3 番目の文脈項目である”c”が結果になります。

(22) local-name

(a) 機能

ノードの局所名を返します。

(b) 形式

```
fn:local-name ( 〔 引数 〕 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを 0 個又は 1 個持ちます。

表 1-128 fn:local-name 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	node	任意のノード （空の XQuery シーケンスを含む）	局所名を取得するノード

2. 結果は xs:string 型の値になります。
3. 引数を指定しない場合，この関数の関数呼出しを評価する時点での文脈項目のノードの局所名を返します。つまり，”fn:local-name(.)”を指定した場合と同じ結果になります。文脈項目がノードでない場合はエラーになります。
4. パラメタ node の値が空の XQuery シーケンスである場合，長さ 0 の文字列を返します。
5. パラメタ node の値が名前を持たないノード（文書ノード，コメントノード，テキストノード）である場合，長さ 0 の文字列を返します。
6. パラメタ node の値が名前を持つノード（要素ノード，属性ノード，処理命令ノード）である場合，そのノードの局所名を返します。

(d) 使用例

XQuery 変数 book が次に示す要素を表したノードを示している場合の関数呼出しの指定と，結果の例を次に示します。

＜ XQuery 変数 book が表す要素＞

```
<bookinfo:書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
</bookinfo:書籍情報>
```

＜関数呼出し指定例と結果＞

項番	関数呼出し	結果（xs:string 型）	説明
1	$book/fn:local-name()	“書籍情報”	引数を指定していないため，文脈項目である「bookinfo:書籍情報」要素ノードの局所名を返します。
2	$book/fn:local-name(./タイトル)	“タイトル”	パラメタ node の値が「タイトル」要素ノードであるため，その局所名を返します。
3	$book/タイトル/fn:local-name(./text())	“” (長さ 0 の文字列)	パラメタ node の値がテキストノードであるため，長さ 0 の文字列を返します。

(23) max

(a) 機能

XQuery シーケンスを構成する基本単位値の最大値を返します。

(b) 形式

```
fn:max（ 引数 ）
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-129 fn:max 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の任意の基本単位型の値を持つ XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は，空の XQuery シーケンス，又は基本単位型の値になります。
3. パラメタ seq の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。
4. 基本単位値の大小比較は，値比較の ge 演算子を使用して行います。パラメタ seq の値である XQuery シーケンスに ge 演算子で比較できない XQuery データ型が混在している場合，エラーになります。
5. 数値データ型の比較では，すべての値を共通の数値データ型に変換してから比較します。このため，パラメタ seq の XQuery シーケンスが数値データ型の値だけで構成されている場合の結果は次に示す XQuery データ型の値になります。

表 1-130　fn:max 関数の結果の XQuery データ型

項番	パラメタ seq の値である XQuery シーケンス中の値の XQuery データ型			結果の XQuery データ型
	xs:double	xs:decimal	xs:int	
1	○	−	−	xs:double
2	×	○	−	xs:decimal
3	×	×	−	xs:int

（凡例）

○：含みます。

×：含みません。

−：結果のデータ型に影響しません。

6. パラメタ seq の値である XQuery シーケンスが NaN（非数）である xs:double 型の値を含む場合，結果は NaN（非数）になります。

(d)　使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(括弧内は XQuery データ型)	説明
1	`fn:max((3,4.0,5))`	5. (xs:decimal)	3，4.0，5 の中での最大値である 5 を xs:decimal 型の値として返します。
2	`fn:max(())`	空の XQuery シーケンス	パラメタ seq の値が空の XQuery シーケンスであるため，空の XQuery シーケンスを返します。
3	`fn:max((1.0E2, 2.0E-3, 100))`	1.0E2 (xs:double)	1.0E2，2.0E-3，100 の中での最大値である 1.0E2 を xs:double 型の値として返します。

(24)　min

(a)　機能

XQuery シーケンスを構成する基本単位値の最小値を返します。

(b)　形式

```
fn:min ( 引数 )
```

(c)　規則

1. この関数は次に示すパラメタを持ちます。

表 1-131　fn:min 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の任意の基本単位型の値を持つ XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は，空の XQuery シーケンス，又は基本単位型の値になります。

3. パラメタ seq の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。

4. 基本単位値の大小比較は，値比較の le 演算子を使用して行います。パラメタ seq の値である XQuery シーケンスに le 演算子で比較できない XQuery データ型が混在している場合，エラーになります。

5. 数値データ型の比較では，すべての値を共通の数値データ型に変換してから比較します。このため，パラメタ seq の XQuery シーケンスが数値データ型の値だけで構成されている場合の結果は次に示す XQuery データ型の値になります。

表 1-132　fn:min 関数の結果の XQuery データ型

項番	パラメタ seq の値である XQuery シーケンス中の値の XQuery データ型			結果の XQuery データ型
	xs:double	xs:decimal	xs:int	
1	○	−	−	xs:double
2	×	○	−	xs:decimal
3		×	−	xs:int

（凡例）

○：含みます。

×：含みません。

−：結果のデータ型に影響しません。

6. パラメタ seq の値である XQuery シーケンスが NaN（非数）である xs:double 型の値を含む場合，結果は NaN（非数）になります。

(d)　使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(括弧内は XQuery データ型)	説明
1	`fn:min((3,4.0,5))`	3. (xs:decimal)	3，4.0，5 の中での最小値である 3 を xs:decimal 型の値として返します。
2	`fn:min(())`	空の XQuery シーケンス	パラメタ seq の値が空の XQuery シーケンスであるため，空の XQuery シーケンスを返します。

項番	関数呼出し	結果(括弧内は XQuery データ型)	説明
3	fn:min((1.0E2, 2.0E3, 100))	1.0E2 (xs:double)	1.0E2，2.0E3，100 の中での最小値である 1.0E2 を xs:double 型の値として返します。

(25) minutes-from-dateTime

(a) 機能

時刻印から分の部分だけを抽出して返します。

(b) 形式

```
fn:minutes-from-dateTime ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-133 fn:minutes-from-dateTime 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dtm	xs:dateTime 型 (空の XQuery シーケンスを含む)	抽出元の時刻印

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。

3. パラメタ dtm の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。

4. パラメタ dtm の値が xs:dateTime 型の値である場合は，その分の部分の値を xs:int 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:minutes-from-dateTime(xs:dateTime(“2006-09-26T18:44:58.153”))	44	パラメタ dtm の値の分の部分である 44 を返します。

(26) minutes-from-time

(a) 機能

時刻から分の部分だけを抽出して返します。

(b) 形式

```
fn:minutes-from-time ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-134 fn:minutes-from-time 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	tim	xs:time 型 (空の XQuery シーケンスを含む)	抽出元の時刻

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。
3. パラメタ tim の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。
4. パラメタ tim の値が xs:time 型の値である場合は，その分の部分の値を xs:int 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:minutes-from-time(xs:time(“18:44:58”))	44	パラメタ tim の値の分の部分である 44 を返します。

(27) month-from-date

(a) 機能

日付から月の部分だけを抽出して返します。

(b) 形式

```
fn:month-from-date ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-135 fn:month-from-date 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dat	xs:date 型	抽出元の日付

項番	対応する引数	パラメタ	XQuery データ型	説明
			(空の XQuery シーケンスを含む)	

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。

3. パラメタ dat の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。

4. パラメタ dat の値が xs:date 型の値である場合は，その月の部分の値を xs:int 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	`fn:month-from-date(xs:date( “2006-09-26” ))`	9	パラメタ dat の値の月の部分である 9 を返します。

(28) month-from-dateTime

(a) 機能

時刻印から月の部分だけを抽出して返します。

(b) 形式

```
fn:month-from-dateTime ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-136 fn:month-from-dateTime 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dtm	xs:dateTime 型 (空の XQuery シーケンスを含む)	抽出元の時刻印

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。

3. パラメタ dtm の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。

4. パラメタ dtm の値が xs:dateTime 型の値である場合は，その月の部分の値を xs:int 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:month-from-dateTime(xs:dateTime(“2006-09-26T18:44:58.153”))	9	パラメタ dtm の値の月の部分である 9 を返します。

(29) name

(a) 機能

ノードの修飾名を返します。

(b) 形式

```
fn:name（ 〔 引数 〕 ）
```

(c) 規則

1. この関数は，次の表に示すパラメタを 0 個又は 1 個持ちます。

表 1-137 fn: name 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	node	任意のノード (空の XQuery シーケンスを含む)	修飾名を取得するノード

2. 結果は xs:string 型の値になります。
3. 引数を指定しない場合，この関数の関数呼出しを評価する時点での文脈項目のノードの修飾名を返します。つまり，”fn:name(.)”を指定した場合と同じ結果になります。文脈項目がノードでない場合はエラーになります。
4. パラメタ node の値が空の XQuery シーケンスである場合，長さ 0 の文字列を返します。
5. パラメタ node の値が名前を持たないノード（文書ノード，コメントノード，テキストノード）である場合，長さ 0 の文字列を返します。
6. パラメタ node の値が名前を持つノード（要素ノード，属性ノード，処理命令ノード）である場合，そのノードの修飾名を返します。

(d) 使用例

XQuery 変数 book が次に示す要素を表したノードを示している場合の関数呼出しの指定と，結果の例を次に示します。

＜ XQuery 変数 book が表す要素＞

```
<bookinfo:書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
```

```
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
</bookinfo:書籍情報>
```

＜関数呼出し指定例と結果＞

項番	関数呼出し	結果（xs:string 型）	説明
1	$book/fn:name()	“bookinfo:書籍情報”	引数を指定していないため，文脈項目である「bookinfo:書籍情報」要素ノードの修飾名を返します。
2	$book/fn:name(./タイトル)	“タイトル”	パラメタ node の値が「タイトル」要素ノードであるため，その修飾名を返します。
3	$book/タイトル /fn:name(./text())	“” (長さ 0 の文字列)	パラメタ node の値がテキストノードであるため，長さ 0 の文字列を返します。

(30) namespace-uri

(a) 機能

ノードの修飾名の XML 名前空間 URI を返します。

(b) 形式

```
fn:namespace-uri （ 〔 引数 〕 ）
```

(c) 規則

1. この関数は，次の表に示すパラメタを 0 個又は 1 個持ちます。

表 1-138 fn: namespace-uri 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	node	任意のノード (空の XQuery シーケンスを含む)	修飾名の XML 名前空間を取得するノード

2. 結果は xs:string 型の値になります。
3. 引数を指定しない場合，この関数の関数呼出しを評価する時点での文脈項目のノードの修飾名の XML 名前空間を返します。つまり，”fn:namespace-uri(.)”を指定した場合と同じ結果になります。文脈項目がノードでない場合はエラーになります。
4. パラメタ node の値が空の XQuery シーケンスである場合，長さ 0 の文字列を返します。
5. パラメタ node の値が要素ノードでも属性ノードでもないノードの場合，長さ 0 の文字列を返します。
6. パラメタ node の値が，展開後修飾名が XML 名前空間 URI を持たない要素ノード又は属性ノードである場合，長さ 0 の文字列を返します。

7. パラメタ node の値が，展開後修飾名が XML 名前空間 URI を持つ要素ノード又は属性ノードである場合，そのノードの修飾名の XML 名前空間 URI を返します。

(d) 使用例

XQuery 変数 book が次に示す要素を表したノードを示している場合の関数呼出しの指定と，結果の例を次に示します。接頭辞 bookinfo に対応する XML 名前空間 URI は” http://www.hirdb-example.com/bookinfo”，要素名及び属性名に対する既定の XML 名前空間 URI は” http://www.hirdb-example.com/default”とします。

＜ XQuery 変数 book が表す要素＞

```
<bookinfo:書籍情報 書籍ID=”452469630”>
  <タイトル>リレーショナルデータベース解説</タイトル>
  <著者>伊藤栄一</著者>
  <著者>木村幸一</著者>
</bookinfo:書籍情報>
```

＜関数呼出し指定例と結果＞

項番	関数呼出し	結果(xs:string 型)	説明
1	$book/fn:namespace-uri()	” http://www.hirdb-example.com/bookinfo”	引数を指定していないため，文脈項目である「bookinfo:書籍情報」要素ノードの修飾名の XML 名前空間 URI を返します。
2	$book/fn:namespace-uri(./タイトル)	” http://www. hirdb-example.com/default”	パラメタ node の値が「タイトル」要素ノードであるため，その修飾名の XML 名前空間 URI を返します。
3	$book/タイトル /fn:namespace-uri(./text())	“” (長さ 0 の文字列)	パラメタ node の値がテキストノードであるため，長さ 0 の文字列を返します。

(31) normalize-space

(a) 機能

文字列に対してホワイトスペースの正規化を行った結果を返します。

ホワイトスペースの正規化では，文字列中の連続するホワイトスペース（半角空白(X’20’)，タブ(X’09’)，NL(X’0A’)，又は CR(X’0D’)）を一つの半角空白に置き換え，先頭及び末尾のホワイトスペースを除去します。

(b) 形式

```
fn:normalize-space（ 〔 引数 〕 ）
```

(c) 規則

1. この関数は，次の表に示すパラメタを 0 個又は 1 個持ちます。

表 1-139 fn:normalize-space 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	str	xs:string 型 （空の XQuery シーケンスを含む）	入力となる文字列

2. 結果は，xs:string 型の値になります。
3. 引数を指定しない場合，この関数の関数呼出しを評価する時点での文脈項目の値を文字列表現した値に対して，ホワイトスペースの正規化を行った結果を返します。つまり，”fn:normalize-space(fn:string(.))”を指定した場合と同じ結果になります。
4. パラメタ str の値が空の XQuery シーケンスである場合，長さ 0 の文字列を返します。
5. パラメタ str の値が xs:string 型の値である場合，その値に対してホワイトスペースの正規化を行った結果を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果(xs:string 型)	説明
1	fn:normalize-space(())	“” (長さ 0 の文字列)	パラメタ str の値が空の XQuery シーケンスであるため，長さ 0 の文字列を返します。
2	fn:normalize-space(“ a b c d ”)	“a b c d”	“ａｂｃｄ “に対してホワイトスペースの正規化を行った結果を返します。

(32) not

(a) 機能

引数に指定した値を xs:boolean 型の値に変換し，その値を否定した値を返します。

(b) 形式

```
fn:not ( 引数 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-140 fn:not 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の XQuery 項目を持つ XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は xs:boolean 型の値になります。
3. 次に示す手順で，結果を返します。
   - パラメタ seq の値に対して fn:boolean 関数を適用し，ブーリアン値を求めます。
   - 求めたブーリアン値が真の場合，偽を返します。求めたブーリアン値が偽の場合，真を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果 (xs:boolean 型)	説明
1	fn:not(fn:true())	偽	パラメタ seq の値が真であるため，その否定である偽を返します。
2	fn:not(0)	真	0 をブーリアン値として評価すると偽になるため，その否定である真を返します。
3	fn:not(“ABC”)	偽	”ABC”をブーリアン値として評価すると真になるため，その否定である偽を返します。

(33) number

(a) 機能

引数に指定した値を xs:double 型に変換した値を返します。

(b) 形式

```
fn:number ( 〔 引数 〕 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを 0 個又は 1 個持ちます。

表 1-141 fn:number 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	item	任意の XQuery 項目 (空の XQuery シーケンスを含む)	入力となる XQuery 項目

2. 結果は xs:double 型の値になります。
3. 引数を指定しない場合，この関数の関数呼出しを評価する時点での文脈項目を基本単位化してから，xs:double 型に変換した値を返します。つまり，”fn:number(.)”を指定した場合と同じ結果になります。
4. パラメタ item の値が空の XQuery シーケンスの場合，NaN（非数）を返します。
5. パラメタ item の値が XQuery 項目である場合，その XQuery 項目を基本単位化し，その結果である基本単位値を xs:double 型に変換した値を返します。変換できない場合は，NaN（非数）を返します。変換の可否については，「変換可能な XQuery データ型」を参照してください。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果（xs:double 型）	説明
1	`fn:number(())`	NaN(非数)	パラメタ item の値が空の XQuery シーケンスであるため，NaN を返します。
2	`fn:number( “abcde” )`	NaN(非数)	“abcde”は xs:double 型に変換できないため，NaN を返します。
3	`fn:number(fn:false())`	0.0E0	xs:boolean 型の真を xs:double 型に変換すると 0.0E0 になるため，その値を返します。
4	`fn:number( “ 15000000 ” )`	1.5E7	“ 15000000 ”を xs:double 型に変換すると 1.5E7 になるため，その値を返します。
5	`fn:number( “INF” )`	正の無限大	“INF”を xs:double 型に変換すると正の無限大になるため，その値を返します。

(34) position

(a) 機能

文脈位置を返します。

(b) 形式

```
fn:position ( )
```

(c) 規則

1. この関数はパラメタを持ちません。
2. 結果は，xs:int 型の値になります。
3. この関数の関数呼出しを評価する時点での文脈項目の文脈位置を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果	説明
1	(“a”, “b”, “c”) [fn:position() ge 2]	(“b”, “c”) ※文脈項目”a”,”b”,”c”に対する fn:position 関数の結果は，それぞれ 1,2,3 になります。	結果は文脈位置が 2 以上である文脈項目”b”,”c”から成る XQuery シーケンスになります。

(35) remove

(a) 機能

XQuery シーケンスから指定した位置の XQuery 項目を削除した結果を返します。

(b) 形式

```
fn:remove ( 引数1 , 引数2 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-142 fn:remove 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	seq	0 個以上の XQuery 項目を含む XQuery シーケンスを含む	入力となる XQuery シーケンス
2	引数 2	pos	xs:int	削除位置

2. 結果は 0 個以上の任意の XQuery 項目から構成される XQuery シーケンスになります。
3. パラメタ seq の値が空の XQuery シーケンスである場合，空の XQuery シーケンスを返します。
4. パラメタ pos には，削除する XQuery 項目の位置を指定します。パラメタ pos の値が 1 より小さい場合，又はパラメタ seq の値である XQuery シーケンスの XQuery 項目数より大きい場合は，パラメタ seq の値である XQuery シーケンスを返します。

5. 結果の XQuery シーケンスは，パラメタ seq の値である XQuery シーケンス中の削除する XQuery 項目より一つ前までの XQuery 項目，パラメタ seq の値である XQuery シーケンス中の削除する XQuery 項目の次以降の XQuery 項目，の順で構成されます。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果	説明
1	fn:remove((“a”, “b”, “c”), 0)	(“a”, “b”, “c”)	パラメタ pos の値が 1 より小さいため，パラメタ seq の値をそのまま返します。
2	fn:remove((“a”, “b”, “c”), 1)	(“b”, “c”)	一つ目の XQuery 項目を削除した XQuery シーケンスを返します。
3	fn:remove((“a”, “b”, “c”), 4)	(“a”, “b”, “c”)	パラメタ pos の値(4)がパラメタ seq の値である XQuery シーケンスの XQuery 項目数(3)より大きいため，パラメタ seq の値をそのまま返します。

(36) reverse

(a) 機能

XQuery シーケンス中の XQuery 項目を逆順に並べ替えた XQuery シーケンスを返します。

(b) 形式

```
fn:reverse ( 引数 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-143 fn:reverse 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の任意の XQuery 項目から成る XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は，0 個以上の任意の XQuery 項目から成る XQuery シーケンスになります。
3. パラメタ seq が空の XQuery シーケンスの場合，空の XQuery シーケンスを返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果	説明
1	fn:reverse((10, 20, 30, 40))	(40,30,20,10)	パラメタ seq の XQuery シーケンス中の XQuery 項目を逆順に並び替えた XQuery シーケンスを返します。
2	fn:reverse((“abcde”))	(“abcde”)	パラメタ seq の XQuery シーケンス中の XQuery 項目が一つであるため，パラメタ seq の XQuery シーケンスをそのまま返します。
3	fn:reverse(())	空の XQuery シーケンス	パラメタ seq の XQuery シーケンスが空の XQuery シーケンスであるため，空の XQuery シーケンスを返します。

(37) round

(a) 機能

引数の値に最も近い整数値を返します。

(b) 形式

```
fn:round ( 引数 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-144 fn:round 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	num	数値データ型 (空の XQuery シーケンスを含む)	入力となる数値

2. 結果は，空の XQuery シーケンス，又は次の表に示す数値データ型の値になります。

表 1-145 fn:round 関数の結果の XQuery データ型

項番	パラメタ num の値の XQuery データ型	結果の XQuery データ型
1	xs:double	xs:double
2	xs:decimal	xs:decimal
3	xs:int	xs:int

3. パラメタ num の値が空の XQuery シーケンスの場合，空の XQuery シーケンスを返します。

4. 該当する値が二つ存在する場合は，大きい方を返します。

5. パラメタ num の値が xs:double 型で，結果が特殊な値になる場合を次の表に示します。

表 1-146　fn:round 関数の結果が特殊な値になる場合

項番	パラメタ num の値	結果の値
1	正の 0	正の 0
2	負の 0	負の 0
3	-0.5 以上かつ 0 未満	負の 0
4	正の無限大	正の無限大
5	負の無限大	負の無限大
6	NaN（非数）	NaN（非数）

(d)　使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果（括弧内は XQuery データ型）	説明
1	`fn:round(2.5)`	3. (xs:decimal)	2.5 に最も近い整数 2，3 のうち大きい方である 3 を返します。
2	`fn:round(2.4999)`	2. (xs:decimal)	2.4999 に最も近い整数 2 を返します。
3	`fn:round(-2.5)`	-2. (xs:decimal)	-2.5 に最も近い整数-2，-3 のうち大きい方である-2 を返します。
4	`fn:round(-4.5E-1)`	負の 0 (xs:double)	-4.5E-1 は 0 未満かつ-0.5 以上であるため，負の 0 を返します。

(38)　seconds-from-dateTime

(a)　機能

時刻印から秒の部分だけを抽出して返します。

(b)　形式

```
fn:seconds-from-dateTime ( 引数 )
```

(c)　規則

1. この関数は次に示すパラメタを持ちます。

表 1-147　fn:seconds-from-dateTime 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dtm	xs:dateTime 型	抽出元の時刻印

項番	対応する引数	パラメタ	XQuery データ型	説明
			(空の XQuery シーケンスを含む)	

2. 結果は空の XQuery シーケンス又は xs:decimal 型の値になります。
3. パラメタ dtm の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。
4. パラメタ dtm の値が xs:dateTime 型の値である場合は，その小数秒部分を含む秒の部分の値を xs:decimal 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果 (xs:decimal 型)	説明
1	fn:seconds-from-dateTime(xs:dateTime(“2006-09-26T18:44:58.153”))	58.153	パラメタ dtm の値の秒の部分である 58.153 を返します。

(39) seconds-from-time

(a) 機能

時刻から秒の部分だけを抽出して返します。

(b) 形式

```
fn:seconds-from-time ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-148 fn:seconds-from-time 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	tim	xs:time 型 (空の XQuery シーケンスを含む)	抽出元の時刻

2. 結果は空の XQuery シーケンス又は xs:decimal 型の値になります。
3. パラメタ tim の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。
4. パラメタ tim の値が xs:time 型の値である場合は，その秒の部分の値を xs: decimal 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果 (xs:decimal 型)	説明
1	fn:seconds-from-time(xs:time(“18:44:58”))	58	パラメタ tim の値の秒の部分である 58 を返します。

(40) starts-with

(a) 機能

引数 1 で指定した文字列が引数 2 で指定した文字列で開始するかどうかを返します。

(b) 形式

```
fn:starts-with ( 引数1, 引数2 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-149 fn:starts-with 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	str1	xs:string 型 (空の XQuery シーケンスを含む)	部分文字列で開始するか検査する文字列
2	引数 2	str2	xs:string 型 (空の XQuery シーケンスを含む)	パラメタ str1 の文字列の最初にあるか検査する部分文字列

2. 結果は xs:boolean 型の値になります。
3. パラメタ str1 の値が空の XQuery シーケンス又は長さ 0 の文字列であり，かつパラメタ str2 の値が長さ 1 以上の文字列である場合は偽を返します。
4. パラメタ str2 の値が空の XQuery シーケンス又は長さ 0 の文字列である場合，パラメタ str1 の値に関係なく真を返します。
5. パラメタ str1 の値及びパラメタ str2 の値が，長さ 1 以上である xs:string 型の値である場合，パラメタ str1 の文字列がパラメタ str2 の文字列で開始していれば，真を返します。それ以外の場合は偽を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果（xs:boolean 型）	説明
1	fn:starts-with(“abcdefg”, “abc”)	真	“abcdefg”は”abc”で開始しているため，真を返します。
2	fn:starts-with(“abcdefg”, “cde”)	偽	“abcdefg”は”cde”で開始していないため，偽を返します。
3	fn:starts-with((), “”)	真	パラメタ str2 の値が長さ 0 の文字列であるため，真を返します。
4	fn:starts-with((), “a”)	偽	パラメタ str1 の値が空の XQuery シーケンスであり，パラメタ str2 の値が長さ 1 以上の文字列であるため，偽を返します。
5	fn:starts-with(“abcdefg”, ())	真	パラメタ str2 の値が空の XQuery シーケンスであるため，真を返します。

(41) string

(a) 機能

XQuery 項目の値を文字列表現した値を xs:string 型の値として返します。

(b) 形式

```
fn:string ( 〔 引数 〕 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを 0 個又は 1 個持ちます。

表 1-150 fn:string 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	item	任意の XQuery 項目 (空の XQuery シーケンスを含む)	入力となる XQuery 項目

2. 結果は xs:string 型の値になります。
3. 引数を指定しない場合，この関数の関数呼出しを評価する時点での文脈項目の値を文字列表現した値を返します。つまり，”fn:string(.)”を指定した場合と同じ結果になります。
4. パラメタ item の値が空の XQuery シーケンスの場合，長さ 0 の文字列を返します。
5. パラメタ item の値がノードの場合，ノード種別に従って，次の表に示す値を返します。

表 1-151　パラメタ item の値がノードである場合，得られる結果

項番	ノード種別	string 関数が返す値
1	文書ノード	文字列値プロパティの値
2	要素ノード	文字列値プロパティの値
3	属性ノード	文字列値プロパティの値
4	処理命令ノード	内容プロパティの値
5	コメントノード	内容プロパティの値
6	テキストノード	内容プロパティの値

6. パラメタ item の値が基本単位値の場合，基本単位値を xs:string 型に変換した値を返します。xs:string 型への変換規則については，「変換可能な XQuery データ型」を参照してください。

(d)　使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果 (xs:string 型)	説明
1	`fn:string(())`	“” (長さ 0 の文字列)	パラメタ item の値が空の XQuery シーケンスであるため，長さ 0 の文字列を返します。
2	`fn:string( “abcde” )`	“abcde”	文字列“abcde”をそのまま返します。
3	`fn:string(1.234E4)`	“12340”	xs:decimal 型の値 1.234E4 を xs:string 型に変換した値”12340”を返します。
4	`fn:string(0.1234E10)`	“1.234E9”	xs:decimal 型の値 0.1234E10 を xs:string 型に変換した値”1.234E9”を返します。
5	`fn:string(fn:true())`	“true”	xs:boolean 型の値である真を xs:string 型に変換した値”true”を返します。

(42)　string-length

(a)　機能

文字列の長さ（文字数）を返します。

(b)　形式

```
fn:string-length （ 〔 引数 〕 ）
```

(c)　規則

1. この関数は，次の表に示すパラメタを 0 個又は 1 個持ちます。

表 1-152　fn:string-length 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	str	xs:string 型 （空の XQuery シーケンスを含む）	入力となる文字列

2. 結果は，xs:int 型の値になります。
3. 引数を指定しない場合，この関数の関数呼出しを評価する時点での文脈項目の値を文字列表現した値の長さを返します。つまり，”fn:string-length(fn:string(.))”を指定した場合と同じ結果になります。
4. パラメタ str の値が空の XQuery シーケンスである場合，0 を返します。
5. パラメタ str の値が xs:string 型の値である場合，その値中の文字数を返します。

(d)　使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果（xs:int 型）	説明
1	`fn:string-length(())`	0	パラメタ str の値が空の XQuery シーケンスであるため，0 を返します。
2	`fn:string-length(“abcde”)`	5	“abcde”の長さ 5 を返します。
3	`fn:string-length(“あいうえお”)`	5	“あいうえお”の長さ 5 を返します。

(43)　subsequence

(a)　機能

XQuery シーケンスの部分 XQuery シーケンスを返却します。

(b)　形式

```
fn:subsequence（ 引数1, 引数2〔, 引数3〕 ）
```

(c)　規則

1. この関数は，次の表に示すパラメタを持ちます。ただし，パラメタ len は省略できます。

表 1-153　fn:subsequence 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	seq	0 個以上の XQuery 項目から構成される XQuery シーケンス	部分 XQuery シーケンスを取り出す XQuery シーケンス

項番	対応する引数	パラメタ	XQuery データ型	説明
2	引数 2	pos	xs:double 型	部分 XQuery シーケンスの取り出し開始位置
3	引数 3	len	xs:double 型	取り出す XQuery 項目数

2. 結果は 0 個以上の XQuery 項目から構成される XQuery シーケンスになります。

3. パラメタ seq が空の XQuery シーケンスである場合，空の XQuery シーケンスを返します。

4. パラメタ pos には，部分 XQuery シーケンスを取り出す開始位置を指定します。パラメタ pos に対して fn:round 関数を評価した結果の整数値が開始位置になります。先頭位置は 1 になります。

   パラメタ pos に対して fn:round 関数を評価した結果の整数値が正の値でない場合は，先頭位置から取り出します。

   パラメタ pos に対して fn:round 関数を評価した結果の整数値がパラメタ seq の XQuery 項目数より大きい場合，空の XQuery シーケンスを返します。

5. パラメタ len には取り出す部分 XQuery シーケンス中の XQuery 項目の数を指定します。パラメタ len に対して fn:round 関数を評価した結果の整数値が，取り出す部分 XQuery シーケンス中の XQuery 項目の数になります。

   パラメタ len に対して fn:round 関数を評価した結果の整数値が正の値でない場合は，空の XQuery シーケンスを返します。

   パラメタ len に対して fn:round 関数を評価した結果の整数値が，パラメタ seq の XQuery シーケンスでの開始位置から末尾までの数より大きい場合，又はパラメタ len を省略した場合は，パラメタ seq の XQuery シーケンスの末尾まで取り出します。

(d) 使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果	説明
1	fn:subsequence((“a”,”b”,”c”,”d”,”e”,”f”,”g”), 5)	(“e”,”f”,”g”)	パラメタ seq の XQuery シーケンス中の 5 番目から末尾までの XQuery 項目から成る XQuery シーケンスを返します。
2	fn:subsequence((“a”,”b”,”c”,”d”,”e”,”f”,”g”), 2, 2.5)	(“b”,”c”,”d”)	2.5 に対して fn:round 関数を評価した結果が 3 となるため，パラメタ seq の XQuery シーケンス中の 2 番目から三つ分の XQuery 項目から成る XQuery シーケンスを返します。
3	fn:subsequence((), 10)	空の XQuery シーケンス	パラメタ seq が空の XQuery シーケンスであるため，空の

項番	関数呼出し	結果	説明
			XQuery シーケンスを返します。
4	fn:subsequence ((“a”,”b”,”c”,”d”,”e”,”f”,”g”), -10, 10)	(“a”,”b”,”c”,”d”,”e”,”f”,”g”)	パラメタ pos の値が負であるため先頭位置から取り出します。また，パラメタ len の値がパラメタ seq の先頭位置から末尾までの XQuery 項目数より大きいため，末尾まで取り出します。

(44) substring

(a) 機能

文字列の部分文字列を返します。

(b) 形式

```
fn:substring ( 引数1, 引数2〔, 引数3〕 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。ただし，パラメタ len は省略できます。

表 1-154 fn:substring 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	str	xs:string 型 （空の XQuery シーケンスを含む）	部分文字列を取り出す文字列
2	引数 2	pos	xs:double 型	部分文字列の取り出し開始位置
3	引数 3	len	xs:double 型	取り出す部分文字列の長さ（文字数）

2. 結果は xs:string 型の値になります。
3. パラメタ str の値が空の XQuery シーケンスである場合，長さ 0 の文字列を返します。
4. パラメタ str の値が長さ 0 の xs:string 型の値である場合，長さ 0 の文字列を返します。
5. パラメタ pos には，部分文字列を取り出す開始位置を指定します。パラメタ pos に対して fn:round 関数を評価した結果の整数値が開始位置になります。先頭位置は 1 になります。

   パラメタ pos に対して fn:round 関数を評価した結果の整数値が正の値でない場合は，先頭位置から取り出します。

   パラメタ pos に対して fn:round 関数を評価した結果の整数値がパラメタ str の値の長さより大きい場合，長さ 0 の文字列を返します。

6. パラメタ len には取り出す部分文字列の長さ（文字数）を指定します。パラメタ len に対して fn:round 関数を評価した結果の整数値が，取り出す部分文字列の長さになります。

   パラメタ len に対して fn:round 関数を評価した結果の整数値が正の値でない場合は，長さ 0 の文字列を返します。

   パラメタ len に対して fn:round 関数を評価した結果の整数値が，パラメタ str の値での開始位置から末尾までの長さより大きい場合，又はパラメタ len を省略した場合は，パラメタ str の値である文字列の末尾まで取り出します。

(d) 使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果（xs:string 型）	説明
1	fn:substring（“abcdefg”, 5）	“efg”	“abcdefg”の 5 文字目から末尾までの部分文字列を返します。
2	fn:substring(“abcdefg”, 2, 2.5)	“bcd”	2.5 に対して fn:round 関数を評価した結果が 3 となるため，“abcdefg”の 2 文字目から 3 文字分の部分文字列を返します。
3	fn:substring((), 10)	“” (長さ 0 の文字列)	パラメタ str の値が空の XQuery シーケンスであるため，長さ 0 の文字列を返します。
4	fn:substring（“abcdefg”,-10, 10）	“abcdefg”	パラメタ pos の値が負であるため先頭位置から取り出します。また，パラメタ len の値が”abcdefg”の先頭位置から末尾までの長さより大きいため，末尾まで取り出します。

(45) substring-after

(a) 機能

引数 1 で指定した文字列中の，引数 2 で指定した文字列が最初に出現する位置より後の部分文字列を返します。

(b) 形式

```
fn:substring-after（ 引数1, 引数2 ）
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-155　fn:substring-after 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	str1	xs:string 型 (空の XQuery シーケンスを含む)	部分文字列を検索する文字列
2	引数 2	str2	xs:string 型 (空の XQuery シーケンスを含む)	パラメタ str1 の文字列中を検索する部分文字列

2. 結果は xs:string 型の値になります。
3. パラメタ str1 の値が空の XQuery シーケンス又は長さ 0 の文字列である場合，パラメタ str2 の値に関係なく長さ 0 の文字列を返します。
4. パラメタ str1 の値が長さ 1 以上の文字列であり，かつパラメタ str2 の値が空の XQuery シーケンス又は長さ 0 の文字列である場合，パラメタ str1 の値を返します。
5. パラメタ str1 の値及びパラメタ str2 の値が，長さ 1 以上である文字列である場合，パラメタ str1 の文字列中にパラメタ str2 の文字列が含まれていれば，その最初の出現位置より後の部分文字列を返します。それ以外の場合は長さ 0 の文字列を返します。

(d)　使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果（xs:string 型）	説明
1	fn:substring-after(“abcdefg”, “efg”)	“” (長さ 0 の文字列)	“abcdefg”中の”efg”が出現する位置より後の部分文字列を返します。
2	fn:substring-after(“abcdefgcde”, “cde”)	“fgcde”	“abcdefgcde”中の”cde”が最初に出現する位置より後の部分文字列を返します。
3	fn:substring-after((), “abc”)	“” (長さ 0 の文字列)	パラメタ str1 の値が空の XQuery シーケンスになるため，長さ 0 の文字列を返します。
4	fn:substring-after(“abcdefg”, ())	“abcdefg”	パラメタ str2 の値が空の XQuery シーケンスになるため，”abcdefg”をそのまま返します。

(46)　substring-before

(a)　機能

引数 1 で指定した文字列中の，引数 2 で指定した文字列が最初に出現する位置より前の部分文字列を返します。

(b) 形式

```
fn:substring-before ( 引数1, 引数2 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-156 fn:substring-before 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	str1	xs:string 型 (空の XQuery シーケンスを含む)	部分文字列を検索する文字列
2	引数 2	str2	xs:string 型 (空の XQuery シーケンスを含む)	パラメタ str1 の文字列中を検索する部分文字列

2. 結果は xs:string 型の値になります。
3. パラメタ str1 の値が空の XQuery シーケンス又は長さ 0 の文字列である場合，パラメタ str2 の値に関係なく長さ 0 の文字列を返します。
4. パラメタ str1 の値が長さ 1 以上の文字列であり，かつパラメタ str2 の値が空の XQuery シーケンス又は長さ 0 の文字列である場合，パラメタ str1 の値を返します。
5. パラメタ str1 の値及びパラメタ str2 の値が，長さ 1 以上である文字列である場合，パラメタ str1 の文字列中にパラメタ str2 の文字列が含まれていれば，その最初の出現位置より前の部分文字列を返します。それ以外の場合は長さ 0 の文字列を返します。

(d) 使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果（xs:string 型）	説明
1	fn:substring-before(“abcdefg”, “abc”)	“” (長さ 0 の文字列)	“abcdefg”中の”abc”が出現する位置より前の部分文字列を返します。
2	fn:substring-before(“abcdefgdef”, “def”)	“abc”	“abcdefgdef”中の”def”が最初に出現する位置より前の部分文字列を返します。
3	fn:substring-before((), “abc”)	“” (長さ 0 の文字列)	パラメタ str1 の値が空の XQuery シーケンスになるため，長さ 0 の文字列を返します。
4	fn:substring-before(“abcdefg”, ())	“abcdefg”	パラメタ str2 の値が空の XQuery シーケンスになるため,”abcdefg”をそのまま返します。

(47) sum

(a) 機能

XQuery シーケンスを構成する基本単位値の合計を返します。

(b) 形式

```
fn:sum ( 引数 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-157 fn:sum 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	seq	0 個以上の数値データ型の値を持つ XQuery シーケンス	入力となる XQuery シーケンス

2. 結果は空の XQuery シーケンス又は数値データ型の値になります。
3. パラメタ seq の値が空の XQuery シーケンスである場合，結果は xs:int 型の 0 を返します。
4. パラメタ seq の値である，XQuery シーケンスごとの結果の XQuery データ型を，次の表に示します。

表 1-158 fn:sum 関数の結果の XQuery データ型

項番	パラメタ seq の値である XQuery シーケンス中の値の XQuery データ型			結果の XQuery データ型
	xs:double	xs:decimal	xs:int	
1	×	×	×	xs:int
2	○	−	−	xs:double
3	×	○	−	xs:decimal
4		×	○	xs:int

（凡例）

○：含みます。

×：含みません。

−：結果のデータ型に影響しません。

5. パラメタ seq の値である XQuery シーケンスが NaN（非数）を含む場合，結果は NaN（非数）になります。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果（括弧内は XQuery データ型）	説明
1	fn:sum((3,4,5))	12 (xs:int)	3，4，5 の合計値 12 を返します。
2	fn:sum(())	0 (xs:int)	パラメタ seq が空の XQuery シーケンスであるため，0 を返します。
3	fn:sum((1.0E2, 2.0E3))	2.1E3(xs:double)	1.0E2，2.0E3 の合計値 2.1E3 を返します。

(48) translate

(a) 機能

引数 1 で指定した文字列中の引数 2 で指定した文字を，引数 3 で指定した文字に置換した結果を返します。

(b) 形式

```
fn:translate ( 引数1, 引数2, 引数3 )
```

(c) 規則

1. この関数は，次の表に示すパラメタを持ちます。

表 1-159 fn:translate 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数 1	str	xs:string 型 （空の XQuery シーケンスを含む）	入力となる文字列
2	引数 2	from	xs:string 型	置換対象の文字から成る文字列
3	引数 3	to	xs:string 型	置換文字から成る文字列

2. 結果は xs:string 型の値になります。
3. パラメタ str の値が空の XQuery シーケンスである場合，長さ 0 の文字列を返します。
4. パラメタ str の値が xs:string 型の値である場合，その値中の文字について，パラメタ from の値である文字列に存在する置換対象の文字は，パラメタ to の値である文字列中の対応する位置の置換文字で置換し，その結果を返します。置換対象の文字がパラメタ from の値である文字列中の N 文字目であった場合，パラメタ to の値である文字列中の N 文字目の文字が置換文字になります。置換対象の文字がパラメタ from の値である文字列中の M 文字目であり，パラメタ to の値である文字列の長さが M 未満であった場合，置換対象の文字を除去します。
5. パラメタ from の値である文字列中に同じ文字が複数存在する場合，最初の位置に対応した置換文字で置換します。

(d) 使用例

関数呼出しの指定と，結果の例を次の表に示します。

項番	関数呼出し	結果（xs:string 型）	説明
1	fn:translate(“abcdef”, “ace”, “ACE”)	“AbCdEf”	“abcdef”中の”a”,”c”,”e”をそれぞれ”A”,”C”,”E”に置換した文字列を返します。
2	fn:translate(“abcdefg”, “aceg”, “ACE”)	“AbCdEf”	“abcdef”中の”a”,”c”,”e”をそれぞれ”A”,”C”,”E”に置換し,”g”を除去した文字列を返します。
3	fn:translate(“abcdef”, “acea”, “ACEB”)	“AbCdEf”	“abcdef”中の”a”,”c”,”e”をそれぞれ”A”,”C”,”E”に置換した文字列を返します。

(49) true

(a) 機能

真を返します。

(b) 形式

```
fn:true ( )
```

(c) 規則

1. この関数はパラメタを持ちません。
2. 結果は，xs:boolean 型の値の真になります。

(50) year-from-date

(a) 機能

日付から年の部分だけを抽出して返します。

(b) 形式

```
fn:year-from-date ( 引数 )
```

(c) 規則

1. この関数は次に示すパラメタを持ちます。

表 1-160　fn:year-from-date 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dat	xs:date 型 (空の XQuery シーケンスを含む)	抽出元の日付

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。

3. パラメタ dat の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。

4. パラメタ dat の値が xs:date 型の値である場合は，その年の部分の値を xs:int 型で返します。

(d)　使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:year-from-date(xs:date(“2006-09-26”))	2006	パラメタ dat の値の年の部分である 2006 を返します。

(51)　year-from-dateTime

(a)　機能

時刻印から年の部分だけを抽出して返します。

(b)　形式

```
fn:year-from-dateTime ( 引数 )
```

(c)　規則

1. この関数は次に示すパラメタを持ちます。

表 1-161　fn:year-from-dateTime 関数のパラメタ

項番	対応する引数	パラメタ	XQuery データ型	説明
1	引数	dtm	xs:dateTime 型 (空の XQuery シーケンスを含む)	抽出元の時刻印

2. 結果は空の XQuery シーケンス又は xs:int 型の値になります。

3. パラメタ dtm の値が空の XQuery シーケンスである場合は，空の XQuery シーケンスを返します。

4. パラメタ dtm の値が xs:dateTime 型の値である場合は，その年の部分の値を xs:int 型で返します。

(d) 使用例

関数呼出しの指定と，結果の例を次に示します。

項番	関数呼出し	結果(xs:int 型)	説明
1	fn:year-from-dateTime(xs:dateTime(“2006-09-26T18:44:58.153”))	2006	パラメタ dtm の値の年の部分である 2006 を返します。

2 構成要素の詳細

この章では，SQL を使用する場合の構成要素の詳細について説明します。

2.1　カーソル指定

2.1.1　カーソル指定　形式 1

(1)　機能

一つ以上の表からのデータの検索，検索結果の並べ替え（ソート）をするために指定します。

カーソル指定はカーソル宣言中，及び動的 SELECT 文中に指定します。

動的 SELECT 文については，「動的 SELECT 文　形式 1（動的検索）」を参照してください。

(2)　使用権限

カーソル指定に含まれるすべての問合せ指定，又は副問合せを使用できるユーザは，カーソル指定を使用できます。

問合せ指定の権限については「問合せ指定」，副問合せの権限については「副問合せ」を参照してください。

(3)　形式

形式 1 ＜一つ以上の表の検索＞

```
問合せ式〔ORDER BY｛列指定｜ソート項目指定番号｝〔 {ASC｜DESC} 〕
          〔, ｛列指定｜ソート項目指定番号｝〔 {ASC｜DESC} 〕〕… 〕
        〔LIMIT ｛〔オフセット行数,〕｛リミット行数｜ALL｝
                ｜ ｛リミット行数｜ALL｝〔OFFSET オフセット行数〕｝〕
```

(4)　オペランド

- 問合せ式

問合せ指定，又は問合せ指定の結果得られる導出表同士の和集合，又は差集合を求めるために指定します。

問合せ式については，「問合せ式」を参照してください。

- ORDER BY ｛列指定｜ソート項目指定番号｝

問合せ式での検索結果を，昇順，又は降順に並べ替える場合のソートの方法を指定します。

ORDER BY 句を省略した場合，導出表中の行の並びは保証しません。

ORDER BY 句についての規則を次に示します。

1. ソートのキーに指定できる列の最大数は 255 個です。

2. 同じ列は 2 回以上指定できません。
3. 最も外側の問合せの選択式に AS 列名を指定している場合，問合せ式で導出される表に重複する列名があるときは，その列名をソートのキーに指定できません。
4. ソートのキーになる項目で，次の項目は指定できません。
   - 繰返し列，及び繰返し列を含む値式
   - 予備列，及び予備列を含む値式
   - BLOB
   - 定義長が 32,001 バイト以上の BINARY
   - BOOLEAN
   - 抽象データ型
   - WRITE 指定
   - GET_JAVA_STORED_ROUTINE_SOURCE 指定
5. ORDER BY 句を指定した場合，選択式には，次に示すデータ型の SQL 変数，SQL パラメタ，及び定数は単独で指定できません。
   - BLOB
   - 定義長が 32,001 バイト以上の BINARY
   - 抽象データ型

列指定

ソートのキーにする列を指定します。

列指定についての規則を次に示します。

1. 問合せ式に UNION〔ALL〕，又は EXCEPT〔ALL〕を含む場合，列指定はできません。ただし，問合せ式で導出される表に重複のない列名がある場合は，その列名を指定できます。
2. 最も外側の問合せに SELECT DISTINCT を指定している場合，ORDER BY 句に指定する列は，検索結果として出力する項目（選択式で指定されている項目）でなければなりません。
3. 最も外側の問合せ指定に SELECT DISTINCT の指定がなく，GROUP BY 句又は集合関数を指定している場合は，ORDER BY 句に指定する列は GROUP 化列でなければなりません。そのほかの場合には，最も外側の問合せ指定で指定している表の，すべての列を指定できます。

ソート項目指定番号

UNION〔ALL〕，又は EXCEPT〔ALL〕を指定した場合は，問合せ式の結果得られた導出表の列のうち，ソートのキーにしたい列が導出表中で何番目にあるのか，その位置を示す番号を指定します。

UNION〔ALL〕，又は EXCEPT〔ALL〕の指定がなく，集合関数，ウィンドウ関数，定数，四則演算，日付演算，スカラ関数，CASE 式，CAST 指定，関数呼出し，コンポネント指定，及び連結演算の結果得られた導出表中の列をソートのキーにする場合は，ソートのキーにしたい列が導出表の中で何番目に指定されているのか，その位置を表す番号を指定します。

選択式に ROW を指定した問合せ式の ORDER BY 句にはソート項目の指定番号は指定できません。

- {ASC ｜ DESC}

ASC

検索結果を昇順に並べ替える場合，指定します。

DESC

検索結果を降順に並べ替える場合，指定します。

- LIMIT {〔オフセット行数,〕{リミット行数｜ ALL} ｜ {リミット行数｜ ALL} 〔OFFSET オフセット行数〕}

問合せ式での検索結果のうち，先頭から読み飛ばす行数，及び取得する行数を指定します。LIMIT 句を指定すると，SQL の検索性能の向上が期待できます。どのような場合に指定するかについては，マニュアル「HiRDB UAP 開発ガイド」の先頭から n 行の検索結果を取得する機能を参照してください。

オフセット行数

オフセット行数は，問合せ式の結果のうち先頭から読み飛ばす行数を指定します。

リミット行数

オフセット行数の指定がない場合，問合せ式の結果から取得する行数を指定します。

オフセット行数の指定がある場合，問合せ式の結果のうち先頭からオフセット行数を読み飛ばしたところから取得する行数を指定します。

ALL

オフセット行数の指定がない場合，問合せ式の結果のすべてを取得します。

オフセット行数の指定がある場合，問合せ式の結果のうち先頭からオフセット行数を読み飛ばした残りのすべてを取得します。

LIMIT 句についての規則を次に示します。

1. オフセット行数，及びリミット行数には，整数定数，埋込み変数，?パラメタ，SQL 変数，及び SQL パラメタを指定できます。
2. オフセット行数，及びリミット行数のデータ型は整数（SMALLINT 又は INTEGER）にしてください。
3. オフセット行数，及びリミット行数にナル値は指定できません。
4. オフセット行数，及びリミット行数に?パラメタ，又は埋込み変数を指定した場合，?パラメタ，又は埋込み変数のデータ型は，INTEGER 型（標識変数なし）が仮定されます。
5. オフセット行数に指定できる値の範囲は 0～2,147,483,647 です。
6. オフセット行数に 0 を指定した場合は，オフセット行数に指定がない場合と同じになります。
7. リミット行数に指定できる値の範囲は－1～2,147,483,647 です。
8. リミット行数に－1 を指定する場合は，定数で指定してください。
9. オフセット行数とリミット行数に指定できる値の合計は，2,147,483,647 までです。

10. リミット行数≧0 の場合，検索結果行数は次のようになります。

    Max（Min（問合せ式の結果行数－オフセット行数，リミット行数），0）

11. リミット行数＝－1，又は ALL を指定した場合，検索結果行数は次のようになります。

    Max（問合せ式の結果行数－オフセット行数，0）

12. オフセット行数を指定した場合，読み飛ばした最終行と同じソートのキーの値を持つ行が複数存在するか，又はソートのキーの指定がないと，検索結果は一意に決まりません。

13. 問合せ式の結果がリミット行数を超える場合，リミット行数の最終行と同じソートのキーの値を持つ行が複数存在するか，又はソートのキーの指定がないと検索結果は一意に決まりません。検索結果が一意に決まらない場合の例を次に示します。

    (例)

    次の SQL で宣言したカーソルを使用して表（ZAIKO）を検索した場合，ソートのキーとなる列（TANKA）に，３番目に小さい値である'3640'を持つ行が複数あるため，検索結果が結果 1 となるか，結果 2 となるかは不定となります。

```
DECLARE CR1 CURSOR FOR
  SELECT * FROM ZAIKO ORDER BY TANKA LIMIT 3
```

表名：ZAIKO

SCODE	SNAME	TANKA	ZSURYO
101M	ブラウス	3500	85
201M	ポロシャツ	3640	29
302S	スカート	3640	65
591S	ソックス	250	280

●結果1

SCODE	SNAME	TANKA	ZSURYO
591S	ソックス	250	280
101M	ブラウス	3500	85
201M	ポロシャツ	3640	29

●結果2

SCODE	SNAME	TANKA	ZSURYO
591S	ソックス	250	280
101M	ブラウス	3500	85
302S	スカート	3640	65

(5) 留意事項

1. ORDER BY を指定すると，HiRDB が作業表を作成することがあります。このとき，作業表の行長によっては，ORDER BY の処理が制限されることがあります。作業表の行長については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(6) 指定例

1. カーソル宣言中でカーソル指定を指定します。

```
DECLARE CR1 CURSOR FOR
  SELECT SNAME,SUM(ZSURYO)
```

```
    FROM ZAIKO
    GROUP BY SNAME
    ORDER BY SNAME ASC
```

2. カーソル宣言中のカーソル指定で LIMIT 句を指定します。

```
DECLARE CR1 CURSOR FOR
  SELECT SCODE,ZSURYO
    FROM ZAIKO
    WHERE ZSURYO>20
    ORDER BY 2,1
    LIMIT 10
```

2.1.2 カーソル指定　形式 2

(1) 機能

リストを介して表を検索するときに指定します。

リストを介した表の検索に対するカーソル指定は，動的 SELECT 文中に指定します。

動的 SELECT 文については，操作系 SQL の「動的 SELECT 文　形式 1（動的検索）」を参照してください。

(2) 使用権限

実表に対する SELECT 権限を持つユーザが，リストを介してその実表を検索するカーソル指定　形式 2 を実行できます。

(3) 形式

```
SELECT  { {値式 | WRITE指定 | GET_JAVA_STORED_ROUTINE_SOURCE指定}
                  〔〔AS〕 列名〕
          〔,  {値式 | WRITE指定 | GET_JAVA_STORED_ROUTINE_SOURCE指定}
                    〔〔AS〕 列名〕〕…
          | *}
    FROM LIST リスト名
    〔ORDER BY {列名 | ソート項目指定番号} 〔 {ASC | DESC} 〕
                  〔, {列名 | ソート項目指定番号} 〔 {ASC | DESC} 〕〕… 〕
    〔LIMIT  {〔オフセット行数,〕 {リミット行数 | ALL}
           |  {リミット行数 | ALL}  〔OFFSET オフセット行数〕} 〕
```

(4) オペランド

- {{値式 | WRITE 指定 | GET_JAVA_STORED_ROUTINE_SOURCE 指定} 〔〔AS〕 列名〕
  〔, {値式 | WRITE 指定 | GET_JAVA_STORED_ROUTINE_SOURCE 指定} 〔〔AS〕 列名〕〕…
  | *}

値式

検索する値式を指定します。

SELECT 句の値式についての規則を次に示します。

1. 列名には検索するリストの基表の列名を指定します。
2. 繰返し列には検索するリストの基表の繰返し列を指定します。
3. 繰返し列を単独で指定する場合，添字に ANY は指定できません。
4. 属性名には検索するリストの基表の抽象データ型の属性を指定します。
5. SELECT 句の値式中に次のものは指定できません。
   - リストの基表の予備列
   - 基表の列指定中の表指定
   - 基表（の列）に対する外への参照
   - 集合関数
   - ウィンドウ関数
   - 副問合せ
6. プラグイン提供関数の受渡し値送信関数の指定なしで，一部の受渡し値受信関数を指定できます。指定できる受渡し値受信関数については，マニュアル「HiRDB UAP 開発ガイド」，及び各種プラグインマニュアルを参照してください。
7. プラグイン提供関数の受渡し値受信関数は二つ以上指定できません。

WRITE 指定

BLOB データの検索結果をファイル出力する場合に指定します。WRITE 指定については，「WRITE 指定」を参照してください。

GET_JAVA_STORED_ROUTINE_SOURCE 指定

JAR ファイルから Java クラスのソースファイルを抽出する場合に指定します。
GET_JAVA_STORED_ROUTINE_SOURCE 指定については，
「GET_JAVA_STORED_ROUTINE_SOURCE 指定」を参照してください。

〔AS〕 列名

値式，WRITE 指定，又は **GET_JAVA_STORED_ROUTINE_SOURCE 指定**に対して名称を付ける場合に指定します。

＊

リストの基表のすべての列を検索します。

- リスト名

検索する集合を格納したリストの名前を指定します。

- ORDER BY {列名｜ソート項目指定番号} {ASC ｜ DESC}

ORDER BY 句の規則については形式 1 を参照してください。

- LIMIT {〔オフセット行数,〕{リミット行数｜ALL} ｜ {リミット行数｜ALL} 〔OFFSET オフセット行数〕}

LIMIT 句の規則については形式 1 を参照してください。

(5) 共通規則

1. リスト作成時にあった行が検索時にない場合には，SQLCODE + 110 を返します。この場合，検索は続行します。ただし，次に示すどちらかの場合には SQLCODE + 110 は返しません。
   - 受渡し値受信関数の引数以外にリストの基表の列名の指定がなく，かつリストの基表の列の * の指定がない
   - ORDER BY 句の指定がある
2. LIMIT 句にオフセット行数の指定がある場合，SQLCODE + 110 を返す行を読み飛ばす行数に含めます。
3. LIMIT 句にリミット行数の指定がある場合，SQLCODE + 110 を返す行を取得する行数に含めます。
4. 同一ユーザが，複数同時に HiRDB と接続してリストを操作できません。

(6) 留意事項

1. リスト作成後に基表の行を削除した場合，検索時にその行は検索されません。ただし，選択式に次に示すものだけを指定した場合，基表の行の削除前のデータが検索されます。
   - 基表の列名を含まない値式
   - 受渡し値受信関数
2. リスト作成後に基表の行を更新した場合，更新後のデータが検索されます。ただし，選択式に指定した受渡し値受信関数については，基表の行の更新前のデータが検索されます。
3. リスト作成後に，基表の行を削除してから行を挿入した場合，挿入した行が検索されることがあります。
4. リストを介して表を検索する SQL を，PREPARE 文で前処理してから OPEN 文を実行するまでの間に，同一リスト名のリストを作成する ASSIGN LIST 文は実行しないでください。
5. WRITE 指定をする場合，第一引数には基表の BLOB 型の列名，又は抽象データ型の BLOB 型の属性名を指定できます。

(7) 指定例（カーソル宣言中でカーソル指定を指定した場合）

```
PREPARE P1 FROM
 'SELECT SCODE,SNAME FROM LIST LIST1'
DECLARE CRL1 CURSOR FOR P1
```

2.2 問合せ式

2.2.1 問合せ式 形式 1（一般問合せ式）

(1) 機能

WITH 句中の導出問合せ式と問合せ式本体との組み合わせを指定します。WITH 句を用いた場合，導出問合せ式の結果得られる導出表を問合せ名とし，問合せ式本体に指定できます。

問合せ式本体では，問合せ指定，又は問合せ指定の結果得られる導出表同士の和集合，又は差集合を求めるため，集合演算を指定します。和集合を求める場合は UNION〔ALL〕，差集合を求める場合は EXCEPT〔ALL〕を指定します。問合せ式は，カーソル宣言，又は動的 SELECT 文中のカーソル指定中に指定できます。

(2) 形式

```
問合せ式：：＝〔WITH 問合せ名〔（列名〔，列名〕…）〕
                                AS（導出問合せ式）
              〔，問合せ名〔（列名〔，列名〕…）〕
                                AS（導出問合せ式）〕…〕
          問合せ式本体

導出問合せ式：：＝  問合せ式本体
問合せ式本体：：＝｛問合せ指定
                  ｜（問合せ式本体）
                  ｜問合せ式本体 {UNION｜EXCEPT} 〔ALL〕
                    {問合せ指定｜（問合せ式本体）}｝
```

(3) オペランド

- WITH 問合せ名〔(列名〔，列名〕…)〕

WITH 問合せ名

問合せ式本体の表式に指定する導出表の名称を指定します。WITH 句中に複数の問合せ名を指定する場合，同じ問合せ名は指定できません。

列名

WITH 句中の一つの導出問合せ式によって指定された導出表から導き出される列に対応させて指定します。

列名についての規則を次に示します。

1. 列名を省略した場合，次のようになります。

- WITH 句中の導出問合せ式に集合演算を指定しない場合，WITH 句中の一つの問合せ指定によって指定された導出表の各列の列名（AS 列名を指定している場合は，AS 句で指定した列名）が WITH 句問合せ名の列名となります。
- WITH 句中の導出問合せ式に集合演算を指定した場合，導出問合せ式の 1 番目の問合せ指定によって指定された導出表の各列の列名（AS 列名を指定している場合は，AS 句で指定した列名）が，WITH 句問合せ名の列名となります。

ただし，WITH 句中の一つの導出問合せ式によって指定された導出表が，列名が同じである二つ以上の列を含む，又は名前のない列を含む場合は省略できません。

2. 導出表の列が次に示す項目から導き出された列で，AS 列名を省略した場合，その列は名前のない列になります。
   - スカラ演算
   - 関数呼出し
   - 集合関数
   - 定数
   - USER 値関数
   - CURRENT_DATE 値関数
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数
   - SQL 変数
   - SQL パラメタ
   - コンポネント指定
3. WITH 句中の一つの問合せ名について，同じ名前は指定できません。
4. WITH 句中の一つの問合せ名について，列名で指定する列の数は，その WITH 句の導出問合せ式の結果で得られる導出表の列と同じ数にしてください。
5. WITH 句中の一つの問合せ名について，列名で指定できる列の数の最大値は 30,000 個です。

- **導出問合せ式：：＝ 問合せ式本体**

表式，問合せ名として導出する列，集合演算，及び行の重複排除の有無を指定します。問合せ式本体の表式に含まれる FROM 句には，表名，ビュー表名のどちらかを指定してください。

- **問合せ式本体：：＝ {問合せ指定｜（問合せ式本体）｜問合せ式本体**
  {UNION ｜ EXCEPT} 〔ALL〕
  {問合せ指定｜（問合せ式本体)}}

システム定義 pd_sql_mode に「1」を指定した場合は，差集合を求める集合演算子として EXCEPT に加えて MINUS も指定できます。

WITH 句を用いた場合，問合せ式本体の表式に含まれる FROM 句には，表名，ビュー表名，又は問合せ名のどれかを指定してください。

WITH 句を用いた場合，問合せ式本体には，1 行 SELECT 文は指定できません。

- **問合せ指定**

表式，選択する列，及び行の重複排除の有無を指定します。

問合せ指定については，「問合せ指定」を参照してください。

- **（問合せ式本体）**

UNION 又は EXCEPT によって結合している問合せ指定の組みが二つ以上指定されている場合，その問合せ式本体中の集合演算の評価順序を指定するために問合せ式本体を括弧で囲んで指定します。

- **問合せ式本体 {UNION ｜ EXCEPT} 〔ALL〕 {問合せ指定 ｜ （問合せ式本体）}**

二つの問合せ指定，又は問合せ式本体の結果から得られる導出表同士の和集合，又は差集合を求める場合に，指定します。集合演算については，「集合演算の共通規則」を参照してください。

(4) WITH 句中の導出問合せ式についての規則

1. WITH 句中の導出問合せ式によって，問合せ名として導出される列の属性（データ型，データ長，及び非ナル値制約の有無）は，WITH 句中の導出問合せ式で指定した導出表の対応する列の属性と同じになります。
2. WITH 句中の導出問合せ式の FROM 句には，実表名又はビュー表名を指定してください。問合せ名は指定できません。
3. WITH 句中の導出問合せ式には，選択式に〔表指定.〕ROW は指定できません。
4. WITH 句中の導出問合せ式には，直接含まれる SELECT 句に添字付きの繰返し列は指定できません。
5. WITH 句中の導出問合せ式には，次に示す値式は指定できません。
   - XML コンストラクタ関数
   - SQL/XML スカラ関数
   - SQL/XML 述語
   - SQL/XML 集合関数
6. WITH 句中の導出問合せ式の GROUP BY 句には，列指定以外の値式は指定できません。
7. WITH 句中の最も外側の問合せ指定の選択式に CASE 式を指定した場合，その CASE 式の探索条件には繰返し列を指定できません。
8. WITH 句中の導出問合せ式では，選択式に次の項目を指定できません。
   - WRITE 指定
   - GET_JAVA_STORED_ROUTINE_SOURCE 指定

- ウィンドウ関数

(5) 問合せ式本体についての規則

1. WITH 句を用いた場合，問合せ式本体の表式に含まれる FROM 句には，表名，ビュー表名，又は問合せ名のどれかを指定してください。
2. WITH 句を用いた場合，問合せ式本体には 1 行 SELECT 文は指定できません。
3. ビュー定義又は WITH 句中の導出問合せ式によって導き出される列が，値式を引数とした集合関数の場合，問合せ式本体ではその導出列を外への参照として用いることができません。

   （例）

```
WITH QRY1(QC1,QC2) AS (SELECT MAX(C1+C2),AVG(C1+C2) FROM T1),
     QRY2(QC1,QC2) AS (SELECT C1+C2,C3-C4 FROM T2)
  SELECT * FROM QRY1 X
     WHERE QC1>(SELECT QC1 FROM QRY2 WHERE QC2 = X.QC2)
```

4. ビュー定義又は WITH 句中の導出問合せ式によって導き出される列が COUNT（*）又は COUNT_FLOAT（*）の場合，問合せ式本体ではその導出列を外への参照として用いることができません。

   （例）

```
WITH QRY1(QC1) AS (SELECT COUNT(*) FROM T1),
     QRY2(QC1,QC2) AS (SELECT C1+C2,C3/C4 FROM T2)
 SELECT * FROM QRY1 WHERE EXISTS
     (SELECT * FROM QRY2 WHERE QC1 = QRY1.QC1)
```

5. WITH 句中の最も外側の問合せ指定の選択式に，次に示す属性の値式を指定して導出した表を問合せ式本体に指定する場合，内部導出表を作成する問合せは指定できません。内部導出表を作成する条件については，「内部導出表」を参照してください。
   - BLOB
   - 最大長が 32,001 バイト以上の BINARY
   - 抽象データ型
   - 繰返し列
6. 次に示す属性の関数呼出し，コンポネント指定，SQL 変数，及び SQL パラメタによって導出された列を，問合せ式本体の選択式に指定した場合，その SELECT 句を直接含む問合せには，FOR READ ONLY を指定できません。
   - BLOB
   - 定義長が 32,001 バイト以上の BINARY
   - 抽象データ型
7. ビュー定義又は WITH 句中の導出問合せ式によって導出される列が列指定以外の値式から導出されている場合，その導出列に対してコンポネント指定は指定できません。

8. ビュー定義又は WITH 句中の導出問合せ式によって導出される列がコンポネント指定によって導出されている場合，問合せ式本体ではその導出列を外への参照として用いることができません。
9. プラグインが提供する関数のうち，受け渡しする値を受信する関数（受渡し値受信関数）と受け渡しする値を送信する関数（受渡し値送信関数）のどちらかを指定して定義したビュー表に対する問合せでは，次のどれかに該当する場合，エラーになります。
   - ビュー定義中の CASE 式，又はスカラ関数 VALUE 中で受渡し値受信関数を指定している。
   - ビュー定義に受渡し値受信関数を指定し，その受渡し値受信関数に対応した受渡し値送信関数を二つ以上指定し，かつ，それらの受渡し値送信関数の第 1 引数が同じである。
   - ビュー定義に受渡し値受信関数又は受渡し値送信関数のどちらかを指定しかつ，そのビュー表を指定した問合せ指定が内部導出表を作成する。内部導出表を作成する条件については，「内部導出表」を参照してください。
   - ビュー定義に指定した受渡し値受信関数に対応する受渡し値送信関数が，そのビュー定義中と，定義したビュー表を指定した問合せ指定のどれにも存在しない。
   - ビュー表を指定した問合せ指定中の受渡し値受信関数に対応する受渡し値送信関数が，その問合せ指定中と，そのビュー表を定義した問合せのどちらにも存在しない。

(6) 集合演算の共通規則

1. 集合演算をする場合，その対象になる二つの問合せ指定，導出問合せ式，又は問合せ式本体の結果から得られる導出表をそれぞれ行の集合とみなして，集合演算（和集合，又は差集合を求める）をします。
   したがって，これらの集合演算の対象になる導出表同士の構成列の列数，及び列の並び順は，同じにしてください。また，対応する列のデータ型は，比較できるデータ型にしてください。ただし，日付データと日付データの文字列表現，時刻データと時刻データの文字列表現，時刻印データと時刻印データの文字列表現，日間隔データと日間隔データの 10 進数表現，及び時間隔データと時間隔データの 10 進数表現とは集合演算できません。
2. 集合演算の対象になる導出表同士の列が文字データ型の場合，対応するそれぞれの列の文字集合はすべて同じにしてください。ただし，2 番目以降の問合せ指定によって指定された導出表の列が次に示す値式の場合，1 番目の問合せ指定によって指定された導出表の列の文字集合に変換されます。
   - 文字列定数
3. 集合演算で導出される表の列名は，問合せ式本体の 1 番目の問合せ指定によって指定された導出表の各列の列名（AS 列名を指定している場合は，AS 句で指定した列名）となります。
4. 導出表の列が次の項目から導き出された列で，AS 列名を省略した場合，その列は名前のない列になります。
   - 四則演算，日付演算，時刻演算，連結演算，又はシステム組込みスカラ関数を含む値式
   - CASE 式
   - CAST 指定
   - 集合関数

- 定数
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- SQL 変数
- SQL パラメタ
- コンポネント指定
- 関数呼出し

5. 集合演算の結果，得られた導出表の構成列の列数，及び列の並び順は演算対象の導出表の構成列と同じになります。
6. 定数，集合関数，日付演算，時刻演算，四則演算，CASE 式，及び CAST 指定の結果は，非ナル値制約なし（ナル値を許します）になります。
7. 集合演算をする場合，問合せ指定中の FROM 句に問合せ式本体は指定できません。
8. 集合演算をする場合，対象となる導出表の構成列には，結果が次のデータ型となる値式は指定できません。
   - BLOB
   - 最大長が 32,001 バイト以上の BINARY
   - BOOLEAN
   - 抽象データ型
9. 集合演算を指定する場合，対象になる導出表の列に，繰返し列を含むことはできません。
10. 集合演算をする場合，対象となる導出表の列に WRITE 指定，GET_JAVA_STORED_ROUTINE_SOURCE 指定，及びウィンドウ関数は指定できません。
11. 集合演算の結果のデータ型，及びデータ長は次のとおりです。

＜文字データ，各国文字データ，混在文字データの場合＞

- 値式のどれかが可変長データの場合の結果のデータは，可変長データになります。
- 結果のデータ長は，一番長いデータ長を持つ値式のデータ長になります。
- 値式に文字データと混在文字データが含まれる場合の結果のデータ型は，混在文字データになります。

＜数値データの場合＞

数値データの場合の，結果のデータ型を次に示します。

オペランド 1	オペランド 2				
	SMALLINT	INTEGER	DECIMAL	SMALLFLT	FLOAT
SMALLINT	SMALLINT	INTEGER	DECIMAL	SMALLFLT	FLOAT

オペランド 1	オペランド 2				
	SMALLINT	INTEGER	DECIMAL	SMALLFLT	FLOAT
INTEGER	INTEGER	INTEGER	DECIMAL	FLOAT	FLOAT
DECIMAL	DECIMAL	DECIMAL	DECIMAL	FLOAT	FLOAT
SMALLFLT	SMALLFLT	FLOAT	FLOAT	SMALLFLT	FLOAT
FLOAT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT

結果のデータ型が DECIMAL の時の精度と位取りを次に示します。p，s をオペランド 1 の精度と位取り，p2，s2 をオペランド 2 の精度と位取りにします。

・精度＝ max（p1－s1，p2－s2）＋ max（s1，s2）

・位取り＝ max（s1，s2）

pd_sql_dec_op_maxprec オペランドの指定値によって，次のように精度が決まります。

pd_sql_dec_op_maxprec オペランドの値	集合演算オペランドの結果列と対応する列の精度	max（p1－s1，p2－s2）＋ max（s1，s2）で求めた精度	集合演算オペランドの結果列の精度
29	すべての列が 29 けた以下	29 けた以下	max（p1－s1，p2－s2）＋ max（s1，s2）で求めた精度
		30 けた以上	エラー
	30 けた以上の列を含む	38 けた以下	max（p1－s1，p2－s2）＋ max（s1，s2）で求めた精度
		39 けた以上	エラー
38	任意	38 けた以下	max（p1－s1，p2－s2）＋ max（s1，s2）で求めた精度
		39 けた以上	エラー

pd_sql_dec_op_maxprec オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

また，INTEGER は DECIMAL（10，0），SMALLINT は DECIMAL（5，0）として扱われます。ただし，非ナル値制約の有無については，対応する構成列がすべて非ナル値制約ありの場合だけ非ナル値制約ありとして扱います。それ以外の場合は非ナル値制約なしとして扱います。

＜時刻印データの場合＞

時刻印データ同士の集合演算ができます。結果のデータ型も同じです。集合演算項が小数秒精度を含む場合の，結果の小数秒精度は，「Q1 集合演算 Q2」での Q1 の小数秒精度を p1，Q2 の小数秒精度を p2 としたとき，MAX（p1，p2）となります。

＜バイナリデータの場合＞

結果のデータ長は，一番長いデータ長を持つ値式のデータ長になります。

12. ALL を指定した場合，重複した行があっても取り除かないでそのまま残し，それぞれの行とします。ALL を省略した場合，重複した行があれば，重複を排除し一つの行とみなします。

「Q1 集合演算 Q2」の結果の重複行の行数は，ALL 指定があるかないかによって異なります。Q1 は問合せ式本体（導出問合せ式），Q2 は問合せ式本体（導出問合せ式），又は問合せ指定です。Q1 中の重複行，Q2 中の重複行，Q1 及び Q2 中の重複行を R とし，重複行 R の Q1 中の行数を m，Q2 中の行数を n とします（m≧0，n≧0）。Q1，Q2，又は Q1 と Q2 の重複した行がない場合は，重複行 R が 1 行ある（m ＝ 1，n ＝ 0 又は m ＝ 0，n ＝ 1）とみなして，集合演算します。各集合演算の結果の各重複行 R の行数を次に示します。

集合演算	ALL 指定なし	ALL 指定あり
UNION	min（1，n ＋ m）	m ＋ n
EXCEPT	1（m ＞ 0 かつ n ＝ 0） 0（m ＝ 0 又は n ＞ 0）	max（m−n，0）

(7) 留意事項

1. 次の条件を満たす場合，HiRDB が作業表を作成することがあります。

   • UNION〔ALL〕，又は EXCEPT〔ALL〕を指定している

   このとき，作業表の行長によっては，上記の処理が制限されることがあります。作業表の行長については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(8) 指定例

<WITH 句を用いた場合>

```
DECLARE CR1 CURSOR FOR
    WITH QRY1(QSCODE,QSNAME,QCOL,QURIAGE) AS
        (SELECT SCODE,SNAME,COL,TANKA*ZSURYO FROM ZAIKO)
            SELECT QSNAME,MAX(QURIAGE) FROM QRY1 GROUP BY QSNAME
```

<WITH 句を用いない場合>

```
DECLARE CR1 CURSOR FOR
    SELECT SCODE,SNAME,COL,TANKA,ZSURYO
        FROM ZAIKO
```

2.2.2 問合せ式　形式 2（繰返し列平坦化問合せ式）

(1) 機能

繰返し列を要素ごとに分割し，それぞれを別の行として検索できます。

また，複数の繰返し列の添字が等しい要素同士を，同じ行として検索できます。これを，**繰返し列の平坦化機能**といいます。

(2) 形式

```
SELECT 〔ALL｜DISTINCT〕 {選択式〔, 選択式〕 …｜*}
 FROM〔認可識別子.〕表識別子
       〔IN （RDエリア名指定）〕
       (FLAT（列名〔, 列名〕 …）) 〔〔AS〕相関名〕
       〔使用インデクスのSQL最適化指定〕
       〔WHERE 探索条件〕
       〔GROUP BY 列指定〔, 列指定〕…〕
       〔HAVING 探索条件〕
```

(3) オペランド

選択式，FLAT，及び探索条件以外のオペランドについては，「問合せ式　形式 1（一般問合せ式）」を参照してください。

- **{選択式〔, 選択式〕…｜*}**

検索結果として出力する項目を指定します。

選択式については，「問合せ指定」を参照してください。

- **〔IN （RD エリア名指定)〕**

IN

アクセス対象の RD エリアを指定します。指定できない場合を次に示します。

- 表識別子に指定した表が読み込み専用のビュー表の場合
- ビュー定義の導出問合せ式中の場合
- 表識別子に指定した表が一時表の場合

RD エリア名指定：：＝値指定

表識別子に指定した表を格納している RD エリアのうち，アクセスする RD エリアの名称を VARCHAR 型，CHAR 型，MVARCHAR 型，又は MCHAR 型の値指定で指定します。複数の RD エリア名を指定する場合はコンマ（,）で区切って指定してください。RD エリア名は重複して指定できません。重複して指定した場合はエラーとなります。値指定で指定する RD エリア名に許される文字については，「名前の指定」を参照してください。また，値指定で指定した RD エリア名の前後の空白は無視されます。RD エリア名を引用符（"）で囲んだ場合は，引用符（"）の外側の空白だけを無視します。

インナレプリカ機能を適用している RD エリアを指定する場合，オリジナル RD エリア名を指定してください。レプリカ RD エリアを対象とする場合は，カレント切り替えコマンド（pddbchg コマンド），又はクライアント環境定義の PDDBACCS オペランドでアクセス対象 RD エリアをレプリカ RD エリアに切り替えてください。

- (FLAT (列名 〔, 列名〕…))

平坦化の対象となる繰返し列又は，繰返し列を含む列の並びを指定します。

複数の繰返し列を指定した場合は，添字が等しい要素が同じ行に含まれるように平坦化されます。

通常の列（繰返し構造でない列）も含む場合は，繰返し列のどの要素に対しても同じ値を持つ行が生成されます。

繰返し列，又は繰返し列を含む列の組を平坦化した後は，通常の列になります。したがって，その問合せ式中では通常の列として指定します。

- 探索条件

探索条件中には，平坦化した後の項目を指定できます。

探索条件については，「探索条件」を参照してください。

(4) 規則

1. 平坦化する列とインデクスとの関係を次に示します。

<table>
<tr><th colspan="4">列名に指定した列とインデクスの関係</th><th>SQL 文
実行可否</th></tr>
<tr><td rowspan="4">インデクス
定義あり</td><td rowspan="3">繰返し列を
含むインデ
クスがある</td><td rowspan="2">FLAT の列名に指定した列のうち，SQL 中に指定した列をすべて包括したインデクスがある</td><td>FLAT の列名中に一つでも繰返し列が存在する</td><td>○</td></tr>
<tr><td>FLAT の列名中に一つも繰返し列が存在しない</td><td>×</td></tr>
<tr><td colspan="2">FLAT の列名に指定した列のうち，SQL 中に指定した列をすべて包括したインデクスがない</td><td>×</td></tr>
<tr><td colspan="3">繰返し列を含むインデクスがない</td><td>×</td></tr>
<tr><td colspan="4">インデクス定義なし</td><td>×</td></tr>
</table>

（凡例）

○：実行可能 ×：エラー

2. 使用インデクスの SQL 最適化指定に WITHOUT INDEX を指定した場合，その最適化指定を無視し，HiRDB が使用できるインデクスを用いて検索します。
3. 使用インデクスの SQL 最適化指定に WITH INDEX を指定する場合，規則 1 に示した SQL 文実行可能な条件を満たすインデクスを指定してください。指定したインデクスが SQL 文実行可能な条件を満たしていない場合，その最適化指定を無視し，HiRDB が利用できるインデクスを用いて検索します。
4. FROM 句の FLAT 指定は，INSERT SELECT の SELECT 文，WITH 句問合せ，ビュー定義，FROM 句の導出表，及び副問合せ中には指定できません。
5. FLAT に指定できる列名は最大 64 個です。

6. 平坦化を指定した SQL 中で指定する列名には，FLAT で指定した列名を指定してください。
7. 表識別子には，ビュー表を指定できません。
8. 平坦化を指定する場合，以下の項目を指定できません。
   - FOR UPDATE 句
   - FOR READ ONLY
   - LIMIT 句
   - コンポネント指定
   - 関数呼出し
   - 副問合せ
   - FLAT 指定のある集合関数
   - 添字を指定した列
9. 平坦化を指定する場合，探索条件中に以下の項目を指定できません。
   - 構造化繰返し述語
10. RD エリア名指定を指定した場合，分割数が表の分割数と異なるインデクスは利用できません。RD エリア名指定を指定する問合せのためにインデクスを定義する場合は，分割数が表の分割数と等しいインデクスを定義してください。

(5) 指定例

繰返し列を用いた成績表を平坦化し，成績が 70 点以上の学生の一覧を検索します。

```
SELECT 氏名,科目名,成績 FROM 成績表 (FLAT(氏名,科目名,成績))
    WHERE 成績>=70
```

成績表

氏名	科目名	成績
山田	数学	90
	国語	65
鈴木	数学	50
	国語	90
佐藤	数学	85
	国語	70

検索結果

氏名	科目名	成績
山田	数学	90
鈴木	国語	90
佐藤	数学	85
佐藤	国語	70

成績表の表定義及びインデクス定義は次のとおりです。

```
表定義：
    CREATE TABLE 成績表(氏名 MCHAR(10),
                        科目名 MCHAR(10) ARRAY[4],
                        成績 SMALLINT ARRAY[4]);
インデクス定義：
    CREATE INDEX 科目成績 ON 成績表(氏名,科目名,成績);
```

2.3 問合せ指定

2.3.1 問合せ指定の形式と規則

(1) 機能

表に対する検索の条件（表式），及び検索した結果を出力する項目（選択式）を指定することで，選択式を列とする検索結果で構成された表を導き出します。この導き出された表のことを導出表といいます。

(2) 使用権限

表に対する SELECT 権限を持つユーザが，その表を検索する問合せ指定を実行できます。

(3) 形式

```
SELECT〔 {ALL | DISTINCT} 〕 {選択式〔, 選択式〕… | *} 表式
```

(4) オペランド

- 〔{ALL | DISTINCT}〕

検索した結果，重複した行（選択式で指定した項目で一つの行を構成し，その行が同一のもの）があった場合，重複した行を排除するかどうかを指定します。

重複した行を排除することを行の重複排除といいます。

ALL
　検索した結果，重複した行があっても重複を許し，そのまま出力します。

DISTINCT
　検索した結果，重複した行があったら，重複を排除し，一つの行とみなして出力します。

- {選択式〔, 選択式〕… | *}

検索結果として出力する項目を指定します。

選択式
　選択式として次に示すものが指定できます。

  - 列指定
  - 〔表指定.〕ROW
  - 集合関数
  - ウィンドウ関数

- スカラ関数※
- CASE 式
- CAST 指定
- 定数
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- SQL 変数又は SQL パラメタ
- 値式※ 〔[AS] 列名〕
- 表指定.*
- コンポネント指定
- 関数呼出し※
- WRITE 指定
- GET_JAVA_STORED_ROUTINE_SOURCE 指定
- スカラ副問合せ

注※

関数呼出しの引数には，？パラメタ，及び埋込み変数を指定できます。指定方法については，「規則」を参照してください。

スカラ関数の LENGTH，SUBSTR 及び POSITION の引数には，？パラメタ及び埋込み変数を指定できます。指定方法についてはそれぞれの関数の規則を参照してください。

選択式についての規則を次に示します。

1. スカラ関数 LENGTH，SUBSTR，及び POSITION の引数には，？パラメタ及び埋込み変数を指定できます。ただし，LENGTH の引数，SUBSTR の第 1 引数，POSITION の第 1 引数，及び第 2 引数にはデータ型が BLOB 型又は BINARY 型，かつ AS データ型を指定した場合だけ指定できます。
2. 選択式中には結果のデータ型が次に示すデータ型の値式は指定できません。
   - BOOLEAN
3. カーソル宣言，動的 SELECT 文，1 行 SELECT 文，FROM 句の導出表，及び述語中の副問合せの選択式中には，結果のデータ型が抽象データ型の値式は指定できません。
4. SELECT DISTINCT を指定した場合，選択式中又は表式中では，DISTINCT 指定の集合関数は指定できません。
5. 選択式に繰返し列を指定した場合，次の指定はできません。
   - SELECT DISTINCT

- その問合せに対する UNION 〔ALL〕
- その問合せに対する EXCEPT 〔ALL〕

6. 〔表指定.〕ROW は，FIX 属性の実表に対してだけ指定できます。ROW は行全体（予備列を含む）を意味し，これを指定することで行全体を一つのデータとして一つの領域に取り出します。取り出されるデータのデータ型は，各列のデータ型にかかわらず ROW 型（ROW 型に対しては，CHAR(n)（n は行長）に対応する変数，又は構造体を指定できます。ただし，構造体中に境界調整による空きがあってはいけません）とします。また，データ長は行長（各列のデータ長の総和）になります。表に予備列が定義してある場合は，取り出した領域中の予備列に対応する部分は予備列に格納されているデータ（予備列の定義長分の 0x00）になります。選択式中に ROW を指定した場合，その問合せ指定に対する次のものと同時に指定できません。
   - 集合関数
   - GROUP BY 句
   - UNION〔ALL〕，又は EXCEPT〔ALL〕
7. 〔表指定.〕ROW は，既定文字集合以外の文字データ型の列を含む表には指定できません。
8. ROW 型を使用する場合は，UAP が動作するプラットフォームと HiRDB サーバが動作するプラットフォームのエンディアンを同じにしてください。異なるエンディアン間では，ROW 型は使用できません。例えば，Windows の UAP で ROW 型を使用する場合は，HiRDB サーバも同じエンディアンの Windows 版を使用してください。
9. 〔表指定.〕ROW は，結合表の内表に対して指定できません。
10. SELECT 句に指定する列は，その表式の FROM 句で指定した表を参照してください。
11. 表式中に集合関数，GROUP BY 句，又は HAVING 句を指定した場合，SELECT 句の列指定は次のどちらかにしてください。
   - グループ化列（GROUP BY 句で指定した値式）
   - 集合関数の引数中で指定
12. 次のデータ型の SQL 変数又は SQL パラメタを選択式に指定した場合，その SELECT 句を直接含む問合せ指定，及びその SELECT 句を含む問合せには，次の機能又は句は指定できません。

   ＜ SQL 変数又は SQL パラメタのデータ型＞
   - BLOB
   - 定義長が 32,001 バイト以上の BINARY
   - BOOLEAN
   - 抽象データ型

   ＜問合せ指定中に同時に指定できない機能又は句＞
   - 集合関数
   - GROUP BY 句
   - HAVING 句

- DISTINCT
- ビュー定義中の導出問合せ式で，上記の機能，句を指定したビュー表を，FROM 句に指定

＜問合せ中に同時に指定できない機能又は句＞

- 集合演算
- ORDER BY 句
- FOR UPDATE 句
- FOR READ ONLY

13. 表指定.＊を指定した場合，指定した表のすべての列（予備列を除く）を表定義時に指定した順序で指定することを意味します。
14. AS 列名は，選択式に対して名称を付ける場合に指定します。
15. n 番目の選択式に AS 列名を指定した場合，その列名は選択式を指定した問合せ指定によって導出される表の n 番目の列名になります。
16. n 番目の選択式に AS 列名を指定しない場合，選択式に指定した列名が問合せ指定によって導出される表の n 番目の列名になります。ただし，選択式が列指定だけでない場合，n 番目の列は名前のない列になります。
17. n 番目の選択式にスカラ副問合せを指定した場合，問合せ指定によって導出される n 番目の列名は，スカラ副問合せの問合せ指定によって導出される列名になります。ただし，AS 列名を指定している場合は，AS 列名で指定した列名になります。また，スカラ副問合せの問合せ指定によって導出される列が名前のない列の場合，問合せ指定によって導出される n 番目の列は，名前のない列になります。
18. WRITE 指定は，選択式に単独で指定できます。
19. GET_JAVA_STORED_ROUTINE_SOURCE 指定は次の箇所で指定できます。
    - 最も外側の問合せ指定の選択式に単独で指定
    - 最も外側の問合せ指定の選択式の，スカラ関数 LENGTH の引数に指定
20. DISTINCT を指定した場合，WRITE 指定及び GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
21. INSERT 文の問合せ指定の選択式には，WRITE 指定及び GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
22. 表式中に列指定以外の値式を指定した GROUP BY 句を指定した場合，
    選択式に指定したスカラ副問合せ中からそのグループ化列を参照することはできません。
23. 選択式にウィンドウ関数を指定する場合，ウィンドウ関数以外の選択式を一つ以上指定してください。
24. スカラ演算中にウィンドウ関数は指定できません。
    ウィンドウ関数を指定した場合，選択式に集合関数を指定できません。
25. 選択式中には，予備列，及び予備列を指定した値式は指定できません。

*

表のすべての列を出力する場合，指定します。

＊は，その問合せ指定の FROM 句で指定したすべての表の，すべての列（予備列を除く）を，FROM 句で指定した表の順序で指定することを意味します。各表中の列の順序は，表定義時に指定した順序になります。

- **表式**

表式については，「表式」を参照してください。

なお，ビュー定義文中の導出問合せ式の制限項目については，「CREATE 〔PUBLIC〕VIEW（ビュー定義，パブリックビュー定義）」を参照してください。

(5) 規則

1. 最も外側の問合せ指定の選択式に，次に示す属性の値式を指定して導出した，名前付きの導出表に対する問合せ指定では，内部導出表を作成する問合せは指定できません。内部導出表を作成する条件については，「内部導出表」を参照してください。
   - BLOB
   - 最大長が 32,001 バイト以上の BINARY
   - 抽象データ型
   - 繰返し列
2. 関数呼出しの引数に？パラメタ，又は埋込み変数を指定する場合，引数は次の形式で指定してください。
   - ？ AS データ型
   - ：埋込み変数〔：標識変数〕 AS データ型

(6) 留意事項

1. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の日付データ型の部分は 4 バイトです。また，形式は X'YYYYMMDD'にしてください。

   行単位（ROW 指定）インタフェースを使用して，日付データを文字表現で受け渡しする場合，列は日付データ型ではなく，CHAR（10）で定義してください。さらに，日付演算は，スカラ関数 DATE を使用し，日付データ型に変換してから指定してください。
2. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の時刻データ型の部分は 3 バイトです。また，形式は X'hhmmss'にしてください。行単位（ROW 指定）インタフェースを使用して，時刻データを文字表現で受け渡しする場合，列は時刻データ型ではなく，CHAR（8）で定義してください。さらに，時刻演算は，スカラ関数 TIME を使用し，時刻データ型に変換してから指定してください。
3. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の時刻印データ型の部分は（7 ＋(p÷2)）バイトです。また，形式は

X'YYYYMMDDhhmmss〔nn…n〕'にしてください。行単位（ROW 指定）インタフェースを使用して，時刻印データを文字表現で受け渡しする場合，列は時刻印データ型ではなく，長さが 19，22，24，又は 26 バイトの CHAR で定義してください。

4. 次のどちらかの条件を満たす場合，HiRDB が作業表を作成することがあります。
   - DISTINCT を指定している
   - 選択式中に集合関数を含む値式を指定している
   - 選択式中にウィンドウ関数を含む値式を指定している

   このとき，作業表の行長によっては，上記の処理が制限されることがあります。作業表の行長については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

2.4 副問合せ

2.4.1 副問合せ指定の形式と規則

(1) 機能

表を検索して得られる値，又は値の集合を指定します。

副問合せの種類を次に示します。

- スカラ副問合せ
  スカラ副問合せとは，結果の列数が 1 列，結果の行数が 1 行以下の副問合せを示します。
- 行副問合せ
  行副問合せとは，結果の列数が 2 列以上，結果の行数が 1 行以下の副問合せを示します。
- 表副問合せ
  表副問合せとは，結果の列数が 1 列以上，結果の行数が 0 行以上の副問合せを示します。

各副問合せは，次に示す箇所に指定します。

スカラ副問合せ

- 値式を指定できる箇所

行副問合せ

- 行値構成子を指定できる箇所
- UPDATE 文の SET 句

表副問合せ

- IN 述語の右側
- 限定述語の右側
- EXISTS 述語
- FROM 句の導出表

(2) 使用権限

表に対する SELECT 権限を持つユーザが，その表を検索する副問合せを実行できます。

(3) 形式

```
スカラ副問合せ：：＝副問合せ
行副問合せ：：＝副問合せ
表副問合せ：：＝副問合せ
```

副問合せ：：＝（〔副問合せ実行方式のSQL最適化指定〕問合せ式本体）

(4) オペランド

- （〔副問合せ実行方式の SQL 最適化指定〕問合せ式本体）

副問合せ実行方式の SQL 最適化指定については「SQL 最適化指定」を，問合せ式本体については「問合せ式」を参照してください。

(5) 述語（IN 述語，比較述語，限定述語，及び EXISTS 述語）中，又は FROM 句の導出表の副問合せの規則

1. 述語中，又は導出表の副問合せの選択式には，結果が次のデータ型となる値式は指定できません。
   - BLOB
   - 最大長が 32,001 バイト以上の BINARY
   - 抽象データ型
2. EXISTS 述語の副問合せで*又は表指定.*を指定した場合は，次のようになります。
   - 集合演算を指定しない場合

   集合関数以外のその副問合せで許される任意の 1 列を意味します。
   - 集合演算を指定した場合

   *を指定した場合は，その問合せ指定の FROM 句で指定したすべての表のすべての列（予備列を除く）を，FROM 句で指定した表の順序で指定することを意味します。各表中の列の順序は，表定義時に指定した順序になります。

   表指定.*を指定した場合は，指定した表のすべての列（予備列を除く）を，表定義時に指定した順序で指定することを意味します。

(6) 規則

1. 副問合せの結果は，非ナル値制約なし（ナル値を許します）になります。ただし，FROM 句の導出表に指定した表副問合せの結果は，その問合せ式の結果の制約となります。
2. 行値構成子に指定した行副問合せの結果の列数は，最大 255 個です。
3. UPDATE の SET 句に指定した行副問合せの結果の列数は，最大 30,000 個です。
4. 表副問合せの結果の列数は，最大 255 個です。ただし，FROM 句の導出表の表副問合せの結果の列数は，最大 30,000 個です。
5. スカラ副問合せ及び行副問合せを指定する場合は，結果の行数は 1 行以下でなければなりません。
6. スカラ副問合せの結果が 0 行の場合，結果の値はナル値になります。
7. 行副問合せの結果が 0 行の場合，結果は構成要素がすべてナル値となる行になります。

8. 副問合せ中の選択式には，結果のデータ型が次に示すデータ型の値式は指定できません。
   - BLOB
   - 最大長が 32,001 バイト以上の BINARY
   - 抽象データ型
   - BOOLEAN

   ただし，UPDATE 文の SET 句中で，更新値として直接指定したスカラ副問合せ及び行副問合せの選択式については，上記制限はありません。

   また，XMLQUERY 関数の XML 問合せ変数の値式に指定したスカラ副問合せ中の選択式には，XML 型の値式として XMLAGG 集合関数が指定できます。
9. 副問合せ中の選択式には，〔表指定.〕ROW は指定できません。
10. 副問合せ中の選択式に WRITE 指定，GET_JAVA_STORED_ROUTINE_SOURCE 指定，及びウィンドウ関数は指定できません。
11. 副問合せ中の選択式には，添字のない繰返し列を指定できません。

    ただし，UPDATE 文の SET 句中で，更新値として直接指定したスカラ副問合せ及び行副問合せの選択式については，上記の制限はありません。
12. 集合関数の引数に指定する値式中に，副問合せは指定できません。
13. 副問合せ中に予備列は指定できません。

(7) 指定例

1. SELECT 文中の限定述語中で表副問合せを指定した場合

```
SELECT DISTINCT SNAME FROM ZAIKO
    WHERE SURYO > ALL
    (SELECT SURYO FROM ZAIKO
    WHERE SNAME = N'ソックス')
```

2. UPDATE 文の SET 句でスカラ副問合せを指定した場合

   在庫表（ZAIKO）の商品コード（SCODE）列が 302S の商品の在庫量（ZSURYO）列を，在庫表と同じ列定義情報を持つ在庫表 2（ZAIKO2）の商品コード（SCODE）列が 302S の商品の在庫量（ZSURYO）に変更します。

```
UPDATE ZAIKO
    SET ZSURYO =
        (SELECT ZSURYO FROM ZAIKO2 WHERE SCODE = '302S')
    WHERE SCODE = '302S'
```

3. UPDATE 文の SET 句で行副問合せを指定した場合

   在庫表（ZAIKO）の商品コード（SCODE）列が 302S の商品の在庫量（ZSURYO）列と単価（TANKA）列を，在庫表と同じ列定義情報を持つ在庫表 2（ZAIKO2）の商品コード（SCODE）列が 302S の商品の在庫量（ZSURYO）と単価（TANKA）に変更します。

```
UPDATE ZAIKO
    SET (TANKA,ZSURYO) =
        (SELECT TANKA,ZSURYO FROM ZAIKO2 WHERE SCODE = '302S')
    WHERE SCODE = '302S'
```

2.5 表式

2.5.1 表式の形式と規則

(1) 機能

検索の対象となる一つ又は複数の表を指定します。また，表を検索，又は結合するときの条件（探索条件やグループ分け）が指定できます。表式は副問合せ及び問合せ指定中，又は 1 行 SELECT 文中に指定します。

(2) 形式

```
FROM 表参照〔，表参照〕…
 〔WHERE 探索条件〕
 〔GROUP BY 値式〔，値式〕…〕
 〔HAVING 探索条件〕
```

(3) オペランド

(a) 副問合せを使用しない場合

- FROM 表参照〔，表参照〕…

検索する，表，問合せ名，導出表，又は結合表を指定します。表参照については，「表参照」を参照してください。

- 〔WHERE 探索条件〕

探索条件を省略すると，表（指定した表，結合表，導出表，及び WITH 句中の導出問合せ式から問合せ名として導出された表）から導き出されるすべての行が検索されます。

探索条件中に，埋込み変数を指定できます。PREPARE 文で前処理をする SELECT 文中では，埋込み変数の代わりに？パラメタを指定します。SQL 手続き中では，SQL 変数又は SQL パラメタを使用します。Java 手続き中については，マニュアル「HiRDB UAP 開発ガイド」の JDBC ドライバを参照してください。

- 〔GROUP BY 値式〔，値式〕…〕

グループ分けでは，GROUP BY 句で指定した値式のすべての結果の値が同一の行をそれぞれ一つのグループとして分け，その結果をグループごとに 1 行にして出力します。

選択式には，グループ分けする列の名称，集合関数，定数，若しくはこれらを一次子とする値式，又はグループ分けをする値式だけ指定できます。すなわち，グループ分けする値式（値式が列指定の場合を除く）を一次子として含む値式は指定できません。なお，GROUP BY 句で指定した値式をグループ化列といいます。

GROUP BY 句についての規則を次に示します。

1. グループ分けの条件となる列にナル値を持つ行がある場合は，ナル値を同じ値として扱いグループ分けします。
2. GROUP BY 句に指定する値式は，同じ形式の値式を重複して指定できません。
3. GROUP BY 句に指定する値式は，集合関数を含むことはできません。
4. GROUP BY 句に指定する値式は，ウィンドウ関数を含むことはできません。
5. GROUP BY 句に指定する値式中には，コンポネント指定を含むことはできません。
6. GROUP BY 句に指定する値式中には，繰返し列は指定できません。
7. GROUP BY 句に指定する値式中には，予備列は指定できません。
8. GROUP BY 句に指定する値式には，副問合せを含むことはできません。

- 〔HAVING 探索条件〕

集合関数（AVG，MAX，MIN，SUM，COUNT 及び COUNT_FLOAT）を指定できます。

HAVING 句についての規則を次に示します。

1. 集合関数以外の探索条件には，グループ分けした値式だけ指定できます。
2. グループ分けした値式は，GROUP BY 句で指定した値式と全く同じ形式にしてください。
3. 探索条件を省略した場合は，すべてのグループを出力します。

(b) 副問合せを使用する場合

- FROM 表参照〔，表参照〕…

検索する，表，問合せ名，導出表，又は結合表を指定します。表参照については，「表参照」を参照してください。

FROM 句に複数の表を追加した場合，各表から 1 行ずつ組み合わせて指定した表の順序に連結した行が FROM 句の結果の表の行になり，その行数は各表の行数の積になります。

- 〔WHERE 探索条件〕

副問合せ中の探索条件中の列指定は，副問合せの外側で指定した表の列を参照できます。

問合せが入れ子の場合，内側の問合せから外側の問合せで指定している表（の列）を参照することを，外への参照をするといいます。

WHERE 句についての規則を次に示します。

1. WHERE 句に COUNT（*）又は COUNT_FLOAT（*）は指定できません。
2. HAVING 句に属する WHERE 句中だけ，WHERE 句に集合関数を指定できます。

3. HAVING 句に属する WHERE 句に集合関数を指定した場合，その集合関数中の列指定は，その HAVING 句に先行する FROM 句に指定した表を参照（外への参照）してください。
4. GROUP BY 句に列指定以外の値式を指定した問合せ指定の WHERE 句の副問合せには，GROUP BY 句に列指定以外の値式を指定できません。

- 〔GROUP BY 値式〔，値式〕…〕

GROUP BY 句に指定する列は，その GROUP BY 句がある表式中の FROM 句で指定している表を参照してください。

グループ分けでは，GROUP BY 句で指定した値式のすべての結果の値が同一の行をそれぞれ一つのグループとして分け，その結果をグループごとに 1 行にして出力します。

選択式には，グループ分けする列の名称，集合関数，定数，若しくはこれらを一次子とする値式，又はグループ分けをする値式だけ指定できます。すなわち，グループ分けする値式（値式が列指定の場合を除く）を一次子として含む値式は指定できません。なお，GROUP BY 句に指定した値式をグループ化列といいます。

また，値式に列指定以外を指定した場合，選択式に指定したスカラ副問合せ中からそのグループ化列を参照することはできません。

GROUP BY 句についての規則を次に示します。

1. グループ分けの条件になる列にナル値を持つ行がある場合，ナル値同士を同じ扱いとしてグループ分けします。
2. GROUP BY 句に指定する値式は，同じ形式の値式を重複して指定できません。
3. GROUP BY 句に指定する値式は，集合関数を含むことはできません。
4. GROUP BY 句に指定する値式は，ウィンドウ関数を含むことはできません。
5. GROUP BY 句に指定する値式中には，コンポネント指定を含むことはできません。
6. GROUP BY 句に指定する値式中には，繰返し列は指定できません。
7. GROUP BY 句に指定する値式中には，予備列は指定できません。
8. GROUP BY 句に指定する値式には，副問合せを含むことはできません。

- 〔HAVING 探索条件〕

HAVING 句は，先行する GROUP BY 句，WHERE 句，又は FROM 句の結果，得られる各グループを選択する条件を指定します。ただし，GROUP BY 句を指定しないと，WHERE 句，又は FROM 句の結果はグループ化列を持たない一つのグループになります。

HAVING 句についての規則を次に示します。

1. HAVING 句中に指定した探索条件の結果が真となるグループが選択されます。

2. HAVING 句を省略すると，先行する GROUP BY 句，WHERE 句，又は FROM 句の結果のすべてのグループが選択されます。
3. グループ分けした値式は，GROUP BY 句で指定した値式と全く同じ形式にしてください。
4. HAVING 句中に列指定を指定する場合，次のようにしてください。
   - その表式中の FROM 句の表を参照するか，又は外側の表式中の FROM 句の表を参照（外への参照）する
   - その表式中の FROM 句の表を参照している場合，グループ化列（GROUP BY 句で指定した値式）か，又は集合関数の引数中に指定する

(4) 共通規則

1. 探索条件中に埋込み変数を指定できます。
2. PREPARE 文で前処理をする SQL 文中では，埋込み変数の代わりに?パラメタを指定します。

(5) 結合についての規則

一つの FROM 句に複数の表，又は問合せ名を指定した検索（複数の表にわたる検索）を結合といいます。

1. FROM 句に，結合するすべての表，又は問合せ名を指定します。WHERE 句に結合条件がない場合は，結合する各表から一つずつ行を取り出して組み合わせたものが結果になります。例えば，l 行，m 行，n 行から成る三つの表を無条件で結合すると l ×m×n 個の行になります。
2. WHERE 句に表間の関係を示す条件（結合条件）を指定した場合は，1.の結合結果から条件に合った行を選択します。
3. 結合条件中に指定する列（結合列）のデータ型は，互いに変換できるようにします。（数値と数値，文字と文字が変換できます）。
4. 結合する列にナル値を持つ行は，どの行に対しても条件を満足しません。
5. 同じ列名の列がある表，又は WITH 句の導出問合せ式から問合せ名として導出される表を結合する場合は，その列を指定するときに表名，又は相関名を付けてその列を一意にして指定します。
6. 結合できる（FROM 句に指定できる）実表，FROM 句の導出表の延べ数は，最大 128 個です。

   また，一つの SQL 文中に指定できる実表の延べ数（副問合せで指定する実表の延べ数も含む）は最大 128 個です。相関名の延べ数は，最大 129 個です。

   1 トランザクションで同時にアクセスできる表数は，システム定義の pd_max_access_tables オペランドの値までです。pd_max_access_tables オペランドのデフォルト値は 64 のため，必要に応じて pd_max_access_tables オペランドの値を変更してください。pd_max_access_tables オペランドの値を増やすとメモリ所要量が増加します。詳細は，マニュアル「HiRDB システム定義」の pd_max_access_tables オペランドを参照してください。

   名前付きの導出表（ビュー表又は WITH 句の問合せ）を使用した場合，その名前付きの導出表一つに対する表の結合数は，ビュー定義文中，又は WITH 句中の導出問合せ式で指定している実表，及び FROM 句の導出表の延べ数になります。同様に，その名前付きの導出表一つに対する SQL 文中で指定

した表数は，ビュー定義文中，又は WITH 句中の導出問合せ式で指定した実表の延べ数になります。その名前付きの導出表一つに対する SQL 文中で指定した相関名数は，ビュー定義文中，又は WITH 句中の導出問合せ式で指定した相関名数になります。

また，ビュー定義文中，又は WITH 句中の導出問合せ式に集合演算を指定して導出した名前付きの導出表を，SQL 文中で指定し，かつ，その名前付きの導出表が内部導出表となる条件をどれも満たさない場合，表数，相関名数は次のようになります。

表数＝

（導出問合せ式中で指定した実表の延べ数）

＋（(導出問合せ式中の集合演算数＋ 1）

×（副問合せ中の実表の延べ数)）

相関名数＝

（導出問合せ式中で指定した相関名の延べ数）

＋（(導出問合せ式中の集合演算数＋ 1）

×（副問合せ中の相関名の延べ数)）

なお，ビュー定義文中の導出問合せ式の制限項目については，「CREATE ［PUBLIC］VIEW（ビュー定義，パブリックビュー定義)」を参照してください。また，WITH 句中の導出問合せ式の制限項目については，「問合せ式」を参照してください。また，名前付きの導出表が内部導出表となる条件は，「内部導出表」を参照してください。

7. 内部導出表を作成する問合せ指定では，内部導出表の作成対象となる一つの名前付きの導出表同士の結合は指定できません。内部導出表を作成する条件については，「内部導出表」を参照してください。

(6) GROUP BY 句，及び HAVING 句についての規則

1. ビュー定義中，又は WITH 句中の導出問合せ式の GROUP BY 句には，列指定の値式を指定してください。
2. 問合せ指定にウィンドウ関数を指定した場合，GROUP BY 句及び HAVING 句は指定できません。

(7) 留意事項

1. 次のどれかの条件を満たす場合，HiRDB が作業表を作成することがあります。
   - 複数の表を結合している
   - GROUP BY 句を指定している
   - 表一次子にビュー表又は問合せ名を指定し，内部導出表を作成している（内部導出表については，「内部導出表」を参照してください）

   このとき，作業表の行長によっては，上記の処理が制限されることがあります。作業表の行長については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(8) 指定例（動的 SELECT 文中で表式を指定した場合）

```
SELECT SUM(ZSURYO) FROM ZAIKO
    WHERE TANKA >= (SELECT AVG(TANKA)
    FROM ZAIKO)
```

(9) 使用例

表 A は商品の単価の表，表 B は受注のあった商品の受注量の表，表 C は先月受注のあった商品の受注量の表です。

＜表名：A＞

商品コード	単価
SCODE	TANKA
10	100
20	200
30	300
40	400
50	500
60	600

＜表名：B＞

商品コード	受注量
SCODE	JSURYO
10	10
40	40
50	50

＜表名：C＞

商品コード	先月受注量
SCODE	SENRYO
10	30
20	20
40	10
50	40

1. 単価が 200 より大きい商品の商品コード，単価，及び受注量を取得します。

```
SELECT A.SCODE,TANKA,JSURYO FROM A LEFT OUTER JOIN B
    ON A.SCODE = B.SCODE WHERE TANKA > 200
```

＜実行結果＞

SCODE	TANKA	JSURYO
30	300	
40	400	40
50	500	50
60	600	

注　受注のない商品の受注量（JSURYO）は，ナル値になります。

2. すべての商品コード，及び単価と，受注量が 40 以上の受注量を取得します。

```
SELECT A.SCODE,TANKA,JSURYO FROM A LEFT OUTER JOIN B
    ON A.SCODE = B.SCODE AND JSURYO >= 40
```

＜実行結果＞

SCODE	TANKA	JSURYO
10	100	
20	200	
30	300	
40	400	40
50	500	50
60	600	

注　受注のない商品の受注量（JSURYO）は，ナル値になります。

3. すべての商品コード，及び単価と，単価が 400 以上の商品の受注量を取得します。

```
SELECT A.SCODE,TANKA,JSURYO FROM A LEFT OUTER JOIN B
    ON A.SCODE = B.SCODE AND TANKA >= 400
```

＜実行結果＞

SCODE	TANKA	JSURYO
10	100	
20	200	
30	300	
40	400	40
50	500	50
60	600	

注　受注のない商品の受注量（JSURYO）は，ナル値になります。

4. 単価が 500 以下の商品について，商品コード及び単価と，40 以上の今月の受注量，30 以下の先月の受注量を取得します。

```
SELECT A.SCODE,A.TANKA,B.JSURYO,C.SENRYO
  FROM A LEFT OUTER JOIN B ON A.SCODE=B.SCODE AND B.JSURYO >= 40
         LEFT OUTER JOIN C ON A.SCODE=C.SCODE AND C.SENRYO <= 30
    WHERE A.TANKA <= 500
```

＜実行結果＞

SCODE	TANKA	JSURYO	SENRYO
10	100		30
20	200		20
30	300		
40	400	40	10
50	500	50	

注　単価が500より大きい商品については検索されません。
受注がないか，今月の受注量が40未満，又は先月の受注量が30を超える商品の受注量（JSURYO，SENRYO）は，ナル値になります。

5. すべての商品コード及び単価と，単価が 400 以下の商品の今月の受注量と，先月の受注量を取得します。

```
SELECT A.SCODE,A.TANKA,B.JSURYO,C.SENRYO
  FROM A LEFT OUTER JOIN B ON A.SCODE = B.SCODE
           AND A.TANKA <= 400
         LEFT OUTER JOIN C ON A.SCODE = C.SCODE
           AND A.TANKA <= 400
```

＜実行結果＞

SCODE	TANKA	JSURYO	SENRYO
10	100	10	30
20	200		20
30	300		
40	400	40	10
50	500		
60	600		

注　受注がないか，又は単価が400より大きい商品の受注量（JSURYO，SENRYO）は，
ナル値になります。

6. すべての商品コードと，各商品の今月の受注量の，先月の受注量に対する比率を求めます。ただし，今月又は先月に受注のなかった商品については，ナル値を出力します。

```
SELECT A.SCODE,100.0*B.JSURYO/C.SENRYO
  FROM A LEFT OUTER JOIN (B INNER JOIN C ON B.SCODE=C.SCODE)
           ON A.SCODE = B.SCODE
```

＜実行結果＞

SCODE	100.0*B.JSURYO/C.SENRYO
10	33.3333333333333333
20	
30	
40	400.0000000000000000
50	125.0000000000000000
60	

7. すべての商品コード及び単価と，今月の受注量と，先月の受注量から，今月と先月の売上を取得します。

```
SELECT SCODE,TANKA*JSURYO,TANKA*SENRYO
    FROM(SELECT A.SCODE,A.TANKA,B.JSURYO,C.SENRYO
           FROM A LEFT OUTER JOIN B ON A.SCODE = B.SCODE
                  LEFT OUTER JOIN C ON A.SCODE = C.SCODE)
    AS DT1(SCODE,TANKA,JSURYO,SENRYO)
```

＜実行結果＞

SCODE	TANKA*JSURYO	TANKA*SENRYO
10	1000	3000
20		4000
30		
40	16000	4000
50	25000	20000
60		

注　受注のない商品の売上(TANKA*JSURYO,TANKA*SENRYO)は，ナル値になります。

2.6 表参照

2.6.1 表参照の形式と規則

(1) 機能

検索する表，又は表の結合を指定します。表参照は表式中に指定します。

(2) 形式

```
表参照：：＝｛表一次子｜結合表｝
表一次子：：＝｛〔認可識別子.〕表識別子
                        〔IN （RDエリア名指定）〕
                                〔〔AS〕相関名〕 〔使用インデクスのSQL最適化指定〕
              ｜問合せ名 〔〔AS〕相関名 〕〔使用インデクスのSQL最適化指定〕
              ｜（結合表）
              ｜導出表〔〔AS〕相関名〔（導出列リスト）〕〕｝
導出表：：＝表副問合せ
導出列リスト：：＝列名リスト
列名リスト：：＝列名〔，列名〕…
結合表：：＝｛表参照〔 {INNER｜ {LEFT｜RIGHT}  〔OUTER〕｝〕 JOIN
              〔結合方式のSQL最適化指定〕
             表参照 ON 探索条件
           ｜表参照 CROSS JOIN 表一次子｝
```

(3) オペランド

- 表一次子：：＝｛〔認可識別子.〕表識別子
  〔IN （RD エリア名指定)〕
  〔〔AS〕相関名〕 〔使用インデクスの SQL 最適化指定〕
  ｜問合せ名〔〔AS〕相関名〕〔使用インデクスの SQL 最適化指定〕
  ｜（結合表）
  ｜導出表〔〔AS〕相関名〔(導出列リスト)〕〕｝

認可識別子

表の所有者の認可識別子を指定します。データディクショナリ表を検索する場合は，認可識別子に MASTER を指定します。

表識別子

検索する表の名称を指定します。

〔IN （RD エリア名指定)〕

IN

アクセス対象の RD エリアを指定します。指定できない場合を次に示します。

- 表識別子に指定した表が読み込み専用のビュー表の場合
- ビュー定義の導出問合せ式中の場合
- 表識別子に指定した表が一時表の場合

RD エリア名指定：：＝値指定

表識別子に指定した表を格納している RD エリアのうち，アクセスする RD エリアの名称を VARCHAR 型，CHAR 型，MVARCHAR 型，又は MCHAR 型の値指定で指定します。複数の RD エリア名を指定する場合はコンマ（,）で区切って指定してください。RD エリア名は重複して指定できません。重複して指定した場合はエラーとなります。値指定で指定する RD エリア名に許される文字については，「名前の指定」を参照してください。また，値指定で指定した RD エリア名の前後の空白は無視されます。RD エリア名を引用符（"）で囲んだ場合は，引用符（"）の外側の空白だけを無視します。

インナレプリカ機能を適用している RD エリアを指定する場合，オリジナル RD エリア名を指定してください。レプリカ RD エリアを対象とする場合は，カレント切り替えコマンド（pddbchg コマンド），又はクライアント環境定義の PDDBACCS オペランドでアクセス対象 RD エリアをレプリカ RD エリアに切り替えてください。

問合せ名

検索の対象となる WITH 句中の導出問合せ式から導出される表の名称を指定します。

〔AS〕相関名

同じ表を結合する場合，又は内側の副問合せで同じ表の列を参照する場合に，表同士を区別するために付ける名称を指定します。AS は省略できます。

一つの FROM 句中で同じ表，又は問合せ名を複数回指定する場合は，相関名を指定してそれぞれの表，及び問合せ名が一意に識別できるようにしてください。また，問合せ名と表名が同じ場合も，相関名を指定して一意に識別できるようにしてください。

相関名には一つの FROM 句中に指定したほかの相関名と同じ名称を指定できません。

相関名には，一つの FROM 句中に指定したその相関名を指定する表名，又は問合せ名以外の表識別子と同じ名称を指定できません。

該当する相関名を指定する表名，又は問合せ名と同一名称の相関名を指定した場合，その FROM 句中に指定した相関名指定のある表名，又は問合せ名は有効範囲がありません。

相関名の有効範囲は，導出表を介さないでその相関名を指定した表参照を FROM 句に含む問合せ指定，1 行 SELECT，及びそれらの内側の副問合せとします。

使用インデクスの SQL 最適化指定

使用インデクスの SQL 最適化指定については「SQL 最適化指定」を参照してください。

(結合表)

結合表の評価順序を指定する場合，結合表を括弧で囲んで指定します。括弧で囲まない場合は，最も左側の表参照から順に評価します。

- 結合表：：＝ {表参照〔{INNER ｜ {LEFT ｜ RIGHT} 〔OUTER〕}〕 JOIN
  〔結合方式の SQL 最適化指定〕

表参照 ON 探索条件
｜表参照 CROSS JOIN 表一次子｝

結合表には，内結合，外結合又は交差結合によって導き出される表を指定します。

内結合では，右側の表と左側の表から一つずつ行を取り出して組み合わせた行のうち，探索条件を満たす行が検索されます。外結合では，外表のすべての行と，探索条件を満たす内表の行が検索されます。交差結合では，右側の表と左側の表から一つずつ行を取り出して組み合わせた行が検索されます。

表参照〔{INNER ｜ {LEFT ｜ RIGHT} 〔OUTER〕}〕 JOIN 表参照

外表と内表の二つの表を突き合わせ処理する（内結合又は外結合する）場合に指定します。

「表参照 1 〔INNER〕 JOIN 表参照 2」の場合，「〔INNER〕 JOIN」の左側に指定する表参照の結果の表（表参照 1）が外表となり，右側に指定する表参照の結果の表（表参照 2）が内表となります。このときの突き合わせの結果のうち，探索条件を満たす行が導き出されます。

「表参照 1 LEFT 〔OUTER〕 JOIN 表参照 2」の場合，「LEFT 〔OUTER〕 JOIN」の左側に指定する表参照の結果の表（表参照 1）が外表となり，右側に指定する表参照の結果の表（表参照 2）が内表となります。外表の行は，突き合わせの結果の真偽に関係なくすべて導き出されます。また，内表の行は，探索条件を満たす行だけ導き出されます。

「表参照 1 RIGHT 〔OUTER〕 JOIN 表参照 2」の場合，「RIGHT 〔OUTER〕 JOIN」の右側に指定する表参照の結果の表（表参照 2）が外表となり，左側に指定する表参照の結果の表（表参照 1）が内表となります。外表の行は，突き合わせの結果の真偽に関係なくすべて導き出されます。また，内表の行は，探索条件を満たす行だけ導き出されます。

結合方式の SQL 最適化指定

結合方式の SQL 最適化指定については「SQL 最適化指定」を参照してください。

ON 探索条件

内結合又は外結合の結合条件を指定します。

探索条件には，外表又は内表の列が指定できます。

また，副問合せ中の探索条件の列指定は，副問合せの外側で指定した表の列を参照できます。問合せが入れ子の場合，内側の問合せから外側の問合せで指定している表，又は列を参照することを外への参照といいます。

探索条件中の列指定を表名で修飾する場合，相関名を指定した表の列に対しては相関名で修飾してください。

副問合せ中の ON 探索条件に COUNT（*），COUNT_FLOAT（*），及びウィンドウ関数は指定できません。また，HAVING 句に属する FROM 句の ON 探索条件中にだけ，ON 探索条件に集合関数を指定できます。HAVING 句に属する FROM 句の ON 探索条件中に集合関数を指定した場合，その集合関数の列指定は，HAVING 句に先行する FROM 句に指定した表を参照（外への参照）してください。

表参照 CROSS JOIN 表一次子

左側に指定した表参照と右側に指定した表一次子との直積を求める場合に指定します。

- 導出表〔〔AS〕相関名〔(導出列リスト)〕〕

表副問合せを指定します。この問合せで導き出された表を FROM 句の導出表と呼びます。表副問合せのｎ番目の列が導出表のｎ番目の列となります。

導出表を含む問合せ指定は読み込み専用となります。

導出表を指定する場合の留意事項を次に示します。

- 表参照に導出表を指定した場合，導出表に対する相関名を指定してください。ただし，最も外側の問合せに次に示す形式を指定した場合だけ，〔AS〕相関名〔(導出列リスト)〕を省略できます。
  - SELECT COUNT(*) FROM 導出表
  - SELECT COUNT_FLOAT(*) FROM 導出表
- 値式を選択式に指定した表副問合せの結果導出される表の構成列に次の構造，又はデータ型を指定できません。
  - 繰返し構造
  - BLOB
  - 32,001 バイト以上の BINARY
  - BOOLEAN
  - 抽象データ型
- 導出表に対して行インタフェース（ROW）は指定できません。
- 導出表中にはウィンドウ関数を指定できません。
- 導出表中には予備列を指定できません。

〔AS〕相関名

導出表の名称を指定します。

〔(導出列リスト)〕

導出表の各列の列名を指定します。

導出列リストを省略した場合，導出表中の最も外側の問合せの結果導出される列名が導出表の列名となります。したがって，選択式の列指定（AS 句を指定している場合 AS 句の列名）が導出表の列名となります。それ以外は，HiRDB が SQL 文中のどの列名とも異なる列名を仮定します。

導出列リストを省略した場合，表副問合せの結果，導出される列名が重複してはいけません。

同じ列名は導出列リスト中に指定できません。

導出列リストを指定する場合，導出列リスト中の列名の数は，導出表によって導出された表の列数と同じにしてください。

導出列リスト中又は表副問合せによって導出される列数は，30,000 以下にしてください。

(4) 共通規則

RD エリア名指定を指定した場合，分割数が表の分割数と異なるインデクスは利用できません。RD エリア名指定を指定する問合せのためにインデクスを定義する場合は，分割数が表の分割数と等しいインデクスを定義してください。

(5) 結合表についての規則

1. 外結合を指定した場合，外結合の結果の内表の列にはナル値を指定できるようになります。
2. 「ON 探索条件」に指定できる列は，外表若しくは内表の列，又は外への参照列です。
3. 結合表は，読み込み専用の表とします。
4. 内結合によって導き出される表は，外表と内表の ON 探索条件を満たす行を連結したもので構成されます。
5. 外結合によって導き出される表は，外表と内表の ON 探索条件を満たす行を連結したものと，外表の ON 探索条件を満たさない行に内表の列数分のナル値を追加したもので構成されます。
6. 結合表を含む問合せ指定の場合，外結合の内表から導き出される選択式には，ROW は指定できません。

(6) FROM 句の導出表についての規則

1. FROM 句導出表の選択式には，ROW は指定できません。
2. FROM 句の導出表の選択式には，添字のない繰返し列は指定できません。
3. FROM 句の導出表には，予備列は指定できません。
4. FROM 句の導出表の選択式には，WRITE 指定及び GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
5. 〔AS〕相関名〔(導出列リスト)〕を省略できる形式（SELECT COUNT(*) FROM 導出表又は SELECT COUNT_FLOAT(*) FROM 導出表）は，WITH 句問合せ，ビュー定義，集合演算，INSERT 文，及び副問合せに指定できません。
6. FROM 句の導出表を指定すると，内部的に作業表を作成します。ただし，FROM 句の導出表に関する次のすべての条件を満たす場合は作業表を作成しません。
   - FROM 句の導出表の条件
     - ・DISTINCT を指定していない
     - ・選択式に副問合せを指定していない
     - ・選択式に BINARY 型の列を指定していない
     - ・選択式にコンポネント指定がない
     - ・FROM 句に 1 表だけを指定している
     - ・GROUP BY 句，HAVING 句，及び集合関数を指定していない
     - ・集合演算を指定していない
     - ・外への参照がある副問合せを指定していない

- FROM 句の導出表の外側の問合せの条件

  ・導出表を含む問合せ指定で，外結合の内表として使用していない

  ・導出表の選択式に列指定以外の値式がある場合，その値式から導出される列指定を，導出表を含む問合せ指定の GROUP BY 句に指定していない

  ・ORDER BY 句に指定している導出表のすべての列指定を，選択式に指定している

- SQL 文全体に対する条件

  ・プラグイン関数を指定していない

  ・SQL/XML スカラ関数，SQL/XML 述語，及び SQL/XML 集合関数を指定していない

(7) 留意事項

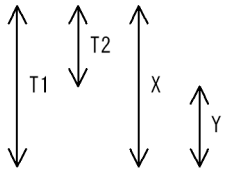
1. FROM 句中で指定した相関名，相関名なしで指定した表名，又は問合せ名の有効範囲は，その FROM 句を含む最も内側の副問合せ，問合せ指定，又は 1 行 SELECT 文中です。なお，それらの内側の副問合せも含みます。
2. 副問合せ中の FROM 句で相関名を指定すると，表名，又は問合せ名は，有効範囲がありません。
3. 最も外側の問合せ指定，1 行 SELECT 文に直接含まれる FROM 句で相関名を指定した場合の表名，又は問合せ名の有効範囲は，内側の副問合せを除いて，問合せ指定，又は 1 行 SELECT 文中です。ただし，FROM 句の表名，又は問合せ名に対して同一名称の相関名を指定した場合，その FROM 句中に指定した相関名指定のある表名，又は問合せ名は有効範囲がありません。

   相関名，表名，又は問合せ名の有効範囲の例を次の図に示します。

   副問合せ中の FROM 句に指定した表名，又は問合せ名に相関名がある場合，その表名，又は問合せ名は有効範囲がありません（図中の T3)。また，自分の問合せで直接指定した相関名のある表（図中の T2 X）を内側の問合せが参照する場合，列名は表名（T2）ではなく，相関名（X）で修飾してください。

図 2-1　相関名，表名，又は問合せ名の有効範囲の例

```
SELECT T1.C1,T2.C1 FROM T1,T2 X
   WHERE T1.C2 =
      ( SELECT Y.C2,FROM T3 Y
        WHERE  Y.C1 = X.C2     )
```

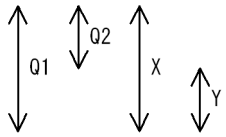


（凡例）

：有効範囲

T1，T2，T3：表名
C1，C2　　：列名
X，Y　　：相関名

```
WITH Q1 (QC1,QC2) AS (SELECT C1,C2 FROM T1),
WITH Q2 (QC1,QC2) AS (SELECT C1,C2 FROM T2),
WITH Q3 (QC1,QC2) AS (SELECT C1,C2 FROM T3)

   SELECT Q1.QC1,Q2.QC1 FROM Q1,Q2 X
      WHERE Q1.QC2 =
        ( SELECT Y.QC2,FROM Q3 Y
      WHERE  Y.QC1 = X.QC2     )
```



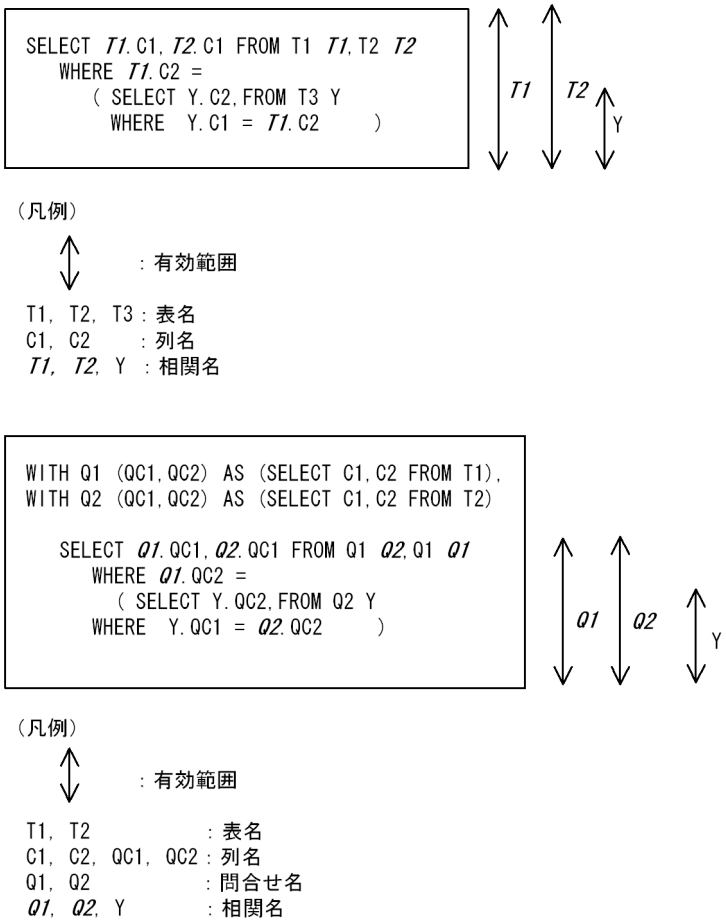
（凡例）

：有効範囲

T1，T2，T3：表名
C1，C2，QC1，QC2：列名
Q1，Q2，Q3　　：問合せ名
X，Y　　：相関名

同じ名称の有効な相関名，表名，又は問合せ名が複数ある範囲でその名称を参照した場合，最も局所的な有効範囲を持つものが指定されます。このとき，外側の表を参照するには，FROM 句で異なる相関名を指定し，その名称で参照してください。

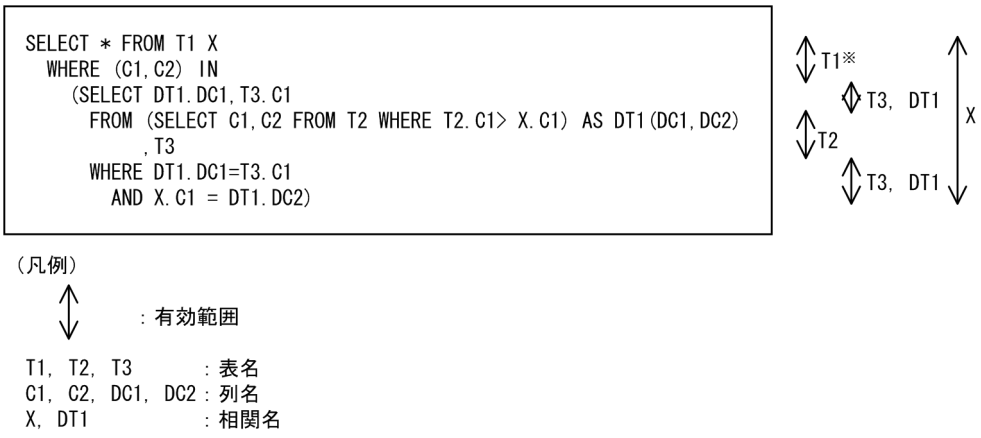
図 2-2　相関名，表名，又は問合せ名の有効範囲の例（FROM 句の表名，又は問合せ名に対して同一名称の相関名を指定した場合）



FROM 句の表名，又は問合せ名に対して同一名称の相関名を指定した場合，その FROM 句中に指定した相関名指定のある表名，又は問合せ名は有効範囲がありません。

4. FROM の導出表を指定した場合の相関名，表名，有効範囲の例を次の図に示します。

図 2-3　FROM の導出表を指定した場合の相関名，表名，有効範囲の例



注※ T1は，ISO SQLでは有効ではないが，旧バージョンの互換性上主問合せ内でだけ有効

2.7 探索条件

2.7.1 機能

- 外結合を指定しない場合は，SQL 中に指定した探索条件によって論理演算を実行し，その結果が真のものだけを検索の対象にします。外結合を指定する場合は，ON 探索条件が真にならない外表の行も検索の対象にします。
- 探索条件は次の箇所に指定します。
  - FROM 句中の ON 探索条件
  - WHERE 句
  - HAVING 句
  - 探索 CASE 式の WHEN
  - トリガ動作条件の WHEN
  - IF 文
  - WHILE 文
  - 検査制約定義

形式

```
探索条件：：＝ {〔NOT〕 {（探索条件）｜述語}
                ｜探索条件 OR {（探索条件）｜述語}
                ｜探索条件 AND {（探索条件）｜述語} }
述語：：＝ {NULL述語｜IN述語｜LIKE述語｜XLIKE述語｜SIMILAR述語｜BETWEEN述語
           ｜比較述語｜限定述語｜EXISTS述語｜論理述語｜構造化繰返し述語}
```

2.7.2 論理演算

次の規則に従って論理演算をします。

- 論理演算の評価順序は，括弧内，NOT，AND，OR の順
- 論理演算の最大ネスト数は，255 個
- 論理演算のネスト数は，論理演算子 AND，又は OR（NOT は除く）の評価順序を表す括弧を省略しないで指定した場合の括弧のネスト数

名前付き導出表を問合せ指定に指定して，かつ指定した名前付き導出表が内部導出表を作成しない場合，その名前付き導出表を導出した問合せの探索条件が AND 論理演算で結合されることで，論理演算の最大ネスト数を超えることがあります。また，論理演算に 16,000 個以上の述語を指定すると SQL エラーになることがあります。

2.7.3　述語の結果

各論理演算をした場合の述語の結果を，次の図に示します。

なお，NULL 述語及び論理述語以外では，ナル値を含む述語の結果は不定になります。不定となった場合，検索の対象にはなりません。

図 2-4　論理演算をした場合の述語の結果

（AND論理演算）

AND	真	偽	不定
真	真	偽	不定
偽	偽	偽	偽
不定	不定	偽	不定

（OR論理演算）

OR	真	偽	不定
真	真	真	真
偽	真	偽	不定
不定	真	不定	不定

（NOT論理演算）

NOT	結果
真	偽
偽	真
不定	不定

2.7.4　述語の共通規則

1. 比較できるデータについては，「変換（代入，比較）できるデータ型」を参照してください。ただし，各国文字データの比較対象として文字列定数を指定した場合は，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字データは長さチェックだけで，文字コードのチェックはされません。
2. 文字データ同士の比較で，文字集合が異なる場合は比較できません。ただし，次の表に示す条件を満たす場合は，比較対象の文字集合に変換して比較できます。

表 2-1　異なる文字集合間の比較ができる条件（EBCDIK と既定文字集合（SJIS）の場合）

EBCDIK の値式	既定文字集合（SJIS）の値式		
	文字列定数	埋込み変数，？パラメタ	文字列定数，埋込み変数，？パラメタ以外
文字列定数	－	－	－
埋込み変数，？パラメタ	×	×	×
文字列定数，埋込み変数，？パラメタ以外	○	○	×

（凡例）

○：文字集合（EBCDIK）の文字コードに変換して比較します。

×：比較できません。

－：サポートしていません。

表 2-2　異なる文字集合間の比較ができる条件（UTF16 と既定文字集合（UTF-8）の場合）

UTF16 の値式	既定文字集合（UTF-8）※の値式		
	文字列定数	埋込み変数，？パラメタ	文字列定数，埋込み変数，？パラメタ以外
文字列定数	－	－	－
埋込み変数，？パラメタ	●	×	●
文字列定数，埋込み変数，？パラメタ以外	○	○	×

注※

混在文字データを含みます。

（凡例）

○：文字集合（UTF16）の文字コードに変換して比較します。

●：文字集合（UTF-8）の文字コードに変換して比較します。

×：比較できません。

－：サポートしていません。

3. 固定長データと固定長データ，又は固定長データと可変長データを比較する場合は，短い方のデータの右側に比較で使用する文字集合の空白を埋めて，文字列長を同じにしてから比較します。

4. 可変長文字データ同士の比較で，比較するデータの長さが異なる場合は，短い方のデータの長さ分だけ左側から比較します。このとき，結果が等しければ更に文字列長を比較します。

5. 数データ同士の比較で，比較するデータのデータ型が異なる場合は，範囲の広い方のデータ型で比較します。範囲の広さを次に示します。

   FLOAT ＞ SMALLFLT ＞ DECIMAL ＞ INTEGER ＞ SMALLINT

   ただし，比較対象の一方が SMALLFLT 型の場合，範囲の広さに関係なく，FLOAT 型で比較します。

6. 時刻印データ同士の比較で，小数秒精度が異なるデータ同士を比較する場合，精度を高い方に合わせて，拡張した小数秒部分に 0 を補います。

7. BINARY 型同士の比較で，比較するデータの長さが異なる場合は，短い方のデータの長さ分だけを左側から比較します。このとき，結果が等しければ更にデータ長を比較します。

8. 予備列は指定できません。

2.7.5　述語

(1)　比較述語

形式

```
行値構成子　{＝｜＜＞｜^＝｜!＝｜＜｜＜＝｜＞｜＞＝} 行値構成子
```

述語が真となる場合

- 行値構成子要素が一つの場合

  左右の行値構成子要素が，比較条件を満たす場合に真となります。

  なお，行値構成子要素のどれかがナル値の場合は不定となります。

- 行値構成子要素が二つ以上の場合

  (＝)

  左右の行値構成子中の対応する要素間の関係が，すべて「＝」である場合に真となります。

  なお，「＜＞」が成立する要素の組み合わせが一つ以上存在するときは偽となります。

  「＜＞」が成立する要素の組み合わせがない場合で，比較する要素にナル値が一つ以上存在するときは不定となります。

  ・(真となる例)

  (1,2,3)＝(1,2,3)

  ('A','B','C')＝('A','B','C')

  (＜＞｜^＝｜!＝)

  左右の行値構成子中の対応する要素を比較し，少なくとも一つ「＜＞」が成立する組み合わせが存在する場合に真となります。

  なお，対応する要素間の関係がすべて「＝」である場合は偽となります。

  「＜＞」が成立する組み合わせが一つも存在しない場合で，比較する要素にナル値が一つ以上存在するときは不定となります。

  ・(真となる例)

  (1,2,3)＜＞(1,5,3)

  ('A','B','C')＜＞('C','A','B')

  (＜)

  左右の行値構成子中の対応する要素を左から順に，「＝」が成立する間比較をします。「＝」が成立しない最初の要素間に「＜」の関係が成立する場合に真となります。

  なお，「＝」が成立しない最初の要素間の関係が「＞」である場合，及び対応する要素間のすべてに「＝」が成立する場合は偽となります。

  「＝」が成立しない最初の要素にナル値が存在する場合は不定となります。

  ・(真となる例)

  (1,2,3)＜(3,1,2)

  1 番目の要素間の関係が 1 ＜ 3 であるため，真となります。

  ('A','B','C','D')＜('A','B','E','A')

  左から順に比較し，「＝」の成立しない最初の要素間の関係が'C'＜'E'であるため，真となります。

  (＞)

  左右の行値構成子中の対応する要素を左から順に，「＝」が成立する間比較をします。「＝」が成立しない最初の要素間に「＞」の関係が成立する場合に真となります。

なお，「＝」が成立しない最初の要素間の関係が「＜」である場合，又は対応する要素間のすべてに「＝」が成立する場合は偽となります。

「＝」が成立しない最初の要素にナル値が存在する場合は不定となります。

・(真となる例)

(1,2,3)＞(1,1,5)

左から順に比較し，「＝」が成立しない最初の要素間の関係が 2 ＞ 1 であるため，真となります。

('A','A',C')＞('A','A','A')

左から順に比較し，「＝」が成立しない最初の要素間の関係が'C'＞'A'であるため，真となります。

(＜＝)

左右の行値構成子中の対応する要素を左から順に，「＝」が成立する間比較をします。「＝」が成立しない最初の要素間に「＜」の関係が成立する場合，又は対応する要素間のすべてに「＝」の関係が成立する場合に真となります。

なお，「＝」が成立しない最初の要素間の関係が「＞」である場合は偽となります。

「＝」が成立しない最初の要素にナル値が存在する場合は不定となります。

(＞＝)

左右の行値構成子中の対応する要素を左から順に，「＝」が成立する間比較をします。「＝」が成立しない最初の要素間に「＞」の関係が成立する場合，又は対応する要素間のすべてに「＝」の関係が成立する場合に真となります。

なお，「＝」が成立しない最初の要素間の関係が「＜」である場合は偽となります。

「＝」が成立しない最初の要素にナル値が存在する場合は不定となります。

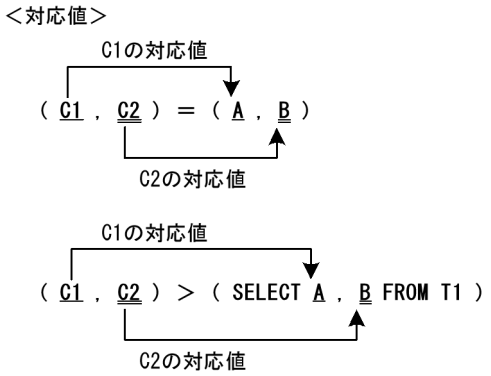
各比較述語は，論理演算を用いた形に展開できます。論理演算を用いて展開した形式を次に示します。

演算子	行値構成子を用いた記述	論理演算を用いた記述
＝	(Rx1,Rx2,…,Rxn)＝(Ry1,Ry2,…,Ryn)	Rx1=Ry1 AND Rx2=Ry2 AND…AND Rxn=Ryn
＜＞	(Rx1,Rx2,…,Rxn)＜＞(Ry1,Ry2,…,Ryn)	Rx1<>Ry1 OR Rx2<>Ry2 OR … OR Rxn<>Ryn
＜	(Rx1,Rx2,Rx3,…,Rxn)＜(Ry1,Ry2,Ry3,…,Ryn)	Rx1<Ry1 OR (Rx1=Ry1 AND Rx2<Ry2) OR (Rx1=Ry1 AND Rx2=Ry2 AND Rx3<Ry3) OR … OR (Rx1=Ry1 AND Rx2=Ry2 AND Rx3=Ry3 AND … AND Rxn-1=Ryn-1 AND Rxn<Ryn)
＞	(Rx1,Rx2,Rx3,…,Rxn)＞(Ry1,Ry2,Ry3,…,Ryn)	Rx1>Ry1 OR (Rx1=Ry1 AND Rx2>Ry2) OR (Rx1=Ry1 AND Rx2=Ry2 AND Rx3>Ry3) OR … OR (Rx1=Ry1 AND Rx2=Ry2 AND Rx3=Ry3 AND … AND Rxn-1=Ryn-1 AND Rxn>Ryn)

規則

1. 比較演算子（＝，＜＞，^＝，!＝，＜，＜＝，＞，＞＝）の両側に定数だけの行値構成子を指定してもかまいません。

2. 演算結果のデータ型が，次のデータ型となる値を指定できません。
   - BLOB
   - 定義長が 32,001 バイト以上の BINARY
   - BOOLEAN
   - 抽象データ型
3. 行値構成子については，「行値構成子」を参照してください。
4. 左右の行値構成子中で，同じ位置にある行値構成子要素を対応値とします。対応値のデータ型は比較可能なデータ型にしてください。



5. 次に示す箇所の比較述語には，副問合せは指定できません。
   - IF 文の探索条件
   - WHILE 文の探索条件
   - CREATE TRIGGER の WHEN 探索条件（トリガ動作条件）
6. 繰返し列を指定する場合は，添字を指定してください。繰返し列を添字付きで指定して，その要素が条件を満たしている場合，対応値との比較結果は真となります。
7. 繰返し列の添字として ANY を指定できます。ANY を指定した場合は，その列の少なくとも一つの要素が条件を満たしていれば，比較結果は真となります。比較結果が真でなく，その列の少なくとも一つの要素に対して指定した条件が不定ならば，その比較結果は不定になります。比較結果が真又は不定でない場合，偽となります。
8. 繰返し列を添字付きで指定した場合，その要素がないと対応値との比較結果は不定になります。
9. 比較対象となる行値構成子は，結果の列数を同じにしてください。
10. うるう秒は 1 分より小さい値として扱われます。

    （例）

    ’00:00:61’ ＜’00:01:00’

(2) NULL 述語

形式

```
値式 IS 〔NOT〕 NULL
```

述語が真となる場合

指定した値式の値がナル値である行に対して，NULL 述語は真になります。NOT を指定した場合は，ナル値でない行に対して真になります。ナル値については，「ナル値」を参照してください。

規則

1. 次のデータ型の値式は，指定できません。
   - BLOB（？パラメタ，又は埋込み変数を単独で指定した場合を除く）
   - 定義長が 32,001 バイト以上の BINARY（？パラメタ，又は埋込み変数を単独で指定した場合を除く）
   - BOOLEAN
2. 繰返し列を添字なしで指定した場合，その列に要素がないとき（NOT 指定の場合は要素が一つ以上あるとき）に，NULL 述語は真となります。列のすべての要素がナル値であっても，NULL 述語は真とはなりません。
3. 繰返し列を添字付きで指定した場合，指定した要素がナル値ならば NULL 述語は真となります。
4. 繰返し列を添字付きで指定した場合，その要素がないと NULL 述語は不定になります。
5. 繰返し列を添字付きで指定する場合は，添字として ANY を指定できます。ANY を指定した場合は，その列の少なくとも一つの要素が条件を満たしていれば，NULL 述語は真となります。

留意事項

- オーバフローエラー抑止が設定されている場合，値式の演算結果がオーバフローによってナル値となった場合にも，NULL 述語は真となります。

(3) IN 述語

形式

```
行値構成子〔IS〕 〔NOT〕 IN
 {（行値構成子〔, 行値構成子〕…）|
  〈表副問合せ〉
  （〔副問合せ実行方式のSQL最適化指定〕
   SELECT〔 {ALL|DISTINCT}〕 {選択式|*}
    〈表 式〉
     FROM 表参照〔, 表参照〕…
      〔WHERE 探索条件〕
      〔GROUP BY 値式〔, 値式〕…〕
      〔HAVING 探索条件〕）}
```

述語が真となる場合

次の条件のどちらかを満たしている場合，IN 述語は真となります。

- 左側の行値構成子が，右側の任意の行値構成子と一致する場合。
- 左側の行値構成子が，表副問合せの任意の結果行と一致する場合。

NOT を指定した場合は，左側の行値構成子が，右側に指定したすべての行値構成子又は表副問合せのすべての結果行と一致しない行に対して，IN 述語は真となります。

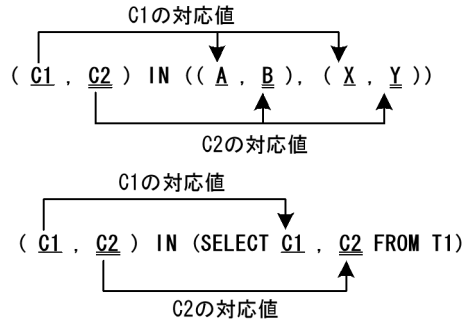
規則

1. 行値構成子又は表副問合せの結果の各データ項目が，次のデータ型となる値を指定できません。
   - BLOB
   - 定義長が 32,001 バイト以上の BINARY
   - BOOLEAN
   - 抽象データ型
2. 右側の行値構成子は，最大 30,000 個指定できます。
3. 右側が表副問合せでない IN 述語の左辺には，値指定だけの行値構成子を指定できません。
4. 表副問合せについては，「副問合せ」を参照してください。
5. IN 述語は，限定述語と同じ意味を表すものがあります。同じ意味を表す述語を次に示します。

IN 述語	限定述語
行値構成子 IN 表副問合せ	行値構成子 ＝ ANY 表副問合せ 又は 行値構成子 ＝ SOME 表副問合せ
行値構成子 NOT IN 表副問合せ	行値構成子 <> ALL 表副問合せ

6. 表副問合せの結果が空集合の場合，IN 述語の結果は偽になります。ただし，NOT を指定した場合は真になります。
7. 次に示す箇所の IN 述語には，表副問合せは指定できません。
   - IF 文の探索条件
   - WHILE 文の探索条件
   - CREATE TRIGGER の WHEN 探索条件（トリガ動作条件）
8. 各行値構成子中で，同じ位置にある行値構成子要素及び表副問合せの選択式を対応値とします。対応値のデータ型は，変換，又は比較できるデータ型にしてください。ただし，左側に指定した行値構成子中の行値構成子要素の結果のデータ型が各国文字データで，その対応値に文字列定数を指定した場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードはチェックされないで，文字データの長さだけがチェックされます。

＜対応値＞



9. 繰返し列を指定する場合は，添字を指定してください。繰返し列を添字付きで指定して，その要素が条件を満たしている場合，IN 述語は真となります。
10. 繰返し列の添字として ANY を指定できます。ANY を指定した場合は，その列の少なくとも一つの要素が条件を満たしていれば，IN 述語は真となります。IN 述語が真でなく，その列の少なくとも一つの要素に対して指定した条件が不定ならば，その IN 述語は不定になります。IN 述語が真又は不定でない場合，偽となります。
11. 右側の行値構成子には，添字に ANY を指定した繰返し列を指定できません。
12. 繰返し列を添字付きで指定した場合，その要素がないと IN 述語は不定になります。
13. 行値構成子の結果がナル値の場合，その対応値の比較結果は不定となります。
14. 比較対象となる行値構成子は，結果の列数を同じにしてください。

留意事項

1. IN 述語に表副問合せを指定した場合，HiRDB が作業表を作成することがあります。このとき，作業表の行長によっては，IN 述語の副問合せの処理が制限されることがあります。作業表の行長については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(4) LIKE 述語

形式

```
値式 〔NOT〕 LIKE パターン文字列 〔ESCAPE エスケープ文字〕
```

述語が真となる場合

指定した値式の値が，パターン文字列の表すパターンと一致する行に対して，LIKE 述語は真になります。NOT を指定した場合は，パターン文字列の表すパターンと一致しない行に対して，述語は真になります。

(値式)

1. 値式には，文字列のパターン比較の対象となる値式を指定します。ただし，SQL 変数，及び SQL パラメタ以外の値指定だけの値式は指定できません。
2. 値式及びパターン文字列に指定できるデータ型は，文字列データ，各国文字列データ，混在文字列データ，又は定義長が 32,000 バイト以下の BINARY です。

3. 繰返し列を指定する場合は，添字を指定してください。繰返し列を添字付きで指定して，その要素が条件を満たしている場合，LIKE 述語は真となります。

4. 繰返し列の添字として ANY を指定できます。ANY を指定した場合は，その列の少なくとも一つの要素が条件を満たしていれば，LIKE 述語は真となります。LIKE 述語が真でなく，その列の少なくとも一つの要素に対して指定した条件が不定ならば，その LIKE 述語は不定になります。LIKE 述語が真又は不定でない場合，偽となります。

5. 繰返し列を添字付きで指定した場合，その要素がないと LIKE 述語は不定になります。

(パターン文字列)

1. パターン文字列には，値指定を指定します。

2. 指定できるパターン文字列のデータ型は，値式に指定できるデータ型と同じです。

3. 値式のデータ型とパターン文字列のデータ型の組み合わせは，次のとおりです。

値式のデータ型		パターン文字列，又はエスケープ文字列のデータ型					
		文字列データ			各国文字列データ	混在文字列データ	バイナリデータ
		既定文字集合	EBCDIK	UTF16			
文字列データ	既定文字集合	○	×	×	×	○	×
	EBCDIK	△※3	○	×		×	
	UTF16	△※3	×	○			
各国文字列データ		△※1	×	×	○	×	×
混在文字列データ		○	×	×	×	○	×
バイナリデータ		△※2	×	×	×	×	○

(凡例)

○：指定できます。

△：制限付きで指定できます。

×：指定できません。

注※1

パターン文字列には文字列定数だけ指定できます。この場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードはチェックされないで，文字データの長さだけがチェックされます。特殊文字を指定する場合は，必ず全角文字で指定してください。

注※2

16 進文字列定数だけ指定できます。

注※3

文字列定数，埋込み変数，又は？パラメタだけ指定できます。

4. 値式，パターン文字列，エスケープ文字列の文字集合は，同じにしてください。ただし，パターン文字列，エスケープ文字列が次に示す場合は，値式の文字集合に変換されるので比較できます。

・文字列定数

・埋込み変数（既定文字集合）

・？パラメタ

5. パターン文字列中の特殊文字には，__（下線），%（パーセント），及びエスケープ文字があります。バイナリデータでの特殊文字は，__，%，及びエスケープ文字を表すコードを指定してください。パターン文字列中の特殊文字__と%の意味を次の表に示します。

表 2-3　パターン文字列中での特殊文字の意味（LIKE 述語）

特殊文字	項目指定のデータ型		特殊文字の意味	特殊文字の表現方法
__（下線）	文字列データ	既定文字集合※1	任意の 1 文字	半角文字の_（下線）
		EBCDIK※1		
		UTF16		
	各国文字列データ			全角文字の__（下線）
	混在文字列データ			半角文字の_（下線）
	バイナリデータ		任意の 1 バイト	5f（下線を表すコード）※3
%（パーセント）	文字列データ	既定文字集合※1	任意の長さ（0 文字以上）の任意の文字列	半角文字の%（パーセント）
		EBCDIK※1		
		UTF16		
	各国文字列データ※2			全角文字の%（パーセント）
	混在文字列データ			半角文字の%（パーセント）
	バイナリデータ		任意の長さ（0 バイト以上）の任意のバイト列	25（パーセントを表すコード）※3

注※1

すべての文字，又はデータ値を 1 バイト文字として扱います。

注※2

すべての文字，又はデータ値を 2 バイト文字として扱います。

注※3

バイナリデータで特殊文字を指定する場合は，HiRDB のセットアップ時に指定した文字コードで，かつ特殊文字（__又は%）を表すコードを指定してください。

6. パターン文字列中に%を含まない場合，列のデータとパターン文字列の長さが異なる場合は，この述語は真になりません。

7. 値式で指定した文字列とパターン文字列が，可変長データ（VARCHAR，NVARCHAR，MVARCHAR，又は BINARY）の場合は，値式のデータとパターン文字列データとの比較のほかに，データ長も比較します。

8. パターン文字列として埋込み変数，SQL 変数，又は SQL パラメタを指定する場合，次のことに注意してください。パターン文字列を固定長の埋込み変数，SQL 変数，又は SQL パラメタにした場合に変数の長さより短いパターン文字列を設定すると，変数の後ろに空白が入ったり，残っていた不当文字が値として設定されてしまう場合があります。このようなパターン文字列で検索すると，後ろに空白，又は不当文字と同じ値がないデータは検索されません。したがって，固定長の変数をパターン文字列として使用する場合は，後ろにすべて%を設定する必要があります。

（例）

変数にパターン文字列として'AB%'，及び'AB%%'を設定した場合，文字列データが'ABCD'のときの比較を次に示します。

変数のデータ型	パターン文字列	文字列データ	比較結果
可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）	'AB%'	'ABCD'	一致します
	'AB%%'	'ABCD'	一致します
固定長文字列（4 バイト）	'AB%△'	'ABCD'	一致しません
	'AB%%'	'ABCD'	一致します

（例）

変数にパターン文字列として X'52454425'を設定した場合，バイナリデータが 52454452554d のときの比較を次に示します。

変数のデータ型	パターン文字列	バイナリデータ	比較結果
BINARY 型	X'52454425'	52454452554d	一致します
	X'5245442525'	52454452554d	一致します

パターン文字列の例

LIKE 述語の代表的なパターン文字列の例を次の表に示します。

表 2-4　LIKE 述語の代表的なパターン文字列の例

項　目	パターン文字列	意　味	例	
			パターン文字列	パターンの一致する文字列
前方一致	nnn%	文字列の先頭部分が nnn です。	'ACT%'	ACT，ACTOR，ACTION など ACT で始まる文字列
後方一致※1	%nnn	文字列の最後の部分が nnn です。	'%ING'	ING，BEING，HAVING など ING で終わる文字列
任意一致	%nnn%	文字列中の任意の部分に nnn を含みます。	N'%日%'	日，日立，昨日，本日中など日を含む文字列※2
完全一致	nnn	文字列が nnn と同じです。	'EQUAL'	EQUAL
部分一致	_…_nnn_…_	文字列中の特定の部分が nnn と同じで，ほかの部分には任意の文字があります。	'_I_'	BIT，HIT，KIT など 3 文字で 2 文字目が I の文字列

項　目	パターン文字列	意　味	例	
			パターン文字列	パターンの一致する文字列
その他	nnn%mmm	文字列の先頭部分が nnn で，最後の部分が mmm です。	'O%N'	ON，OWN，ORIGIN など O で始まり N で終わる文字列
	%nnn%mmm%	文字列中の任意の部分に nnn を含み，その部分より後の任意の部分に mmm を含みます。	'%O%N%'	ON，ONE，DOWN，COUNT など O を含み，その部分より後に N を含む文字列
	nnn_…_mmm%	先頭部分が nnn で始まり，後方に mmm がある文字列です。	'CO_ _ECT%'	CORRECT，CONNECTOR，CONNECTION など CO で始まり 5 文字目から 7 文字目が ECT の文字列

注

nnn，mmm は，%，_を含まない任意の文字列

注※1

空白も比較対象の文字として扱うため，後方に空白のあるデータと比較した場合，結果は偽になります。

注※2

各国文字列中での特殊文字（%，_）は，各国文字の「%」，「_」を使用します。

エスケープ文字

パターン文字列中に __（下線）や%（パーセント）を記述すると，無条件に特殊文字とみなされ，通常の文字として扱えません。特殊文字を通常の文字として扱いたい場合は，エスケープ文字を指定します。キーワード ESCAPE の後に，任意の 1 文字（エスケープ文字）を指定しておくと，パターン文字列中に記述したエスケープ文字の直後の特殊文字を通常の文字として扱えます。

(例 1)

'5%'，'25%'など，文字列中に'5%'を含む文字列の場合

```
'%5?%' ESCAPE '?'
```

(例 2)

'SQLPRINT_REC'など，文字列の最後が'PRINT_REC'となる文字列の場合

```
'%PRINT@_REC' ESCAPE '@'
```

(例 3)

X'48695244425f'など，バイナリ列中に X'48695244425f'を含む 16 進文字列の場合

```
X'4869524442ee5f' ESCAPE X'ee'
```

エスケープ文字に指定できる文字を次に示します。

項目のデータ型		指定できる文字
文字データ（CHAR，VARCHAR）	既定文字集合	任意の 1 文字（1 バイト文字）※1

項目のデータ型		指定できる文字
	EBCDIK	
	UTF16	任意の 1 文字※2
混在文字データ（MCHAR，MVARCHAR）		任意の 1 文字※2
各国文字データ（NCHAR，NVARCHAR）		任意の 1 文字（2 バイト文字）※3
バイナリデータ（BINARY）		任意の 1 バイトの値

注

エスケープ文字の次には，必ず特殊文字を指定してください。

注※1

すべての文字，又はデータ値を 1 バイト文字として扱います。

注※2

文字でないデータ値の場合は，その文字コードの最小の長さ（UTF16 の場合は長さ 2，UTF8，SJIS などの場合は長さ 1）の文字として扱います。

注※3

すべての文字，又はデータ値を 2 バイト文字として扱います。

留意事項

1. LIKE の左辺の値式で使用する列の定義長は，255 バイト以下（CHAR，VARCHAR，MCHAR，MVARCHAR，及び BINARY）又は，127 文字以下（NCHAR，及び NVARCHAR）の方が性能が良くなります。
2. CHAR 型及び VARCHAR 型の列にマルチバイト文字が格納されている場合，そのマルチバイト文字は 1 バイトずつ評価されます。したがって，パターン文字列中に指定した 1 バイト文字の文字コードが，マルチバイト文字の文字コードに含まれる場合，LIKE 述語の結果は真となります。

   (例)

   sjis の文字コードでセットアップし，表 T1 に CHAR 型の列 C1 があり，列 C1 中に「ア」という行がある状態で，次の問合せを実行します。

   SELECT C1 FROM T1 WHERE C1 LIKE '%A%' ;

   「ア」は，文字コードを 16 進数で表すと 8341 となります。パターン文字列中の「A」は，文字コードを 16 進数で表すと 41 となります。したがって，マルチバイト文字である「ア」の文字コードの中に，1 バイト文字「A」の文字コードを含んでいるため，LIKE 述語の結果は真となります。

(5) XLIKE 述語

形式

```
値式 〔NOT〕 XLIKE パターン文字列 〔ESCAPE エスケープ文字〕
```

述語が真となる場合

指定した値式の値が，パターン文字列の表すパターンと一致する行に対して，XLIKE 述語は真になります。NOT を指定した場合は，パターン文字列の表すパターンと一致しない行に対して，述語は真になります。ただし，比較するときに大文字と小文字が区別されないため，同等に扱われます。

(値式)

1. 値式には，文字列のパターン比較の対象となる列を指定します。ただし，SQL 変数，及び SQL パラメタ以外の値指定だけの値式は指定できません。
2. 指定できる値式のデータ型は，文字列データ，各国文字データ，又は混在文字列データです。
3. 繰返し列を指定する場合は，添字を指定してください。繰返し列を添字付きで指定して，その要素が条件を満たしている場合，XLIKE 述語は真となります。
4. 繰返し列の添字として ANY を指定できます。ANY を指定した場合は，その列の少なくとも一つの要素が条件を満たしていれば，XLIKE 述語は真となります。XLIKE 述語が真でなく，その列の少なくとも一つの要素に対して指定した条件が不定ならば，その XLIKE 述語は不定になります。XLIKE 述語が真又は不定でない場合，偽となります。
5. 繰返し列を添字付きで指定した場合，その要素がないと XLIKE 述語は不定になります。

(パターン文字列)

1. パターン文字列には，値指定を指定します。
2. 指定できるパターン文字列のデータ型は，値式に指定できるデータ型と同じです。
3. 値式のデータ型とパターン文字列のデータ型の組み合わせは，次のとおりです。

値式のデータ型		パターン文字列，又はエスケープ文字のデータ型				
		文字列データ			各国文字列データ	混在文字列データ
		既定文字集合	EBCDIK	UTF16		
文字列データ	既定文字集合	○	×	×	×	○
	EBCDIK	△※2	○	×		×
	UTF16	△※2	×	○		
各国文字列データ		△※1	×	×	○	×
混在文字列データ		○	×	×	×	○

(凡例)

○：指定できます。

△：制限付きで指定できます。

×：指定できません。

注※1

パターン文字列には文字列定数だけ指定できます。この場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードはチェックされないで，文字データの長さだけがチェックされます。特殊文字を指定する場合は，必ず全角文字で指定してください。

注※2

文字列定数，埋込み変数，又は？パラメタだけ指定できます。

4. 値式，パターン文字列，エスケープ文字列の文字集合は，同じにしてください。ただし，パターン文字列，エスケープ文字列が次に示す場合は，値式の文字集合に変換されるので比較できます。
   - 文字列定数
   - 埋込み変数（既定文字集合）
   - ？パラメタ
5. パターン文字列中の特殊文字には，__（下線），%（パーセント），及びエスケープ文字があります。パターン文字列中の特殊文字のうち__と%の意味を次の表に示します。

表 2-5　パターン文字列中での特殊文字の意味（XLIKE 述語）

特殊文字	項目指定のデータ型		特殊文字の意味	特殊文字の表現方法
__（下線）	文字列データ	既定文字集合※1	任意の 1 文字	半角文字の_（下線）
		EBCDIK※1		
		UTF16		
	各国文字列データ			全角文字の__（下線）
	混在文字列データ			半角文字の_（下線）
%（パーセント）	文字列データ	既定文字集合※1	任意の長さ（0 文字以上）の任意の文字列	半角文字の%（パーセント）
		EBCDIK※1		
		UTF16		
	各国文字列データ※2			全角文字の%（パーセント）
	混在文字列データ			半角文字の%（パーセント）

注※1

すべての文字，又はデータ値を 1 バイト文字として扱います。

注※2

すべての文字，又はデータ値を 2 バイト文字として扱います。

6. パターン文字列データの比較時に，大文字と小文字の区別をしない文字を次に示します。
   - 'A'～'Z'と'a'～'z'
   - 'ア'～'オ'，'ヤ'，'ユ'，'ヨ'，'ツ'と'ァ'～'ォ'，'ャ'，'ュ'，'ョ'，'ッ'
   - '-'（ハイフン）と'ー'（長音）
   - 'あ'～'お'，'や'，'ゆ'，'よ'，'つ'と'ぁ'～'ぉ'，'ゃ'，'ゅ'，'ょ'，'っ'

     （各国文字及び混在文字）
7. パターン文字列中に%を含まない場合，列のデータとパターン文字列の長さが異なる場合は，この述語は真になりません。

8. 値式で指定した文字列とパターン文字列が，可変長文字列（VARCHAR，NVARCHAR，又はMVARCHAR）の場合は，文字列データとパターン文字列データとの比較のほかに文字列長も比較します。

9. パターン文字列として埋込み変数，SQL 変数，又は SQL パラメタを指定する場合，次のことに注意してください。パターン文字列を固定長の埋込み変数，SQL 変数，又は SQL パラメタにした場合に変数の長さより短いパターン文字列を設定すると，変数の後ろに空白が入ったり，残っていた不当文字が値として設定されたりしてしまう場合があります。このようなパターン文字列で検索すると，後ろに空白，又は不当文字と同じ値がないデータは検索されません。したがって，固定長の変数をパターン文字列として使用する場合は，後ろにすべて%を設定する必要があります。

パターン文字列の例

XLIKE 述語の代表的なパターン文字列の例を次の表に示します。

表 2-6　XLIKE 述語の代表的なパターン文字列の例

項　目	パターン文字列	意　味	例	
			パターン文字列	パターンの一致する文字列
前方一致	nnn%	文字列の先頭部分が nnn です。	'ACT%'	ACT，Actor，Action など ACT※1 で始まる文字列
後方一致	%nnn	文字列の最後の部分が nnn です。	'%ING'	Ing，Being，HAVING など ING※2 で終わる文字列
任意一致	%nnn%	文字列中の任意の部分に nnn を含みます。	'%or%'	OR，More，CoLor など or※3 を含む文字列
完全一致	nnn	文字列が nnn と同じです。	'MAX'	MAX，max，mAx※4 などの文字列
部分一致	_…_nnn_…_	文字列中の特定の部分が nnn と同じで，ほかの部分には任意の文字があります。	'_I_'	Bit，HIT，Kit など 3 文字で 2 文字目が I の文字列
その他	nnn%mmm	文字列の先頭部分が nnn で，最後の部分が mmm です。	'O%N'	on，Own，ORIGIN など O か o で始まり N か n で終わる文字列
	%nnn%mmm%	文字列中の任意の部分に nnn を含み，その部分より後の任意の部分に mmm を含みます。	'%O%N%'	ON，one，DowN，Count など O か o を含み，その部分より後に N か n を含む文字列
	nnn_…_mmm%	先頭部分が nnn で始まり，後方に mmm がある文字列です。	'CO_ _ECT%'	correct，Connector，CONNECTION など CO※5 で始まり 5 文字目から 7 文字目が ECT※6 の文字列

注 1

nnn，mmm は，%，_を含まない任意の文字列

注 2

空白も比較対象の文字として扱うため，後方に空白のあるデータと比較した場合，結果は偽になります。

注 3

各国文字列中での特殊文字（%，__）は，各国文字の「%」，「__」を使用します。

注※1

ACT，ACt，AcT，Act，aCT，aCt，acT，act のどれかの文字列

注※2

ING，INg，Ing，InG，iNG，iNg，inG，ing のどれかの文字列

注※3

OR，Or，oR，or のどれかの文字列

注※4

MAX，MAx，Max，MaX，mAX，mAx，maX，max のどれかの文字列

注※5

CO，Co，cO，co のどれかの文字列

注※6

ECT，ECt，Ect，EcT，eCT，eCt，ecT，ect のどれかの文字列

エスケープ文字

パターン文字列中に __（下線）や %（パーセント）を記述すると，無条件に特殊文字とみなされ，通常の文字として扱えません。特殊文字を通常の文字として扱いたい場合は，エスケープ文字を指定します。キーワード ESCAPE の後に，任意の 1 文字（エスケープ文字）を指定しておくと，パターン文字列中に記述したエスケープ文字の直後の特殊文字を通常の文字として扱えます。

（例 1）

'5%'，'25%'など，文字列中に'5%'を含む文字列の場合

```
'%5?%' ESCAPE '?'
```

（例 2）

'SQLPRINT_REC'など，文字列の最後が'PRINT_REC'となる文字列の場合

```
'%PRINT@_REC' ESCAPE '@'
```

エスケープ文字に指定できる文字を次に示します。

項目のデータ型		指定できる文字
文字データ（CHAR，VARCHAR）	既定文字集合	任意の 1 文字（1 バイト文字）※1
	EBCDIK	
	UTF16	任意の 1 文字※2
混在文字データ（MCHAR，MVARCHAR）		任意の 1 文字※2

項目のデータ型	指定できる文字
各国文字データ（NCHAR，NVARCHAR）	任意の 1 文字（2 バイト文字）※3

注

エスケープ文字の次には，必ず特殊文字を指定してください。

注※1

すべての文字，又はデータ値を 1 バイト文字として扱います。

注※2

文字でないデータ値の場合は，その文字コードの最小の長さ（UTF16 の場合は長さ 2，UTF8，SJIS などの場合は長さ 1）の文字として扱います。

注※3

すべての文字，又はデータ値を 2 バイト文字として扱います。

留意事項

1. XLIKE の左辺の値式で使用する列の定義長は，255 バイト以下（CHAR，VARCHAR，MCHAR，及び MVARCHAR），又は 127 文字以下（NCHAR，及び NVARCHAR）の方が性能が良くなります。
2. CHAR 型及び VARCHAR 型の列にマルチバイト文字が格納されている場合，そのマルチバイト文字は 1 バイトずつ評価されます。したがって，パターン文字列中に指定した 1 バイト文字の文字コードが，マルチバイト文字の文字コードに含まれる場合，XLIKE 述語の結果は真となります。

（例）

sjis の文字コードでセットアップし，表 T1 に CHAR 型の列 C1 があり，列 C1 中に「ア」という行がある状態で，次の問合せを実行します。

SELECT C1 FROM T1 WHERE C1 XLIKE '%A%' ;

「ア」は，文字コードを 16 進数で表すと 8341 となります。パターン文字列中の「A」は，文字コードを 16 進数で表すと 41 となります。したがって，マルチバイト文字である「ア」の文字コードの中に，1 バイト文字「A」の文字コードを含んでいるため，XLIKE 述語の結果は真となります。

(6) SIMILAR 述語

形式

```
値式 〔NOT〕 SIMILAR TO パターン文字列 〔ESCAPE エスケープ文字〕
```

述語が真となる場合

指定した値式の値がパターン文字列の表すパターンと一致する行に対して，SIMILAR 述語は真になります。NOT を指定した場合は，パターン文字列の表すパターンと一致しない行に対して，述語は真になります。ただし，パターン文字列の長さが 0 の場合，値式の長さが 0 のときに SIMILAR 述語は真になります。

規則

(値式)

1. 値式には，文字列のパターン比較の対象となる値式を指定します。ただし，？パラメタ，及び埋込み変数の値指定だけの値式は指定できません。
2. 値式及びパターン文字列に指定できるデータ型は，文字列データ，各国文字列データ，混在文字列データ，又は定義長が 32,000 バイト以下の BINARY です。
3. 繰返し列を指定する場合は，添字を指定してください。繰返し列を添字付きで指定して，その要素が条件を満たしている場合，SIMILAR 述語は真となります。
4. 繰返し列の添字として ANY を指定できます。ANY を指定した場合は，その列の少なくとも一つの要素が条件を満たしていれば，SIMILAR 述語は真となります。SIMILAR 述語が真でなく，その列の少なくとも一つの要素に対して指定した条件が不定の場合，その SIMILAR 述語は不定になります。SIMILAR 述語が真又は不定でない場合，偽となります。
5. 繰返し列を添字付きで指定した場合，その要素がないと SIMILAR 述語は不定になります。

(パターン文字列)

1. パターン文字列には，値式を指定します。
2. 値式に繰返し列は指定できません。
3. 値式のデータ型とパターン文字列のデータ型の組み合わせを，次の表に示します。

値式のデータ型		パターン文字列，又はエスケープ文字のデータ型					
		文字列データ			各国文字列データ	混在文字列データ	バイナリデータ
		既定文字集合	EBCDIK	UTF16			
文字列データ	既定文字集合	○	×	×	×	○	×
	EBCDIK	△※3	○	×		×	
	UTF16	△※3	×	○			
各国文字列データ		△※1	×	×	○	×	×
混在文字列データ		○	×	×	×	○	×
バイナリデータ		△※2	×	×	×	×	○

(凡例)

○：指定できます。

△：制限付きで指定できます。

×：指定できません。

注※1

パターン文字列には文字列定数だけ指定できます。この場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードはチェックされないで，文字データの長さだけがチェックされます。なお，特殊文字を指定する場合は，必ず全角文字で指定してください。

注※2

16 進文字列定数だけ指定できます。

注※3

文字列定数，埋込み変数，又は？パラメタだけ指定できます。

4. 値式，パターン文字列，エスケープ文字列の文字集合は，同じにしてください。ただし，パターン文字列，エスケープ文字列が次に示す場合は，値式の文字集合に変換されるので比較できます。

・文字列定数

・埋込み変数（既定文字集合）

・？パラメタ

5. パターン文字列に指定する正規表現の形式を次に示します。

```
正規表現 ：：＝ 正規項 ｜ 正規表現 垂直棒 正規表現
正規項 ：：＝ 正規因子｜ 正規項 正規因子
正規因子 ：：＝  正規一次子
               ｜ 正規一次子 *
               ｜ 正規一次子 +
               ｜ 正規一次子 ？
               ｜ 正規一次子 繰返し因子
繰返し因子 ：：＝ 左波括弧 下限値 ［ 上限指定 ］ 右波括弧
上限指定 ：：＝ コンマ ［ 上限値 ］
正規一次子 ：：＝  文字指定子
                 ｜ %
                 ｜ 正規文字集合
                 ｜ 正規文字集合識別子指定
                 ｜ 左括弧 正規表現 右括弧
文字指定子 ：：＝  エスケープされない文字
                 ｜ エスケープされた文字
正規文字集合 ：：＝
                   ｜ 左角括弧 文字列挙… 右角括弧
                   ｜ 左角括弧 ^ 文字列挙… 右角括弧
文字列挙 ：：＝  文字指定子
               ｜ 文字指定子 負符号 文字指定子
               ｜ 正規文字集合識別子指定
正規文字集合識別子指定 ：：＝ 左角括弧 コロン 正規文字集合識別子 コロン 右角括弧
```

6. パターン文字列に指定する正規表現の構文規則を次に示します。

・正規文字集合識別子には次のどれかを指定します。

’ALPHA’,’UPPER’,’LOWER’,’DIGIT’,’ALNUM’,’SPACE’,’WHITESPACE’

・エスケープされない文字には，特殊文字を除く任意の 1 文字を指定します。特殊文字を次に示します。

特殊文字	バイナリデータで指定するコード		
	文字集合なし	文字集合あり	
		EBCDIK	UTF16
_（下線文字）	X’5F’	X’6D’	U+005F
%（パーセント）	X’25’	X’6C’	U+0025
*（アスタリスク）	X’2A’	X’5C’	U+002A
＋（正符号）	X’2B’	X’4E	U+002B

特殊文字	バイナリデータで指定するコード		
	文字集合なし	文字集合あり	
		EBCDIK	UTF16
?（疑問符）	X’3F’	X’6F’	U+003F
｜（垂直棒）	X’7C’	X’4F’	U+007C
(（左括弧）	X’28’	X’4D’	U+0028
)（右括弧）	X’29’	X’5D’	U+0029
{（左波括弧）	X’7B’	X’C0’	U+007B
}（右波括弧）	X’7D’	X’D0’	U+007D
[（左角括弧）	X’5B’	X’4A’	U+005B
]（右角括弧）	X’5D’	X’5A’	U+005D
エスケープ文字	ESCAPE に指定した値		
-（負符号）※	X’2D’	X’60’	U+002D
:（コロン）※	X’3A’	X’7A’	U+003A
^（サーカムフレックス）※	X’5E’	X’5F’	U+005E

注※　文字列挙中でだけ特殊文字として扱います。

・エスケープされた文字には，特殊文字そのものを指定したい場合に，エスケープ文字に続けてエスケープする特殊文字を指定します。

・下限値と上限値には，0≦下限値≦上限値≦256 を満たす整数を指定します。

7. パターン文字列に指定する正規表現の各指定の意味を，次の表に示します。

表 2-7　正規表現の各指定の意味

正規表現の指定	意味
文字指定子	文字指定子で指定された文字（長さ 1 の文字列）を意味します。
_（下線文字）	長さが 1 の任意の文字を意味します。
%（パーセント）	長さが 0 以上の任意の文字列を意味します。
正規一次子*	直前の正規一次子の，0 回以上の繰り返しを意味します。
正規一次子+	直前の正規一次子の，1 回以上の繰り返しを意味します。
正規一次子?	直前の正規一次子の，0 回又は 1 回の繰り返しを意味します。
正規表現｜正規表現	｜の左右に指定した正規表現のうちのどちらかを意味します。
(正規表現)	( )内で指定した正規表現のグループ化を意味します。 正規表現を使用する際に，正規表現であることを明確にする場合に使用します。主に「｜」を使用するときに利用します。

正規表現の指定	意味
正規一次子{n} 正規一次子{n,m} 正規一次子{n,}	直前の正規一次子の繰り返しを意味します。繰り返し回数の指定方法と意味を次に示します。 {n}　：直前の正規表現の，n回の繰り返し {n,m}：直前の正規表現の，n回以上m回以下の繰り返し {n,}　：直前の正規表現の，n回以上の繰り返し
[文字列挙…]	列挙された文字のうちの任意の文字を意味します。
[^文字列挙…]	列挙された文字を除く任意の文字を意味します。
文字指定子 1-文字指定子 2	文字列挙中に指定した場合，文字指定子 1 が示す文字から文字指定子 2 が示す文字までの任意の文字（文字コードによる範囲）を意味します。
[:ALPHA:]	任意の英大文字（¥，@，#は含まない），又は英小文字
[:UPPER:]	任意の英大文字（¥，@，#は含まない）
[:LOWER:]	任意の英小文字
[:DIGIT:]	任意の数字
[:ALNUM:]	任意の英大文字（¥，@，#は含まない），英小文字，又は数字
[:SPACE:]	半角の空白文字（ただし，値式が各国文字列データの場合は，全角の空白文字）
[:WHITESPACE:]	次の表に示す文字コードの文字のうち，任意の 1 文字（文字コード種別によって[:WHITESPACE:]が意味する文字は異なる）

[:WHITESPACE:]に含まれる文字の文字コードを次の表に示します。

Unicode(UTF-8)		シフト JIS 漢字		EUC 日本語漢字，EUC 中国語漢字	中国語漢字(GB18030)	LANG-C	Unicode 規格で定められた文字の名前
既定文字集合	UTF16	既定文字集合	EBCDIK				
X’09’	U+0009	X’09’	X’05’	X’09’	X’09’	X’09’	Horizontal Tabulation
X’0A’	U+000A	X’0A’	X’15’	X’0A’	X’0A’	X’0A’	Line Feed
X’0B	U+000B	X’0B	X’0B’	X’0B	X’0B	X’0B	Vertical Tabulation
X’0C’	U+000C	X’0C’	X’0C’	X’0C’	X’0C’	X’0C’	Form Feed
X’0D’	U+000D	X’0D’	X’0D’	X’0D’	X’0D’	X’0D’	Carriage Return
X’20’	U+0020	X’20’	X’40’	X’20’	X’20’	X’20’	Space
X’C285’	U+0085	－	－	－	X’81308135’	－	Next Line※
X’C2A0’	U+00A0	－	－	－	X’81308432’	－	No-Break Space※

Unicode(UTF-8)		シフト JIS 漢字		EUC 日本語漢字, EUC 中国語漢字	中国語漢字(GB18030)	LANG-C	Unicode 規格で定められた文字の名前
既定文字集合	UTF16	既定文字集合	EBCDIK				
X'E19A80'	U+1680	−	−	−	X'8134AC34'	−	Ogham Space Mark※
X'E28080'	U+2000	−	−	−	X'8136A336'	−	En Quad※
X'E28081'	U+2001	−	−	−	X'8136A337'	−	Em Quad※
X'E28082'	U+2002	−	−	−	X'8136A338'	−	En Space※
X'E28083'	U+2003	−	−	−	X'8136A339'	−	Em Space※
X'E28084'	U+2004	−	−	−	X'8136A430'	−	Three-Per-Em Space※
X'E28085'	U+2005	−	−	−	X'8136A431'	−	Four-Per-Em Space※
X'E28086'	U+2006	−	−	−	X'8136A432'	−	Six-Per-Em Space※
X'E28087'	U+2007	−	−	−	X'8136A433'	−	Figure Space※
X'E28088'	U+2008	−	−	−	X'8136A434'	−	Punctuation Space※
X'E28089'	U+2009	−	−	−	X'8136A435'	−	Thin Space※
X'E2808A'	U+200A	−	−	−	X'8136A436'	−	Hair Space※
X'E280A8'	U+2028	−	−	−	X'8136A635'	−	Line Separator※

Unicode(UTF-8)		シフト JIS 漢字		EUC 日本語漢字，EUC 中国語漢字	中国語漢字(GB18030)	LANG-C	Unicode 規格で定められた文字の名前
既定文字集合	UTF16	既定文字集合	EBCDIK				
X’E280A9’	U+2029	−	−	−	X’8136A636’	−	Paragraph Separator※
X’E280AF’	U+202F	−	−	−	X’8136A732’	−	Narrow No-Break Space※
X’E38080’	U+3000	0x8140	−	X’A1A1’	X’A1A1’	−	Ideographic Space※

（凡例）

−：該当しません。

注※

値式が文字列型の場合は，[:WHITESPACE:]に含まれません。

8. 正規文字集合識別子指定はバイナリデータでは指定できません。

9. パターン文字列として埋込み変数，SQL 変数，又は SQL パラメタを指定する場合，次のことに注意してください。

パターン文字列を固定長の埋込み変数，SQL 変数，又は SQL パラメタにした場合に，変数の長さより短いパターン文字列を設定すると，変数の後ろに空白が入ったり，残っていた不当文字が値として設定されてしまうことがあります。このようなパターン文字列で検索すると，後ろに空白，又は不当文字と同じ値がないデータは検索されません。したがって，固定長の変数をパターン文字列として使用する場合は，後ろにすべて%を設定してください。

（例 1）

変数にパターン文字列として'AB%'，及び'AB%%'を設定した場合，文字列データが'ABCD'のときの比較結果を次に示します。

変数のデータ型	パターン文字列	文字列データ	比較結果
可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）	'AB%'	'ABCD'	一致します。
	'AB%%'	'ABCD'	一致します。
固定長文字列（4 バイト）	'AB%△'	'ABCD'	一致しません。
	'AB%%'	'ABCD'	一致します。

（例 2）

変数にパターン文字列として X'52454425'を設定した場合，バイナリデータが 52454452554d のときの比較を次に示します。

変数のデータ型	パターン文字列	バイナリデータ	比較結果
BINARY 型	X'52454425'	52454452554d	一致します。

変数のデータ型	パターン文字列	バイナリデータ	比較結果
	X'5245442525'	52454452554d	一致します。

不正なパターン文字列

1. パターン文字列が不正（KFPA11424-E メッセージを出力）となる条件を次に示します。

関連する項目	条件	不正なパターン文字列の例
正規一次子* 正規一次子+ 正規一次子?	* , + , ?の直前の正規一次子が指定されていない。	‘(*)’, ‘(+)’,’ (?)’
正規表現｜正規表現	｜の前後のどちらかに正規表現が指定されていない。	‘a ｜’,’ (a ｜)’,’ (a ｜｜ b)’
(正規表現)	()内に正規表現が指定されていない。	‘()’
	(と)が対応していない。	‘(abc’,’ abc)’
正規一次子{n} 正規一次子{n,m} 正規一次子{n,}	繰返し因子の直前の正規一次子が指定されていない。	‘{4}’
	繰返し因子の繰り返し回数の指定が不正。	‘a{-1}’,’ a{4,2}’
	{と}が対応していない。	‘a{4’,’ a4}’
[文字列挙…] [^文字列挙…]	文字列挙にエスケープされていない特殊文字を含んでいる。ただし，正規文字集合識別指定は指定できる。	‘[a%c]’
	-の前後の文字指定が不正。	‘[-]’, ‘[c-a]’, ‘[a--] ‘
	[]内に文字列挙の指定がない。	‘[]’, ‘[^]’
	[と]が対応していない。	‘[a-c’,’ a-c]’
エスケープ文字	パターン文字列の最後の文字がエスケープ文字である。	‘abc¥’（エスケープ文字として’¥’を指定した場合）
正規文字集合識別指定	正規文字集合識別子が不正である。	‘[:INVALID:]’

パターン文字列の例

SIMILAR 述語の代表的なパターン文字列の例を次の表に示します。

表 2-8　SIMILAR 述語の代表的なパターン文字列の例

項　目	パターン文字列	意　味	例	
			パターン文字列	パターンの一致する文字列
前方一致	nnn%	文字列の先頭部分が nnn です。	'ACT%'	ACT，ACTOR，ACTION など ACT で始まる文字列
後方一致※1	%nnn	文字列の最後の部分が nnn です。	'%ING'	ING，BEING，HAVING など ING で終わる文字列
任意一致	%nnn%	文字列中の任意の部分に nnn を含みます。	N'%日%'	日，日立，昨日，本日中など日を含む文字列※2
完全一致	nnn	文字列が nnn と同じです。	'EQUAL'	EQUAL

項　目	パターン文字列	意　味	例	
			パターン文字列	パターンの一致する文字列
部分一致	_…_nnn_…_	文字列中の特定の部分が nnn と同じで，ほかの部分には任意の文字があります。	'_I_'	BIT，HIT，KIT など 3 文字で 2 文字目が I の文字列
1 回以上の繰り返し	mmm[0-9]+ 又は mmm[:DIGIT:]+	文字列の先頭部分が mmm で，その後ろが数値です。	'KFPA11[0-9]+-E' 又は 'KFPA11[:DIGIT:]+-E"	KFPA11104-E,KFPA11901-E など KFPA11 で始まり 7 文字目から数値で数値の後が-E の文字列
幾つかの文字の選択	mmm(n|o) 又は mmm[no]	文字列の先頭部分が mmm で，i 文字目が n 又は o です（i は任意の数字）。	'KFPA%-(W | E)' 又は 'KFPA%-[WE]'	KFPA20008-W,KFPA11901-E など KFPA で始まり最後の 2 文字が-W 又は-E の文字列
0 回以上，1 回以下の繰り返し	nnno?mmm	文字列の先頭部分が nnn，最後の部分が mmm でその間に o がある又はないです。	'OW?N'	ON，OWN など O で始まり N で終わる文字列の間に W がある文字列とない文字列
0 回以上の繰り返し	nnno*mmm	文字列の先頭部分が nnn，最後の部分が mmm でその間に o が 0 回以上繰り返します。	10*1	11，101，1001 など最初の文字が 1 でそれ以降に 0 が 0 回以上繰り返され最後に 1 の文字列
n 回の繰り返し	mmm{n}	文字列の先頭部分が mmm で n 回繰り返します。	[1-9]0{3}	1000，2000，3000 など最初の文字が 1～9 でそれ以降 0 が 3 回繰り返される文字列
その他	nnn%mmm	文字列の先頭部分が nnn で，最後の部分が mmm です。	'O%N'	ON，OWN，ORIGIN など O で始まり N で終わる文字列
	%nnn%mmm%	文字列中の任意の部分に nnn を含み，その部分から後の任意の部分に mmm を含みます。	'%O%N%'	ON，ONE，DOWN，COUNT など O を含み，その部分より後に N を含む文字列
	nnn_…_mmm%	先頭部分が nnn で始まり，後方に mmm がある文字列です。	'CO_ _ECT%'	CORRECT， CONNECTOR， CONNECTION など CO で始まり 5 文字目から 7 文字目が ECT の文字列

注

nnn，mmm は，特殊文字を含まない任意の文字列

注※1

空白も比較対象の文字として扱うため，後方に空白のあるデータと比較した場合，結果は偽になります。

注※2

各国文字列中での特殊文字は，各国文字に合わせた特殊文字を使用します。

(エスケープ文字)

パターン文字列中の特殊文字は通常の文字として扱えません。特殊文字を通常の文字として扱いたい場合は，エスケープ文字を指定します。キーワード ESCAPE の後に，任意の 1 文字（エスケープ文字）を指定しておくと，パターン文字列中に記述したエスケープ文字の直後の特殊文字を，通常の文字として扱えます。エスケープ文字には，文字定数，？パラメタ，埋込み変数，SQL 変数名，SQL パラメタ名を指定できます。

(例 1)

'5%'，'25%'など，文字列中に'5%'を含む文字列の場合

```
'%5¥%%' ESCAPE '¥'
```

(例 2)

'SQLPRINT_REC'など，文字列の最後が'PRINT_REC'となる文字列の場合

```
'%PRINT¥_REC' ESCAPE '¥'
```

(例 3)

X'48695244425f'など，バイナリ列中に X'48695244425f'を含む 16 進文字列の場合

```
X'4869524442ee5f' ESCAPE X'ee'
```

エスケープ文字に指定できる文字を次に示します。

項目のデータ型		指定できる文字
文字データ（CHAR，VARCHAR）	既定文字集合	任意の 1 文字（1 バイト文字）※1
	EBCDIK	
	UTF16	任意の 1 文字※2
混在文字データ（MCHAR，MVARCHAR）		任意の 1 文字※2
各国文字データ（NCHAR，NVARCHAR）		任意の 1 文字（2 バイト文字）※3
バイナリデータ（BINARY）		任意の 1 バイトの値

注

エスケープ文字の次には，必ず特殊文字を指定してください。

注※1

すべての文字，又はデータ値を 1 バイト文字として扱います。

注※2

文字でないデータ値の場合は，その文字コードの最小の長さ（UTF16 の場合は長さ 2，UTF8，SJIS などの場合は長さ 1）の文字として扱います。

注※3

すべての文字，又はデータ値を 2 バイト文字として扱います。

留意事項

1. SIMILAR の左辺の値式で使用する列の定義長は，255 バイト以下（CHAR，VARCHAR，MCHAR，MVARCHAR，及び BINARY）又は，127 文字以下（NCHAR，及び NVARCHAR）の方が性能が良くなります。
2. CHAR 型及び VARCHAR 型の列にマルチバイト文字が格納されている場合，そのマルチバイト文字は 1 バイトずつ評価されます。したがって，パターン文字列中に指定した 1 バイト文字の文字コードが，マルチバイト文字の文字コードに含まれる場合，SIMILAR 述語の結果は真となります。

   （例）

   sjis の文字コードでセットアップし，表 T1 に CHAR 型の列 C1 があり，列 C1 中に「ア」という行がある状態で，次の問合せを実行します。

   SELECT C1 FROM T1 WHERE C1 SIMILAR TO '%A%' ;

   「ア」は，文字コードを 16 進数で表すと 8341 となります。パターン文字列中の「A」は，文字コードを 16 進数で表すと 41 となります。したがって，マルチバイト文字である「ア」の文字コードの中に，1 バイト文字「A」の文字コードを含んでいるため，SIMILAR 述語の結果は真となります。
3. パターン文字列が極端に長い場合や，特殊文字{}を連続して指定した場合，検索性能が劣化したり，メモリ使用量が増大したりすることがあります。

(7) BETWEEN 述語

形式

```
行値構成子1 〔NOT〕 BETWEEN 行値構成子2 AND 行値構成子3
```

述語が真となる場合

次の条件を満足する行に対して，BETWEEN 述語は真になります。

行値構成子 2 ≦ 行値構成子 1 ≦ 行値構成子 3

NOT を指定した場合は，条件を満足しない行に対して，この述語は真になります。

規則

（行値構成子 1）

1. 値指定だけの行値構成子要素は指定できません。
2. 繰返し列を指定する場合は，添字を指定してください。繰返し列を添字付きで指定して，その要素が条件を満たしている場合，BETWEEN 述語は真となります。
3. 繰返し列の添字として ANY を指定できます。ANY を指定した場合は，その列の少なくとも一つの要素が条件を満たしていれば，BETWEEN 述語は真となります。BETWEEN 述語が真でなく，その列の少なくとも一つの要素に対して指定した条件が不定ならば，その BETWEEN 述語は不定になります。BETWEEN 述語が真又は不定でない場合，偽となります。

4. 繰返し列を添字付きで指定した場合，その要素がないと BETWEEN 述語は不定になります。

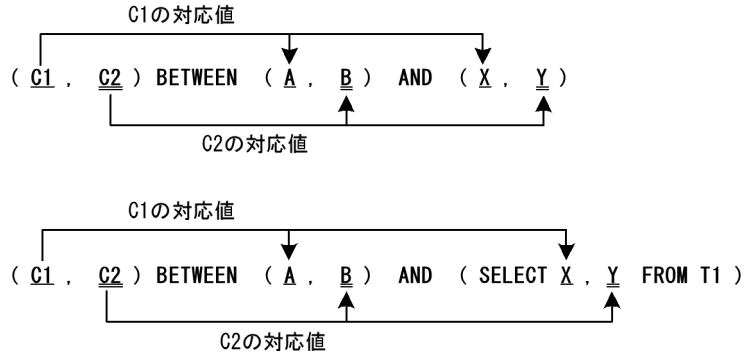
(行値構成子 2 及び行値構成子 3)

1. 繰返し列を指定できません。

(共通)

1. 各行値構成子中で，同じ位置にある行値構成子要素を対応値とします。対応値のデータ型は，それぞれ変換できるデータ型にしてください。ただし，行値構成子 1 に各国文字データを指定し，行値構成子 2，又は行値構成子 3 の対応値に文字列定数を指定した場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードはチェックされないで，文字データの長さだけがチェックされます。

＜対応値＞



2. 行値構成子 1，行値構成子 2 及び行値構成子 3 には，次のデータ型の値を指定できません。
   - BLOB
   - 最大長が 32,001 バイト以上の BINARY
   - BOOLEAN
   - 抽象データ型
3. 行値構成子 1，行値構成子 2 及び行値構成子 3 は，結果の列数を同じにしてください。

(8) 限定述語

形式

```
行値構成子 {=|<>|^=|!=|<|<=|>|>=}
       { {ANY|SOME} |ALL}
   〈表副問合せ〉
   (〔副問合せ実行方式のSQL最適化指定〕
    SELECT〔 {ALL|DISTINCT} 〕 {選択式|*}
       〈表 式〉
      FROM 表参照〔, 表参照〕…
         〔WHERE 探索条件〕
         〔GROUP BY 値式〔, 値式〕…〕
         〔HAVING 探索条件〕)
```

述語が真となる場合

ANY，又は SOME を指定した場合，表副問合せの結果の任意の行が一つでも行値構成子との比較条件を満たしていれば，限定述語の結果は真になります。

ALL を指定した場合，表副問合せの結果のすべての行が行値構成子との比較条件を満たしているか，又は表副問合せの結果が空集合であれば，限定述語の結果は真になります。

規則

1. 演算結果のデータ型が，次のデータ型となる値を指定できません。
   - BLOB
   - 定義長が 32,001 バイト以上の BINARY
   - BOOLEAN
   - 抽象データ型
2. 行値構成子の結果がナル値である行に対して，この述語は不定になります。
3. 表副問合せについては，「副問合せ」を参照してください。
4. 限定述語の SOME 指定と ANY 指定のどちらを指定しても処理は同じです。
5. 限定述語は，IN 述語と同じ意味を表すものがあります。同じ意味を表す述語を次に示します。

限定述語	IN 述語
行値構成子 = ANY 表副問合せ 又は 行値構成子 = SOME 表副問合せ	行値構成子 IN 表副問合せ
行値構成子 <> ALL 表副問合せ	行値構成子 NOT IN 表副問合せ

6. ANY，又は SOME を指定した限定述語の結果を次の表に示します。表副問合せの結果の任意の行が一つでも条件を満たしていれば真になります。すべての行の比較結果がすべて偽であるか，又は表副問合せの結果が空集合であれば偽になります。どちらでもなければ不定になります。

表 2-9　ANY，又は SOME を指定した限定述語の結果

副問合せの各行に対する比較結果		限定述語の結果（ANY 又は SOME）
真の行あり		真
真の行なし	不定あり	不定
	不定なし	偽
空集合		偽

7. ALL を指定した限定述語の結果を次の表に示します。表副問合せの結果のすべての行の比較結果が真であるか，又は表副問合せの結果が空集合であれば真になります。任意の行が一つでも偽であれば偽になります。どちらでもなければ不定になります。

表 2-10　ALL を指定した限定述語の結果

副問合せの各行に対する比較結果		限定述語の結果（ALL）
偽の行あり		偽
偽の行なし	不定あり	不定
	不定なし	真
空集合		真

8. 次に示す箇所には，限定述語は指定できません。
   - IF 文の探索条件
   - WHILE 文の探索条件
   - CREATE TRIGGER の WHEN 探索条件（トリガ動作条件）
9. 行値構成子要素に繰返し列を指定する場合は，添字を指定してください。繰返し列を添字付きで指定して，その要素が条件を満たしている場合，限定述語は真となります。
10. 繰返し列の添字として ANY を指定できます。ANY を指定した場合は，その列の少なくとも一つの要素が条件を満たしていれば，限定述語は真となります。限定述語が真でなく，その列の少なくとも一つの要素に対して指定した条件が不定ならば，その限定述語は不定になります。限定述語が真又は不定でない場合，偽となります。
11. 繰返し列を添字付きで指定した場合，その要素がないと限定述語は不定になります。
12. 限定述語の左辺に指定した行値構成子の結果の列数と，表副問合せの結果の列数は同じにしてください。

留意事項

1. 限定述語を指定した場合，HiRDB が作業表を作成することがあります。このとき，作業表の行長によっては，限定述語の処理が制限されることがあります。作業表の行長については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(9)　EXISTS 述語

形式

```
EXISTS
   〈表副問合せ〉
   (〔副問合せ実行方式のSQL最適化指定〕
    SELECT〔 {ALL|DISTINCT}〕 {選択式|*}
       〈表 式〉
      FROM 表参照〔, 表参照〕…
         〔WHERE 探索条件〕
         〔GROUP BY 値式〔, 値式〕…〕
         〔HAVING 探索条件〕)
```

述語が真となる場合

表副問合せの結果が空集合でなければ，EXISTS 述語の結果は真になります。

規則

1. 表副問合せについては，「副問合せ」を参照してください。
2. EXISTS 述語は，表副問合せの結果が空集合でないかどうかをテストする述語です。
3. EXISTS 述語の結果を次の表に示します。表副問合せの結果が 1 行以上であれば真になります。空集合であれば偽になります。

表 2-11　EXISTS 述語の結果

副問合せの結果，問合せ条件に合った行の数	EXIST 述語の結果
1 行以上	真
0 行	偽

4. 次に示す箇所には，EXISTS 述語は指定できません。
   - IF 文の探索条件
   - WHILE 文の探索条件
   - CREATE TRIGGER の WHEN 探索条件（トリガ動作条件）

(10)　論理述語

形式

```
値式 IS 〔NOT〕 {TRUE | FALSE | UNKNOWN}
```

述語が真となる場合

値式の論理値が，指定した真（TRUE），偽（FALSE），又は不定（UNKNOWN）と一致する場合，論理述語は真になります。NOT を指定した場合は，値式の論理値が，指定した真（TRUE），偽（FALSE），又は不定（UNKNOWN）と一致しない場合に真となります。

規則

1. 論理述語を評価した場合の述語の結果を次の表に示します。NOT を指定した場合は，表中の各論理値が反転します。

表 2-12　論理述語を評価した場合の述語の結果

値式の論理値	IS TRUE	IS FALSE	IS UNKNOWN
真	真	偽	偽
偽	偽	真	偽
不定	偽	偽	真

2. 値式には，次のデータ型の値を指定できます。
   - BOOLEAN

3. 論理値が不定であることと，論理値がナル値であることは等価です。

(11) 構造化繰返し述語

形式

```
ARRAY（列指定〔，列指定〕…）［ANY］（探索条件）
```

述語が真となる場合

ARRAY（列指定〔，列指定〕…）に指定した繰返し列を，添字が同じ要素の組とした複数項目の繰り返しとみなし，そのどれかの要素が探索条件を満たす場合，構造化繰返し述語は真となります。

規則

ARRAY（列指定〔，列指定〕…）

1. 列指定には，構造化する繰返し列を指定します。
2. 列指定は，一つのインデクス構成列にすべて含まれるようにしてください。
3. 列指定に次に示す列は指定できません。
   ・異なる表の列
   ・異なる表から導出した列
   ただし，異なる表には，実表が同じで相関名が異なるものも含みます。
4. 列指定は，外への参照をする列を指定できません。
5. 列指定は，同じ列を 2 回以上指定できません。
6. 列指定は，最大 64 個指定できます。

探索条件

探索条件を指定します。ただし，構造化繰返し述語についての次の規則があります。

1. 探索条件には，次のものを指定できません。
   ・添字付きの列指定
   ・ARRAY（列指定〔，列指定〕…）で指定した繰返し列以外の列
   ・システム定義スカラ関数，関数呼出し，及び IS_USER_CONTAINED_IN_HDS_GROUP を含む述語
   ・構造化繰返し述語
   ・XMLEXISTS 述語
   ・列指定を含まない述語
   ・副問合せ
2. NULL 述語を指定した場合，その列に要素がないときは，NULL 述語は不定になります。指定した要素がナル値ならば NULL 述語は真になります。

共通

1. 構造化繰返し述語を含む探索条件は，NOT で否定できません。
2. IF 文，WHILE 文の探索条件に，構造化繰返し述語は指定できません。
3. HAVING 句に，構造化繰返し述語は指定できません。
4. CASE 式の探索条件に，構造化繰返し述語は指定できません。
5. 構造化繰返し述語を含んでいて，かつオペランドの探索条件に次に示す列（外への参照をする列を除く）がある OR は指定できません。
   ・異なる表の列
   ・異なる表から導出した列
   ただし，異なる表には，実表が同じで相関名が異なるものも含みます。
6. ビュー定義の導出問合せ式中の探索条件には指定できません。
7. 外結合の結合表を含む問合せ指定の ON 探索条件に構造化繰返し述語を指定する場合，列指定には内表の列を指定してください。
8. 外結合の結合表を含む問合せ指定の WHERE 句に構造化繰返し述語を指定する場合，列指定には外結合の最も外側の表の列を指定してください。

使用例

成績表から，数学の成績が 85 点以上の氏名を求めます。なお，成績表は科目名，成績が要素 10 の繰返し列となっています。

```
SELECT 氏名 FROM 成績表
    WHERE ARRAY(科目名,成績)[ANY](科目名='数学' AND 成績>=85)
```

成績表

氏名	科目名	成績
山田	数学	90
	国語	65
鈴木	数学	50
	国語	90
佐藤	数学	85
	国語	70

⇩

検索結果

氏名
山田
佐藤

注　成績表の表定義，インデクス定義は次のとおりです。

```
表定義：
  CREATE TABLE 成績表(氏名 MCHAR(10),
                      科目名 MCHAR(10) ARRAY[4],
                      成績 SMALLINT  ARRAY[4]);
インデクス定義：
  CREATE INDEX 科目成績 ON 成績表(科目名, 成績);
```

(12) XMLEXISTS 述語

HiRDB XML Extension と連携することで使用できる述語です。

詳細は，「XMLEXISTS 述語」を参照してください。

2.8　行値構成子

2.8.1　行値構成子の形式と規則

(1)　機能

行，又は順序付けられた列の並びを指定します。

(2)　形式

```
行値構成子：：＝｛ 行値構成子要素
                  ｜（行値構成子要素 〔，行値構成子要素〕 …）
                  ｜行副問合せ｝
行値構成子要素：：＝値式
```

(3)　規則

1. 行値構成子に指定できる行値構成子要素の数は，最大 255 個です。
2. 行値構成子の結果の列は，次のデータ型を指定できません。
   - BLOB
   - 32,001 バイト以上の BINARY
   - 抽象データ型
   - BOOLEAN
3. 行値構成子要素を 2 個以上指定する場合で，行値構成子要素に繰返し列を指定したとき，添字に ANY を指定できません。
4. 行副問合せについては，「副問合せ」を参照してください。

2.9 値式，値指定，及び項目指定

2.9.1 値式，値指定，及び項目指定の形式と規則

(1) 機能

SQL 中に値を指定する場合，次の形式で値を指定できます。

(2) 形式

```
値式：：＝｛〔＋｜－〕一次子｜値式｛＋｜－｜＊｜／｝〔｛＋｜－｝〕一次子
          ｜値式 ＊ 一次子｜値式 ／ 一次子｜値式 ‖ 一次子｝
一次子：：＝｛（値式）｜項目指定｜符号なし値指定｜集合関数｜ウィンドウ関数｜スカラ関数
            ｜CASE式｜CAST指定｜ラベル付き間隔｜関数呼出し
            ｜スカラ副問合せ｜NEXT VALUE式｝
値指定：：＝｛定数｜?パラメタ｜：埋込み変数〔：標識変数〕
            ｜USER｜CURRENT_USER｜CURRENT_DATE｜CURRENT_TIME
            ｜CURRENT_TIMESTAMP〔(p)〕
            ｜CURRENT DATE｜CURRENT TIME
            ｜CURRENT TIMESTAMP〔(p)〕
            ｜〔文ラベル.〕SQL変数名
            ｜〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
            ｜SQLCODE｜SQLCOUNT｜SQLCODE_OF_LAST_CONDITION｜SQLERRM_OF_LAST_CONDITION｝
符号なし値指定：：＝
          ｛符号なし数定数｜一般定数｜?パラメタ｜：埋込み変数〔：標識変数〕
            ｜USER｜CURRENT_USER｜CURRENT_DATE｜CURRENT_TIME
            ｜CURRENT_TIMESTAMP〔(p)〕
            ｜CURRENT DATE｜CURRENT TIME
            ｜CURRENT TIMESTAMP〔(p)〕
            ｜〔文ラベル.〕SQL変数名
            ｜〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
            ｜SQLCODE｜SQLCOUNT｜SQLCODE_OF_LAST_CONDITION｜SQLERRM_OF_LAST_CONDITION｝
定数：：＝｛数定数｜一般定数｝
一般定数：：＝｛文字列定数｜16進文字列定数
              ｜各国文字列定数｜混在文字列定数｝
項目指定：：＝｛列指定
              ｜〔文ラベル.〕SQL変数名
              ｜〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
              ｜コンポネント指定｝
```

(3) 共通規則

1. 値式中で指定できる演算のうち，四則演算，日付演算，時刻演算，連結演算，CASE 式，CAST 指定，ウィンドウ関数，スカラ関数，及び NEXT VALUE 式を総称して，スカラ演算といいます。
2. 値式は，比較述語，比較値，BETWEEN 述語，IN 述語，LIKE 述語，XLIKE 述語，限定述語，論理述語，コンポネント指定，関数呼出し，列指定，更新値，一次子，集合関数，又はスカラ演算で指定します。
3. スカラ演算の評価順序を次に示します。

1. 括弧内
2. ＊又は／
3. ＋，－，又は‖

ただし，評価順序の同じスカラ演算が値式中に複数ある場合は，左から右に評価されます。

4. スカラ演算のネストの最大数は，255 です。スカラ演算のネストの数は，演算子＋，－，＊，／，又は‖の評価順序を表す括弧を省略しないで指定した場合の括弧のネスト数です。なお，ネスト数はスカラ関数の種類によって次のようになります。

   - スカラ関数 SUBSTR，VARCHAR_FORMAT，TIMESTAMP_FORMAT，DATE（日時書式を指定したとき），TIME（日時書式を指定したとき），及び TIMESTAMP（機能 3.のとき）を指定した場合は 2 となります。
   - スカラ関数 VALUE は引数の値式の数＋ 1 となります。
   - そのほかのスカラ関数は 1 となります。

   また，単純 CASE 式，及び探索 CASE 式は WHEN の数，CASE 略式の COALESCE は引数の値式の数＋ 1，CASE 略式の NULLIF は 2 としてそれぞれ数えます。

   名前付き導出表の列をオペランドとするスカラ演算を指定して，その列がスカラ演算で導出され，かつその名前付き導出表が内部導出表を作成しない場合，名前付き導出表の列を導出する，スカラ演算をオペランドとするスカラ演算を指定することと等価となり，スカラ演算の最大ネスト数を超えることがあります。

   値式中にスカラ演算と関数呼出しを指定する場合，スカラ演算のネスト数と関数呼出しのネスト数の合計は，最大 255 です。

5. 指定したデータがナル値の場合は，四則演算，日付演算，時刻演算，又は連結演算の結果もナル値になります。
6. 四則演算，日付演算，又は時刻演算の除算で，第 2 演算項の値に 0 を指定した場合，エラーになります。
7. 演算途中でオーバフローが発生した場合は，エラーになります。値式については，「四則演算」，「日付演算」，「時刻演算」，及び「連結演算」を参照してください。
8. SQL パラメタ名は，手続き定義の中でその SQL 手続きの SQL パラメタを参照するために使用できます。
9. SQL 変数名は，手続き定義又は関数定義の中の複合文中で，その複合文中で宣言した SQL 変数を参照するために使用できます。
10. 同じ名前の列，SQL 変数，又は SQL パラメタがあるとき，それらの名前は表指定，文ラベル，又は〔認可識別子.〕ルーチン識別子で修飾して指定します。修飾しないか，又は修飾子が同じ場合には，それらの名前の有効範囲の狭い順に優先されて識別されます。すなわち，列，SQL 変数，SQL パラメタの順に優先されます。列名として有効であるときは，列として識別されます。列としては無効で，SQL 変数として有効であるときは，SQL 変数として識別されます。列，及び SQL 変数としては無効で，SQL パラメタとして有効であるときは，SQL パラメタとして識別されます。列，SQL 変数，及び SQL パラメタとして無効であるときは，文法上エラーとなります。

    ただし，ハンドラ宣言中は，そのハンドラ宣言の外側の文ラベル，又は〔認可識別子.〕ルーチン識別子を継承しません。したがって，ハンドラ宣言中で SQL 変数又は SQL パラメタを，文ラベル又は〔認可識別子.〕ルーチン識別子で修飾した場合，該当するハンドラ宣言中に宣言した文ラベルだけ有効と

なります。列，SQL 変数，及び SQL パラメタの有効範囲の例を次に示します。INTEGER 型の列 Y 及び Z がある表 T1 を定義しているものとします。

```
                                                                範囲A
CREATE PROCEDURE PPP(OUT Y INTEGER)
  BEGIN
    DECLARE X INTEGER;

    ┌──────────────────────────────────────────────────────────────┐
                                                          範囲B
    AAA: BEGIN
        DECLARE X INTEGER;
        DECLARE Z INTEGER;
        DECLARE CN1 CONDITION FOR SQLCODE VALUE -800;
        DECLARE CN2 CONDITION FOR SQLCODE VALUE -210;

      ┌──────────────────────────────────────────────────────────┐
                                                        範囲C
      DECLARE EXIT HANDLER FOR CN1
      AAA: BEGIN
          DECLARE X INTEGER;
                :
          SET X = 10;  ..................1
          SET Z = 10;  ..................2
          SET AAA.X = 10;  ..............3
          SET AAA.Z = 10;  ..............4
          SET PPP.Y = 10;  ..............5
                :
      END AAA;
      └──────────────────────────────────────────────────────────┘

      ┌──────────────────────────────────────────────────────────┐
                                                        範囲D
      DECLARE EXIT HANDLER FOR CN2
      BEGIN
          SET Y = 10;  ..................6
          SET AAA.X = 10;  ..............7
          SET PPP.Y = 10;  ..............8
                :
      END;
      └──────────────────────────────────────────────────────────┘

                              :
      SELECT Y INTO X FROM T1 WHERE Y = Z;  .......9

      ┌──────────────────────────────────────────────────────────┐
                                                        範囲E
      BBB: BEGIN
          DECLARE X INTEGER;
          DECLARE Y INTEGER;
                :
          SET X = 10;  ..................10
          SET Y = 10;  ..................11
                :

        ┌──────────────────────────────────────────────────┐
                                                  範囲F
        CCC: BEGIN
            SET X = 10;  ..............12
            SET Y = 10;  ..............13
            SET PPP.Y = 10;  ..........14
                  :
        END CCC;
        └──────────────────────────────────────────────────┘

                :
      END BBB;
      └──────────────────────────────────────────────────────────┘

                              :
    END AAA;
    └──────────────────────────────────────────────────────────────┘
END
```

〔説明〕

1.範囲 C で定義された SQL 変数「X」に 10 が代入されます。

2.範囲 B で定義された SQL 変数「Z」に 10 が代入されます。

3.範囲 C で定義された SQL 変数「X」に 10 が代入されます。

4.エラー（該当するハンドラ宣言中に有効な「Z」がない）となります。

5.エラー（ハンドラ宣言中のため，「PPP」は継承しない）となります。

6.SQL パラメタ「Y」に 10 が代入されます。

7.エラー（ハンドラ宣言中のため，「AAA」は継承しない）となります。

8.エラー（ハンドラ宣言中のため，「PPP」は継承しない）となります。

9.範囲 B で定義された SQL 変数「X」に，表 T1 の列「Y」の値が代入されます。また，1 行 SELECT 文中の WHERE 句中に指定している「Y」，及び「Z」は，表 T1 の列「Y」，及び「Z」が使用されます。

10.範囲 E で定義された SQL 変数「X」に 10 が代入されます。

11.範囲 E で定義された SQL 変数「Y」に 10 が代入されます。

12.範囲 E で定義された SQL 変数「X」に 10 が代入されます。

13.範囲 E で定義された SQL 変数「Y」に 10 が代入されます。

14.SQL パラメタ「Y」に 10 が代入されます。

11. SQL 手続き文中を除く，ルーチン制御 SQL 中だけで使用できる値指定は次のとおりです。括弧内は，それぞれのデータ型を示します。

    - SQLCODE（INTEGER）
    - SQLCODE_OF_LAST_CONDITION（INTEGER）
    - SQLCOUNT（INTEGER）
    - SQLERRM_OF_LAST_CONDITION（VARCHAR(254)）

    SQLCODE 及び SQLCOUNT は，ルーチン制御 SQL を除く直前の SQL 手続き文を実行した結果のリターンコードが 100 であることの判定や，更新行数の参照に使用します。

    SQLCODE_OF_LAST_CONDITION 又は SQLERRM_OF_LAST_CONDITION は，ルーチン制御 SQL を除く SQL 手続き文を実行した結果のリターンコードが 0 以外になった最後のリターンコード及びメッセージを，例外ハンドラ中で参照する場合に使用します。

12. 添字のない繰返し列は，CASE 式の探索条件に直接指定した，繰返し列を除くスカラ演算に指定できません。

13. 添字 ANY は，CASE 式の探索条件に指定した，繰返し列を除くスカラ演算に指定できません。

14. スカラ副問合せについては，「副問合せ」を参照してください。

2.10　四則演算

2.10.1　四則演算の形式と規則

(1)　四則演算の種類と機能

SQL の値式中に四則演算を使用した検索ができます。四則演算の種類と機能を次の表に示します。

表 2-13　四則演算の種類と機能

四則演算		意味	機能
＋	（単項演算）	正符号	符号を反転しません。
	（2 項演算）	加算	第 1 演算項に第 2 演算項を加えます。
－	（単項演算）	負符号	符号を反転します。
	（2 項演算）	減算	第 1 演算項から第 2 演算項を減らします。
＊	（2 項演算）	乗算	第 1 演算項に第 2 演算項を掛けます。
／	（2 項演算）	除算	第 1 演算項を第 2 演算項で割ります。

(2)　四則演算結果のデータ型

第 1 演算項データ型が，SMALLINT，INTEGER，DECIMAL，SMALLFLT，及び FLOAT の場合の，四則演算（2 項演算）の演算項のデータ型と演算結果のデータ型の関係を次の表に示します。

表 2-14　四則演算（2 項演算）の演算項のデータ型と演算結果のデータ型の関係

第 1 演算項データ型	第 2 演算項データ型				
	SMALLINT	INTEGER	DECIMAL	SMALLFLT	FLOAT
SMALLINT	INTEGER	INTEGER	DECIMAL	SMALLFLT	FLOAT
INTEGER	INTEGER	INTEGER	DECIMAL	FLOAT	FLOAT
DECIMAL	DECIMAL	DECIMAL	DECIMAL	FLOAT	FLOAT
SMALLFLT	SMALLFLT	FLOAT	FLOAT	SMALLFLT	FLOAT
FLOAT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT

第 1 演算項データ型が，CHAR，VARCHAR，MCHAR，及び MVARCHAR の場合，又は第 1 演算項データ型が SMALLINT，INTEGER，DECIMAL，SMALLFLT，及び FLOAT の場合で，第 2 演算項データ型が，CHAR，VARCHAR，MCHAR，及び MVARCHAR のときの，四則演算（2 項演算）の演算項のデータ型と演算結果のデータ型の関係について説明します。

文字データはその値の文字列表現によって次の数データ型に変換されます。演算結果のデータ型は変換後の数データ型を用いて，表「四則演算（2 項演算）の演算項のデータ型と演算結果のデータ型の関係」によって決定します。

- 整数の文字列表現は，INTEGER 型に変換されます。
- 10 進数の文字列表現は，DECIMAL 型に変換されます。
- 浮動小数点数の文字列表現は，FLOAT 型に変換されます。

四則演算は，既に示したデータ型で演算されます。単項演算の場合の演算結果のデータ型は，演算項のデータ型と同じです。四則演算の結果のデータ型が，DECIMAL の場合の結果の精度と位取りを次の表に示します。

表 2-15　四則演算の結果のデータ型が DECIMAL の場合の結果の精度と位取り

四則演算結果のデータ型	精度と位取り		
	加減算の場合	乗算の場合	除算の場合
DECIMAL（p，s）	p = 1 + max（p1−s1，p2−s2）+ s s = max（s1，s2）	p = p1 + p2 s = s1 + s2	P = max_prec s = max（0，max_prec−((p1−s1) + s2)）
DECIMAL（p'，s'） （p > max_prec の場合）	p'= max_prec s'= max（s1，s2）		該当しません。

注 1

第 1 演算項のデータ型：DECIMAL（p1，s1）

第 2 演算項のデータ型：DECIMAL（p2，s2）

注 2

max_prec は DECIMAL 型の精度の最大値です。max_prec について，次の表に示します。なお，pd_sql_dec_op_maxprec オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

表 2-16　DECIMAL 型の精度の最大値

システム共通定義 pd_sql_dec_op_maxprec オペランドの値	第 1 演算項の精度 p1 と第 2 演算項の精度 p2	max_prec の値
29	p1≦29 かつ p2≦29	29
	p1 > 29 又は p2 > 29	38
38	任意	38

注 3

INTEGER は，DECIMAL（10，0）として扱われます。

SMALLINT は，DECIMAL（5，0）として扱われます。

(3) 規則

四則演算を使用する場合の規則については，次に示す規則と，「値式，値指定，及び項目指定」の共通規則を参照してください。

1. 四則演算は，数データに対して実行します。文字データを指定した場合，文字データを数データに変換してから四則演算を実行します。
2. 次の結果が文字データとなる値式は四則演算に指定できません。
   - USER
   - SQLERRM_OF_LAST_CONDITION
3. 文字データを含む四則演算を次の場所に指定する場合，文字データとして指定できるのは文字列定数，又は混在文字列定数だけです。
   - 選択式
   - 集合関数の引数
   - ユーザ定義関数（システム定義スカラ関数を含む）の引数
   - GROUP BY 句
   - 分割列に対する VALUES 句の値式
   - 新値相関名で参照する INSERT 文の VALUES 句，又は UPDATE 文の SET 句の値式
   - 旧値相関名で参照する UPDATE 文 SET 句の値式
4. 四則演算子（＋，－，＊，／）の両側に埋込み変数，又は?パラメタだけの値式は指定できません。
5. 異なるデータ型の四則演算をネストさせる場合には，途中の演算結果に注意する必要があります。
6. 演算項中にウィンドウ関数は指定できません。
7. 演算項中に予備列は指定できません。

(4) 留意事項

1. 次に示す四則演算の規則は，オーバフローエラー抑止が設定されている場合は，エラーになりません。
   - 除算で第 2 演算項の値に 0 を指定した場合は，エラーになります。
   - 計算途中でオーバフローが発生した場合は，エラーになります。

   なお，オーバフローエラー抑止が設定されている場合の演算結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。
2. DECIMAL の除算で，被除数の整数部のけた数（p1－s1）と，除数の小数部のけた数（s2）の合計が max_prec 以上の場合（(p1－s1）＋ s2≧max_prec)，除算の結果の位取りは 0 となります。小数点以下のけたを求めたい場合，除算前にスカラ関数 DECIMAL を使用して，被除数の精度（p1)，又は除数の位取り（s2）を小さくしてください。

（参考）

表「四則演算の結果のデータ型が DECIMAL の場合の結果の精度と位取り」から，次の式が求められます。

p1 ＝ max_prec－s ＋ s1－s2

s2 ＝ max_prec－s ＋ s1－p1

（例）

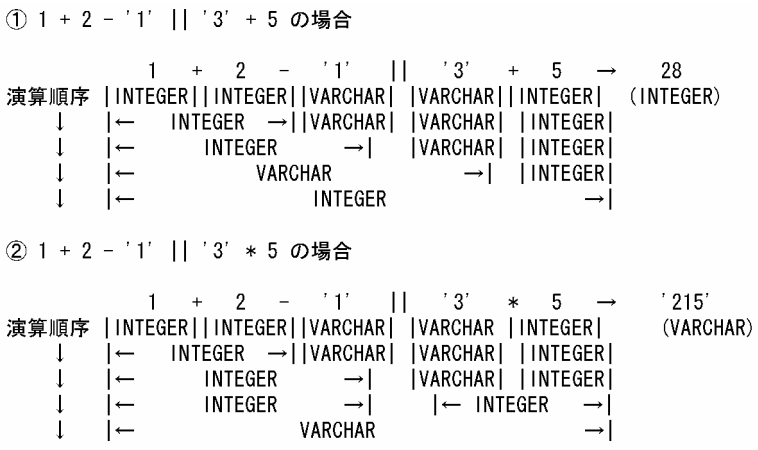
システム共通定義の pd_sql_dec_op_maxprec オペランドの値が 29 の場合，表 T1 の列 C1（DECIMAL（12，2））及び列 C2（DECIMAL（12，2））のそれぞれの合計値（SUM）の除算を求めると，DEC（29，2）と DEC（29，2）による除算となり，その結果は DEC（29，0）となります。小数点以下 2 けたまで求めたい場合は，スカラ関数 DECIMAL を使用して，被除数の精度を小さくします。

```
SELECT DEC(SUM(C1),27,2)/SUM(C2) FROM T1
```

3. 四則演算と連結演算の混在する場合，結果のデータ型は演算順序，各演算の結果によって決まります。

（例）

図 2-5　結果のデータ型



2.11　日付演算

2.11.1　日付演算の形式と規則

(1)　機能

SQL 中に日付演算を指定すれば，日付，又は日間隔データの演算を使用した検索・更新ができます。

日付同士の減算，日付の日間隔の加減算，日間隔同士の加減算，及び日間隔に対する整数の乗除算ができます。

(2)　演算できるデータ

日付データ，又は日間隔データと日付演算ができる値を次の表に示します。

表 2-17　日付演算できるデータ

演算の種類	日付演算できる値	
	日付データ	日間隔データ
加算	• 日間隔データ（INTERVAL YEAR TO DAY） • 日間隔を 10 進数表現した定数 • ラベル付き間隔（YEAR〔S〕，MONTH〔S〕，DAY〔S〕）	• 日付データ（DATE） • 日付を既定の文字列表現で指定した定数 • 日間隔データ（INTERVAL YEAR TO DAY） • 日間隔を 10 進数表現した定数
減算	• 日付データ（DATE） • 日付を既定の文字列表現で指定した定数 • 日間隔データ（INTERVAL YEAR TO DAY） • 日間隔を 10 進数表現した定数 • ラベル付き間隔（YEAR〔S〕，MONTH〔S〕，DAY〔S〕）	• 日間隔データ（INTERVAL YEAR TO DAY） • 日間隔を 10 進数表現した定数
乗算，除算	指定できません。	• 整数データ（INTEGER，SMALLINT）
単項演算	日付データ，及びラベル付き間隔には，指定できません。	

(3)　ラベル付き間隔

ラベル付き間隔は，日付演算をする場合に，数値とそれに続く間隔キーワードで特定の時間単位を表します。ラベル付き間隔は，日付データに対する日間隔データの加減算の第 2 演算項にだけ指定できます。

(4) 形式

```
（値式）｛YEAR〔S〕｜MONTH〔S〕｜DAY〔S〕｝
```

(5) 説明

1. 値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式のデータ型は，整数（SMALLINT，INTEGER）にしてください。
3. YEAR〔S〕，MONTH〔S〕，DAY〔S〕は，それぞれ年，月，日の単位を表し，語尾の S は指定しなくてもかまいません。指定例を次に示します。

   （例）

   1 か年：1 YEAR

   11 か月：11 MONTHS

   100 か日：100 DAYS
4. 値式に指定できる値の範囲を次に示します。

   YEAR〔S〕：－9998～9998

   MONTH〔S〕：－199987～119987

   DAY〔S〕：－3652058～3652058

(6) 日付演算の形式と結果のデータ型

日付演算の形式と結果のデータ型の関係を次の表に示します。

表 2-18　日付演算の形式と結果のデータ型の関係

演算の形式	結果のデータ型
日付データ － 日付データ	日間隔データ型
日付データ｛＋｜－｝{日間隔データ｜ラベル付き間隔}	日付データ型
日間隔データ ＋ 日付データ	日付データ型
日間隔データ｛＋｜－｝日間隔データ	日間隔データ型
日間隔データ｛＊｜／｝整数データ	日間隔データ型

注

日付データ，又は日間隔データとの日付演算には，埋込み変数，標識変数，及び?パラメタは指定できません。

(7)　日付演算の規則

日付演算をする場合は，次に示す規則，及び「値式，値指定，及び項目指定」の共通規則に従ってください。

(a)　日付データ同士を減算する場合の規則

- 結果のデータ型は，年数，月数，及び日数を表す日間隔データ型になります。
- 演算式を（日付 1－日付 2）としたときの結果は，次の規則に従い計算します。

日付 1≧日付 2 の場合

結果＝日付 1－日付 2

日付 1 ＜日付 2 の場合

結果＝－（日付 2－日付 1）※

日付 1 の日≧日付 2 の日の場合

結果の日＝日付 1 の日－日付 2 の日

日付 1 の日＜日付 2 の日の場合

結果の日＝日付 1 の日－日付 2 の日＋日付 2 の月の最終日

日付 2 の月＝日付 2 の月＋ 1

日付 1 の月≧日付 2 の月の場合

結果の月＝日付 1 の月－日付 2 の月

日付 1 の月＜日付 2 の月の場合

結果の月＝日付 1 の月－日付 2 の月＋ 12

日付 2 の年＝日付 2 の年＋ 1

結果の年＝日付 1 の年－日付 2 の年

注※

減算結果は，日付 2－日付 1 の結果に負符号を付けた値になります。

（例）

DATE（'1995-10-15'）-DATE（'1989-12-16'）の減算結果を求める場合

```
  1995-10-15 ← 日付1
－1989-12-16 ← 日付2
------------------------------------
  0005 か年 09 か月 30 日 ← 演算結果
```

(b)　計算手順の説明

① 結果の日＝ 15 － 16 ＋ 31 ＝ 30

12月の最終日

日付2の月＝ 12 ＋ 1 ＝ 13

② 結果の月＝ 10 － 13 ＋ 12 ＝ 9

① で求めた日付2の月

日付2の年＝ 1989 ＋ 1 ＝ 1990

③ 結果の年＝ 1995 － 1990 ＝ 5

② で求めた日付2の年

④ 上記の ①，② 及び ③ の計算式から，結果は5か年9か月30日の日間隔になります。

(8)　日付データと日間隔データとを加減算する場合の規則

1. 結果のデータ型は，日付データ型になります。
2. 演算の結果の範囲は，0001 年 01 月 01 日～9999 年 12 月 31 日の間にしてください。
3. 日間隔データ（ラベル付き間隔でない）の場合，年，月，日の順に演算します。
4. 年，月の演算の結果が，存在しない日付（小の月の 31 日，及びうるう年以外の年の 2 月 29 日）になった場合の結果は，その月の最終日に修正※されます。存在しない日付が発生し，日付が修正された場合は，SQLWARNA に'W'が設定されます。
5. 日の演算の結果がその月の最終日を超えたり，初日（1 日）を下回った場合の結果は，それぞれ年，月のけた上げ，けた下げをします。

注※

任意の月の最終日から数か月後が，必ずその月の最終日になるわけではありません。ある日付に任意の月数を加算した後，その結果の日付から同じ月数を引いても，必ず元の日付になるとは限らないので注意してください。

（例）

DATE（'1995-01-31'） ＋ 1MONTH → DATE（'1995-02-28'）

DATE（'1995-02-28'） － 1MONTH → DATE（'1995-01-28'）

1995年1月31日に1か月加算して，その結果の日付1995年2月28日から
1か月減算しても，演算の結果は元の日付1995年1月31日にはならないで，
1995年1月28日になります。

(9) 日間隔データ同士の加減算及び日間隔データの整数データによる乗除算の場合の規則

1. 結果のデータ型は，日間隔データ型になります。
2. 日数について，演算の結果が，00～99 の範囲外であれば，SQLWARNC に'W'が設定されます。また，この時の演算結果の日数には，00 未満の場合は 00 が，99 より大きい場合は 99 が設定されます。
3. 除算の結果が小数点以下となる場合，小数点以下は切り捨てられます。

(10) 留意事項

次に示す日付演算の規則で，オーバフローエラー抑止が設定されている場合，エラーになりません。

- 日付データ型のオーバフロー
- 日間隔データ型のオーバフロー
- ラベル付き間隔のオーバフロー

なお，オーバフローエラー抑止が設定されている場合の演算結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

2.12　時刻演算

2.12.1　時刻演算の形式と規則

(1)　機能

SQL 中に時刻演算を指定すれば，時刻，又は時間隔データの演算を使用した検索・更新ができます。

時刻同士の減算，時刻に対する時間隔の加減算，時間隔同士の加減算，及び時間隔に対する整数の乗除算ができます。

(2)　演算できるデータ

時刻データ，又は時間隔データと時刻演算ができる値を次の表に示します。

表 2-19　時刻演算ができる値

演算の種類	時刻演算できる値	
	時刻データ	時間隔データ
加算	• 時間隔データ（INTERVAL HOUR TO SECOND） • 時間隔を 10 進数表現した定数 • ラベル付き間隔（HOUR〔S〕，MINUTE〔S〕，SECOND〔S〕）	• 時刻データ（TIME） • 時刻を既定の文字列表現で指定した定数 • 時間隔データ（INTERVAL HOUR TO SECOND） • 時間隔を 10 進数表現した定数
減算	• 時刻データ（TIME） • 時刻を既定の文字列表現で指定した定数 • 時間隔データ（INTERVAL HOUR TO SECOND） • 時間隔を 10 進数表現した定数 • ラベル付き間隔（HOUR〔S〕，MINUTE〔S〕，SECOND〔S〕）	• 時間隔データ（INTERVAL HOUR TO SECOND） • 時間隔を 10 進数表現した定数
乗算，除算	指定できません。	• 整数データ（INTEGER，SMALLINT）
単項演算	時刻データ，及びラベル付き間隔には，指定できません。	

(3)　ラベル付き間隔

ラベル付き間隔は，時刻演算をする場合に，数値とそれに続く間隔キーワードで特定の時間単位を表します。ラベル付き間隔は，時刻データに対する時間隔データの加減算の第 2 演算項にだけ指定できます。

(4) 形式

```
（値式）｛HOUR〔S〕｜MINUTE〔S〕｜SECOND〔S〕｝
```

(5) 説明

1. 値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式のデータ型は，整数（SMALLINT，INTEGER）にしてください。
3. HOUR〔S〕，MINUTE〔S〕，SECOND〔S〕は，それぞれ時間，分，秒の単位を表し，語尾の S は指定してもしなくてもかまいません。指定例を次に示します。

   （例）

   1 時間：1 HOUR

   23 分：23 MINUTES

   100 秒：100 SECONDS
4. 値式に指定できる値の範囲を次に示します。

   HOUR〔S〕：－23～23

   MINUTE〔S〕：－1439～1439

   SECOND〔S〕：－86399～86399

(6) 時刻演算の形式と結果のデータ型

時刻演算の形式と結果のデータ型の関係を次の表に示します。

表 2-20　時刻演算の形式と結果のデータ型の関係

演算の形式	結果のデータ型
時刻データ － 時刻データ	時間隔データ型
時刻データ｛＋｜－｝{時間隔データ｜ラベル付き間隔}	時刻データ型
時間隔データ ＋ 時刻データ	時刻データ型
時間隔データ｛＋｜－｝時間隔データ	時間隔データ型
時間隔データ｛＊｜／｝整数データ	時間隔データ型

注

時刻データ，又は時間隔データとの日付演算には，埋込み変数，標識変数，及び?パラメタは指定できません。

(7)　時刻演算の規則

時刻演算をする場合は，次に示す規則，及び「値式，値指定，及び項目指定」の共通規則に従います。

(a)　時刻データ同士を減算する場合の規則

- 結果のデータ型は，時，分，秒を表す時間隔データ型になります。
- 時刻 1 又は時刻 2 の秒が 60 秒以上の場合，演算する前に 59 秒に修正されます。
- 演算式を（時刻 1－時刻 2）としたときの結果は，次の規則に従い計算します。

時刻 1≧時刻 2 の場合

結果＝時刻 1－時刻 2

時刻 1 ＜時刻 2 の場合

結果＝－（時刻 2－時刻 1）

時刻 1 の秒≧時刻 2 の秒の場合

結果の秒＝時刻 1 の秒－時刻 2 の秒

時刻 1 の秒＜時刻 2 の秒の場合

結果の秒＝時刻 1 の秒－時刻 2 の秒＋ 60

時刻 2 の分＝時刻 2 の分＋ 1

時刻 1 の分≧時刻 2 の分の場合

結果の分＝時刻 1 の分－時刻 2 の分

時刻 1 の分＜時刻 2 の分の場合

結果の分＝時刻 1 の分－時刻 2 の分＋ 60

時刻 2 の時＝時刻 2 の時＋ 1

結果の時＝時刻 1 の時－時刻 2 の時

（例）TIME（'13:10:15'）-TIME（'11:50:59'）の減算結果を求める場合

```
  13:10:15 ← 時刻１
－11:50:59 ← 時刻２
-----------------------
  1時間 19分 16秒 ← 演算結果
```

(b)　計算手順の説明

① 結果の秒＝ 15 － 59 ＋ 60 ＝ 16
　　└ 1分間

時刻2の分＝ 50 ＋ 1 ＝ 51

② 結果の分＝ 10 － 51 ＋ 60 ＝ 19
　　└ ① で求めた時刻2の分

時刻2の時＝ 11 ＋ 1 ＝ 12

③ 結果の時＝ 13 － 12 ＝ 1
　　└ ② で求めた時刻2の時

④ 上記の ①，② 及び ③ の計算式から，結果は1時間19分16秒の時間隔になります。

(8)　時刻データと時間隔データとを加減算する場合の規則

1. 結果のデータ型は，時刻データ型になります。
2. 演算の結果の範囲は，0 時 0 分 0 秒～23 時 59 分 59 秒の間にしてください。
3. 時間隔データ（ラベル付き間隔以外）の場合，時，分，秒の順に演算します。
4. うるう秒を含む時刻データの場合，演算結果は次のようになります。
   - 秒が 60 秒以上の場合，演算する前に 59 秒に修正されます。
     （例）
     ’12:34:60’ ＋ 1 MINUTE の演算結果は’12:35:59’
     ’12:34:60’ ＋ 1 SECOND の演算結果は’12:35:00’
   - 演算結果には，うるう秒を含みません。

(9)　時間隔データ同士の加減算及び時間隔データの整数データによる乗除算の場合の規則

1. 結果のデータ型は，時間隔データ型になります。
2. 演算結果の範囲は，－99 時間 59 分 59 秒間～99 時間 59 分 59 秒間にしてください。
3. 分，秒が 60 を超える場合は，それぞれ時，分にけた上げをします。

4. 除算は，時，分，秒を秒に通算した形に変換し，結果の小数点以下は切り捨てられます。

(10) 留意事項

次に示す時刻演算の規則で，オーバフローエラー抑止が設定されている場合は，エラーになりません。

- 時刻データ型のオーバフロー
- 時間隔データ型のオーバフロー
- ラベル付き間隔のオーバフロー

なお，オーバフローエラー抑止が設定されている場合の演算結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

2.13　連結演算

2.13.1　連結演算の形式と規則

(1)　機能

値式中に連結演算を指定すると，複数のデータ列（文字列又はバイナリ列）を指定した順に連結して一つのデータ列にします。

BLOB 型，又は最大長が 32,001 バイト以上の BINARY 型の連結演算は，UPDATE 文の SET 句の更新値にだけ指定できます。BLOB 型，又は定義長が 32,001 バイト以上の BINARY 型の連結演算については，「4.36　UPDATE 文　形式 1（データ更新）」の「連結演算を使用して BLOB 型，又は定義長が 32,001 バイト以上の BINARY 型の列を更新する場合の規則」を参照してください。形式は，文字列データの連結演算と同じです。

文字列データ，及び最大長が 32,000 バイト以下の BINARY 型データの連結演算について，連結できるデータ型と演算結果のデータ型を次の表に示します。

表 2-21　連結できるデータ型と演算結果のデータ型（1/2）

第 1 演算項のデータ型		第 2 演算項のデータ型						
		文字データ		各国文字データ		混在文字データ		バイナリデータ
		CHAR	VARCHAR	NCHAR	NVARCHAR	MCHAR	MVARCHAR	BINARY
文字データ	CHAR	CHAR※1	VARCHAR	×	×	MCHAR※1	MVARCHAR	×
	VARCHAR	VARCHAR	VARCHAR	×	×	MVARCHAR	MVARCHAR	BINARY※2
各国文字データ	NCHAR	×	×	NCHAR※1	NVARCHAR	×	×	×
	NVARCHAR	×	×	NVARCHAR	NVARCHAR	×	×	×
混在文字データ	MCHAR	MCHAR※1	MVARCHAR	×	×	MCHAR※1	MVARCHAR	×
	MVARCHAR	MVARCHAR	MVARCHAR	×	×	MVARCHAR	MVARCHAR	×
バイナリデータ	BINARY	×	BINARY※2	×	×	×	×	BINARY

第 1 演算項のデータ型		第 2 演算項のデータ型						
		文字データ		各国文字データ		混在文字データ		バイナリデータ
		CHAR	VARCHAR	NCHAR	NVARCHAR	MCHAR	MVARCHAR	BINARY
数データ※3※4	INTEGER，SMALLINT，及び DECIMAL	VARCHAR	VARCHAR	×	×	MVARCHAR	MVARCHAR	×
	FLOAT，及び SMALLFLT	CHAR※1	VARCHAR	×	×	MCHAR※1	MVARCHAR	×

（凡例）

×：指定できません。

注※1

連結後の長さが CHAR，NCHAR，又は MCHAR の最大長を超える場合，VARCHAR，NVARCHAR，又は MVARCHAR になります。

注※2

BINARY 型には，16 進文字列定数だけ演算項に指定できます。

注※3

数データは次のように変換されます。

- INTEGER 型の数データは，VARCHAR(11)の文字データに変換されます。
- SMALLINT 型の数データは，VARCHAR(6)の文字データに変換されます。
- DECIMAL(p,0)型の数データは，VARCHAR(p ＋ 1)の文字データに変換されます。
- DECIMAL(p,s)型の数データは，VARCHAR(p ＋ 2)の文字データに変換されます。
- FLOAT 型又は SMALLFLT 型の数データは，CHAR(23)の文字データに変換されます。

注※4

数データを連結する場合，数データは次のように変換されます。

- 数データと文字データの連結の場合，数データは文字データの文字集合に変換されます。
- 数データと混在文字データの連結の場合，数データは混在文字データに変換されます。
- 数データと数データの連結の場合，数データは既定文字集合に変換されます。

表 2-22　連結できるデータ型と演算結果のデータ型（2/2）

第 1 演算項のデータ型		第 2 演算項のデータ型	
		数データ※2※3	
		INTEGER，SMALLINT，及び DECIMAL	FLOAT，及び SMALLFLT
文字データ	CHAR	VARCHAR	CHAR※1
	VARCHAR	VARCHAR	VARCHAR
各国文字データ	NCHAR	×	×
	NVARCHAR	×	×
混在文字データ	MCHAR	MVARCHAR	MCHAR※1
	MVARCHAR	MVARCHAR	MVARCHAR
バイナリデータ	BINARY	×	×
数データ※2	INTEGER，SMALLINT，及び DECIMAL	VARCHAR	VARCHAR
	FLOAT，及び SMALLFLT	VARCHAR	CHAR※1

（凡例）

×：指定できません。

注※1

表「連結できるデータ型と演算結果のデータ型（1/2）」の注※1 を参照してください。

注※2

表「連結できるデータ型と演算結果のデータ型（1/2）」の注※3 を参照してください。

注※3

表「連結できるデータ型と演算結果のデータ型（1/2）」の注※4 を参照してください。

演算結果のデータ長を次の表に示します。

表 2-23　連結演算の結果のデータ長

演算結果のデータ型	データ長（可変長データの場合，最大長）
CHAR(n)	n = n1 + n2（ただし，n > 255 の場合，結果のデータ型は VARCHAR）
VARCHAR(n)	n = n1 + n2（ただし，n > 32,000 の場合はエラー）
NCHAR(n)	n = n1 + n2（ただし，n > 127 の場合，結果のデータ型は NVARCHAR）
NVARCHAR(n)	n = n1 + n2（ただし，n > 16,000 の場合はエラー）
MCHAR(n)	n = n1 + n2（ただし，n > 255 の場合，結果のデータ型は MVARCHAR）
MVARCHAR(n)	n = n1 + n2（ただし，n > 32,000 の場合はエラー）

演算結果のデータ型	データ長（可変長データの場合，最大長）
BINARY(n)	n = n1 + n2（ただし，n > 32,000 の場合，UPDATE 文の SET 句以外ではエラー）

（凡例）

n1：第 1 演算項のデータ長

n2：第 2 演算項のデータ長

注 1

n は，演算結果のデータ型が CHAR，VARCHAR，MCHAR，MVARCHAR，及び BINARY の場合はバイト数を，NCHAR，及び NVARCHAR の場合は文字数を示します。

注 2

第 1 演算項，又は第 2 演算項が長さ 0 の文字列定数（各国，及び混在を含む）の場合，n1，又は n2 を 0 としてください。ただし，すべての連結演算の結果データ長が 0 の場合，n は 1 になります。

(2) 形式

値式||一次子

(3) 規則

文字列データの連結演算をする場合は，「値式，値指定，及び項目指定」の共通規則にも従ってください。

1. 連結演算の演算項に，埋込み変数，又は?パラメタを直接指定できません。
2. 連結演算の演算項に，予備列を指定できません。
3. 演算結果は，一次子，又は値式の非ナル値制約の有無に関係なく，ナル値を許します。
4. 連結演算の結果のデータ長（最大長）が可変長データの最大長を超えた場合はエラーになります。
5. 連結演算の結果のデータ型が BINARY 型の場合，結果のデータ長（最大長）が 32,000 バイトを超えるときは，UPDATE 文の SET 句以外ではエラーになります。
6. 文字集合が異なる場合，文字データ同士の連結として連結できません。ただし，次に示す値式は，連結相手の文字集合に変換して連結できます。
   - 文字列定数

2.14　集合関数

2.14.1　集合関数の形式と規則

(1)　機能

集合関数を SQL 中に指定すれば，平均値，合計値，最大値，最小値，及び行数の算出，並びに XML 型の値の結合ができます。

XMLAGG 集合関数の詳細については，「XMLAGG」を参照してください。

集合関数の機能を次の表に示します。

表 2-24　集合関数の機能

項 目	集合関数						
	AVG	SUM	MAX	MIN	COUNT	COUNT_FLOAT	XMLAGG
機能	平均値の算出	合計値の算出	最大値の算出	最小値の算出	行数の算出	行数の算出	XML 型の値の結合
ナル値の扱い	無視	無視	無視	無視	無視※1	無視※1	無視
DISTINCT 指定の意味	指定した値式の中で，値が重複する行を除いた後の平均値	指定した値式の中で，値が重複する行を除いた後の合計値	指定しても意味がない	指定しても意味がない	指定した値式の中で，値が重複する行を除いた行数	指定した値式の中で，値が重複する行を除いた行数	指定できません
適用対象が 0 件※3，又はナル値だけの集合の場合の関数値	ナル値	ナル値	ナル値	ナル値	0	0	ナル値
引数中での繰返し列の指定可否※2	指定できません	指定できません	指定できます	指定できます	指定できません	指定できません	指定できません

注※1

集合関数では，基本的にナル値を無視しますが，COUNT（＊）及び COUNT_FLOAT（＊）の場合はナル値に関係なく，条件を満足するすべての行数を算出します。

注※2

集合関数 MAX 及び MIN については，FLAT 指定を記述することで，引数に繰返し列を指定できます。繰返し列を指定した場合，集合関数 MAX 及び MIN は適用対象である行中すべての要素から，最

大値又は最小値を算出します。なお，列全体の値がナル値（要素数が 0 の繰返し列）の場合は，算出の対象外となります。

注※3

GROUP BY 句又は HAVING 句の指定がある場合，行数が 0 のグループの算出結果は出力しません。

集合関数の引数のデータ型と集合関数の値のデータ型の関係を次の表に示します。

表 2-25　列指定での列のデータ型と集合関数の値のデータ型の関係

集合関数の引数のデータ型	AVG	SUM	MAX 及び MIN	COUNT	COUNT_FLOAT	XMLAGG
INTEGER	INTEGER	INTEGER	INTEGER	INTEGER	FLOAT	×
SMALLINT	INTEGER	INTEGER	SMALLINT	INTEGER	FLOAT	×
DECIMAL（p，s）	DECIMAL（max_prec※3，max_prec※3−p + s）	DECIMAL（max_prec※3，s）	DECIMAL（p，s）	INTEGER	FLOAT	×
FLOAT	FLOAT	FLOAT	FLOAT	INTEGER	FLOAT	×
SMALLFLT	SMALLFLT	SMALLFLT	SMALLFLT	INTEGER	FLOAT	×
INTERVAL YEAR TO DAY	×	×	INTERVAL YEAR TO DAY	INTEGER	FLOAT	×
INTERVAL HOUR TO SECOND	×	×	INTERVAL HOUR TO SECOND	INTEGER	FLOAT	×
CHAR(n)	×	×	CHAR(n) ※1	INTEGER	FLOAT	×
VARCHAR(n)	×	×	VARCHAR(n) ※1	INTEGER	FLOAT	×
NCHAR(n)	×	×	NCHAR(n)	INTEGER	FLOAT	×
NVARCHAR(n)	×	×	NVARCHAR(n)	INTEGER	FLOAT	×
MCHAR(n)	×	×	MCHAR(n)	INTEGER	FLOAT	×
MVARCHAR(n)	×	×	MVARCHAR(n)	INTEGER	FLOAT	×
DATE	×	×	DATE	INTEGER	FLOAT	×
TIME	×	×	TIME	INTEGER	FLOAT	×
TIMESTAMP	×	×	TIMESTAMP	INTEGER	FLOAT	×
BINARY(n)※2	×	×	BINARY(n)	INTEGER	FLOAT	×

集合関数の引数のデータ型	AVG	SUM	MAX 及び MIN	COUNT	COUNT_FLOAT	XMLAGG
BLOB(n)	×	×	×	×	×	×
BOOLEAN	×	×	×	×	×	×
XML	×	×	×	×	×	XML
抽象データ型（XML 型を除く）	×	×	×	×	×	×

表 2-26　定数指定での定数のデータ型と集合関数の値のデータ型の関係

集合関数の引数のデータ型	AVG	SUM	MAX 及び MIN	COUNT	COUNT_FLOAT	XMLAGG
INTEGER	INTEGER	INTEGER	INTEGER	INTEGER	FLOAT	×
DECIMAL（p，s）	DECIMAL（max_prec※3，max_prec※3−p + s）	DECIMAL（max_prec※3，s）	DECIMAL（p，s）	INTEGER	FLOAT	×
FLOAT	FLOAT	FLOAT	FLOAT	INTEGER	FLOAT	×
その他	×	×	×	×	×	×

（凡例）

×：使用できません。

注※1

結果の文字集合は集合関数の引数に指定した値式の文字集合になります。

注※2

n は 32,000 以下にしてください。

注※3

max_prec は DECIMAL 型の精度の最大値です。次の表に，max_prec の値について示します。なお，pd_sql_dec_op_maxprec オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

表 2-27　DECIMAL 型の精度の最大値

システム共通定義 pd_sql_dec_op_maxprec オペランドの値	集合関数の引数の精度 p	max_prec の値
29	p≦29	29
	p > 29	38
38	任意	38

(2) 形式

```
  集合関数：：＝ {COUNT（＊） | COUNT_FLOAT（＊） | ALL集合関数 | DISTINCT集合関数 | XMLAGG集合関数}
  ALL集合関数：：＝ {AVG | SUM | MAX | MIN | COUNT | COUNT_FLOAT} （〔ALL〕 {値式 | FLAT指定}）
  DISTINCT集合関数：：＝ {AVG | SUM | MAX | MIN | COUNT | COUNT_FLOAT} （DISTINCT {値式 | FLAT指定}）
  FLAT指定：：＝FLAT（列指定）
```

XMLAGG 集合関数の形式については，「XMLAGG」を参照してください。

(3) 副問合せを使用しない場合の規則

1. GROUP BY 句，WHERE 句，又は FROM 句のうち，最後に指定された句の結果として得られる各グループが，集合関数の入力になります。ただし，GROUP BY 句を指定しない場合，WHERE 句，又は FROM 句の結果がグループ化列（GROUP BY 句で指定した値式）を持たない一つのグループになります。また，集合関数の演算結果はグループごとに得られます。
2. 集合関数は，SELECT 句，及び HAVING 句に指定できます。
3. GROUP BY 句，HAVING 句，又は集合関数を指定した場合，SELECT 句，及び HAVING 句中の列指定は，次のどちらかにしてください。
   - グループ化列（GROUP BY 句で指定した値式）
   - 集合関数の引数中に指定
4. SELECT DISTINCT を指定した場合，DISTINCT 集合関数は指定できません。

(4) 副問合せを使用する場合の規則

1. 集合関数を副問合せ中に指定した場合，集合関数の入力対象になる集合を次に示します。
   - 集合関数は，問合せ指定（副問合せの括弧内の指定も含む）ごとに指定できます。その問合せ指定ごとに集合関数の入力対象の集合が決まります。
   - COUNT（＊）又は COUNT_FLOAT（＊）の入力対象になる集合は，COUNT（＊）又は COUNT_FLOAT（＊）を直接含む問合せ指定で決まります。そのほかの集合関数の入力対象になる集合は，その引数中で参照する表を FROM 句で指定した問合せ指定で決まります。
   - 問合せ指定中の GROUP BY 句，WHERE 句，又は FROM 句のうち，最後に指定された句の結果として得られる各グループが集合関数の入力になります。

   ただし，GROUP BY 句を指定しないと，WHERE 句，又は FROM 句の結果が，グループ化列（GROUP BY 句で指定した値式）を持たない一つのグループになります。また，集合関数の演算結果は，グループごとに得られます。
2. ある問合せ指定 Q に対する集合関数は，その問合せ指定 Q の SELECT 句，又は HAVING 句中にだけ指定できます。その HAVING 句の副問合せ中の FROM 句中の ON 探索条件，WHERE 句，及び HAVING 句にも，問合せ指定 Q の表の列を引数として参照（外への参照）すると問合せ指定 Q に対する集合関数を指定できます。

3. 問合せ指定中で GROUP BY 句，若しくは HAVING 句を指定する場合，又は SELECT 句に集合関数を指定する場合，その問合せ指定の SELECT 句，及び HAVING 句中の列指定は次に示す条件が必要です。

   SELECT 句中の列指定の場合

   - その問合せ指定中の FROM 句の表を参照している。
   - グループ化列である，又は集合関数の引数中に指定している。

   HAVING 句中の列指定の場合

   - その問合せ指定中の FROM 句の表を参照しているか，又は外側の問合せ指定中の FROM 句の表を参照（外側への参照）している。
   - その問合せ指定中の FROM 句の表を参照している場合で，グループ化列，又は集合関数の引数中に指定している。

4. 問合せ指定中に SELECT DISTINCT を指定すると，その問合せ指定に対する DISTINCT 指定の集合関数は指定できません。DISTINCT 指定の集合関数を DISTINCT 集合関数といいます。
5. 副問合せ中で集合関数の引数に，外への参照する列を含む演算は指定できません。

(5) 共通規則

1. 集合関数は，SET 句，IF 文，WHILE 文，SET 文，RETURN 文，WRITE LINE 文，ADD 句，及び GROUP BY 句に指定できません。
2. 集合関数の引数中に，埋込み変数，及び?パラメタは指定できません。

   ただし，XMLAGG 集合関数の値式に XMLQUERY 関数を指定した場合，その XMLQUERY 関数の XML 問合せ変数中の値式には埋込み変数又は？パラメタを指定できます。
3. 集合関数の引数中の値式には次を指定してください。
   - 列指定（予備列を除く）を含む値式
   - 数定数
   - 結果のデータ型が数データ型となる四則演算
   - 結果のデータ型が数データ型となるスカラ関数
   - 結果のデータ型が数データ型となる CASE 式
   - 結果のデータ型が数データ型となる CAST 指定
   - 結果のデータ型が数データ型となる関数呼び出し
4. 集合関数の引数中に，集合関数及びウィンドウ関数は指定できません。
5. 文字列データの大小関係は，集合関数の引数に指定した値式の文字集合で使用する文字コードに従います。
6. 各国文字列データの大小関係は，使用する各国文字コード（シフト JIS コード，EUC 日本語漢字コード，又は EUC 中国語漢字コード）に従います。

7. 混在文字列データの大小関係は，ASCII コードと使用する各国文字コード（シフト JIS コード，EUC 日本語漢字コード，EUC 中国語漢字コード，中国語漢字コード（GB18030），又は Unicode（UTF-8））に従います。
8. 平均値（AVG）は，有効数字以下が切り捨てられます。
9. 計算途中でオーバフローが発生した場合はエラーになります。ただし，オーバフローエラー抑止が設定されている場合は，エラーにはなりません。オーバフローエラー抑止が設定されている場合，対象になる集合関数を次に示します。
   - AVG
   - SUM
   - COUNT
   - COUNT_FLOAT

   なお，オーバフローエラー抑止が設定されている場合の演算結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。
10. COUNT_FLOAT の値が 2 の 53 乗（16 けた）以上になると，値は丸められます。
11. 集合関数の引数中にコンポネント指定は指定できません。
12. 集合関数の引数中に繰返し列を指定する場合，次の規則に従ってください。
    - 繰返し列は，FLAT 指定を記述した集合関数 MAX 又は MIN だけ指定できます。
    - FLAT 指定に記述する列指定は，添字のない繰返し列だけ指定できます。
    - GROUP BY 句に列指定以外の値式を指定した問合せ指定には，FLAT 指定を記述した集合関数は指定できません。
13. 問合せ指定ごとに DISTINCT 集合関数は二つ以上指定できません。

    ただし，次の場合には，DISTINCT 集合関数を二つ以上指定できます。
    - 集合関数 MAX 及び MIN には，異なる値式を指定できます。
    - 集合関数 AVG，SUM，COUNT 及び COUNT_FLOAT には，同じ形式で記述した値式を指定できます。
14. 集合関数の引数に指定する値式中に，副問合せは指定できません。

(6) 使用例

1. 在庫表（ZAIKO）のすべての商品の在庫量（ZSURYO）の平均を求めます。

```
SELECT AVG(ZSURYO) FROM ZAIKO
```

2. 在庫表（ZAIKO）の商品名（SNAME）がスカートの行の在庫量（ZSURYO）の合計を求めます。

```
SELECT N'スカート',SUM(ZSURYO)
    FROM ZAIKO
    WHERE SNAME = N'スカート'
```

3. 在庫表（ZAIKO）から単価（TANKA）と在庫量（ZSURYO）の積（金額）の合計を求めます。

```
SELECT SUM(TANKA * ZSURYO)
    FROM ZAIKO
```

4. 在庫表（ZAIKO）から在庫量（ZSURYO）の最大値，最小値，及び件数の合計を求めます。

```
SELECT COUNT(*),MAX(ZSURYO),MIN(ZSURYO)
    FROM ZAIKO
```

5. 次の競技結果表から，種目コードが'0001'の種目の最高点（最大値）及び最低点（最小値）を求めます。なお，得点列は要素数３の繰返し列です。

競技結果表

選手コード	種目コード	得点
A001	0001	3.0
		2.5
		4.5
A001	0002	6.5
		6.5
		7.0
A002	0001	5.0
		6.8
		6.0
A002	0002	6.5
		6.5
		7.0

実行するSQL

```
SELECT MAX(FLAT(得点)),MIN(FLAT(得点))
    FROM 競技結果表
    WHERE 種目コード='0001'
```

検索結果

MAX(EXP)	MIN(EXP)
6.8	2.5

2.15　ウィンドウ関数

2.15.1　ウィンドウ関数の形式と規則

(1)　機能

ウィンドウ関数に関連づけられたウィンドウ指定のウィンドウ枠を対象にして，結果を求める関数です。ウィンドウ関数の機能を，次の表に示します。

ウィンドウ枠には，行の集合のどの部分を集計対象とするかを指定します。ウィンドウ指定として（）だけをサポートしているバージョンでは，ウィンドウ枠は WHERE 句，又は FROM 句の結果として導出される表のすべての範囲を示します。

表 2-28　ウィンドウ関数の機能

ウィンドウ関数の種別	内容
COUNT(*)	ウィンドウ枠に対する入力行数を求める。

(2)　形式

```
ウィンドウ関数：：＝COUNT(*)
                    OVER　ウィンドウ指定
ウィンドウ指定：：＝（）
```

(3)　オペランド

- ウィンドウ関数：：＝ COUNT(*)
  OVER　ウィンドウ指定

  ウィンドウ指定：：＝（）

ウィンドウ関数についての規則を示します。

1. ウィンドウ関数は，選択式中に指定できます。
2. ウィンドウ関数 COUNT(*) OVER()の結果のデータ型は INTEGER 型になります。
3. 計算途中でオーバフローが発生した場合はエラーになります。ただし，オーバフローエラー抑止が設定されている場合はエラーになりません。なお，オーバフローエラー抑止が設定されている場合の演算結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。
4. ウィンドウ関数は，次に示す箇所には指定できません。
   - ・INSERT 文の問合せ式本体
   - ・副問合せ

- ・導出表
- ・ビュー定義の導出問合せ式
- ・WITH 句中の導出問合せ式

5. 集合演算対象となる問合せ指定，導出問合せ式，及び問合せ式本体に指定できません。
6. ウィンドウ関数を指定した場合，GROUP BY 句，及び HAVING 句は指定できません。
7. 選択式にウィンドウ関数を指定する場合，ウィンドウ関数以外の選択式を一つ以上指定してください。
8. スカラ演算中にウィンドウ関数は指定できません。
9. ウィンドウ関数を指定した場合，選択式に集合関数を指定できません。

(4) 留意事項

1. 関数の入力が空集合の場合，集合関数 COUNT(*)は 0 を出力しますが，ウィンドウ関数 COUNT(*) OVER()を指定した場合は，検索結果行そのものが出力されません。

(5) 使用例

ウィンドウ関数の使用例を次に示します。ここで使用する表の構成は次のとおりです。

得点表（TOKUTEN）

ID	KAI	TEN
A	1回目	5
A	2回目	6
B	1回目	3
B	2回目	7
C	1回目	5
C	2回目	5

1. 得点表（TOKUTEN）から得点（TEN）の高い順に，番号（ID），得点（TEN）及び総数を求める。

```
SELECT “ID”, “TEN”, COUNT(*) OVER() AS “SOUSU”
  FROM “TOKUTEN” ORDER BY “TEN” DESC
```

＜実行結果＞

ID	TEN	SOUSU
B	7	6
A	6	6
A	5	6
C	5	6
C	5	6
B	3	6

2.16　スカラ関数

データ型の変数，データの部分抽出，値の変換などを SQL の問合せ式中の選択式，又は探索条件で指定します。

スカラ関数には，次の 3 種類があります。

- **システム組込みスカラ関数**
  システム組込みスカラ関数は，スカラ関数が指定できる箇所に指定できます。
- **システム定義スカラ関数**
  システム定義スカラ関数は，関数呼出しが指定できる箇所に指定できます。
  システム定義スカラ関数には，追加で定義できる関数（拡張システム定義スカラ関数）があります。データベースの初期設定時には定義されないで，pdextfunc コマンドを実行することで定義され，指定できるようになる関数です。pdextfunc コマンドについては，マニュアル「HiRDB コマンドリファレンス」を参照してください。
- **プラグイン定義スカラ関数**
  プラグイン定義スカラ関数は，関数呼出しが指定できる箇所に指定できます。

スカラ関数の一覧を次の表に示します。

表 2-29　スカラ関数の一覧

分　類	スカラ関数	機　能	スカラ関数の種別
変換関数	INTEGER	数データを整数データに変換します。	組込み
	DECIMAL	数データを 10 進数データに変換します。	組込み
	FLOAT	数データを浮動小数点データに変換します。	組込み
	DIGITS	整数，10 進数，日間隔データ，又は時間隔データの数字の部分を抽出して，文字列表現に変換します。	組込み
	NUMEDIT	数値を編集して，文字列表現に変換します。	定義
	STRTONUM	数値の文字列表現を，数データ型に変換します。	定義
	CHARACTER	日付データ，時刻データ，又は時刻印データを，文字列表現に変換します。	組込み
	VARCHAR_FORMAT	日付データ，時刻データ，及び時刻印データを，指定した書式の文字列表現に変換します。	組込み
	DATE	指定した書式の日付の文字列表現を，日付データに変換します。 西暦 1 年 1 月 1 日からの通算日数をそれが示す日付データに変換します。	組込み

分　類	スカラ関数	機　能	スカラ関数の種別
	DAYS	日付データ，又は時刻印データを西暦 1 年 1 月 1 日からの通算日数に変換します。	組込み
	TIME	指定した書式の時刻の文字表現を，時刻データに変換します。	組込み
	TIMESTAMP	時刻印の既定の文字列表現を，時刻印データに変換します。西暦 1 年 1 月 1 日からの通算日数を，それが示す時刻印データに変換します。日付データと時刻データから，それらを組み合わせた時刻印データに変換します。	組込み
	TIMESTAMP_FORMAT	指定した書式に従った時刻印の文字列表現を，時刻印データに変換します。	組込み
	MIDNIGHTSECONDS	深夜 0 時から，指定した時刻までの秒数を求めます。	定義
	HEX	値式を 16 進文字列表現に変換します。	組込み
	ASCII	文字をアスキーコードに変換します。	定義
	CHR	アスキーコードを文字に変換します。	定義
	RADIANS	角度を，度数からラジアンに変換します。	定義
	DEGREES	角度を，ラジアンから度数に変換します。	定義
	CAST 指定	値式のデータを，指定したデータ型に変換します。 CAST 指定については，「CAST 指定」を参照してください。	該当しません
抽出関数	YEAR	日付データ，時刻印データ，又は日間隔データから年の部分を抽出します。	組込み
	MONTH	日付データ，時刻印データ，又は日間隔データから月の部分を抽出します。	組込み
	DAY	日付データ，時刻印データ，又は日間隔データから日の部分を抽出します。	組込み
	HOUR	時刻データ，時刻印データ，又は時間隔データから時間の部分を抽出します。	組込み
	MINUTE	時刻データ，時刻印データ，又は時間隔データから分の部分を抽出します。	組込み
	SECOND	時刻データ，時刻印データ，又は時間隔データから秒の部分を抽出します。	組込み
数学関数	ABS	値式の絶対値を返します。	組込み
	MOD	剰余を返します。	組込み
	CEIL	その数値以上の最小の整数を求めます。	定義
	FLOOR	その数値以下の最大の整数を求めます。	定義
	TRUNC	数値の，指定したけた未満を切り捨てます。	定義

分　類	スカラ関数	機　能	スカラ関数の種別
	ROUND	切り上げと切り捨ての境界値を指定して，指定したけたに数値を丸めます。又は，四捨五入します。	定義
	SIGN	数値の符号を 1（正），0，－1（負）で求めます。	定義
	SQRT	数値の平方根を求めます。	定義
	POWER	数値の累乗を求めます。	定義
	EXP	自然対数の底の累乗を求めます。	定義
	LN	自然対数を求めます。	定義
	LOG10	常用対数を求めます。	定義
	SIN	ラジアンで指定した角度の正弦（三角関数）を求めます。	定義
	COS	ラジアンで指定した角度の余弦（三角関数）を求めます。	定義
	TAN	ラジアンで指定した角度の正接（三角関数）を求めます。	定義
	ASIN	指定した数値の逆正弦（三角関数）をラジアンで求めます。	定義
	ACOS	指定した数値の逆余弦（三角関数）をラジアンで求めます。	定義
	ATAN	指定した数値の逆正接（三角関数）をラジアンで求めます。	定義
	ATAN2	点（x，y）の逆正接（三角関数）をラジアンで求めます。	定義
	SINH	双曲線正弦を求めます。	定義
	COSH	双曲線余弦を求めます。	定義
	TANH	双曲線正接を求めます。	定義
	PI	円周率 π を求めます。	定義
文字列操作関数	SUBSTR	データ列（文字列又はバイナリ列）中の指定した位置から，指定した文字数又は長さの部分データ列を求めます。	組込み
	LEFTSTR	文字列中の先頭から，指定した文字数の部分文字列を求めます。	定義
	RIGHTSTR	文字列中の最後から，指定した文字数の部分文字列を求めます。	定義
	UPPER	文字列データ中の英小文字を英大文字に変換します。	組込み
	LOWER	文字列データ中の英大文字を英小文字に変換します。	組込み
	TRIM	左，右，又は左右から，空白又は指定した文字を繰り返し削除します。	組込み
	TRANSL（TRANSL_LONG）	文字列中の指定した文字を，対応するほかの文字に文字変換します。	定義
	LTRIM	左から，空白又は指定した文字を繰り返し削除します。	定義

分　類	スカラ関数	機　能	スカラ関数の種別
	RTRIM	右から，空白又は指定した文字を繰り返し削除します。	定義
	LTRIMSTR	左から，指定した文字列を繰り返し削除します。	定義
	RTRIMSTR	右から，指定した文字列を繰り返し削除します。	定義
	REPLACE （REPLACE_LONG）	文字列中の部分文字列を，ほかの文字列に繰り返し置き換えます。	定義
	INSERTSTR （INSERTSTR_LONG）	指定した位置から，指定した文字数の部分文字列を文字列中から削除して，その位置に別の文字列を挿入します。	定義
	POSSTR	文字列中に，指定した位置以降に n 回数目に出現する，指定した部分文字列の文字位置を求めます。	定義
	REVERSESTR	左右を反転した文字列を求めます。	定義
	POSITION	データ列（文字列又はバイナリ列）中に，指定した位置以降に最初に出現する，指定した部分データ列の位置を求めます。	組込み
日付操作関数	NEXT_DAY	指定した日付の次の，指定した曜日の日付を求めます。	定義
	LAST_DAY	指定した日付の，年月の最終日を求めます。	定義
	DAYOFWEEK	指定した日付の曜日が，その週の何日目かを求めます。	定義
	DAYOFYEAR	指定した日付が，その年の何日目かを求めます。	定義
	DAYNAME	指定した日付の曜日を，英語で求めます。	定義
	WEEK	指定した日付が，その年の第何週目かを求めます。	定義
	WEEKOFMONTH	指定した日付が，その月の第何週目かを求めます。	定義
	MONTHNAME	指定した日付の月の，英語名を求めます。	定義
	ROUNDMONTH	切り上げの最小の日を指定し，日付の日を端数として，年月に丸めた日付を求めます。	定義
	TRUNCYEAR	年度の最初の月日を指定して，その年度の最初の日付を求めます。	定義
	QUARTER	年度の最初の月日を指定して，それが第何四半期かを求めます。	定義
	HALF	年度の最初の月日を指定して，それが上期か下期かを求めます。	定義
	CENTURY	指定した日付の世紀を求めます。	定義
	MONTHS_BETWEEN	日付間の月数を，実数で求めます。	定義
	YEARS_BETWEEN	日付間の年数を，実数で求めます。	定義
日時操作関数	DATE_TIME	日付データと時刻データとを連結して，時刻印の既定の文字列表現に変換します。	定義

分　類	スカラ関数	機　能	スカラ関数の種別
	INTERVAL_DATETIMES	既定の文字列表現の時刻印間の，日時間隔を求めます。	定義
	ADD_INTERVAL	既定の文字列表現の時刻印に，日時間隔を加算します。	定義
検査関数	ISDIGITS	文字列中のすべての文字が，数字かどうかを検査します。	定義
	IS_DBLBYTES	文字列中のすべての文字が，マルチバイト文字かどうかを検査します。	定義
	IS_SNGLBYTES	文字列中のすべての文字が，シングルバイト文字かどうかを検査します。	定義
XML 型値操作関数	XMLQUERY	XQuery を評価し，評価結果の XML 型の値を生成します。	プラグイン
	XMLSERIALIZE	XML 型の値から VARCHAR 型又は BINARY 型の値を生成します。	プラグイン
	XMLPARSE	XML 文書から XML 型の値を生成します。	プラグイン
そのほかの関数	LENGTH	値式のデータの長さを求めます。	組込み
	VALUE	値式の並びの中からナル値でない最初の値式を抽出します。	組込み
	NVL	ナル値をナル値でない値に変換します。	拡張定義
	GREATEST	引数中の最大値を求めます。	定義
	LEAST	引数中の最小値を求めます。	定義
	BIT_AND_TEST	指定した引数同士のビットごとの論理積を求め，結果を真偽で返します。	組込み
	CASE 式	条件付けられた値を指定します。 CASE 式については，「CASE 式」を参照してください。	該当しません

（凡例）

組込み：システム組込みスカラ関数

定義：システム定義スカラ関数

拡張定義：拡張システム定義スカラ関数

プラグイン：プラグイン定義スカラ関数

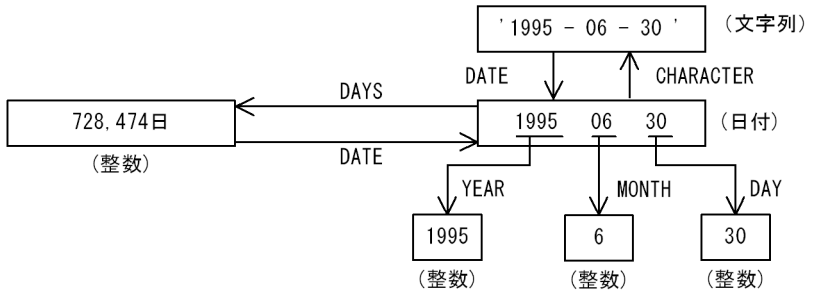
2.16.1　システム組込みスカラ関数

ここでは，システム組込みスカラ関数の文法について説明します。

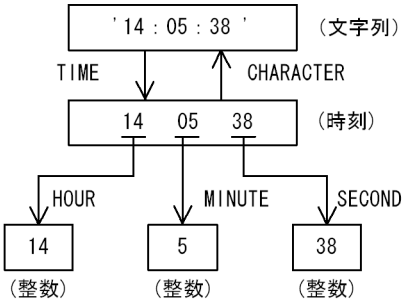
日付データ，時刻データ，時刻印データ，及び数データへのシステム組込みスカラ関数の実行例を次の図に示します。

図 2-6　日付データ，時刻データ，時刻印データ，及び数データへのシステム組込みスカラ関数の実行例

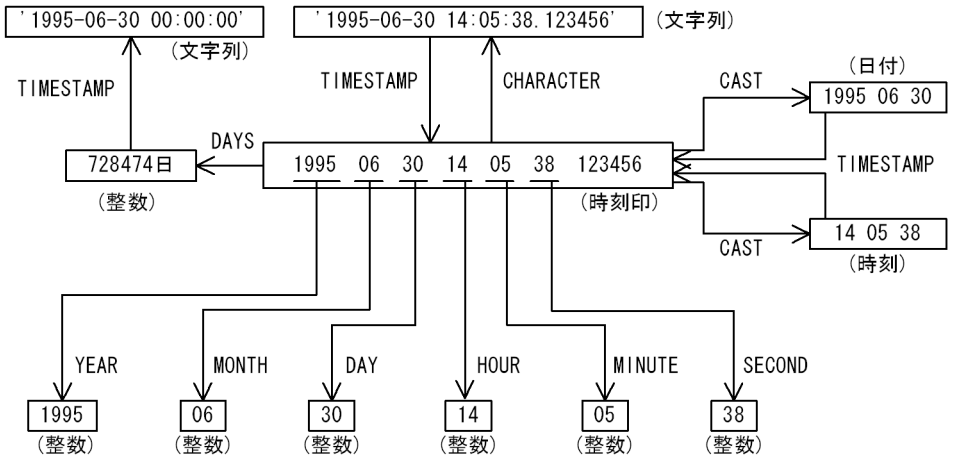
●日付（1995年6月30日の例）



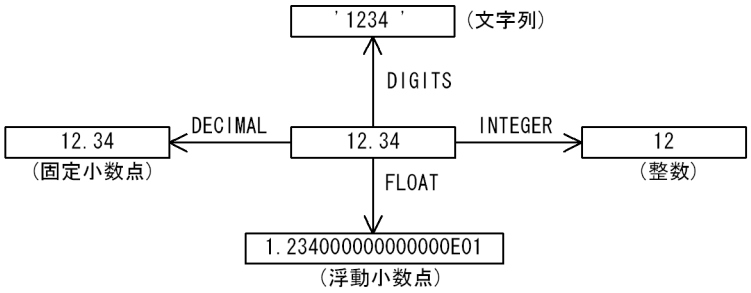
●時刻（14時5分38秒の例）



●時刻印（1995年6月30日14時5分38.123456秒の場合）



●　数データ（12.34の例）



（凡例）（　）：データ型を示します。

システム組込みスカラ関数に関係する規則を次に示します。

1. 埋込み変数，及び?パラメタを単独で値式には指定できません。ただし，四則演算などの値式（単項演算子式を除く）中には指定できます。
2. 繰返し列を引数の値式として指定する場合，添字を指定してください。ただし，添字として ANY は指定できません。
3. 予備列は引数中に指定できません。

(1) ABS

(a) 機能

値式の絶対値を返します。

(b) 形式

```
ABS（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式のデータ型は数データ型，日間隔データ，又は時間隔データにしてください。
3. 結果のデータ型は，値式のデータ型になります。
4. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。
5. 結果は値式の絶対値で表現できる値にしてください。表現できない値を指定するとオーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(2) BIT_AND_TEST

(a) 機能

値式 1 と値式 2 の間でビットごとの論理積を求め，論理積の結果のビットのどれかが 1 の場合に BOOLEAN 値（TRUE）を返します。

(b) 形式

```
BIT_AND_TEST（値式1，値式2）
```

(c) 規則

1. 値式 1 及び値式 2 に指定できるものを次に示します。
   - 定数
   - USER 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 埋込み変数，又は?パラメタ
   - 関数呼出し
   - スカラ副問合せ
2. 値式 1 及び値式 2 のデータ型は，文字データ型（CHAR 又は VARCHAR），又は最大長が 32,000 バイト以下の BINARY にしてください。値式 1 と値式 2 に指定できるデータ型の組み合わせを次の表に示します。

表 2-30 値式 1 と値式 2 に指定できるデータ型の組み合わせ（システム組込みスカラ関数 BIT_AND_TEST）

値式 1 のデータ型	値式 2 のデータ型	
	文字データ（CHAR 及び VARCHAR）	バイナリデータ（BINARY）
文字データ（CHAR 及び VARCHAR）	○	×※

値式1のデータ型	値式2のデータ型	
	文字データ (CHAR及びVARCHAR)	バイナリデータ (BINARY)
バイナリデータ（BINARY）	×※	○

（凡例）

○：指定できます。

×：指定できません。

注※

文字データの値式として，16進文字列定数だけ指定できます。

3. 値式1及び値式2の両方に，埋込み変数，又は?パラメタだけの値式は指定できません。
4. 一方の値式が埋込み変数，又は?パラメタの場合，埋込み変数，又は?パラメタのデータ型はもう一方の値式のデータ型が文字データのときはVARCHARを想定し，もう一方の値式のデータ型がバイナリデータのときはBINARYを想定します。また，データ長はもう一方の値式のデータ長を想定します。
5. 値式1と値式2が両方とも文字列データ型の場合，値式1と値式2の文字集合は同じにしてください。ただし，値式1と値式2のどちらかが次に示す値式である場合，対応する値式の文字集合に変換します。
   - 文字列定数
   - 埋込み変数（既定文字集合）
   - ?パラメタ
6. 値式1と値式2のデータ長が異なる場合，短い方のデータの右側にX'00'を埋めて，文字列長を同じにしてから，ビットごとの論理積を求めます。
7. 結果のデータ型は，BOOLEAN型になります。
8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式1又は値式2のどちらかがナル値であれば，結果もナル値になります。
9. 値式1と値式2の間でビットごとの論理積を求め，論理積の結果のビットのどれかが1の場合はTRUEとなり，そうでない場合は結果はFALSEとなります。
10. 値式1及び値式2の両方が長さ0の文字列の場合，結果はFALSEになります。

(d) 留意事項

BIT_AND_TESTは，次の箇所に指定できます。

- 探索条件中の論理述語の値式
- 戻り値データ型がBOOLEAN型のCREATE FUNCTION中のRETURN文

(e) 使用例

表T1の列（データ型：VARCHAR(2)）で，C1にビットがあるかどうかを調べます。

```
SELECT * FROM T1
    WHERE BIT_AND_TEST(C1,X'FFFF') IS TRUE
```

(3) CHARACTER

(a) 機能

日付データ，時刻データ，又は時刻印データを，文字列表現に変換します。

(b) 形式

```
CHAR〔ACTER〕（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数
   - 日付データ型の列，時刻データ型の列，及び時刻印データ型の列
   - 列指定
   - コンポネント指定
   - SQL 変数又は SQL パラメタ
   - 演算結果が日付データ型となる日付演算
   - 演算結果が時刻データ型となる時刻演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式のデータ型は，日付データ型（DATE），時刻データ型（TIME），又は時刻印データ型（TIMESTAMP）にしてください。
3. 結果のデータ型を次に示します。

   値式が日付データ型の場合：

   CHAR(10)

値式が時刻データ型の場合：

CHAR(8)

値式が時刻印データ型の場合：

CHAR(19)，CHAR(22)，CHAR(24)，又は CHAR(26)

4. 結果の値は，値式のデータ型の，既定の文字列表現となります。
5. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。
6. 結果の文字集合は，既定の文字集合となります。

(d) 使用例

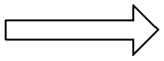
1. 表 T1 の C1 列（データ型：CHAR）のうち，6 か月より以前の日付を現在（1995-02-06）の日付に更新します。

```
UPDATE T1
    SET C1=CHAR(CURRENT_DATE)
    WHERE CHAR(CURRENT_DATE - 6 MONTHS) > C1
```

表名：T1

C1列（CHAR(10)）

C1列（CHAR(10)）
1994 - 02 - 24
1994 - 05 - 17
1994 - 08 - 19
1994 - 11 - 21
1994 - 09 - 04



実行結果

C1列（CHAR(10)）
1995 - 02 - 06
1995 - 02 - 06
1994 - 08 - 19
1994 - 11 - 21
1994 - 09 - 04

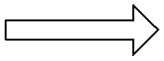
2. 表 T2 の C1 列（データ型：CHAR）のうち，0 時 0 分 0 秒のものを現在（14:24:45）の時刻に更新します。

```
UPDATE T2
    SET C1=CHAR(CURRENT_TIME)
    WHERE C1='00:00:00'
```

表名：T2

C1列（CHAR(10)）

C1列（CHAR(10)）
00:00:00
08:40:10
10:03:00
00:00:00
13:50:30



実行結果

C1列（CHAR(10)）
14:24:45
08:40:10
10:03:00
14:24:45
13:50:30

(4) DATE

(a) 機能

1. 指定した書式の日付の文字列表現を，日付データに変換します。
2. 西暦 1 年 1 月 1 日からの通算日数を，日付データに変換します。

(b) 形式

機能 1.の形式

```
DATE（値式〔，日時書式〕）
```

機能 2.の形式

```
DATE（値式）
```

(c) 機能 1.の規則

1. 値式に指定できるものを次に示します。
   - 日付の文字列表現の定数
   - CURRENT_DATE 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 演算結果が日付データ型となる日付演算
   - 連結演算
   - 集合関数（MAX，MIN）
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式は，次のデータ型にしてください。
   - 日時書式を指定した場合

     ・定義長が 8～255 バイトの文字データ型（CHAR，VARCHAR）。ただし，文字集合が UTF16 の場合は，定義長が 16～510 バイトの文字データ型。

     ・混在文字データ型（MCHAR，MVARCHAR）
   - 日時書式を指定しない場合

・定義長が 10 バイトの文字データ型（CHAR，VARCHAR）。ただし，文字集合が UTF16 の場合は，定義長が 20 バイトの文字データ型（CHAR，VARCHAR）。

・日付データ型（DATE）

3. 値式のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。
4. 値式は，日時書式で指定した書式の，日付の文字列表現にしてください。日時書式を省略した場合は，日付の既定の文字列表現にしてください。

   (例)

   日時書式'YYYY/MM/DD'の場合→'1995/06/30'

   日時書式を省略した場合→'1995-06-30'
5. 値式が日付データ型の場合，結果はその日付になります。
6. 日時書式については，「日時書式の指定」を参照してください。
7. 日時書式を指定した場合，値式と日時書式の文字集合は同じにしてください。ただし，日時書式が次に示す値式である場合，値式の文字集合に変換します。
   - 文字列定数

(d) 機能 2.の規則

1. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - コンポネント指定
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式のデータ型は，整数（INTEGER）にしてください。また，四則演算，集合関数，又は CASE 式を指定したときは，演算の結果を整数のデータ型にする必要があります。
3. 値式の範囲は，1～3652059 の間にしてください。
4. 結果は西暦 1 年 1 月 1 日から，（指定した数値－1）日後の日付になります。

   （例） 値式が 35 の場合：西暦 1 年 2 月 4 日

(e) 共通規則

1. 結果のデータ型は，日付データ型（DATE）になります。
2. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式，又は日時書式がナル値であれば，結果もナル値になります。

(f) 使用例

1. スカラ関数 CHARACTER の使用例と同様の処理を，スカラ関数 DATE を用いて処理します。

```
UPDATE T1
    SET C1=CHAR(CURRENT_DATE)
    WHERE CURRENT_DATE - 6 MONTHS > DATE(C1)
```

2. 表 T2 の C1 列（データ型：CHAR）の日付の，既定の文字列表現以外の書式（'DD/MON/YYYY'）で表現した文字列から，日付データを取得します。

```
SELECT DATE(C1,'DD/MON/YYYY') FROM T2
```

表名：T2

C1列（CHAR(11)）

C1列（CHAR(11)）
01/JAN/2000
16/JUL/2002

実行結果

実行結果
2000-01-01
2002-07-16

(5) DAY

(a) 機能

日付データ，時刻印データ，又は日間隔データから日の部分だけを抽出します。

(b) 形式

```
DAY（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 日付演算
   - 集合関数（MAX，MIN）

- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式のデータ型は，日付データ型（DATE），時刻印データ型（TIMESTAMP），又は日間隔データ型（INTERVAL YEAR TO DAY）にしてください。
3. 結果のデータ型は，整数（INTEGER）になります。
4. 値式が日付データ型，又は時刻印データ型であれば，結果は 1～31 になります。
5. 値式が日間隔データ型であれば，結果は-99～99 になります。結果が 0 以外の場合，値式の符号と結果の符号は同じになります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

(d) 使用例

表 T1 の C1 列（データ型：日付データ型）のうち，現在と同じ日付の行をすべて検索します。

```
SELECT * FROM T1
    WHERE DAY(C1)=DAY(CURRENT_DATE)
```

(6) DAYS

(a) 機能

日付データ，又は時刻印データを，西暦 1 年 1 月 1 日からの通算日数に変換します。

(b) 形式

```
DAYS（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 日付の既定の文字列表現の定数
   - CURRENT_DATE 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定

- SQL 変数，又は SQL パラメタ
- 演算結果が日付データ型となる日付演算
- 演算結果が，日付の既定の文字列表現の定数となる連結演算
- 集合関数（MAX，MIN）
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式のデータ型は，日付データ型（DATE），又は時刻印データ型（TIMESTAMP）にしてください。
3. 結果のデータ型は，整数（INTEGER）にしてください。
4. 結果は，西暦 1 年 1 月 1 日から指定した値式の日付を含めた日までの通算の日数になります。
5. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

(d) 使用例

表 T1 の C1 列の値から，現在（1995-06-30）までの日数を求めます。

```
SELECT DAYS(CURRENT_DATE) - DAYS(C1)
    FROM T1
```

(7) DECIMAL

(a) 機能

数データを 10 進数データに変換します。

(b) 形式

```
DEC〔IMAL〕 （値式〔，精度〔，位取り〕〕）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ

- 四則演算
- 集合関数
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式のデータ型として指定できるものを次に示します。
   - 数データ型
3. 精度は，1～38 の整数にしてください。精度の指定を省略した場合の仮定値は，指定した値式のデータ型によって異なります。省略時に仮定される精度を次の表に示します。

表 2-31　スカラ関数 DECIMAL の精度の省略値

データ型	精度（けた）
INTEGER	10
SMALLINT	5
DECIMAL	15
FLOAT	システム共通定義の pd_sql_dec_op_maxprec が 29 の場合※：29 システム共通定義の pd_sql_dec_op_maxprec が 38 の場合※：38
SMALLFLT	

注※

システム共通定義の pd_sql_dec_op_maxprec オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

4. 位取りは，0～精度に指定した値の範囲で指定します。また，位取りは，整数，又は整数の文字列表現にしてください。位取りを省略すると，0 が仮定されます。
5. 結果のデータ型を次の表に示します。

表 2-32　スカラ関数 DECIMAL の結果のデータ型

データ型	結果の精度	結果の位取り
DECIMAL	引数で指定した精度。省略した場合は，表「スカラ関数 DECIMAL の精度の省略値」に示すとおりです。	引数で指定した位取り。省略した場合は 0 になります。

6. 値式の整数部は，指定した，精度及び位取りで表現できる値にしてください。精度を超えると，オーバフローエラーになります。
7. 結果の値で，指定した位取りよりも下のけたの数値は切り捨てられます。

8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

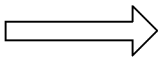
(d) 使用例

システム共通定義の pd_sql_dec_op_maxprec が 29 の場合，表 T1 の C1 列（データ型：DECIMAL（10，0））を C2 列（データ型：INTEGER）で除算した結果は 10 進数データ DEC（29，19）になりますが，不要なけたを削除して DEC（4，2）にするためにスカラ関数 DECIMAL を使用して変換します。

```
SELECT DECIMAL(C1/C2, 4, 2)
    FROM T1
```

表名：T1

C1列 (DECIMAL(10,0))	C2列 (INTEGER)
12	7
5	8
7	3



実行結果

1.71
0.62
2.33

(8) DIGITS

(a) 機能

整数，10 進数，日間隔データ，又は時間隔データの数字の部分を抽出して文字列表現に変換します。

(b) 形式

```
DIGITS（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 整数定数，又は 10 進数定数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 演算結果が日間隔データ型となる日付演算
   - 演算結果が時間隔データ型となる時刻演算
   - 集合関数
   - スカラ関数
   - CASE 式

- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式のデータ型は，次に示すどれかにしてください。
   - 整数（INTEGER，SMALLINT）
   - 固定小数点数（DECIMAL）
   - 日間隔データ型（INTERVAL YEAR TO DAY）
   - 時間隔データ型（INTERVAL HOUR TO SECOND）
3. 結果のデータ型は，固定長文字列（CHAR）になります。
4. 結果のデータ長は，値式のデータ型によって異なります。結果のデータ長を次の表に示します。

表 2-33　スカラ関数 DIGITS の結果のデータ長

データ長	結果のデータ長
INTEGER	10
SMALLINT	5
DECIMAL（p，s）	p
INTERVAL YEAR TO DAY	8
INTERVAL HOUR TO SECOND	6

（凡例）

p：精度

s：位取り

5. 結果は値式の絶対値の文字列表現になり，符号，及び小数点は含みません。また，実際の値のけた数が小さい場合，先頭に'0'が設定されます。

   （例）変換前のデータ型が DECIMAL（4，1）の場合

   15. → '0150'

   −12.4→ '0124'
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。
7. 結果の文字集合は，既定の文字集合になります。

(d)　使用例

表 T3 の C1 列（データ型：CHAR）の値と，C2 列（データ型：DECIMAL）の値が，等しいデータを検索します。

```
SELECT * FROM T3
    WHERE C1 = DIGITS(C2)
```

表名：T3

C1列（CHAR(3)）　C2列（DECIMAL(3,0)）

'△10 '	10
'020 '	20
'030 '	-30

DIGITS（C2）列（CHAR(3)）

'010 '
'020 '
'030 '

実行結果

'020 '	20
'030 '	-30

(9)　FLOAT

(a)　機能

数データを，浮動小数点データに変換します。

(b)　形式

```
FLOAT（値式）
```

(c)　規則

1. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ

2. 値式のデータ型として指定できるものを次に示します。
   - 数データ型
3. 結果のデータ型は，倍精度浮動小数点数（FLOAT）になります。
4. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

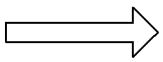
(d) 使用例

表 T1 の C1 列（データ型：INTEGER）を C2 列（データ型：INTEGER）で除算した結果を浮動小数点データで取得するために，除算をする前にスカラ関数 FLOAT を使用してどちらかのオペランドを FLOAT 型に変換します。

```
SELECT FLOAT(C1)/C2 FROM T1
```

表名：T1

C1列（INTEGER）	C2列（INTEGER）
1	3
8	100
-1	4



実行結果

C1／C2列（FLOAT）
3.333333333333333E-01
8.000000000000000E-02
-2.500000000000000E-01

(10) HEX

(a) 機能

値式を 16 進文字列表現に変換します。

(b) 形式

```
HEX（値式）
```

(c) 規則

1. システム内部で表現されている値式の形式を，16 進文字列表現に変換します。内部表現形式と実行結果の例を次の表に示します。

表 2-34 スカラ関数 HEX の内部表現形式と実行結果の例

値　式	内部の表現形式	HEX（値式）
'#AB12'	データ型：CHAR（5） 23 ｜ 41 ｜ 42 ｜ 31 ｜ 32	'2341423132'
1234	データ型：INTEGER Windows の場合：D2 ｜ 04 ｜ 00 ｜ 00 UNIX の場合：00 ｜ 00 ｜ 04 ｜ D2	Windows の場合：'D2040000' UNIX の場合：'000004D2'

値　式	内部の表現形式	HEX（値式）
1234.	データ型：DECIMAL（4，0） 01 ｜ 23 ｜ 4C	'01234C'

2. 値式に指定できるものを次に示します。

- 定数
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- 列指定
- コンポネント指定
- SQL 変数，又は SQL パラメタ
- 四則演算
- 日付演算
- 時刻演算
- 連結演算
- 集合関数
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

3. 値式に指定できるデータ型と結果のデータ型，及びデータ長の関係を次の表に示します。

表 2-35　スカラ関数 HEX の値式のデータ型と結果のデータ型，及びデータ長の関係

値　式			実行結果		
データ型	定義長	実長	データ型	定義長	実長
CHAR(n)	1≦n＜128	－	CHAR	n＊2	－
	128≦n≦16,000	－	VARCHAR	n＊2	n＊2
NCHAR(n)	1≦n＜64	－	CHAR	n＊4	－
	64≦n≦8,000	－	VARCHAR	n＊4	n＊4
MCHAR(n)	1≦n＜128	－	CHAR	n＊2	－
	128≦n≦16,000	－	VARCHAR	n＊2	n＊2

値　式			実行結果		
データ型	定義長	実長	データ型	定義長	実長
VARCHAR(n)	1≦n≦16,000	〔r〕	VARCHAR	n＊2	r＊2
NVARCHAR(n)	1≦n≦8,000	〔r〕	VARCHAR	n＊4	r＊4
MVARCHAR(n)	1≦n≦16,000	〔r〕	VARCHAR	n＊2	r＊2
INTEGER	－	－	CHAR	8	－
SMALLINT	－	－	CHAR	4	－
DECIMAL（P,S）	1≦P≦38 0≦S≦38 S≦P	－	CHAR	（↓P/2↓＋1）＊2	－
FLOAT	－	－	CHAR	16	－
SMALLFLT	－	－	CHAR	8	－
DATE	－	－	CHAR	8	－
TIME	－	－	CHAR	6	－
TIMESTAMP(p)	p=0,2,4,又は 6	－	CHAR	（7 ＋ p/2）＊2	－
INTERVAL YEAR TO DAY	－	－	CHAR	10	－
INTERVAL HOUR TO SECOND	－	－	CHAR	8	－
BINARY(n)	1≦n≦16,000	〔r〕	VARCHAR	n＊2	r＊2

（凡例）

P：精度

S：位取り

p：小数秒精度

－：該当しません。

注

値式に長さ 0 の文字列定数（各国，及び混在を含む）だけが指定された場合，実行結果の定義長は 1 になります。

4. 値式のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。

5. 値式の演算結果が次に示すデータ型の場合は指定できません。

- 最大長が 16,001 バイト以上の CHAR，VARCHAR，MCHAR，及び MVARCHAR
- 最大長が 8,001 文字以上の NCHAR，及び NVARCHAR
- BLOB
- 最大長が 16,001 バイト以上の BINARY

- BOOLEAN

6. 埋込み変数，及び?パラメタを含む値式は指定できません。
7. 値式中の演算項，又は関数の引数が定数だけの場合，結果の長さが 256 バイト以上は指定できません。
8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。
9. Windows 版の場合，結果の値（DECIMAL 型を除いた数データ）はリトルエンディアンの内部表現となります。具体的には，INTEGER 型の 1234 の内部表現は"D2 04 00 00"となり，実行結果は"D2040000"となります。
10. UNIX 版の場合，結果の値はサーバのプラットフォームの内部表現に依存します。例えば，Intel 系 CPU で動作する Linux の場合，DECIMAL 型を除いた数データはリトルエンディアンの内部表現となります。具体的には，INTEGER 型の 1234 の内部表現は"D2 04 00 00"となり，実行結果は"D2040000"となります。
11. 結果のデータ型が文字データとなる場合，結果の文字集合は既定文字集合になります。

(11) HOUR

(a) 機能

時刻データ，時刻印データ，又は時間隔データから時間の部分だけを抽出します。

(b) 形式

```
HOUR（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 時刻演算
   - 集合関数（MAX，MIN）
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し

- スカラ副問合せ

2. 値式のデータ型は，時刻データ型（TIME），時刻印データ型（TIMESTAMP），又は時間隔データ型（INTERVAL HOUR TO SECOND）にしてください。
3. 結果のデータ型は，整数（INTEGER）になります。
4. 値式が時刻データ型，又は時刻印データ型であれば，結果は 0～23 になります。
5. 値式が時間隔データ型であれば，結果は－99～99 になります。結果が 0 以外の場合，値式の符号と結果の符号は同じになります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

(d) 使用例

表 T1 の C1 列（データ型：時刻データ型）のうち，現在と同じ時間の行をすべて検索します。

```
SELECT * FROM T1
    WHERE HOUR (C1) = HOUR(CURRENT_TIME)
```

(12) INTEGER

(a) 機能

数データを，整数に変換します。

(b) 形式

```
INT〔EGER〕（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定

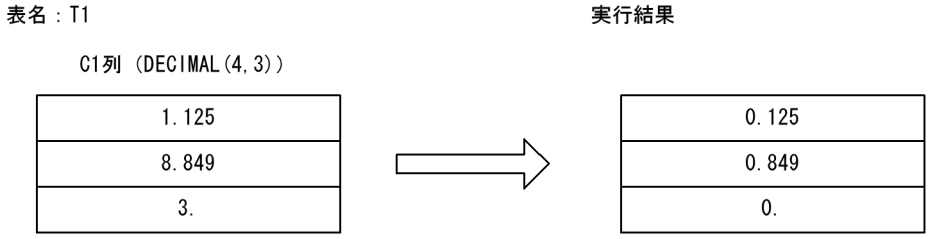
- 関数呼出し
- スカラ副問合せ

2. 値式のデータ型として指定できるものを次に示します。
   - 数データ型
3. 結果のデータ型は，整数（INTEGER）になります。
4. 結果は，INTEGER で表現できる値にしてください。
5. 結果の値で，小数点以下の数値は切り捨てられます。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

(d) 使用例

表 T1 の C1 列（データ型：DECIMAL（4，3））のうち，小数部分だけを取得します。

```
SELECT C1 - INTEGER(C1) FROM T1
```



(13) LENGTH

(a) 機能

値式のデータの長さを求めます。

(b) 形式

```
LENGTH（ {値式 | GET_JAVA_STORED_ROUTINE_SOURCE指定} ）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 定数
   - USER 値関数
   - CURRENT_DATE 値関数
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数

- 列指定
- コンポネント指定
- SQL 変数，又は SQL パラメタ
- 四則演算
- 日付演算
- 時刻演算
- 連結演算
- 集合関数
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ
- ：埋込み変数［：標識変数］ AS データ型 （データ型は BLOB 型，又は BINARY 型だけ）
- ？ AS データ型 （データ型は BLOB 型，又は BINARY 型だけ）

2. 値式には，次のデータ型は指定できません。
   - BOOLEAN
   - 抽象データ型
3. 値式に埋込み変数，又は？パラメタを指定する場合は，AS 句でそのデータ型を指定してください。埋込み変数又は？パラメタに与えるデータの実長（位置付け子の場合は位置付け子に割り当てられたデータの実長）が AS 句で指定したデータ型の最大長より大きい場合はエラーとなります。
4. 結果のデータ型は，整数（INTEGER）になります。
5. 値式のデータ型によって，実行結果が異なります。値式のデータ型による実行結果を次の表に示します。

表 2-36　スカラ関数 LENGTH の値式のデータ型による実行結果

値式のデータ型	実行結果
固定長文字データ	既定文字集合の場合，定義長のバイト数 既定文字集合以外の場合，定義長のその文字集合での文字数
可変長文字データ	既定文字集合の場合，実際のデータのバイト数 既定文字集合以外の場合，実際のデータのその文字集合での文字数
固定長各国文字データ	定義長の文字数
可変長各国文字データ	実際のデータの文字数
固定長混在文字データ	実際のデータの文字数
可変長混在文字データ	実際のデータの文字数

値式のデータ型	実行結果
数データ	定義長のバイト数 ただし，DECIMAL は，（↓精度で指定したけた数/2↓＋1）になります。 詳細については，「データ型」を参照してください。
日付，時刻，時刻印データ	定義長のバイト数
日間隔，時間隔データ	詳細については，「データ型」を参照してください。
長大データ	実際のデータのバイト数
バイナリデータ	実際のデータのバイト数

6. 値式が文字データ型の場合，空白も 1 文字として数えます。
7. 値式が定数の場合，HiRDB で解釈されたデータ型に従って処理されます。詳細については，「定数」を参照してください。
8. 値式が USER 値関数，CURRENT_DATE 値関数，CURRENT_TIME 値関数，又は CURRENT_TIMESTAMP 値関数の場合，HiRDB で解釈されたデータ型に従って処理されます。詳細については，「USER 値関数，CURRENT_DATE 値関数，CURRENT_TIME 値関数，及び CURRENT_TIMESTAMP 値関数」を参照してください。
9. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

(14) LOWER

(a) 機能

文字データ，各国文字データ，又は混在文字データ中の英大文字を英小文字に変換します。

(b) 形式

```
LOWER（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 定数
   - USER 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 連結演算
   - 集合関数

- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式に，NULL，埋込み変数，及び?パラメタは指定できません。
3. 値式のデータ型は，文字データ型（CHAR，VARCHAR），各国文字データ型（NCHAR，NVARCHAR），又は混在文字データ型（MCHAR，MVARCHAR）にしてください。
4. 値式のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。
5. 結果のデータ型，及びデータ長は，値式のデータ型，及びデータ長と同じになります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。
7. 結果の文字集合は，引数に指定した値式の文字集合となります。

(15) MINUTE

(a) 機能

時刻データ，時刻印データ，又は時間隔データから，分の部分だけを抽出します。

(b) 形式

```
MINUTE（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 時刻演算
   - 集合関数（MAX，MIN）
   - スカラ関数
   - CASE 式
   - CAST 指定

- 関数呼出し
- スカラ副問合せ

2. 値式のデータ型は，時刻データ型（TIME），時刻印データ型（TIMESTAMP），又は時間隔データ型（INTERVAL HOUR TO SECOND）にしてください。
3. 結果のデータ型は，整数（INTEGER）になります。
4. 値式が時刻データ型，又は時刻印データ型であれば，結果は 0～59 になります。
5. 値式が時間隔データ型であれば，結果は-59～59 になります。結果が 0 以外の場合，値式の符号と結果の符号は同じになります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

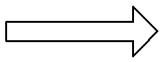
(d) 使用例

表 T1 の C2 列（データ型：時刻データ型）と C3 列（データ型：時刻データ型）の差が，30 分以内のデータを検索します。

```
SELECT C1 FROM T1
    WHERE MINUTE(C3-C2) <= 30
```

表名：T1

C1列	C2列（時刻データ型）	C3列（時刻データ型）
A01	08:00:00	08:45:00
A02	09:00:00	09:28:00
A03	10:00:00	10:30:00



実行結果

C1列
A02
A03

(16) MOD

(a) 機能

剰余を返します。

(b) 形式

```
MOD（値式1，値式2）
```

(c) 規則

1. 値式 1，及び値式 2 に指定できるものを次に示します。
   - 整数定数，又は 10 進数定数
   - 列指定
   - コンポネント指定

- SQL 変数，又は SQL パラメタ
- 四則演算
- 集合関数
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式 1 には被除数を，値式 2 には除数を指定します。
3. 値式 1，又は値式 2 のデータ型は，次に示すどちらかにしてください。
   - 整数（INTEGER，SMALLINT）
   - 固定小数点数（DECIMAL）
4. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式 1，又は値式 2 がナル値であれば，結果もナル値になります。
5. 値式 1，又は値式 2 に小数部を含む場合，結果の値にも小数部を含みます。
6. 結果の符号は，値式 1 の符号と同じになります。
7. 値式 2 が 0 の場合，エラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。
8. 次の式が成立する場合，計算途中でオーバフローが発生するため，オーバフローエラーになります。

   （p1－s1）＋ s2 ＞ max_prec

   p1：値式 1 の値の実行的な精度

   s1：値式 1 の値の実行的な位取り

   p2：値式 2 の値の実行的な精度

   s2：値式 2 の値の実行的な位取り

   max_prec：精度の最大値です。精度の最大値 max_prec を次の表に示します。

表 2-37　精度の最大値 max_prec

システム共通定義 pd_sql_dec_op_maxprec オペランドの値 ※	p1 と p2	max_prec の値
29	p1≦29 かつ p2≦29	29
	p1 ＞ 29 又は p2 ＞ 29	38
38	任意	38

注※

システム共通定義の pd_sql_dec_op_maxprec オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

9. 結果のデータ型と値式 1，及び値式 2 のデータ型の関係を次の表に示します。

表 2-38　結果のデータ型と値式 1，及び値式 2 のデータ型の関係

値式 1 のデータ型	値式 2 のデータ型		
	SMALLINT	INTEGER	DECIMAL
SMALLINT	SMALLINT	INTEGER	DECIMAL
INTEGER	SMALLINT	INTEGER	DECIMAL
DECIMAL（p，0）	SMALLINT	INTEGER	DECIMAL
DECIMAL（p，s） （s ＞ 0）	DECIMAL	DECIMAL	DECIMAL

10. 結果のデータ型が DECIMAL の場合の結果の精度と位取りを次の表に示します。

表 2-39　結果のデータ型が DECIMAL の場合の結果の精度と位取り

剰余結果の精度と位取り	精度と位取り
精度（p）	p ＝ MIN（p2－s2 ＋ s，max_prec）
位取り（s）	s ＝ MAX（s1，s2）

注 1

値式 1 のデータ型：DECIMAL（p1，s1）

値式 2 のデータ型：DECIMAL（p2，s2）

注 2

INTEGER は，DECIMAL（10，0）として扱われます。

SMALLINT は，DECIMAL（5，0）として扱われます。

注 3

max_prec は，精度の最大値で，表「精度の最大値 max_prec」のようになります。

(17)　MONTH

(a)　機能

日付データ，時刻印データ，又は日間隔データから月の部分だけを抽出します。

(b) 形式

```
MONTH（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 日付演算
   - 集合関数（MAX，MIN）
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式のデータ型は，日付データ型（DATE），時刻印データ型（TIMESTAMP），又は日間隔データ型（INTERVAL YEAR TO DAY）にしてください。
3. 結果のデータ型は，整数（INTEGER）になります。
4. 値式が日付データ型，又は時刻印データ型であれば，結果は 1～12 になります。
5. 値式が日間隔データ型であれば，結果は－11～11 になります。結果が 0 以外の場合，値式の符号と結果の符号は同じになります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

(d) 使用例

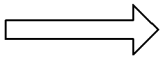
表 T1 のうち，当月（9 月）でない行をすべて削除します。

```
DELETE FROM T1
    WHERE MONTH(C1) < > MONTH(CURRENT_DATE)
```

表名：T1

C1列（日付データ型）

1995 - 05 - 23
1995 - 09 - 27
1995 - 04 - 14
1995 - 02 - 03



実行結果

1995 - 09 - 27

(18) POSITION

(a) 機能

データ列（文字列又はバイナリ列）中で，部分データ列に最初に一致する部分の開始位置を求めます。

(b) 形式

```
POSITION（値式1 IN 値式2〔 FROM 値式3〕）
```

(c) 規則

1. 値式 1 には検索する部分データ列を指定します。値式 2 には検索対象のデータ列を指定します。値式 1，及び値式 2 に指定できるものを次に示します。ただし，値式 1 及び値式 2 のデータ型の組み合わせによって指定できるものが異なります。指定できるものの組み合わせは規則 2. を参照してください。
   - 定数(文字列，各国文字列，混在文字列，又は 16 進文字列)
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
   - ：埋込み変数〔：標識変数〕 AS データ型 （データ型は BLOB 型，又は BINARY 型だけ）
   - ? AS データ型 （データ型は BLOB 型，又は BINARY 型だけ）
2. 値式 1，及び値式 2 に指定できるデータ型の組み合わせを次の表に示します。

表 2-40　スカラ関数 POSITION の値式 1 及び値式 2 に指定できるデータ型の組み合わせ

値式 1	値式 2					
	文字列データ型	各国文字列データ型	混在文字列データ型	BLOB 型	最大長が 32,000 バイト以下の BINARY 型	最大長が 32,001 バイト以上の BINARY 型
文字列データ型	○※3,※5	×	○※4,※5	△※1,※6	△※1,※5	△※1,※6
各国文字列データ型	×	○※5	×	×	×	×
混在文字列データ型	○※4,※5	×	○※5	×	×	×
BLOB 型	△※2,※7	×	×	○※6	○※7	○※6
最大長が 32,000 バイト以下の BINARY 型	△※2,※5	×	×	○※6	○※5	○※6
最大長が 32,001 バイト以上の BINARY 型	△※2,※7	×	×	○※6	○※7	○※6

（凡例）

○：指定できます。

△：制限付きで指定できます。

×：指定できません。

(表番号)：指定できる場合の，対応する項目の組み合わせ表

注※1

値式 1 が 16 進文字列定数の場合だけ指定できます。

注※2

値式 2 が 16 進文字列定数の場合だけ指定できます。

注※3

値式 1 と値式 2 が両方とも文字列データ型の場合，値式 1 と値式 2 の文字集合は同じにしてください。ただし，値式 1 が次に示す値式である場合，値式 2 の文字集合に変換します。

・文字列定数

注※4

文字列データ型の値式の文字集合が既定文字集合以外の場合は指定できません。

注※5

項目の組み合わせについては，表「スカラ関数 POSITION の値式 1 及び値式 2 に指定できる項目の組み合わせ(値式 1，及び値式 2 が共に文字列データ型，各国文字列データ型，混在文字列データ型，又は最大長が 32,000 バイト以下の BINARY 型の場合)」を参照してください。

注※6

項目の組み合わせについては，表「スカラ関数 POSITION の値式 1 及び値式 2 に指定できる項目の組み合わせ(値式 2 が BLOB 型，又は最大長が 32,001 バイト以上の BINARY 型の場合)」を参照してください。

注※7

項目の組み合わせについては，表「スカラ関数 POSITION の値式 1 及び値式 2 に指定できる項目の組み合わせ(値式 1 が BLOB 型，又は最大長が 32,001 バイト以上の BINARY 型，値式 2 が文字列データ型，又は最大長が 32,000 バイト以下の BINARY 型の場合)」を参照してください。

3. 値式 1，及び値式 2 の指定できる項目の組み合わせを次に示します。

表 2-41　スカラ関数 POSITION の値式 1 及び値式 2 に指定できる項目の組み合わせ(値式 1，及び値式 2 が共に文字列データ型，各国文字列データ型，混在文字列データ型，又は最大長が 32,000 バイト以下の BINARY 型の場合)

値式 1	値式 2											
	定数	列指定	コンポネント指定	SQL 変数，SQL パラメタ	連結演算	集合関数	スカラ関数	CASE 式	CAST 指定	関数呼出し	スカラ副問合せ	埋込み変数，？パラメタ※
定数	○	○	○	○	○	○	○	○	○	○	○	○
列指定	○	○	○	○	○	○	○	○	○	○	○	○
コンポネント指定	○	○	○	○	○	○	○	○	○	○	○	○
SQL 変数，SQL パラメタ	○	○	○	○	○	○	○	○	○	○	○	○
連結演算	○	○	○	○	○	○	○	○	○	○	○	○
集合関数	○	○	○	○	○	○	○	○	○	○	○	○
スカラ関数	○	○	○	○	○	○	○	○	○	○	○	○
CASE 式	○	○	○	○	○	○	○	○	○	○	○	○
CAST 指定	○	○	○	○	○	○	○	○	○	○	○	○
関数呼出し	○	○	○	○	○	○	○	○	○	○	○	○
埋込み変数，？パラメタ※	○	○	○	○	○	○	○	○	○	○	○	○

(凡例)

○：指定できます。

注※

埋込み変数，？パラメタは BINARY 型の場合だけ指定できます。

表 2-42　スカラ関数 POSITION の値式 1 及び値式 2 に指定できる項目の組み合わせ(値式 2 が BLOB 型，又は最大長が 32,001 バイト以上の BINARY 型の場合)

値式 1	値式 2											
	定数	列指定	コンポネント指定	SQL 変数，SQL パラメタ	連結演算	集合関数	スカラ関数	CASE 式	CAST 指定	関数呼出し	スカラ副問合せ	埋込み変数，？パラメタ
定数※	×	○	×	○	×	×	×	×	×	×	×	○
列指定	×	×	×	×	×	×	×	×	×	×	×	×
コンポネント指定	×	×	×	×	×	×	×	×	×	×	×	×
SQL 変数，SQL パラメタ	×	○	×	○	×	×	×	×	×	×	×	○
連結演算	×	×	×	×	×	×	×	×	×	×	×	×
集合関数	×	×	×	×	×	×	×	×	×	×	×	×
スカラ関数	×	×	×	×	×	×	×	×	×	×	×	×
CASE 式	×	×	×	×	×	×	×	×	×	×	×	×
CAST 指定	×	×	×	×	×	×	×	×	×	×	×	×
関数呼出し	×	×	×	×	×	×	×	×	×	×	×	×
埋込み変数，？パラメタ	×	○	×	○	×	×	×	×	×	×	×	○

(凡例)

○：指定できます。

×：指定できません。

注※

定数は文字列データ型（16 進文字列定数）の場合だけ指定できます。

表 2-43　スカラ関数 POSITION の値式 1 及び値式 2 に指定できる項目の組み合わせ(値式 1 が BLOB 型，又は最大長が 32,001 バイト以上の BINARY 型，値式 2 が文字列データ型，又は最大長が 32,000 バイト以下の BINARY 型の場合)

値式 1	値式 2											
	定数	列指定	コンポネント指定	SQL 変数，SQL パラメタ	連結演算	集合関数	スカラ関数	CASE 式	CAST 指定	関数呼出し	スカラ副問合せ	埋込み変数，？パラメタ※
定数	×	×	×	×	×	×	×	×	×	×	×	×
列指定	×	×	×	×	×	×	×	×	×	×	×	×
コンポネント指定	×	×	×	×	×	×	×	×	×	×	×	×
SQL 変数，SQL パラメタ	○	○	○	○	○	○	○	○	○	○	○	○
連結演算	×	×	×	×	×	×	×	×	×	×	×	×
集合関数	×	×	×	×	×	×	×	×	×	×	×	×
スカラ関数	×	×	×	×	×	×	×	×	×	×	×	×
CASE 式	×	×	×	×	×	×	×	×	×	×	×	×
CAST 指定	×	×	×	×	×	×	×	×	×	×	×	×
関数呼出し	×	×	×	×	×	×	×	×	×	×	×	×
埋込み変数，？パラメタ	○	○	○	○	○	○	○	○	○	○	○	○

(凡例)

○：指定できます。

×：指定できません。

注※

埋込み変数，？パラメタは BINARY 型の場合だけ指定できます。

4. 値式 1 及び値式 2 に埋込み変数，又は？パラメタを指定する場合は，AS 句でそのデータ型を指定してください。埋込み変数又は？パラメタに与えるデータの実長（位置付け子の場合は位置付け子に割り当てられたデータの実長）が AS 句で指定したデータ型の最大長より大きい場合はエラーとなります。
5. 値式 3 は検索開始位置を指定します。次に，値式 2 と値式 3 の単位と値の範囲の関係を示します。

- 値式 2 が，文字集合が既定文字集合の文字列データ型，BLOB 型，又は BINARY 型の場合
  値式 3 はバイト単位で指定します。値式 3 の値は次の範囲にしてください。

・値式 2 のデータ型が，文字集合が既定文字集合の文字列データ型の場合は，1≦（値式 3）≦（値式 2 の最大長）

・値式 2 のデータ型が BLOB 型又は BINARY 型の場合は，1≦（値式 3）

- 値式 2 が，各国文字列データ型又は混在文字列データ型の場合

  値式 3 は文字単位で指定します。値式 3 の値は 1≦（値式 3）≦（値式 2 の最大長）の範囲にしてください。なお，混在文字列データ型の場合，値式 2 の最大長の単位はバイトです。

- 値式 2 が，文字集合が既定文字集合以外の文字列データ型の場合

  値式 3 は文字単位で指定します。値式 3 の値は 1≦（値式 3）≦（値式 2 の最大長）÷c の範囲にしてください。c の値は，その文字集合の文字の最小バイト数で，EBCDIK の場合は 1，UTF16 の場合は 2 となります。なお，値式 2 の最大長の単位はバイトです。

値式 3 を省略した場合，検索開始位置は 1 を仮定します。

6. 値式 3 に指定できるものを次に示します。
   - 符号なし整数定数
   - 列指定
   - コンポネント指定
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
   - SQL 変数又は SQL パラメタ
   - 埋込み変数又は？パラメタ

   ただし，値式 1，又は値式 2 が BLOB 型，又は最大長が 32,001 バイト以上の BINARY 型の場合，値式 3 には次のものが指定できます。
   - 符号なし整数定数
   - SQL 変数又は SQL パラメタ
   - 埋込み変数又は？パラメタ
7. 値式 3 のデータ型は，整数（INTEGER，SMALLINT）にしてください。
8. 結果のデータ型は，整数（INTEGER）になります。
9. 結果は値式 2 が，文字集合が既定文字集合の文字列データ型，BLOB 型，又は BINARY 型の場合はバイト単位，文字集合が既定文字集合以外の文字列データ型，各国文字列データ型又は混在文字列データ型の場合は文字単位の位置となります。

10. 値式 1 の実長が 0 の場合，結果は値式 3 の値になります。ただし，値式 3 を省略した場合は 1 になります。
11. 値式 1 の実長が 0 より大きく，値式 3 が値式 2 の実長より大きい値の場合，結果は 0 になります。
12. 値式 1 の部分データ列が値式 2 のデータ列中の検索開始位置以降に出現しない場合は，結果は 0 になります。
13. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式 1，値式 2，又は値式 3 がナル値であれば，結果もナル値になります。

(d) 使用例

表 T1 の C1 列（データ型：CHAR）の 6 バイト目以降に，最初に’TIME:’が出現する位置を検索します。

```
SELECT POSITION( ‘TIME:’ IN C1 FROM 6) FROM T1
```

表名：T1

C1列（VARCHAR(50)）

TIME:10:15,TIME:12:00	（実長21）	12
TIME:13:00	（実長10）	0
*	（ナル値）	*

実行結果

（凡例） * ：ナル値を示します。

(19) SECOND

(a) 機能

時刻データ，時刻印データ，又は時間隔データから，秒の部分だけを抽出します。

(b) 形式

```
SECOND（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 時刻演算

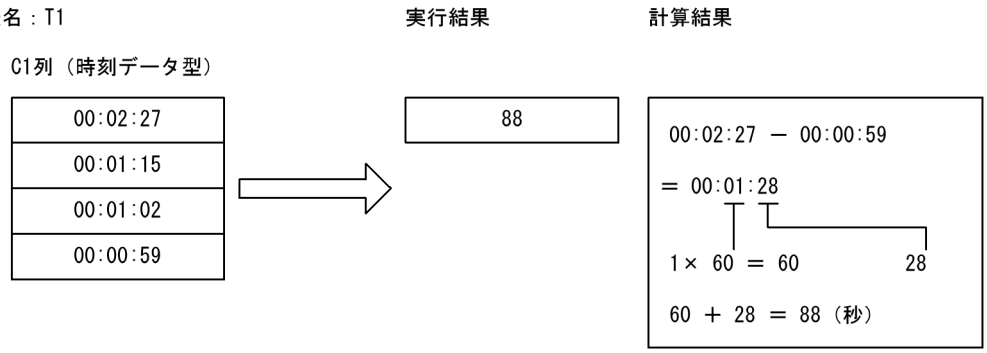
- 集合関数（MAX，MIN）
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式のデータ型は，時刻データ型（TIME），時刻印データ型（TIMESTAMP），又は時間隔データ型（INTERVAL HOUR TO SECOND）にしてください。
3. 結果のデータ型は，整数（INTEGER）になります。
4. 値式が時刻データ型，又は時刻印データ型であれば，結果は 0～59 になります。また，うるう秒を含んだ時刻データ又は時刻印データであれば，結果は 0～61 になります。なお，うるう秒を含んだ時刻データ又は時刻印データを指定する方法については，マニュアル「HiRDB システム定義」の pd_leap_second オペランドの説明を参照してください。
5. 値式が時間隔データ型であれば，結果は－59～59 になります。結果が 0 以外の場合，値式の符号と結果の符号は同じになります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

(d) 使用例

表 T1 の C1 列（データ型：時刻データ型）のうち，最も早い時刻と最も遅い時刻の差を秒単位で取得します。

```
SELECT MINUTE(MAX(C1)-MIN(C1))
        *60+SECOND(MAX(C1)
        -MIN(C1))
    FROM T1
```



(20) SUBSTR

(a) 機能

文字列データ，各国文字列データ，混在文字列データ，又はバイナリデータの一部を抽出します。

(b) 形式

```
SUBSTR（値式1，値式2〔，値式3〕）
```

(c) 規則

1. 値式 1 には，処理対象となるデータ列（文字列又はバイナリ列）を指定します。値式 1 に指定できるものを次に示します。ただし，値式 1 のデータ型によって指定できるものが異なります。
   - 定数（文字列，各国文字列，又は混在文字列）
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
   - ：埋込み変数〔：標識変数〕 AS データ型 （データ型は BLOB 型，又は BINARY 型だけ）
   - ? AS データ型 （データ型は BLOB 型，又は BINARY 型だけ）

   各データ型で指定できるものを次の表に示します。

表 2-44 スカラ関数 SUBSTR の値式 1 のデータ型による指定できる項目

項目	値式 1 のデータ型		
	文字列データ型， 各国文字列データ型， 混在文字列データ型	最大長が 32,000 バイト以下の BINARY 型	BLOB 型， 最大長が 32,001 バイト以上の BINARY 型
定数	○	×	×
列指定	○	○	○
コンポネント指定	○	○	×

項目	値式 1 のデータ型		
	文字列データ型， 各国文字列データ型， 混在文字列データ型	最大長が 32,000 バイト以下の BINARY 型	BLOB 型， 最大長が 32,001 バイト以上の BINARY 型
SQL 変数 SQL パラメタ	○	○	○
連結演算 集合関数 スカラ関数 CASE 式 CAST 指定	○	○	×
関数呼出し	○	○	○
スカラ副問合せ	○	○	×
埋込み変数 ？パラメタ	×	○	○

(凡例)

○：指定できます。

×：指定できません。

2. 値式 1 のデータ型は，次のどれかを指定してください。
   - 任意の文字集合の文字列データ型（CHAR，VARCHAR）
   - 各国文字列データ型（NCHAR，NVARCHAR）
   - 混在文字列データ型（MCHAR，MVARCHAR）
   - BLOB 型，又は BINARY 型
3. 値式 1 に埋込み変数又は？パラメタを指定する場合は，AS 句でそのデータ型を指定してください。埋込み変数又は？パラメタに与えるデータの実長（位置付け子の場合は位置付け子に割り当てられたデータの実長）が AS 句で指定したデータ型の最大長より大きい場合はエラーとなります。
4. 値式 2 には，取り出す部分データ列の開始位置を正の整数で指定します。次に，値式 1 と値式 2 の単位と値の範囲の関係を示します。
   - 値式 1 が，文字集合が既定文字集合の文字列データ型，BLOB 型，又は BINARY 型の場合

     値式 2 はバイト単位で指定します。値式 2 の値は次の範囲にしてください。

     ・値式 1 のデータ型が，文字集合が既定文字集合の文字列データ型の場合は，1≦（値式 2）≦（値式 1 の最大長）

     ・値式 1 のデータ型が BLOB 型又は BINARY 型の場合は，1≦（値式 2）
   - 値式 1 が，各国文字列データ型又は混在文字列データ型の場合

     値式 2 は文字単位で指定します。値式 2 の値は 1≦（値式 2）≦（値式 1 の最大長）の範囲にしてください。なお，混在文字列データ型の場合，値式 1 の最大長の単位はバイトです。

- 値式 1 が，文字集合が既定文字集合以外の文字列データ型の場合

  値式 2 は文字単位で指定します。値式 2 の値は 1≦（値式 2）≦（値式 1 の最大長）÷c の範囲にしてください。c の値は，その文字集合の文字の最小バイト数で，EBCDIK の場合は 1，UTF16 の場合は 2 です。なお，値式 1 の最大長の単位はバイトです。

5. 値式 3 には，取り出す部分データ列の長さを正の整数で指定します。次に，値式 1 と値式 3 の単位と値の範囲の関係を示します。

   - 値式 1 が，文字集合が既定文字集合の文字列データ型，BLOB 型，又は BINARY 型の場合

     値式 3 はバイト単位で指定します。値式 3 の値は次の範囲にしてください。

     ・値式 1 のデータ型が，文字集合が既定文字集合の文字列データ型の場合は，0≦（値式 3）≦（値式 1 の最大長）－（値式 2）＋ 1

     ・値式 1 のデータ型が BLOB 型又は BINARY 型の場合は，0≦（値式 3）

   - 値式 1 が，各国文字列データ型又は混在文字列データ型の場合

     値式 3 は文字単位で指定します。値式 3 の値は 0≦（値式 3）≦（値式 1 の最大長）－（値式 2）＋ 1 の範囲にしてください。なお，混在文字列データ型の場合，値式 1 の最大長の単位はバイトです。

   - 値式 1 が，文字集合が既定文字集合以外の文字列データ型の場合

     値式 3 は文字単位で指定します。値式 3 の値は 0≦（値式 3）≦（値式 1 の最大長）÷c－（値式 2）＋ 1 の範囲にしてください。c の値は，その文字集合の文字の最小バイト数で，EBCDIK の場合は 1，UTF16 の場合は 2 です。なお，値式 1 の最大長の単位はバイトです。

   値式 3 には定数の 0 は指定できません。

6. 値式 2，及び値式 3 に指定できるものを次に示します。

   - 符号なし整数定数
   - 列指定
   - コンポネント指定
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - SQL 変数又は SQL パラメタ
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

   ただし，値式 1 が BLOB 型，又は最大長が 32,001 バイト以上の BINARY 型の場合，値式 2 及び値式 3 には次のものが指定できます。

- 符号なし整数定数
- SQL 変数又は SQL パラメタ
- 埋込み変数又は？パラメタ

7. 値式 2，及び値式 3 のデータ型は，整数（INTEGER，SMALLINT）にしてください。
8. 値式 2 が，値式 1 の実長より大きい値の場合，結果はナル値（結果の長さは 0）になります。
9. 結果のデータの長さが 0 の場合，その結果はナル値になります。
10. 値式 3 を省略した場合，値式 1 が固定長データのときは，値式 2 の開始位置から定義長の最後の文字まで取り出します。また，値式 1 が可変長データの場合は，値式 2 の開始位置から実長の最後の文字まで取り出します。
11. 値式 1 が BLOB 型で値式 3 を省略しない場合，指定した範囲のデータ列の中で実データの含まれていない部分があるときは，実データの含まれている部分だけを取り出します。
12. 結果の値は，非ナル値制約なし（ナル値を許します）になります。
13. 値式 1，値式 2，値式 3 のどれかがナル値であれば，結果もナル値になります。
14. 結果のデータ型と長さを次の表に示します。

- スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合の文字列型，各国文字型，混在文字型，BLOB 型又はバイナリ型で，値式 3 を指定した場合）
- スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合の文字列型，各国文字型，混在文字型，BLOB 型又はバイナリ型で，値式 3 を省略した場合）
- スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合以外の文字列型で，値式 3 を指定した場合）
- スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合以外の文字列型で，値式 3 を省略した場合）

表 2-45　スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合の文字列型，各国文字型，混在文字型，BLOB 型又はバイナリ型で，値式 3 を指定した場合）

部分列を抽出する文字列（値式 1）のデータ型	実長	長さ L（値式 3）		
		定数（文字列）		定数以外
		$L_1 \leqq 255$	$L_1 \geqq 256$	
CHAR(n)	－	CHAR（L）	VARCHAR（L）〔L〕	VARCHAR(n)〔L〕
VARCHAR(n)	〔r〕			
NCHAR(n)	－	NCHAR（L）	NVARCHAR（L）〔L〕	NVARCHAR(n)〔L〕
NVARCHAR(n)	〔r〕			
MCHAR(n)	－	MCHAR（L_1）	MVARCHAR（L_1）〔L_2〕	MVARCHAR(n)〔L_2〕
MVARCHAR(n)	〔r〕			

部分列を抽出する文字列（値式 1）のデータ型	実長	長さ L（値式 3）		
		定数（文字列）		定数以外
		$L_1 \leqq 255$	$L_1 \geqq 256$	
BLOB(n)	〔r〕	BLOB（L）〔k_5〕	BLOB（L）〔k_5〕	BLOB（n）〔k_5〕
BINARY(n)	〔r〕	BINARY(L)〔k5〕	BINARY(L)〔k_5〕	BINARY(n)〔k_5〕

（凡例）

〔 〕：実長を示します。

L_1：

- 文字列データ型の場合，値式 3 の長さ L（バイト単位）
- 各国文字列データ型の場合，値式 3 の長さ L（文字単位）×2
- 混在文字列データ型の場合，min（値式 3 の長さ L（文字単位）×c，n）

L_2：

抽出した L 文字の部分文字列のバイト数（$L \leqq L_2 \leqq L_1$）

n：

処理対象となるデータ列（値式 1）の定義長
文字列データ型，混在文字列データ型，及び BLOB 型の場合はバイト単位
各国文字列データ型の場合は文字単位

k_5：

min（L，r−S ＋ 1）

c：

1 文字を表現する最大バイト数
次の表に，文字コードごとの最大バイト数を示します。

pdntenv 及び pdsetup で指定する文字コード種別	最大バイト数（byte）
sjis	2
ujis	2
chinese	2
chinese-gb18030	4
lang-c	2
utf-8※	3～6
utf-8_ivs※	3～10

注※

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，又は utf-8_ivs を指定した場合は次の指定に従います。

- システム共通定義 pd_substr_length
- クライアント環境定義 PDSUBSTRLEN
- SQL コンパイルオプション SUBSTR LENGTH

－：

該当しません。

注

値式 1 が可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）の場合，取り出そうとする文字列中で実データの含まれていない部分には，空白が設定されます。

（例）

VARCHAR（8）〔5〕の文字列 1 に対して，SUBSTR（文字列 1，3，5）を実行すると，取り出す文字列の右側 2 文字には空白が設定されます。

なお，値式 1 が BLOB 型又は BINARY 型で値式 3 を省略しない場合，指定した範囲のバイナリ列の中で，実データが含まれていない部分があるときは，実データの含まれている部分だけを取り出します。空白は設定されません。

（例）

BLOB(1024)〔512〕のバイナリデータ 1 に対して，SUBSTR（バイナリデータ 1，101，600）を実行すると，バイナリデータ 1 の 101 バイト目から 512 バイト目までが結果となります。

表 2-46　スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合の文字列型，各国文字型，混在文字型，BLOB 型又はバイナリ型で，値式 3 を省略した場合）

部分列を抽出する文字列（値式 1）のデータ型	実長	開始位置 S（値式 2）			
		定 数（文字列）			定数以外
		値式 1 が固定長		値式 1 が可変長	
		$k_0 \leqq 255$	$k_0 \geqq 256$		
CHAR(n)	－	CHAR（k_1）	VARCHAR(n)〔k_1〕	－	VARCHAR(n)〔k_1〕
VARCHAR(n)	〔r〕	－	－	VARCHAR(n)〔k_2〕	VARCHAR(n)〔k_2〕
NCHAR(n)	－	NCHAR（k_1）	NVARCHAR(n)〔k_1〕	－	NVARCHAR(n)〔k_1〕
NVARCHAR(n)	〔r〕	－	－	NVARCHAR(n)〔k_2〕	NVARCHAR(n)〔k_2〕
MCHAR(n)	－	MCHAR（k_1）	MVARCHAR(n)〔k_3〕	－	MVARCHAR(n)〔k_3〕

部分列を抽出する文字列（値式 1）のデータ型	実長	開始位置 S（値式 2）			
		定 数（文字列）			定数以外
		値式 1 が固定長		値式 1 が可変長	
		k_0≦255	k_0≧256		
MVARCHAR(n)	〔r〕	−	−	MVARCHAR(n)〔k_4〕	MVARCHAR(n)〔k_4〕
BLOB(n)	〔r〕			BLOB(n)〔k_2〕	BLOB(n)〔k_2〕
BINARY(n)	〔r〕			BINARY(n)〔k_2〕	BINARY(n)〔k_2〕

（凡例）

〔 〕：実長を示します。

n：

- 文字列データ型の場合は，バイト単位
- 混在文字列データ型の場合は，バイト単位
- 各国文字列データ型の場合は，文字単位

k_0：

文字列データ型，又は混在文字列データ型の場合，n−S ＋ 1

各国文字列データ型の場合，（n−S ＋ 1 ）× 2

k_1：

n−S ＋ 1

k_2：

max（r−S ＋ 1 ，0）

k_3：

S 文字目から n バイト目までの部分文字列のバイト数

max（n−（S−1）×c，0）≦k_3≦n−S ＋ 1

k_4：

S 文字目から r バイト目までの部分文字列のバイト数

max（r−（S−1）×c，0））≦k_4≦max（r−S ＋ 1 ，0）

c ：

1 文字を表現する最大バイト数

詳細については，「スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合の文字列型，各国文字型，混在文字型，BLOB 型又はバイナリ型で，値式 3 を指定した場合）」の，凡例の c を参照してください。

－：

該当しません。

注

値式 1 が可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）で値式 3 を省略すると，k_2 又は k_4 が 0 であれば結果はナル値になります。

表 2-47　スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合以外の文字列型で，値式 3 を指定した場合）

部分列を抽出する文字列（値式 1）のデータ型	実長	長さ L（値式 3）		
		定数（文字列）		定数以外
		$L_1 \leqq 255$	$L_1 \geqq 256$	
CHAR(n)	－	CHAR（L_1）	VARCHAR（L_1）〔L_2〕	VARCHAR(n)〔L_2〕
VARCHAR(n)	－			

（凡例）

〔 〕：実長を示します。

L_1：

min（値式 3 の長さ L×c，n）

L_2：

抽出した L 文字の部分文字列のバイト数（$L \leqq L_2 \leqq L_1$）

c：

値式結果の文字集合が EBCDIK の場合は 1，UTF16 の場合は 4

n：

処理対象となるデータ列（値式 1）の定義長（バイト単位）

－：

該当しません。

注

値式 1 が可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）の場合，取り出そうとする文字列中で実データの含まれていない部分には，その文字集合の空白が設定されます。

表 2-48　スカラ関数 SUBSTR の結果のデータ型と長さ（値式 1 のデータ型が既定文字集合以外の文字列型で，値式 3 を省略した場合）

部分列を抽出する文字列（値式 1）のデータ型	実長	開始位置 S（値式 2）			
		定 数（文字列）			定数以外
		値式 1 が固定長		値式 1 が可変長	
		$k_0 \leqq 255$	$k_0 \geqq 256$		
CHAR(n)	−	CHAR（k_0）	VARCHAR(n)〔k_1〕	−	VARCHAR(n)〔k_1〕
VARCHAR(n)	〔r〕	−	−	VARCHAR(n)〔k_2〕	VARCHAR(n)〔k_2〕

（凡例）

〔 〕：実長を示します。

n：

処理対象となるデータ列（値式 1）の定義長（バイト単位）

k_0：

$n-(S-1) \times c_1$

k_1：

S バイト目から n バイト目までの部分列のバイト数 $\max（n-（S-1）\times c_2，0）\leqq k_1 \leqq （n-(S-1) \times c_1）$

k_2：

S バイト目から r バイト目までの部分列のバイト数 $\max（r-（S-1）\times c_2，0）\leqq k_2 \leqq \max（r-(S-1)\times c_1，0）$

c_1：

1 文字を表現する最小バイト数

値式結果の文字集合が EBCDIK の場合は 1，UTF16 の場合は 2

c_2：

1 文字を表現する最大バイト数

値式結果の文字集合が EBCDIK の場合は 1，UTF16 の場合は 4

−：

該当しません。

注

値式 1 が可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）で値式 3 を省略すると，k_2 が 0 であれば結果はナル値になります。

(d) 使用例

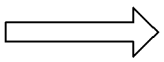
表 T1 の C1 列（データ型：CHAR）のうち，入社年度（C1 列の 2 文字目から 2 文字分）が'95 年の行を検索します。

```
SELECT C1 FROM T1
    WHERE SUBSTR(C1,2,2) = '95'
```

表名：T1

C1列（CHARACTER(6)）

C1列（CHARACTER(6)）
A95157
B93045
A94105
C95009
B95063



実行結果

C1列（CHARACTER(6)）
A95157
C95009
B95063

(21) TIME

(a) 機能

指定した書式の時刻の文字列表現を，時刻データに変換します。

(b) 形式

```
TIME（値式〔，日時書式〕）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 時刻の文字列表現の定数
   - CURRENT_TIME 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 演算結果が時刻データとなる時刻演算
   - 連結演算
   - 集合関数（MAX，MIN）
   - スカラ関数
   - CASE 式
   - CAST 指定

- 関数呼出し
- スカラ副問合せ

2. 値式は，次のデータ型にしてください。
   - 日時書式を指定した場合

     ・定義長が 6～255 の文字データ型（CHAR，VARCHAR）。ただし，文字集合が UTF16 の場合は，定義長が 12～510 の文字データ型（CHAR，VARCHAR）。

     ・混在文字データ型（MCHAR，MVARCHAR）
   - 日時書式を指定しない場合

     ・定義長が 8 バイトの文字データ型（CHAR，VARCHAR）。ただし，文字集合が UTF16 の場合は，定義長が 16 バイトの文字データ型（CHAR，VARCHAR）。

     ・時刻データ型（TIME）
3. 値式は，日時書式で指定した書式の，時刻の文字列表現にしてください。日時値式のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。
4. 値式は，日時書式で指定した書式の，時刻の文字列表現にしてください。日時書式を省略した場合は，時刻の既定の文字列表現にしてください。

   (例)

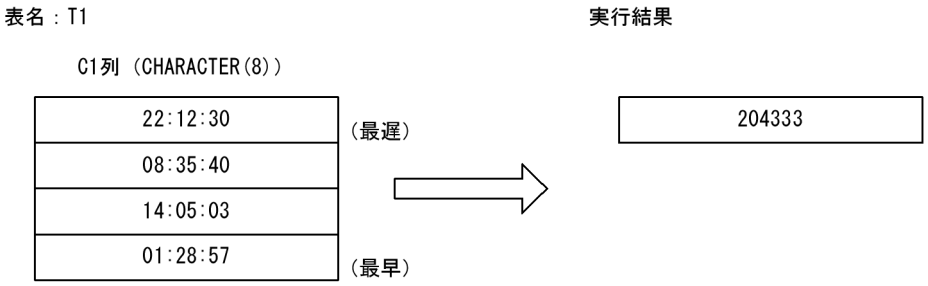
   日時書式'HH-MI-SS'の場合→'13-45-17'

   日時書式を省略した場合→'13:45:17'
5. 値式が時刻データ型の場合，結果はその時刻になります。
6. 日時書式については，「日時書式の指定」を参照してください。
7. 結果のデータ型は，時刻データ型（TIME）になります。
8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式，又は日時書式がナル値であれば，結果もナル値になります。
9. 日時書式を指定した場合，値式と日時書式の文字集合は同じにしてください。ただし，日時書式が次に示す値式である場合，値式の文字集合に変換します。
   - 文字列定数

(d) 使用例

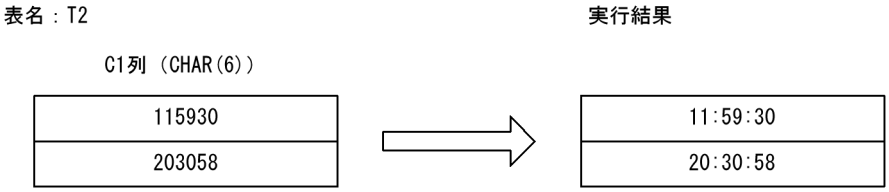
1. 表 T1 の C1 列（データ型：CHAR）のうち，最も早い時刻と最も遅い時刻の差を取得します。

```
SELECT MAX(TIME(C1))-MIN(TIME(C1))
    FROM T1
```



2. 表 T2 の C1 列（データ型：CHAR）の，時刻の既定の文字列表現以外の書式（'HHMISS'）で表現した文字列から，時刻データを取得します。

```
SELECT TIME(C1,'HHMISS') FROM T2
```



(22) TIMESTAMP

(a) 機能

1. 時刻印の既定の文字列表現を，時刻印データに変換します。
2. 西暦 1 年 1 月 1 日からの通算日数を，それが示す時刻印データに変換します。
3. 日付データと時刻データから，それらを組み合わせた時刻印データに変換します。

(b) 形式

機能 1.の形式

```
TIMESTAMP（値式）
```

機能 2.の形式

```
TIMESTAMP（値式）
```

機能 3.の形式

```
TIMESTAMP（値式1，値式2）
```

(c) 機能 1.の規則

1. 値式に指定できるものを次に示します。
   - 時刻印の，既定の文字列表現の定数
   - CURRENT_TIMESTAMP 値関数

- 列指定
- コンポネント指定
- SQL 変数又は SQL パラメタ
- 連結演算
- 集合関数
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 指定できるデータ型を次に示します。
   - 定義長が 19～26 バイトの文字データ型（CHAR，VARCHAR）。ただし，文字集合が UTF16 の場合は，定義長が 38～52 バイトの文字データ型（CHAR，VARCHAR）。
   - 時刻印データ型（TIMESTAMP）
3. 値式のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。
4. 値式に文字データ型を指定する場合は，時刻印の既定の文字列表現にしてください。
5. 値式が時刻印データ型の場合，結果はその時刻印になります。
6. 結果のデータ型は，時刻印データ型（TIMESTAMP）になり，次の小数秒精度になります。
   - 値式が文字データ型の場合は，値式の時刻印の既定の文字列表現に従う小数秒精度となります。
   - 値式が時刻印データの場合は，その時刻印データの小数秒精度となります。

(d) 機能 2.の規則

1. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - コンポネント指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し

- スカラ副問合せ

2. 値式のデータ型は，整数（INTEGER）にしてください。また，四則演算，スカラ関数，CASE 式，CAST 指定，関数呼出し，又は集合関数を指定した場合，演算の結果を整数のデータ型にする必要があります。
3. 値式の範囲は，1～3,652,059 にしてください。
4. 結果は，西暦 1 年 1 月 1 日から，（指定した数値－1）日後の時刻印になります。また，結果の時刻部分は 0 時 0 分 0 秒になります。

   (例)

   値式が 35 の場合→西暦 1 年 2 月 4 日 0 時 0 分 0 秒

(e) 機能 3.の規則

1. 値式 1 に指定できるものを次に示します。
   - 日付の既定の文字列表現の定数
   - CURRENT_DATE 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数又は SQL パラメタ
   - 演算結果が日付データ型となる日付演算
   - 連結演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式 1 に指定できるデータ型を次に示します。
   - 定義長が 10 バイトの文字データ型（CHAR，VARCHAR）。ただし，文字集合が UTF16 の場合は，定義長が 20 バイトの文字データ型（CHAR，VARCHAR）。
   - 日付データ型（DATE）
3. 値式 1 のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。
4. 値式 1 は，日付の既定の文字列表現にしてください。

   (例)

   '1995-06-30'
5. 値式 2 に指定できるものを次に示します。

- 時刻の既定の文字列表現の定数
- CURRENT_TIME 値関数
- 列指定
- コンポネント指定
- SQL 変数又は SQL パラメタ
- 演算結果が時刻データとなる時刻演算
- 連結演算
- 集合関数
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

6. 値式 2 に指定できるデータ型を次に示します。
   - 定義長が 8 バイトの文字データ型（CHAR，VARCHAR）。ただし，文字集合が UTF16 の場合は，定義長が 16 バイトの文字データ型（CHAR，VARCHAR）。
   - 時刻データ型（TIME）
7. 値式 2 のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。
8. 値式 2 は，時刻の既定の文字列表現にしてください。

   (例)

   '13:45:17'
9. 値式 1 と値式 2 に指定する文字集合は同じにしてください。ただし，値式 1 と値式 2 のどちらかが次に示す値式の場合，対応する値式の文字集合に変換します。
   - 文字列定数

(f) 共通規則

1. 結果のデータ型は，時刻印データ型（TIMESTAMP）になります。
2. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。

(g) 使用例

1. 表 T1 の C1 列（データ型：時刻印データ型）のうち，指定した時刻印以降のデータを検索します。

```
SELECT C1 FROM T1
    WHERE C1 >= TIMESTAMP('2000-01-01 00:00:00.00')
```

表名：T1

C1列（時刻印データ型）
2002-01-01 00:00:00.00
2001-12-31 23:59:59.59
1999-12-31 23:59:40.35
1995-01-24 10:10:10.00

実行結果

C1列（時刻印データ型）
2002-01-01 00:00:00.00
2001-12-31 23:59:59.59

2. 表 T1 の C1 列（データ型：数データ型）を時刻印データに変換して，表 T2 の C1 列（データ型：時刻印データ型）に挿入します。

```
INSERT INTO T2(C1) SELECT TIMESTAMP(C1) FROM T1
```

表名：T1

C1列（数データ型）
730000
735000
740000
745000

実行結果
表名：T2

C1列（時刻印データ型）
1999-09-03 00:00:00
2013-05-12 00:00:00
2027-01-19 00:00:00
2040-09-27 00:00:00

3. 表 T1 の C1 列（データ型：日付データ型）と，表 T1 の C2 列（データ型：時刻データ型）とを組み合わせたデータを，表 T2 の C1 列（データ型：時刻印データ型）に挿入します。

```
INSERT INTO T2(C1) SELECT TIMESTAMP(C1,C2) FROM T1
```

表名：T1

C1列（日付データ型）	C2列（時刻データ型）
2002-04-20	00:00:00
2002-05-10	10:10:10
2002-04-26	15:16:20
2002-04-29	18:03:12

実行結果
表名：T2

C1列（時刻印データ型）
2002-04-20 00:00:00
2002-05-10 10:10:10
2002-04-26 15:16:20
2002-04-29 18:03:12

(23) TIMESTAMP_FORMAT

(a) 機能

指定した日時書式に従った時刻印の文字列表現を，時刻印データに変換します。

(b) 形式

```
TIMESTAMP_FORMAT （値式，日時書式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 時刻印の文字列表現の定数
   - 列指定
   - コンポネント指定

- SQL 変数又は SQL パラメタ
- 連結演算
- 集合関数
- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式は次のデータ型にしてください。
   - 定義長が 14～255 バイトの文字データ型（CHARACTER，VARCHAR）。ただし，文字集合が UTF16 の場合は，定義長が 28～510 バイトの文字データ型（CHARACTER，VARCHAR）。
   - 混在文字データ型（MCHAR，MVARCHAR）
3. 値式のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。
4. 値式は，日時書式で指定した書式の，時刻印の文字列表現にしてください。
   (例)
   日時書式'YYYY/MM/DD HH-MI-SS.NNNN'の場合
   →'2002/06/30 10-45-30.1523'
5. 日時書式については，「日時書式の指定」を参照してください。
6. 結果のデータ型は時刻印データ型（TIMESTAMP）となります。また，小数秒精度は 6 になります。
7. 値式と日時書式に指定する文字集合は同じにしてください。ただし，日時書式が次の場合，対応する値式の文字集合に変換します。
   - 文字列定数

(d) 共通規則

1. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式，又は日時書式がナル値であれば，結果もナル値になります。

(e) 使用例

表 T1 の C1 列（データ型：CHAR）の，時刻印の既定の文字列表現以外の書式('DD/MON/YYYY HH-MI-SS NNNN')で表現した文字列から，時刻印データを取得します。

```
SELECT TIMESTAMP_FORMAT(C1,'DD/MON/YYYY HH-MI-SS NNNN') FROM T1
```

表名：T1

C1列（CHAR(24)）

C1列（CHAR(24)）
01/JAN/2000 10-23-02 5620
16/JUL/2002 18-43-37 3415

実行結果

2000-01-01 10:23:02.562000
2002-07-16 18:43:37.341500

(24) TRIM

(a) 機能

TRIM もとに指定した文字列から，TRIM 文字に指定した文字を取り除きます。TRIM 文字で指定した文字以外が現れるまで繰り返し実行されます。

TRIM 指定に LEADING を指定した場合は左側から，TRAILING を指定した場合は右側から，BOTH を指定又は指定を省略した場合は左右から取り除きます。

(b) 形式

```
TRIM（〔〔TRIM指定〕〔TRIM文字〕 FROM 〕TRIMもと）
  TRIM指定：：＝ {LEADING｜TRAILING｜BOTH}
```

(c) 規則

1. TRIM 文字，及び TRIM もとに指定できるものを次に示します。
   - 定数（文字列，各国文字列，混在文字列，又は 16 進文字列）
   - USER 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. FROM 句を指定する場合は，TRIM 指定若しくは TRIM 文字，又は両方を指定してください。
3. TRIM 文字を省略すると，文字列型，各国文字列型，及び混在文字列型の場合は結果のデータ型に応じた空白文字が仮定されます。BINARY 型の場合は，X'00'が仮定されます。
4. TRIM 文字及び TRIM もとに指定できるデータ型の組み合わせを次の表に示します。

表 2-49　TRIM 文字及び TRIM もとに指定できるデータ型の組み合わせ

TRIM もと		TRIM 文字					
		CHAR 又は VARCHAR			NCHAR 又は NVARCHAR	MCHAR 又は MVARCHAR	最大長が 32,000 バイト以下の BINARY
		既定	EBCDIK	UTF16			
CHAR 又は VARCHAR	既定	○	×	×	×	△※2	△※4
	EBCDIK	△※1	○	×	×	×	×
	UTF16	△※1	×	○	×	×	×
NCHAR 又は NVARCHAR		×	×	×	○	×	×
MCHAR 又は MVARCHAR		○※5	×	×	×	○	×
最大長が 32,000 バイト以下の BINARY		△※3	×	×	×	×	○

（凡例）

○：指定できます。

△：制限付きで指定できます。

×：指定できません。

注※1

TRIM 文字が文字列定数又は 16 進文字列定数の場合だけ指定できます。

注※2

TRIM 文字が 1 バイト文字 1 文字の場合だけ指定できます。

注※3

TRIM 文字が 16 進文字列定数の場合だけ指定できます。

注※4

TRIM もとが 16 進文字列定数の場合だけ指定できます。

注※5

TRIM 文字を MCHAR，又は MVARCHAR として扱います。

5. 結果のデータ型を次の表に示します。

表 2-50　システム組込みスカラ関数 TRIM の結果のデータ型

TRIM もとのデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)
BINARY(n)	BINARY(n)

6. 結果のデータ型が文字列型の場合，TRIM もとの文字集合が結果の文字集合となります。

7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。TRIM 文字又は TRIM もとの値式がナル値であれば，結果もナル値になります。
8. TRIM 文字の値の長さは，文字データ型，混在文字データ型，及び各国文字データ型の場合，文字単位の長さが 1 になるように指定してください。BINARY 型の場合は 1 バイトになるように指定してください。

(d) 使用例

TRIM の使用例を次に示します。

```
TRIM(LEADING 'a' FROM 'aaaadata base')   ==> 'data base'
TRIM(TRAILING 'a' FROM 'data baseaaaaaaa')   ==> 'data base'
TRIM(BOTH 'a' FROM 'aaaaadata baseaaaaaaa')   ==> 'data base'
TRIM('a' FROM 'aaaaadata baseaaaaaaa')   ==> 'data base'
TRIM('    data base    ')              ==> 'data base'
```

(25) UPPER

(a) 機能

文字データ，各国文字データ，又は混在文字データ中の英小文字を英大文字に変換します。

(b) 形式

```
UPPER（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 定数
   - USER 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数（HEX，LOWER，SUBSTR，UPPER）
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ

2. 値式に，NULL，埋込み変数，及び?パラメタは指定できません。
3. 値式のデータ型は，文字データ型（CHAR，VARCHAR），各国文字データ型（NCHAR，NVARCHAR），又は混在文字データ型（MCHAR，MVARCHAR）にしてください。
4. 値式のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は任意です。
5. 結果のデータ型，及びデータ長は，値式のデータ型，及びデータ長と同じになります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式がナル値であれば，結果もナル値になります。
7. 結果の文字集合は，引数に指定した値式の文字集合となります。

(26) VALUE

(a) 機能

値式の並びの中からナル値でない最初の値式の示す値を抽出します。

(b) 形式

```
VALUE（値式〔，値式〕…）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - 定数
   - USER 値関数
   - CURRENT_DATE 値関数
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 日付演算
   - 時刻演算
   - 連結演算
   - 集合関数
   - スカラ関数
   - CASE 式

- CAST 指定
- ?パラメタ，又は埋込み変数
- 関数呼出し
- スカラ副問合せ

2. 値式の数は 255 個以内で指定します。
3. 値式の演算結果が次に示すデータ型の場合は指定できません。
   - BLOB
   - 最大長が 32,001 バイト以上の BINARY
   - BOOLEAN
   - 抽象データ型
4. 値式に，NULL は指定できません。
5. 最初の値式に，?パラメタ，及び埋込み変数は単独（単項演算式に指定した場合も含む）で指定できません。
6. 値式のデータ型は，それぞれ比較できるデータ型にしてください。
   （例）
   値式の一つのデータ型が CHAR の場合，ほかの値式も CHAR にしてください。
   比較できるデータ型については，「データ型」を参照してください。
   ただし，次に示すデータは比較できません。
   - 日付データと日付データの文字列表現
   - 時刻データと時刻データの文字列表現
   - 時刻印データと時刻印データの文字列表現
   - 日間隔データと日間隔データの 10 進数表現
   - 時間隔データと時間隔データの 10 進数表現
   - バイナリデータと 16 進文字列定数
7. VALUE の一つ以上の値式が，?パラメタ，又は埋込み変数の場合，?パラメタ，又は埋込み変数のデータ型として最初の値式のデータ型を想定します。
8. 値式の並びは左から右に順に評価され，ナル値でない最初の値が結果になります。
9. 結果のデータ型，及びデータ長は，集合演算（UNION ALL，又は EXCEPT ALL）の結果のデータ型，及びデータ長と同じになります。詳細については，「問合せ式」を参照してください。
10. 結果の値は，非ナル値制約なし（ナル値を許します）になります。すべての値式がナル値であれば，結果もナル値になります。
11. 値式が文字データ型の場合，値式の文字集合はすべて同じにしてください。ただし，二つ目以降の引数に指定した値式が次に示す値式である場合，先頭の引数で指定した値式の文字集合に変換されます。
    - 文字列定数

- 埋込み変数（既定文字集合）
- ?パラメタ

12. 結果のデータ型が文字データ型となる場合，先頭の引数で指定した値式の文字集合が結果の文字集合となります。

(d) 使用例

表 T5 の列 C2 からナル値を抽出して，0 を設定します。

```
SELECT VALUE(C1,C2,C3,0)
    FROM T5
```

表名：T5

C1列	C2列	C3列
*	20	*
*	*	*

実行結果

20
0

注　* はナル値を示します。

(27) VARCHAR_FORMAT

(a) 機能

日付データ，時刻データ，又は時刻印データを指定した日時書式に従って，文字列表現に変換します。

(b) 形式

```
VARCHAR_FORMAT （値式，日時書式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数又は SQL パラメタ
   - 演算結果が日付データ型となる日付演算
   - 演算結果が時刻データ型となる時刻演算
   - 集合関数

- スカラ関数
- CASE 式
- CAST 指定
- 関数呼出し
- スカラ副問合せ

2. 値式のデータ型は，日付データ型（DATE），時刻データ型（TIME），又は時刻印データ型（TIMESTAMP）にしてください。
3. 日時書式については，「日時書式の指定」を参照してください。
4. 結果のデータ型を次に示します。

   日時書式が文字データ型（CHAR 又は VARCHAR）の場合：

   VARCHAR(n)

   日時書式が混在文字データ型（MCHAR 又は MVARCHAR）の場合：

   MVARCHAR(n)

   定義長 n は次の値となります。

   ・値式を定数以外で指定し，かつ日時書式を定数で指定した場合は，指定した書式に従って変換できる文字列の最大長となります。

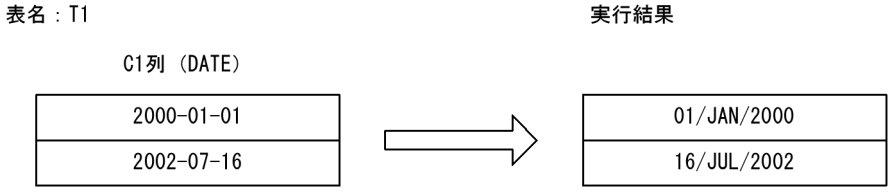
   ・値式を定数で指定し，かつ日時書式を定数で指定した場合は，書式に従って変換した後の文字列長となります。

   ・日時書式を定数以外で指定した場合は，日時書式のデータ型の定義長＋ 15 となります。ただし，日時書式のデータ型が，文字集合が UTF16 の文字データ型の場合，日時書式のデータ型の定義長＋ 30 となります。
5. 結果の値は，値式のデータ型の，既定の文字列表現になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。値式，又は日時書式がナル値であれば，結果もナル値になります。
7. 結果の文字集合は，引数に指定した日時書式の文字集合となります。

(d) 使用例

表 T1 の C1 列（データ型：DATE）から，指定した日時書式（'DD/MON/YYYY'）で表現した文字列で結果を取得します。

```
SELECT VARCHAR_FORMAT(C1,'DD/MON/YYYY') FROM T1
```



(28) YEAR

(a) 機能

日付データ，時刻印データ，又は日間隔データから，年の部分だけを抽出します。

(b) 形式

```
YEAR（値式）
```

(c) 規則

1. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - コンポネント指定
   - SQL 変数，又は SQL パラメタ
   - 日付演算
   - 集合関数（MAX，MIN）
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 関数呼出し
   - スカラ副問合せ
2. 値式のデータ型は，日付データ型（DATE），時刻印データ型（TIMESTAMP），又は日間隔データ型（INTERVAL YEAR TO DAY）にしてください。
3. 結果のデータ型は，整数（INTEGER）になります。
4. 値式が日付データ型，又は時刻印データ型であれば，結果は 1～9999 になります。
5. 値式が日間隔データ型であれば，結果は－9999～9999 になります。結果が 0 以外の場合，値式の符号と結果の符号は同じになります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。すべての値式がナル値であれば，結果もナル値になります。

(d) 使用例

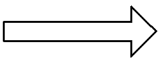
表 T1 の C2 列（データ型：INTEGER）の値が 221140 の C1 列（データ型：日付データ型）の年度を取得します。ただし，C1 列は年度末を表すため，表示の際は 1 年前の年を出力します。

```
SELECT SUBSTR(DIGITS(YEAR(C1)-1),7,4),N'年度' FROM T1
    WHERE C2=221140
```

表名：T1

C1列（日付データ型）　C2列（INTEGER）

1993 - 03 - 20	221140
1994 - 03 - 20	218030
1995 - 03 - 19	220100



実行結果

1992年度

2.16.2　システム定義スカラ関数

ここでは，システム定義スカラ関数の文法について説明します。

なお，システム定義スカラ関数を使用する場合は，事前に必要となる作業及び規則がありますので注意してください。

事前に必要となる作業

- システム定義スカラ関数を使用する場合，pdinit 又は pdmod でデータディクショナリ LOB 用 RD エリアを作成しておく必要があります。pdinit 及び pdmod については，マニュアル「HiRDB コマンドリファレンス」を参照してください。
- 拡張システム定義スカラ関数を使用する場合は，あらかじめ pdextfunc コマンドで関数を定義してください。
- 拡張システム定義スカラ関数を使用する場合，ディクショナリ表に行が追加されるため，データディクショナリ用 RD エリアの容量，及びシステムログ量が増加します。増加する容量は，あらかじめ見積もっておいてください。容量の見積もりについては，マニュアル「HiRDB システム導入・設計ガイド」の「データディクショナリ用 RD エリアの容量の見積もり」及び「拡張システム定義スカラ関数の定義時に出力されるシステムログ量」を参照してください。なお，不要になった拡張システム定義スカラ関数は，pdextfunc コマンドで削除できます。そうすることで，データディクショナリ用 RD エリアの空き容量を増やせます。

規則

- 繰返し列を引数の値式として指定する場合，添字を指定してください。ただし，添字として ANY は指定できません。
- 予備列は引数中に指定できません。
- 埋込み変数又は？パラメタを，単独で値式に指定する場合は，AS データ型を必ず指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。
- システム定義スカラ関数を指定して導出した，名前付き導出表に対する問合せでは，ホールダブルカーソルを使用できません。
- システム定義スカラ関数は，ビュー定義中の導出問合せ式には指定できません。

(1) ACOS

(a) 機能

引数の逆余弦である角度を，0～π の範囲（ラジアン単位）で返します。

(b) 形式

```
〔MASTER.〕ACOS（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数値が定義されていない値を引数に指定すると，定義域エラー（domain error occurs）になります。

(2) ADD_INTERVAL

(a) 機能

引数 1 で指定した既定の文字列表現（'YYYY-MM-DD hh:mm:ss'）の時刻印に，引数 2 で指定した 10 進数表現（±YYYYMMDDhhmmss.）の日時間隔を加算した，時刻印の既定の文字列表現を返します。

(b) 形式

```
〔MASTER.〕ADD_INTERVAL（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - 文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 の値式に指定できるものを次に示します。
   - 10 進数定数又は整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式

- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は文字データ型（CHAR 又は VARCHAR）にしてください。また，値の長さは 19 バイト以下にしてください。
6. 引数 1 のデータ型が文字データ型（CHAR，VARCHAR）の場合，そのデータ型の文字集合は UTF16 以外にしてください。引数 1 に，文字集合が UTF16 の文字データ型を指定する場合，CAST 指定を用いて既定文字集合の文字データ型に変換してください。
7. 引数 2 のデータ型は DECIMAL，INTEGER，又は SMALLINT にしてください。また，14 けた以下の整数値を指定してください。なお，小数点以下を指定しても無視されます。
8. 結果のデータ型は CHAR(19)になります。
9. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
10. 関数の結果の文字集合は，引数 1 の文字集合となります。
11. 引数 1 の値には，有効な時刻印の既定の文字列表現（'YYYY-MM-DD hh:mm:ss'）を指定してください。時刻印の既定の文字列表現には，小数秒を指定しないでください。
12. 引数 2 の値式には，日時間隔の 10 進数表現（±YYYYMMDDhhmmss.）を指定してください。
    - YYYY：年数
    - MM：月数
    - DD：日数
    - hh：時数
    - mm：分数
    - ss：秒数

    引数 2 の値が正の場合，引数 1 の時刻印への日時間隔（YYYYMMDDhhmmss.）の加算になります。
    引数 2 の値が負の場合，引数 1 の時刻印への日時間隔（YYYYMMDDhhmmss.）の減算になります。
13. 日時間隔の加減算は，年，月，日，時，分，秒の順に演算します。年又は月の演算結果が存在しない日付（小の月の 31 日，及びうるう年以外の年の 2 月 29 日）になった場合の結果は，その月の最終日に修正されます。

    なお，任意の月の最終日から数か月後が，必ずその月の最終日になるわけではありません。また，ある日付に任意の月数を加算した後，その結果の日付から同じ月数を引いても，必ず元の日付になるとは限らないので注意してください。

    同じ月数を引いても，必ず元の日付になるとは限らないので注意してください。

14. 時刻印の既定の文字列表現でうるう秒を指定した場合，演算結果は次のようになります。
    - 分以上の演算で，結果の秒が 60 秒以上になった場合は，59 秒に修正されます。
    - 秒の演算では，60 秒を 1 分，61 秒を 1 分 1 秒として演算します。
    - 関数の結果には，うるう秒を含みません。
15. 関数の結果が，0001 年 01 月 01 日 00 時 00 分 00 秒～9999 年 12 月 31 日 23 時 59 分 59 秒の範囲でない場合，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の演算結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。
16. 関数の結果は，時刻印の既定の文字列表現になります。

(d) 使用例

ADD_INTERVAL の使用例を次に示します。

```
ADD_INTERVAL('1999-12-31 23:59:59',10000000001.)
  ==> '2001-01-01 00:00:00'

ADD_INTERVAL('2001-01-01 00:00:00',-10000000001.)
  ==> '1999-12-31 23:59:59'

ADD_INTERVAL('1956-06-07 03:15:30',400313115450.)
  ==> '1996-09-20 15:10:20'

ADD_INTERVAL('1998-12-31 13:59:59',10200030405.)
  ==> '2000-02-29 17:04:04'

ADD_INTERVAL('2000-02-29 17:04:04',-10200030405.)
  ==> '1998-12-28 13:59:59'

ADD_INTERVAL('2000-03-05 17:04:60',100.)
  ==> '2000-03-05 17:05:59'
  システム共通定義 pd_leap_secondが Yの場合です。

ADD_INTERVAL('2000-03-05 17:04:60',-60.)
  ==> '2000-03-05 17:04:00'
  システム共通定義 pd_leap_secondがYの場合です。

ADD_INTERVAL(CAST(CURRENT_TIMESTAMP AS CHAR(19)),-100000000.)
  ==> '2008-09-29 10:17:48'
  CURRENT TIMESTAMPが，2008年10月29日10時17分48秒の場合です。
```

(3) ASCII

(a) 機能

引数で指定した文字列の，最初の文字の ASCII コードを整数値で返します。

(b) 形式

```
〔MASTER.〕ASCII（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は文字データ型（CHAR 又は VARCHAR）にしてください。
5. 引数のデータ型の文字集合は既定文字集合にしてください。
6. 結果のデータ型は INTEGER になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
8. 引数の値の長さが 0 バイトの場合，結果はナル値になります。

(d) 使用例

ASCII の使用例を次に示します。

```
ASCII('ABC')       ==> 65
CHR(ASCII('ABC'))  ==> 'A'
```

(4) ASIN

(a) 機能

引数の逆正弦である角度を，－π/2～π/2 の範囲（ラジアン単位）で返します。

(b) 形式

```
〔MASTER.〕ASIN（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数値が定義されていない値を引数に指定すると，定義域エラー（domain error occurs）になります。

(5) ATAN

(a) 機能

引数の逆正接である角度を，－π/2～π/2 の範囲（ラジアン単位）で返します。

(b) 形式

```
〔MASTER.〕ATAN（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(6) ATAN2

(a) 機能

x/y の逆正接である角度を，－π～π の範囲（ラジアン単位）で返します。なお，x は引数 1 とし，y は引数 2 とします。

(b) 形式

```
〔MASTER.〕ATAN2（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。

(7) CEIL

(a) 機能

引数の値以上の，最小の整数値を返します。

(b) 形式

```
〔MASTER.〕CEIL（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型を次の表に示します。

表 2-51　システム定義スカラ関数 CEIL の結果のデータ型

引数のデータ型	結果のデータ型
SMALLINT	INTEGER
INTEGER	INTEGER
DECIMAL(p,s)	DECIMAL(p,s)

引数のデータ型	結果のデータ型
SMALLFLT	FLOAT
FLOAT	FLOAT

6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(8) CENTURY

(a) 機能

引数で指定された日付の世紀を返します。

(b) 形式

```
〔MASTER.〕CENTURY（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は DATE にしてください。
5. 結果のデータ型は INTEGER になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d) 使用例

CENTURY の使用例を次に示します。

```
CENTURY(DATE('1900-12-31'))    ==> 19
CENTURY(DATE('1901-01-01'))    ==> 20
CENTURY(DATE('1999-12-31'))    ==> 20
CENTURY(DATE('2000-12-31'))    ==> 20
CENTURY(DATE('2001-01-01'))    ==> 21
```

(9) CHR

(a) 機能

引数で指定した整数値が示す，ASCII コードの文字を返します。引数の値が 0～255 の範囲外の値ならば，ナル値を返します。

(b) 形式

```
〔MASTER.〕CHR（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し

- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は，INTEGER 又は SMALLINT にしてください。
5. 結果のデータ型は CHAR(1)になります。
6. 結果の文字集合は既定の文字集合になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d) 使用例

CHR の使用例を次に示します。

```
CHR(65)           ==> 'A'
ASCII(CHR(65))    ==> 65
```

(10) COS

(a) 機能

ラジアンで角度を指定した，引数の余弦（三角関数 COS）を返します。

(b) 形式

```
〔MASTER.〕COS（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算

- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(11) COSH

(a) 機能

引数の双曲線余弦を返します。

(b) 形式

```
〔MASTER.〕COSH（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し

- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(12) DATE_TIME

(a) 機能

引数 1 で指定した DATE 型の日付と，引数 2 で指定した TIME 型の時刻とを，時刻印の既定の文字列表現('YYYY-MM-DD hh:mm:ss')に変換して返します。

(b) 形式

```
〔MASTER.〕DATE_TIME（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し

- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 引数 2 の値式に指定できるものを次に示します。
   - CURRENT_TIME 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 時刻演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は DATE にしてください。引数 2 のデータ型は TIME にしてください。
6. 結果のデータ型は CHAR(19)になります。
7. 結果の文字集合は，既定の文字集合となります。
8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
9. DATE_TIME（日時，時刻）の結果である時刻印の既定の文字列表現は，日付の既定の文字列表現，1 文字の空白，及び時刻の既定の文字列表現を連結した値になります。

```
CHAR(日付) || ' ' || CHAR(時刻)
```

(d) 使用例

DATE_TIME の使用例を次に示します。

```
DATE_TIME(DATE('1999-12-31'), TIME('23:59:59'))
  ==> '1999-12-31 23:59:59'
DATE_TIME(CURRENT_DATE, CURRENT_TIME)
  ==> '1999-07-27 11:05:20'
```

(13) DAYNAME

(a) 機能

引数で指定した日付の，曜日を示す文字列（'Sunday'，'Monday'などの英語の文字列）を返します。

(b) 形式

```
〔MASTER.〕DAYNAME（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は DATE にしてください。
5. 結果のデータ型は VARCHAR(18)になります。
6. 結果の文字集合は，既定の文字集合となります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d) 使用例

DAYNAME の使用例を次に示します。

```
DAYNAME(DATE('1999-06-06'))    ==> 'Sunday'
DAYNAME(DATE('1999-06-07'))    ==> 'Monday'
DAYNAME(DATE('1999-06-08'))    ==> 'Tuesday'
DAYNAME(DATE('1999-06-09'))    ==> 'Wednesday'
DAYNAME(DATE('1999-06-10'))    ==> 'Thursday'
DAYNAME(DATE('1999-06-11'))    ==> 'Friday'
DAYNAME(DATE('1999-06-12'))    ==> 'Saturday'
```

(14) DAYOFWEEK

(a) 機能

引数で指定した日付の，曜日を示す 1～7 の整数値（その週の第何日目かを示す値）を返します。なお，週の最初の日は日曜日とし，1 は日曜日を示します。

(b) 形式

```
〔MASTER.〕DAYOFWEEK（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は DATE にしてください。

5. 結果のデータ型は INTEGER になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d) 使用例

DAYOFWEEK の使用例を次に示します。

```
DAYOFWEEK(DATE('1999-06-06'))    ==> 1
DAYOFWEEK(DATE('1999-06-07'))    ==> 2
DAYOFWEEK(DATE('1999-06-08'))    ==> 3
DAYOFWEEK(DATE('1999-06-11'))    ==> 6
DAYOFWEEK(DATE('1999-06-12'))    ==> 7
  なお，'1999-06-06' は，日曜日です。
```

(15) DAYOFYEAR

(a) 機能

引数で指定した日付が，その年の第何日目かを示す 1～366 の整数値を返します。

(b) 形式

```
〔MASTER.〕DAYOFYEAR（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は DATE にしてください。
5. 結果のデータ型は INTEGER になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d) 使用例

DAYOFYEAR の使用例を次に示します。

```
DAYOFYEAR(DATE('1999-01-01'))    ==>   1
DAYOFYEAR(DATE('1999-02-28'))    ==>  59
DAYOFYEAR(DATE('1999-06-07'))    ==> 158
DAYOFYEAR(DATE('1999-12-31'))    ==> 365
DAYOFYEAR(DATE('2000-12-31'))    ==> 366
```

(16) DEGREES

(a) 機能

引数で指定した角度を，ラジアンから度数に変換します。

(b) 形式

```
〔MASTER.〕DEGREES（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式

- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(17) EXP

(a) 機能

自然対数の底の累乗を求めます。

(b) 形式

```
〔MASTER.〕EXP（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定

- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(18) FLOOR

(a) 機能

引数の値以下の，最大の整数を返します。

(b) 形式

```
〔MASTER.〕FLOOR（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ

- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型を次の表に示します。

表 2-52　システム定義スカラ関数 FLOOR の結果のデータ型

引数のデータ型	結果のデータ型
SMALLINT	INTEGER
INTEGER	INTEGER
DECIMAL(p, s)	DECIMAL(p, s)
SMALLFLT	FLOAT
FLOAT	FLOAT

6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(19)　GREATEST

(a)　機能

すべての引数中の最大値を返します。

(b)　形式

```
〔MASTER.〕GREATEST（引数，引数〔，引数〕）
```

(c)　規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数の数は最大 3 個です。
3. 値式に指定できるものを次に示します。
   - 定数
   - USER 値関数，CURRENT_DATE 値関数，又は CURRENT_TIME 値関数

- 列指定
- SQL 変数又は SQL パラメタ
- 四則演算，日付演算，時刻演算，又は連結演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-53　システム定義スカラ関数 GREATEST（引数 3 を省略する場合）の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	指定可否
INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	○
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
DATE	DATE	○
TIME	TIME	○
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

表 2-54　システム定義スカラ関数 GREATEST（引数 3 を指定する場合）の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	指定可否
INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	○

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
DATE	DATE	DATE	○
TIME	TIME	TIME	○
上記以外のデータ型の組み合わせ			×

(凡例)

○：指定できます。

×：指定できません。

6. 引数に指定した値式のデータ型が文字データ型の場合，値式の文字集合はすべて同じにしてください。

7. 結果のデータ型を次の表に示します。

表 2-55　システム定義スカラ関数 GREATEST の結果のデータ型

引数のデータ型	結果のデータ型
INTEGER 又は SMALLINT	INTEGER
DECIMAL〔,INTEGER 又は SMALLINT〕	DECIMAL※1
FLOAT,SMALLFLT〔,DECIMAL,INTEGER,又は SMALLINT〕	FLOAT
CHAR 又は VARCHAR	VARCHAR※2
MCHAR 又は MVARCHAR	MVARCHAR※2
NCHAR 又は NVARCHAR	NVARCHAR※2
DATE	DATE
TIME	TIME

注※1

精度と位取りを次に示します。pi 及び si を，それぞれ i 番目の引数の精度及び位取りとします。

精度＝ max(p1-s1, p2-s2,…)＋ max(s1, s2,…)

位取り＝ max(s1, s2,…)

結果の精度が 38 を超えた場合はエラーになります。

また，INTEGER は DECIMAL(10,0)，SMALLINT は DECIMAL(5,0)として扱われます。

注※2

最大長を次に示します。ni を i 番目の引数の最大長（固定長のデータ型の場合は定義長）とします。

最大長＝ max(n1, n2,…)

8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式のどれかがナル値であれば，結果もナル値になります。
9. 結果のデータ型が文字データ型となる場合，結果の文字集合は引数に指定した値式の文字集合となります。

(20) HALF

(a) 機能

引数 2 で指定した月，及び引数 3 で指定した日を各年度の開始月日として，引数 1 で指定された日付が上期か下期かを 1～2 の整数で返します。

(b) 形式

```
〔MASTER.〕HALF（引数1〔，引数2〔，引数3〕〕）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 及び引数 3 の値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算

- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は DATE にしてください。引数 2 及び引数 3 のデータ型は，INTEGER 又は SMALLINT にしてください。
6. 結果のデータ型は INTEGER になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式のどれかがナル値であれば，結果もナル値になります。
8. 引数 2 を省略すると，1 を仮定します。引数 2 に指定できる値の範囲は，1～12 です。

   引数 3 を省略すると，1 を仮定します。引数 3 に指定できる値の範囲は，引数 2 の値が 2 の場合は，1～29 です。それ以外の場合は，1～（引数 2 で指定した月の最終日）です。
9. 結果は，次の規則によって求められます。
   - 引数 1 の日＜引数 3 の場合

     月数＝引数 1 の月－引数 2－1
   - 引数 1 の月≧引数 3 の場合

     月数＝引数 1 の月－引数 2
   - 月数＜ 0 の場合

     結果＝(月数＋ 12)÷2 ＋ 1
   - 月数≧0 の場合

     結果＝月数÷2 ＋ 1

(d) 使用例

HALF の使用例を次に示します。

```
HALF(DATE('1999-01-01'))          ==> 1
HALF(DATE('1999-09-10'))          ==> 2
HALF(DATE('1999-12-31'))          ==> 2
HALF(DATE('1999-04-01'), 4)       ==> 1
HALF(DATE('1999-09-10'), 4)       ==> 1
HALF(DATE('1999-03-31'), 4)       ==> 2
HALF(DATE('1999-03-21'), 3, 21)   ==> 1
```

```
HALF(DATE('1999-09-20'), 3, 21)   ==> 1
HALF(DATE('1999-03-20'), 3, 21)   ==> 2
```

(21) INSERTSTR（INSERTSTR_LONG）

(a) 機能

引数 1 で指定した文字列に対して，引数 2 で指定した文字位置から引数 3 で指定した文字数の部分文字列を削除し，その位置に引数 4 で指定した文字列を挿入した文字列を返します。

(b) 形式

```
〔MASTER.〕INSERTSTR（引数1，引数2，引数3，引数4）
〔MASTER.〕INSERTSTR_LONG（引数1，引数2，引数3，引数4）
```

(c) 規則

1. 引数 1，引数 2，引数 3，及び引数 4 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 及び引数 4 の値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 及び引数 3 の値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数

- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 及び引数 4 に指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-56　システム定義スカラ関数 INSERTSTR，INSERTSTR_LONG の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 4 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

6. 引数 1 及び引数 4 の両方に文字列型（CHAR 又は VARCHAR)の値式を指定する場合，両方の引数の文字集合は同じにしてください。
7. 引数 4 の値の最大長を次に示します。

表 2-57　システム定義スカラ関数 INSERTSTR の引数 4 の最大長

引数 1（及び引数 4）のデータ型		引数 4 の最大長
CHAR 又は VARCHAR	既定文字集合，及び文字集合(UTF16 以外)	255
	文字集合(UTF16)	510
MCHAR 又は MVARCHAR		255
NCHAR 又は NVARCHAR		127

表 2-58　システム定義スカラ関数 INSERTSTR_LONG の引数 4 の最大長

引数 1（及び引数 4）のデータ型	引数 4 の最大長
CHAR 又は VARCHAR	32000

引数 1（及び引数 4）のデータ型	引数 4 の最大長
MCHAR 又は MVARCHAR	32000
NCHAR 又は NVARCHAR	16000

8. 引数 2 及び引数 3 のデータ型は，INTEGER 又は SMALLINT にしてください。

9. 引数 3 に 0 を指定すると，部分文字列は，削除されません。

10. 結果のデータ型を次に示します。

表 2-59　システム定義スカラ関数 INSERTSTR の結果のデータ型

引数 1 のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

表 2-60　システム定義スカラ関数 INSERTSTR_LONG の結果のデータ型

引数 1 のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(32000)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(32000)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(16000)

11. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式のどれかがナル値であれば，結果もナル値になります。

12. 結果のデータ型が文字データ型の場合，結果の文字集合は引数 1 の文字集合となります。

13. 引数 2 及び引数 3 に指定できる値の範囲を次に示します。なお，m は引数 2 の値です。

表 2-61　システム定義スカラ関数 INSERTSTR の引数 2 及び引数 3 の値の範囲

引数 1 のデータ型		引数 2 の値の範囲	引数 3 の値の範囲
CHAR(n) 又は VARCHAR(n)	既定文字集合，及び文字集合(UTF16 以外)	1～n	0～（n＋1－m）
	文字集合(UTF16)	1～n／2	0～（n／2＋1－m）
MCHAR(n)又は MVARCHAR(n)		1～n	0～（n＋1－m）
NCHAR(n)又は NVARCHAR(n)		1～n	0～（n＋1－m）

表 2-62　システム定義スカラ関数 INSERTSTR_LONG の引数 2 及び引数 3 の値の範囲

引数 1 のデータ型		引数 2 の値の範囲	引数 3 の値の範囲
CHAR(n) 又は VARCHAR(n)	既定文字集合，及び文字集合(UTF16 以外)	1～32000	0～（32001－m）

引数 1 のデータ型		引数 2 の値の範囲	引数 3 の値の範囲
	文字集合(UTF16)	1～16000	0～（16001－m）
MCHAR(n)又は MVARCHAR(n)		1～32000	0～（32001－m）
NCHAR(n)又は NVARCHAR(n)		1～16000	0～（16001－m）

14. 引数 1 で指定した文字列の長さ（文字数）が，引数 2 で指定した文字位置 m よりも短い場合は，文字数が（m－1）になるまで引数 1 の文字集合の空白（引数 1 のデータ型が NCHAR 又は NVARCHAR の場合は全角空白）が埋められ，どの部分文字列も削除されません。

    引数 1 で指定した文字列の長さ（文字数）が m 以上で，（m－1 ＋ nd）未満の場合は，文字位置 m から最後の文字までが削除されます。なお，nd は引数 3 で指定した文字数とします。

15. 結果の長さは，結果のデータ型の最大長を超えないようにしてください。結果の長さが引数 1 の長さより長くなる場合は，INSERTSTR_LONG を使用してください。

(d) 使用例

INSERTSTR，INSERTSTR_LONG の使用例を次に示します。

```
INSERTSTR('data warehouse system', 6, 9, 'base')
  ==> 'data base system'
INSERTSTR_LONG('data system', 6, 0, 'warehouse ')
  ==> 'data warehouse system'
INSERTSTR('data base management system', 11, 11, '')
  ==> 'data base system'
  引数4に指定しているのは，長さ0の文字列です。

INSERTSTR_LONG('data base system', 31, 0, '')
  ==> 'data base system
  なお，結果の値には，'system'の後に14文字の空白を含みます。
```

(22) INTERVAL_DATETIMES

(a) 機能

引数で指定した二つの既定の文字列表現（'YYYY-MM-DD hh:mm:ss'）の，時刻印の間の日時間隔を，10 進数表現（±YYYYMMDDhhmmss.）で返します。引数 1 の時刻印＜引数 2 の時刻印の場合，結果は負の値になります。

(b) 形式

```
〔MASTER.〕INTERVAL_DATETIMES（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
    - 値式〔AS データ型〕

2. 値式に指定できるものを次に示します。
   - 文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数 1 及び引数 2 のデータ型は，文字列型（CHAR 又は VARCHAR）にしてください。
5. 引数 1 及び引数 2 の値式に指定する値式の文字集合は同じにしてください。また，文字集合は UTF16 以外にしてください。引数 1 及び引数 2 に文字集合が UTF16 の文字データ型を指定する場合，CAST 指定を用いて既定文字集合の文字データ型に変換してください。
6. 結果のデータ型は DECIMAL(14)になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
8. 引数 1 及び引数 2 の値には，有効な時刻印の既定の文字列表現（'YYYY-MM-DD hh:mm:ss'）を指定してください。時刻印の既定の文字列表現には，小数秒を指定してないでください。
9. INTERVAL_DATETIMES（時刻印 1，時刻印 2）の結果は，次の規則に従い計算されます。
   - 時刻印 1 ＜時刻印 2 の場合
     結果＝－INTERVAL_DATETIMES（時刻印 2 ，時刻印 1）
   - 時刻印 1≧時刻印 2 の場合
     ・時刻印 1 の秒≧時刻印 2 の秒のとき
     結果の秒＝時刻印 1 の秒－時刻印 2 の秒
     ・時刻印 1 の秒＜時刻印 2 の秒のとき
     結果の秒＝時刻印 1 の秒－時刻印 2 の秒＋ 60
     時刻印 2 の分＝時刻印 2 の分＋ 1
     ・時刻印 1 の分≧時刻印 2 の分のとき

結果の分＝時刻印１の分－時刻印２の分

・時刻印１の分＜時刻印２の分のとき

結果の分＝時刻印１の分－時刻印２の分＋ 60

時刻印２の時＝時刻印２の時＋ 1

・時刻印１の時≧時刻印２の時のとき

結果の時＝時刻印１の時－時刻印２の時

・時刻印１の時＜時刻印２の時のとき

結果の時＝時刻印１の時－時刻印２の時＋ 24

時刻印２の日＝時刻印２の日＋ 1

・時刻印１の日≧時刻印２の日のとき

結果の日＝時刻印１の日－時刻印２の日

・時刻印１の日＜時刻印２の日のとき

結果の日＝時刻印１の日－時刻印２の日＋時刻印２の月の最終日

時刻印２の月＝時刻印２の月＋ 1

・時刻印１の月≧時刻印２の月のとき

結果の月＝時刻印１の月－時刻印２の月

・時刻印１の月＜時刻印２の月のとき

結果の月＝時刻印１の月－時刻印２の月＋ 12

時刻印２の年＝時刻印２の年＋ 1

結果の年＝時刻印１の年－時刻印２の年

- 結果＝（結果の年×10000000000 ＋結果の月×100000000
  ＋結果の日×1000000 ＋結果の時×10000 ＋結果の分×100
  ＋結果の秒)

(d) 使用例

INTERVAL_DATETIMES の使用例を次に示します。

```
INTERVAL_DATETIMES('2001-01-01 00:00:00', '1999-12-31 23:59:59')
  ==>    10000000001.
INTERVAL_DATETIMES('1999-12-31 23:59:59', '2001-01-01 00:00:00')
  ==>   -10000000001.
INTERVAL_DATETIMES('1996-09-20 15:10:20', '1956-06-07 03:15:30')
  ==>   400313115450.
NUMEDIT(INTERVAL_DATETIMES('1996-09-20 15:10:20',
                           '1956-06-07 03:15:30'),
        '<9990" years "90" months "90" days "90" hours "
         90" minutes "90" seconds"')
  ==>   '40 years 3 months 13 days 11 hours 54 minutes 50 seconds'
```

(23) ISDIGITS

(a) 機能

引数の文字列が数字だけかどうかを，BOOLEAN 値で返します。

(b) 形式

```
〔MASTER.〕ISDIGITS（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数又は混在文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は，文字列型（CHAR 又は VARCHAR），又は混在文字列型（MCHAR 又は MVARCHAR）にしてください。また，引数の値の長さは 60 バイト以下にしてください。
5. 引数のデータ型が文字列型（CHAR 又は VARCHAR）の場合，そのデータ型の文字集合は UTF16 以外にしてください。引数に文字集合が UTF16 の文字データ型を指定する場合，CAST 指定を用いて既定文字集合の文字データ型に変換してください。
6. 結果のデータ型は BOOLEAN になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

8. 引数に指定された文字列が数字（'0'～'9'）だけならば，結果は TRUE になり，そうでなければ，結果は FALSE になります。

(d) 使用例

ISDIGITS の使用例を次に示します。

```
ISDIGITS('1234567890')   ==> true
ISDIGITS('123ABC')       ==> false
ISDIGITS('')             ==> false
```

(24) IS_DBLBYTES

(a) 機能

引数の文字列がマルチバイト文字だけかどうかを，BOOLEAN 値で返します。

(b) 形式

```
〔MASTER.〕IS_DBLBYTES（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数又は混在文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。

4. 引数のデータ型は，文字列型（CHAR 又は VARCHAR），又は混在文字列型（MCHAR 又は MVARCHAR）にしてください。
5. 引数のデータ型が文字列型（CHAR 又は VARCHAR）の場合，そのデータ型の文字集合は既定文字集合にしてください。
6. 結果のデータ型は BOOLEAN になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
8. 引数に指定された文字列がマルチバイト文字だけならば，結果は TRUE になり，そうでなければ，結果は FALSE になります。

(d) 使用例

IS_DBLBYTES の使用例を次に示します。

```
IS_DBLBYTES(M'data base management system HiRDB')   ==>  false
IS_DBLBYTES(M'データベース管理システムHiRDB')       ==>  false
IS_DBLBYTES(M'データベース管理システム')             ==>  true
```

(25) IS_SNGLBYTES

(a) 機能

引数の文字列がシングルバイト文字だけかどうかを，BOOLEAN 値で返します。

(b) 形式

```
〔MASTER.〕IS_SNGLBYTES（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数又は混在文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し

- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は，文字列型（CHAR 又は VARCHAR），又は混在文字列型（MCHAR 又は MVARCHAR）にしてください。
5. 引数のデータ型が文字列型（CHAR 又は VARCHAR）の場合，そのデータ型の文字集合は既定文字集合にしてください。
6. 結果のデータ型は BOOLEAN になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
8. 引数に指定された文字列がシングルバイト文字だけならば，結果は TRUE になり，そうでなければ，結果は FALSE になります。

(d) 使用例

IS_SNGLBYTES の使用例を次に示します。

```
IS_SNGLBYTES(M'data base management system HiRDB')   ==> true
IS_SNGLBYTES(M'データベース管理システムHiRDB')       ==> false
IS_SNGLBYTES(M'データベース管理システム')             ==> false
```

(26) LAST_DAY

(a) 機能

引数で指定した日付の，年月の最後の日付を返します。

(b) 形式

```
〔MASTER.〕LAST_DAY（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数

- 列指定
- SQL 変数又は SQL パラメタ
- 日付演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は DATE にしてください。
5. 結果のデータ型は DATE になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d) 使用例

LAST_DAY の使用例を次に示します。

```
LAST_DAY(DATE('1999-01-01'))   ==> '1999-01-31'
LAST_DAY(DATE('1999-02-16'))   ==> '1999-02-28'
LAST_DAY(DATE('1999-06-10'))   ==> '1999-06-30'
LAST_DAY(DATE('1999-12-25'))   ==> '1999-12-31'
LAST_DAY(DATE('2000-02-03'))   ==> '2000-02-29'
```

(27) LEAST

(a) 機能

すべての引数中の，最小値を返します。

(b) 形式

```
〔MASTER.〕LEAST（引数，引数〔，引数〕）
```

(c) 規則

1. 引数に指定できるものを次に示します。

- 値式〔AS データ型〕

2. 引数の数は，最大 3 個です。

3. 値式に指定できるものを次に示します。
   - 定数
   - USER 値関数，CURRENT_DATE 値関数，又は CURRENT_TIME 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算，日付演算，時刻演算，又は連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。

5. 指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-63　システム定義スカラ関数 LEAST（引数 3 を省略する場合）の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	指定可否
INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	○
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
DATE	DATE	○
TIME	TIME	○
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

表 2-64　システム定義スカラ関数 LEAST（引数 3 を指定する場合）の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	指定可否
INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	INTEGER，SMALLINT，DECIMAL，FLOAT，又は SMALLFLT	○
CHAR，又は VARCHAR	CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
DATE	DATE	DATE	○
TIME	TIME	TIME	○
上記以外のデータ型の組み合わせ			×

(凡例)

○：指定できます。

×：指定できません。

6. 引数に指定した値式のデータ型が文字データ型の場合，値式の文字集合はすべて同じにしてください。

7. 結果のデータ型を次の表に示します。

表 2-65　システム定義スカラ関数 LEAST の結果のデータ型

引数のデータ型	結果のデータ型
INTEGER 又は SMALLINT	INTEGER
DECIMAL〔,INTEGER 又は SMALLINT〕	DECIMAL※1
FLOAT,SMALLFLT〔,DECIMAL,INTEGER,又は SMALLINT〕	FLOAT
CHAR 又は VARCHAR	VARCHAR※2
MCHAR 又は MVARCHAR	MVARCHAR※2
NCHAR 又は NVARCHAR	NVARCHAR※2
DATE	DATE
TIME	TIME

注※1

精度と位取りを次に示します。pi 及び si を，それぞれ i 番目の引数の精度及び位取りとします。

精度＝ max（p1－s1，p2－s2，…）＋ max（s1，s2，…）

位取り＝ max（s1，s2，…）

結果の精度が 38 を超えた場合はエラーになります。また，INTEGER は DECIMAL(10,0)，SMALLINT は DECIMAL(5,0)として扱われます。

注※2

最大長を次に示します。ni を i 番目の引数の最大長（固定長のデータ型の場合は定義長）とします。

最大長＝ max（n1，n2，…）

8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式のどれかがナル値であれば，結果もナル値になります。
9. 結果のデータ型が文字列データ型の場合，引数の文字集合が結果の文字集合となります。

(28) LEFTSTR

(a) 機能

引数 1 の文字列の先頭（最も左）から，引数 2 で指定された文字数の部分文字列を抽出します。

(b) 形式

```
〔MASTER.〕LEFTSTR（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 の値式に指定できるものを次に示します。
   - 整数定数

- 列指定
- SQL 変数又は SQL パラメタ
- 四則演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は，次のどれかを指定してください。
   - 任意の文字集合の文字列データ型（CHAR，VARCHAR）
   - 各国文字列データ型（NCHAR，NVARCHAR）
   - 混在文字列データ型（MCHAR，MVARCHAR）
6. 引数 2 のデータ型は，INTEGER 又は SMALLINT にしてください。
7. 引数 2 に指定できる値の範囲を次の表に示します。

表 2-66　システム定義スカラ関数 LEFTSTR の引数 2 の値の範囲

引数 1 のデータ型		引数 2 の値の範囲
CHAR(n) 又は VARCHAR(n)	既定文字集合，及び文字集合(UTF16 以外)	0～n
	文字集合(UTF16)	0～n／2
MCHAR(n)又は MVARCHAR(n)		0～n
NCHAR(n)又は NVARCHAR(n)		0～n

8. 結果のデータ型を次の表に示します。

表 2-67　システム定義スカラ関数 LEFTSTR の結果のデータ型

引数 1 のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

9. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
10. 引数 1 の文字列の値の文字数が，引数 2 で指定した文字数よりも少ない場合，結果は引数 1 と同じ値になります。
11. 結果のデータ型が文字列型の場合，引数 1 の文字集合が結果の文字集合となります。

(d) 使用例

LEFTSTR の使用例を次に示します。

```
LEFTSTR('data base system', 9)    ==> 'data base'
LEFTSTR('data system', 0)         ==> ''  (長さ0の文字列)
```

(29) LN

(a) 機能

引数の自然対数を返します。

(b) 形式

```
〔MASTER.〕LN（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(30) LOG10

(a) 機能

引数の常用対数を返します。

(b) 形式

```
〔MASTER.〕LOG10（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。

5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(31) LTRIM

(a) 機能

引数 1 の文字列の左から，引数 2 で指定された文字列を構成する文字を取り除きます。引数 2 で指定された文字列中の文字以外が現れるまで，繰り返し取り除きます。

(b) 形式

```
〔MASTER.〕LTRIM（引数1〔，引数2〕）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数 1 のデータ型は，次のどれかを指定してください。
   - 任意の文字集合の文字列型（CHAR，VARCHAR）
   - 各国文字列型（NCHAR，NVARCHAR）

- 混在文字列型（MCHAR，MVARCHAR）

5. 引数 2 を指定する場合，指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-68　システム定義スカラ関数 LTRIM の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

6. 引数 2 の値の長さは，文字列データ型又は混在文字列データ型の場合は 30 バイト以下（文字集合が UTF16 の文字データ型の場合は 60 バイト以下），各国文字列データ型の場合は 30 文字以下にしてください。

7. 引数 1 及び引数 2 のデータ型が両方とも文字列型の場合，値式の文字集合は同じにしてください。

8. 結果のデータ型を次の表に示します。

表 2-69　システム定義スカラ関数 LTRIM の結果のデータ型

引数 1（及び引数 2）のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

9. 結果のデータ型が文字列型の場合，引数 1 の文字集合が結果の文字集合となります。

10. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。

11. 引数 2 を省略した場合は，次に示す値を仮定します。

表 2-70　引数 2 を省略した場合の仮定値

引数 1 のデータ型	引数 2 を省略した場合の仮定値
CHAR(n)又は VARCHAR(n)	半角空白 1 文字からなる文字列
MCHAR(n)又は MVARCHAR(n)	半角空白と全角空白からなる文字列
NCHAR(n)又は NVARCHAR(n)	全角空白 1 文字からなる文字列

(d) 使用例

LTRIM の使用例を次に示します。

```
LTRIM('abcabcabdata base', 'abc')   ==> 'data base'
LTRIM('    data base')              ==> 'data base'
```

(32) LTRIMSTR

(a) 機能

引数 1 の文字列の左から，引数 2 で指定された文字列と一致する部分文字列を，一致しなくなるまで繰り返し取り除きます。

(b) 形式

```
〔MASTER.〕LTRIMSTR（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-71　システム定義スカラ関数 LTRIMSTR の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

5. 引数 1 及び引数 2 のデータ型が両方とも文字列型の場合，値式の文字集合は同じにしてください。

6. 引数 2 の最大長を次の表に示します。

表 2-72　システム定義スカラ関数 LTRIMSTR の引数 2 の最大長

引数 1（及び引数 2）のデータ型		引数 2 の最大長
CHAR 又は VARCHAR	既定文字集合，及び文字集合(UTF16 以外)	255
	文字集合(UTF16)	510
MCHAR 又は MVARCHAR		255
NCHAR 又は NVARCHAR		127

7. 結果のデータ型を次の表に示します。

表 2-73　システム定義スカラ関数 LTRIMSTR の結果のデータ型

引数 1（及び引数 2）のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

8. 結果のデータ型が文字列型の場合，引数 1 の文字集合が結果の文字集合となります。

9. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。

(d)　使用例

LTRIMSTR の使用例を次に示します。

```
LTRIMSTR('abcabcabdata base', 'abc')   ==> 'abdata base'
```

(33) MIDNIGHTSECONDS

(a) 機能

引数で指定された時刻を，深夜 0 時からの秒数で返します。

(b) 形式

```
〔MASTER.〕MIDNIGHTSECONDS（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_TIME 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 時刻演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は TIME にしてください。
5. 結果のデータ型は INTEGER になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 結果は，次の計算式から求められます。
   結果＝（((引数の時×60）＋引数の分）×60）＋引数の秒

(d) 使用例

MIDNIGHTSECONDS の使用例を次に示します。

```
MIDNIGHTSECONDS(TIME('23:59:59'))   ==> 86399
MIDNIGHTSECONDS(TIME('14:14:14'))   ==> 51254
```

(34) MONTHNAME

(a) 機能

引数で指定した日付の，月の名前を示す文字列（'January'，'February'などの英語の文字列）を返します。

(b) 形式

```
〔MASTER.〕MONTHNAME（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は DATE にしてください。
5. 結果のデータ型は VARCHAR(18)になります。

6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 結果の文字集合は既定の文字集合となります。

(d) 使用例

MONTHNAME の使用例を次に示します。

```
MONTHNAME(DATE('1999-01-01'))   ==> 'January'
MONTHNAME(DATE('1999-02-28'))   ==> 'February'
MONTHNAME(DATE('1999-03-03'))   ==> 'March'
MONTHNAME(DATE('1999-04-01'))   ==> 'April'
MONTHNAME(DATE('1999-05-05'))   ==> 'May'
MONTHNAME(DATE('1999-06-07'))   ==> 'June'
MONTHNAME(DATE('1999-07-07'))   ==> 'July'
MONTHNAME(DATE('1999-08-15'))   ==> 'August'
MONTHNAME(DATE('1999-09-23'))   ==> 'September'
MONTHNAME(DATE('1999-10-10'))   ==> 'October'
MONTHNAME(DATE('1999-11-11'))   ==> 'November'
MONTHNAME(DATE('1999-12-31'))   ==> 'December'
```

(35) MONTHS_BETWEEN

(a) 機能

引数で指定した二つの日付間の月数を，実数（FLOAT 型）で返します。

(b) 形式

```
〔MASTER.〕MONTHS_BETWEEN（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し

- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数 1 及び引数 2 のデータ型は DATE にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
7. MONTHS_BETWEEN（日付 1，日付 2）の結果の月数は，次の規則に従い計算されます。
   - 日付 1 ＜日付 2 の場合

     結果＝－MONTHS_BETWEEN（日付 2，日付 1）
   - 日付 1≧日付 2 の場合

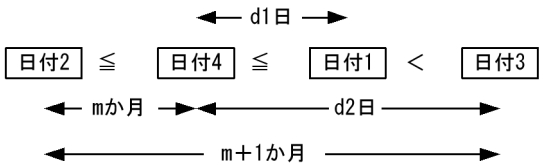
     m を 0 以上の整数とし，日付 4≦日付 1 ＜日付 3 を満たす（日付 2 ＋ m か月）の日付，及び（日付 2 ＋（m ＋ 1）か月）の日付を，それぞれ日付 4 及び日付 3 とします。

     日付 4 と日付 1 の間の日数（DAYS（日付 1）－DAYS（日付 4)），及び日付 4 と日付 3 の間の日数（DAYS（日付 3）－DAYS（日付 4））を，それぞれ d1 及び d2 とします。d2 の値は，28，29，30，又は 31 のどれかになります。

     結果の月数は，（m ＋ d1÷d2）か月になります。

   これらの日付 1，日付 2，日付 3，及び日付 4 の関係を次の図に示します。

図 2-7　日付 1，日付 2，日付 3，及び日付 4 の関係（MONTHS_BETWEEN）



(d)　使用例

MONTHS_BETWEEN の使用例を次に示します。

```
MONTHS_BETWEEN(DATE('1999-07-10'), DATE('1999-06-10'))
  ==> 1
MONTHS_BETWEEN(DATE('1999-07-11'), DATE('1999-06-10'))
  ==> 1.032258…
MONTHS_BETWEEN(DATE('1999-06-11'), DATE('1999-05-10'))
  ==> 1.033333…
MONTHS_BETWEEN(DATE('1999-02-11'), DATE('1999-01-10'))
  ==> 1.035714…
```

```
MONTHS_BETWEEN(DATE('2000-02-11'), DATE('2000-01-10'))
  ==> 1.034482…
MONTHS_BETWEEN(DATE('1999-09-09'), DATE('1999-06-10'))
  ==> 2.967741…
MONTHS_BETWEEN(DATE('1999-06-10'), DATE('1999-09-09'))
  ==> -2.967741…
```

(36) NEXT_DAY

(a) 機能

引数 1 で指定した日付よりも後で最も近い，引数 2 の整数で指定した曜日の日付を返します。

(b) 形式

```
〔MASTER.〕NEXT_DAY（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 の値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算

- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は DATE にしてください。引数 2 のデータ型は INTEGER 又は SMALLINT にしてください。
6. 結果のデータ型は DATE になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
8. 引数 2 で指定した整数値の意味を次の表に示します。

表 2-74　曜日を表す引数 2 の整数値の意味

引数 2 の整数値	意味（曜日）
1	日曜日（Sunday）
2	月曜日（Monday）
3	火曜日（Tuesday）
4	水曜日（Wednesday）
5	木曜日（Thursday）
6	金曜日（Friday）
7	土曜日（Saturday）

9. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(d)　使用例

NEXT_DAY の使用例を次に示します。

```
NEXT_DAY(DATE('1999-06-10'), 1)   ==> '1999-06-13'
NEXT_DAY(DATE('1999-06-10'), 4)   ==> '1999-06-16'
NEXT_DAY(DATE('1999-06-10'), 5)   ==> '1999-06-17'
NEXT_DAY(DATE('1999-06-10'), 6)   ==> '1999-06-11'
```

```
NEXT_DAY(DATE('1999-06-10'), 7)   ==> '1999-06-12'
  なお，'1999-06-10'は木曜日です。
```

(37) NUMEDIT

(a) 機能

引数 1 で指定した数値を，引数 2 で指定した形式の文字列表現に編集します。

(b) 形式

```
〔MASTER.〕NUMEDIT（引数1, 引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 の値式に指定できるものを次に示します。
   - 文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し

- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は数データ型にしてください。
6. 引数 2 のデータ型は文字列型（CHAR 又は VARCHAR）にしてください。また，引数 2 の値の長さは 250 バイト以下にしてください。
7. 引数 2 のデータ型の文字集合は UTF16 以外にしてください。引数 2 に文字集合が UTF16 の文字データ型を指定する場合，CAST 指定を用いて既定文字集合の文字データ型に変換してください。
8. 結果のデータ型は VARCHAR(255)になります。
9. 結果の文字集合は，引数 2 の文字集合となります。
10. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
11. 引数 2 の値は，次の形式で指定してください。

```
〔"文字列"〕〔+〕
     { 〔 {<|>} 〕 {9|0|,|"文字列"} … 〔. {9|0|,|"文字列"} …〕
      | {E|e} .整数 〔 "文字列" 〕 }
```

"文字列"

引用符で囲まれた文字列は，その文字列自身を表現します。

引用符で囲まれた文字列中で引用符そのものを指定する場合は，連続する二つの引用符で表現します。

＋

〔+〕は，符号の表現方法を指定します。

'+'は，引数 1 が負の値ならばマイナス符号(−)，そうでなければプラス符号(+)で表現します。

'+'を省略すると，引数 1 が負の値ならば，マイナス符号(−)，そうでなければ，一つの空白文字で表現します。

{< | >}

{< | >}は，数値の文字表現の詰め込み方を指定します。

'<'を指定すると，すべての空白文字を取り除き，文字表現を左にシフトします。

'>'を指定すると，すべての空白文字を取り除き，文字表現を右にシフトし，左側に空白文字を埋めます。

'<'，'>'のどちらも指定しない場合，文字表現をシフトしません。

{9 | 0 | , | "文字列"} … 〔. {9 | 0 | , | "文字列"}〕 …

'9'，'0'は数値の一つのけたを表現し，引数 1 の数値に対応するけたを編集します。

'9'及び'0'の総個数は，編集後の数値の精度を示します。

'0'を指定すると，その'0'の編集結果は対応するけたの数字になります。

','を指定すると，その','の両側の編集結果が数字ならば，','の編集結果はそれ自身になり，そうでなければ一つの空白文字になります。

'.' は，小数点を表現します。

'.'より後の'9'及び'0'の総個数は，編集後の数値の位取りを示します。

'.'より前に'9'を指定すると，対応するけたが 0 で，その'9'が最初のけたである，又はその'9'の編集結果の左の編集結果が数字でない（ただし，コンマに対応するときは除く）の場合，その'9'の編集結果は一つの空白文字になります。そうでなければ，その'9'の編集結果は対応するけたの数字になります。

'.'より後に'9'を指定すると，対応するけたが 0 で，その'9'が最後のけたである，又はその'9'の編集結果の右の編集結果が数字でない（ただし，コンマに対応するときは除く）の場合，その'9'の編集結果は空（0 個）の文字になります。そうでなければ，その'9'の編集結果は対応するけたの数字になります。

{E | e} .整数

{E | e}.整数は，浮動小数点数形式の書式で指定します。

編集結果は整数で指定された仮数の位取りを持つ，浮動小数点数形式になります。仮数の位取りは，30 未満にしてください。

12. 引数 1 の値が引数 2 で指定された形式で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

13. 編集によって下位のけたが失われる場合，結果の値は丸められます。

(d) 使用例

NUMEDIT の使用例を次に示します。

```
NUMEDIT(1234567.89, '99,999,990.00"$"')   ==> '  1,234,567.89$'
NUMEDIT(1000, '99,999,990.00"$"')         ==> '      1,000.00$'
NUMEDIT(1234567.89, '" $"99,999,990.00')  ==> ' $  1,234,567.89'
NUMEDIT(1000, '" $"99,999,990.00')        ==> ' $      1,000.00'
NUMEDIT(1234567.89, '"$"+99,999,990,00')  ==> '$+ 1,234,567.89'
NUMEDIT(-1000, '"$"+99,999,990.00')       ==> '$-     1,000.00'
NUMEDIT(1234567.89, '"$">99,999,990.00')  ==> '  $1,234,567.89'
NUMEDIT(1000, '"$">99,999,990.00')        ==> '      $1,000.00'
NUMEDIT(1234567.89, '"$"<99,999,990.00')  ==> '$1,234,567.89'
NUMEDIT(1000, '"$"<99,999,990.00')        ==> '$1,000.00'
NUMEDIT(0.5, '"$"<99,999,990.00')         ==> '$0.50'
NUMEDIT(1234567.89, '+E.10"$"')           ==> '+1.2345678900E+0.6$'
NUMEDIT(1234567.89, '"$"+e.10')           ==> '$+1.2345678900e+0.6'
```

(38) NVL

(a) 機能

ナル値をナル値でない値に変換します。

(b) 形式

```
NVL（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 定数
   - USER 値関数
   - CURRENT_DATE 値関数
   - CURRENT_TIME 値関数
   - CURRENT_TIMESTAMP 値関数
   - 列指定
   - SQL 変数，又は SQL パラメタ
   - 四則演算
   - 日付演算
   - 時刻演算
   - 連結演算
   - 集合関数
   - スカラ関数
   - CASE 式
   - CAST 指定
   - 埋込み変数，又は?パラメタ
   - 関数呼出し
   - スカラ副問合せ
3. 引数 1，及び引数 2 の演算結果が次に示すデータ型の場合は指定できません。
   - LOB
   - BOOLEAN

- 抽象データ型

4. 引数 1，及び引数 2 に，NULL は指定できません。
5. 引数 1，及び引数 2 にバイナリデータを指定する場合は，実長が 32,001 バイト以上の値は指定できません。
6. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
7. 引数 1，及び引数 2 のデータ型は，それぞれ比較できるデータ型にしてください。

   比較できるデータ型については，「データ型」を参照してください。

   ただし，次に示すデータは比較できません。

   - 日付データと日付データの文字列表現
   - 時刻データと時刻データの文字列表現
   - 時刻印データと時刻印データの文字列表現
   - 日間隔データと日間隔データの 10 進数表現
   - 時間隔データと時間隔データの 10 進数表現
   - バイナリデータと 16 進文字列定数
8. 結果の値は，引数 1 がナル値でない場合は引数 1 になり，引数 1 がナル値の場合は引数 2 になります。
9. 結果のデータ型，及びデータ長は次のとおりです。

   ＜文字データ，各国文字データ，混在文字データの場合＞

   - 結果のデータ型は，可変長データになります。
   - 結果のデータ長は，一番長いデータ長を持つ引数のデータ長になります。
   - 引数に文字データと混在文字データが含まれる場合の結果のデータ型は，混在文字データになります。

   ＜バイナリデータの場合＞

   - 結果のデータ型は，バイナリデータになります。
   - 結果のデータ長は，一番長いデータ長を持つ引数のデータ長になります。ただし，一番長いデータ長が 32,001 バイト以上の場合は，結果のデータ長は 32,000 バイトになります。

   ＜文字データ，各国文字データ，混在文字データ，バイナリデータ以外の場合＞

   結果のデータ型，及びデータ長は，集合演算（UNION ALL，又は EXCEPT ALL）の結果のデータ型，及びデータ長と同じになります。詳細については，「問合せ式」を参照してください。
10. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 及び引数 2 の両方ともナル値であれば，結果もナル値になります。
11. 引数 1，及び引数 2 が文字データ型の場合，文字集合は同じにしてください。

(d) 使用例

表 T1 の C1 の値を取得する際，C1 がナル値の場合は 0 を取得します。

```
SELECT NVL(C1,0) FROM T1
```

(39) PI

(a) 機能

円周率 π の値を返します。

(b) 形式

```
〔MASTER.〕PI ()
```

(c) 規則

1. 結果のデータ型は，FLOAT 型になります。
2. 結果の値は，非ナル値制約なし（ナル値を許します）になります。ただし，常に円周率の値が返されるため，ナル値が返されることはありません。

(d) 使用例

PI の使用例を次に示します。

```
PI()   ==> 3.14159265358979323846
```

(40) POSSTR

(a) 機能

引数 1 で指定した文字列中の，引数 3 で指定した文字位置以降に，引数 2 で指定した部分文字列が引数 4 で指定した回数（nd）以上出現した場合，その回数 nd 番目の部分文字列の開始位置を文字単位で返します。

(b) 形式

```
〔MASTER.〕POSSTR（引数1，引数2〔，引数3〔，引数4〕〕）
```

(c) 規則

1. 引数 1，引数 2，引数 3，及び引数 4 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 及び引数 2 の値式に指定できるものを次に示します。

- 文字列定数，混在文字列定数，又は各国文字列定数
- 列指定
- SQL 変数又は SQL パラメタ
- 連結演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 引数 3 及び引数 4 の値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 及び引数 2 に指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-75　システム定義スカラ関数 POSSTR の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○

引数 1 のデータ型	引数 2 のデータ型	指定可否
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

6. 引数 1 及び引数 2 のデータ型が両方とも文字列データ型（CHAR 又は VARCHAR）の場合，値式の文字集合は同じにしてください。

7. 引数 2 の値の最大長を次の表に示します。

表 2-76　システム定義スカラ関数 POSSTR の引数 2 の最大長

引数 1（及び引数 2）のデータ型		引数 2 の最大長
CHAR 又は VARCHAR	既定文字集合，及び文字集合(UTF16 以外)	255
	文字集合(UTF16)	510
MCHAR 又は MVARCHAR		255
NCHAR 又は NVARCHAR		127

8. 引数 3 及び引数 4 のデータ型は，INTEGER 又は SMALLINT にしてください。

9. 結果のデータ型は INTEGER になります。

10. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式のどれかがナル値であれば，結果もナル値になります。

11. 引数 3 及び引数 4 に指定できる値の範囲を次の表に示します。

表 2-77　システム定義スカラ関数 POSSTR の引数 3 及び引数 4 の値の範囲

引数 1（及び引数 2）のデータ型		引数 3 の値の範囲	引数 4 の値の範囲
CHAR(n) 又は VARCHAR(n)	既定文字集合，及び文字集合(UTF16 以外)	1～n	1～n
	文字集合(UTF16)	1～n／2	1～n／2
MCHAR(n)又は MVARCHAR(n)		1～n	1～n
NCHAR(n)又は NVARCHAR(n)		1～n	1～n

12. 引数 3 を省略すると，1 が仮定されます。また，引数 4 を省略しても 1 が仮定されます。

13. 引数 2 の文字列の長さ（文字数）が 0 の場合，結果は引数 3 の値になります。

14. 引数 2 の文字列と一致する部分文字列が，引数 1 の文字列中の引数 3 の文字位置（m）以降に，引数 4 で指定した回数（nd）以上出現する場合，結果は m 以上の値で，nd 番目の部分文字列が開始する前の引数 1 の文字列内の文字数に 1 を加えた値になります。

なお，引数 2 の文字列と一致するこれらの部分文字列は，重ならないものとします。引数 2 の文字列と一致する部分文字列が，引数 1 の文字列中の引数 3 の文字位置（m）以降に，引数 4 で指定した回数（nd）以上出現しない場合，結果は 0 になります。

(d) 使用例

POSSTR の使用例を次に示します。

```
POSSTR('data base system', 'a')        ==> 2
POSSTR('data base system', '')         ==> 1（引数2は長さ0の文字列）
POSSTR('data base system', 'a', 5)     ==> 7
POSSTR('data base system', 'st')       ==> 13
POSSTR('data base system', 'a', 1, 3)  ==> 7
POSSTR('data base system', 'manager')  ==> 0
```

(41) POWER

(a) 機能

引数 1 の値の n 乗を返します。なお，n は引数 2 の値とします。

(b) 形式

```
〔MASTER.〕POWER（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型を次の表に示します。

表 2-78　システム定義スカラ関数 POWER の結果のデータ型

引数 1 のデータ型	引数 2 のデータ型	
	SMALLINT 又は INTEGER	DECIMAL(p,s)，SMALLFLT，又は FLOAT
SMALLINT 又は INTEGER	INTEGER	FLOAT
DECIMAL(p,s)，SMALLFLT，又は FLOAT	FLOAT	FLOAT

6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
7. 引数 1 に負の値を指定し，引数 2 に整数でない値を指定すると，定義域エラー（domain error occurs）になります。
8. 引数 1 に値 0 を指定し，引数 2 に正でない値を指定すると，0 除算（division by zero）になります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。
9. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(42)　QUARTER

(a)　機能

引数 2 で指定した月，及び引数 3 で指定した日を各年度の開始の月日として，引数 1 で指定した日付が第何四半期かを示す 1～4 の整数で返します。

(b)　形式

```
〔MASTER.〕QUARTER（引数1〔，引数2〔，引数3〕〕）
```

(c)　規則

1. 引数 1，引数 2，及び引数 3 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。

- CURRENT_DATE 値関数
- 列指定
- SQL 変数又は SQL パラメタ
- 日付演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 引数 2 及び引数 3 の値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は DATE にしてください。
6. 引数 2 及び引数 3 のデータ型は，INTEGER 又は SMALLINT にしてください。
7. 結果のデータ型は INTEGER になります。
8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式のどれかがナル値であれば，結果もナル値になります。
9. 引数 2 を省略すると，1 が仮定されます。引数 2 に指定できる値の範囲は，1～12 です。

10. 引数３を省略すると，１が仮定されます。引数３に指定できる値の範囲は，引数２の値が２の場合は，１～29 です。それ以外の場合は，１～（引数２で指定した月の最終日）です。

11. 結果は，次の規則によって求められます。

- 引数１の日＜引数３の場合
  月数＝引数１の月－引数２－1
- 引数１の月≧引数３の場合
  月数＝引数１の月－引数２
- 月数＜０の場合
  結果＝（月数＋ 12）÷4 ＋ 1
- 月数≧0 の場合
  結果＝月数÷4 ＋ 1

(d) 使用例

QUARTER の使用例を次に示します。

```
QUARTER(DATE('1999-01-01'))          ==> 1
QUARTER(DATE('1999-09-10'))          ==> 3
QUARTER(DATE('1999-12-31'))          ==> 4
QUARTER(DATE('1999-04-01'), 4)       ==> 1
QUARTER(DATE('1999-09-10'), 4)       ==> 2
QUARTER(DATE('1999-03-31'), 4)       ==> 4
QUARTER(DATE('1999-03-21'), 3, 21)   ==> 1
QUARTER(DATE('1999-09-20'), 3, 21)   ==> 2
QUARTER(DATE('1999-03-20'), 3, 21)   ==> 4
```

(43) RADIANS

(a) 機能

引数で指定した角度を，度数からラジアンに変換します。

(b) 形式

```
〔MASTER.〕RADIANS（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定

- SQL 変数又は SQL パラメタ
- 四則演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(44) REPLACE（REPLACE_LONG）

(a) 機能

引数 1 の文字列中の，引数 2 の文字列と一致するすべての部分文字列を，引数 3 の文字列に置き換えます。

(b) 形式

```
〔MASTER.〕REPLACE（引数1，引数2〔，引数3〕）
〔MASTER.〕REPLACE_LONG（引数1，引数2，引数3）
```

(c) 規則

1. 引数 1，引数 2，及び引数 3 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算

- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-79　システム定義スカラ関数 REPLACE（引数 3 を省略する場合）の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

表 2-80　システム定義スカラ関数 REPLACE（引数 3 を指定する場合），REPLACE_LONG の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ			×

(凡例)

○：指定できます。

×：指定できません。

5. 引数 2 及び引数 3 の値の最大長を次の表に示します。

表 2-81　システム定義スカラ関数 REPLACE，REPLACE_LONG の引数 2，引数 3 の最大長

引数 1（及び引数 2，引数 3）のデータ型		引数 2 及び引数 3 の最大長
CHAR 又は VARCHAR	既定文字集合，及び文字集合(UTF16 以外)	255
	文字集合(UTF16)	510
MCHAR 又は MVARCHAR		255
NCHAR 又は NVARCHAR		127

6. 引数 1，引数 2，及び引数 3 のデータ型が文字列型（CHAR 又は VARCHAR）の場合，引数に指定する値式の文字集合は同じにしてください。

7. 引数 3 を省略すると，空の文字列を仮定し，引数 1 の文字列中から引数 2 の文字列と一致するすべての部分文字列を削除します。

8. 結果のデータ型を次に示します。

表 2-82　システム定義スカラ関数 REPLACE の結果のデータ型

引数 1 のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

表 2-83　システム定義スカラ関数 REPLACE_LONG の結果のデータ型

引数 1 のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(32000)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(32000)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(16000)

9. 結果のデータ型が文字列型の場合，引数 1 の文字集合が結果の文字集合となります。

10. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式のどれかがナル値であれば，結果もナル値になります。

11. 結果の長さは，結果のデータ型の最大長を超えないようにしてください。結果の長さが引数 1 の長さより長くなる場合は，REPLACE_LONG を使用してください。

(d)　使用例

REPLACE の使用例を次に示します。

```
REPLACE('a big dog and a small dog', 'dog', 'cat')
  ==> 'a big cat and a small cat'
```

```
REPLACE('a big dog and a small dog', 'big ')
  ==> 'a dog and a small dog'
```

(45) REVERSESTR

(a) 機能

引数で指定した文字列中の，左右を反転した文字列を返します。

(b) 形式

```
〔MASTER.〕REVERSESTR（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は，次のどれかを指定してください。
   - 任意の文字集合の文字列型（CHAR，VARCHAR）
   - 各国文字列型（NCHAR，NVARCHAR）
   - 混在文字列型（MCHAR，MVARCHAR）
5. 結果のデータ型を次の表に示します。

表 2-84　システム定義スカラ関数 REVERSESTR の結果のデータ型

引数のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

6. 結果のデータ型が文字列型の場合，引数の文字集合が結果の文字集合となります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d)　使用例

REVERSESTR の使用例を次に示します。

```
REVERSESTR('data base')       ==> 'esab atad'
REVERSESTR('esab atad')       ==> 'data base'
REVERSESTR('   esab atad')    ==> 'data base   '
```

(46)　RIGHTSTR

(a)　機能

引数 1 の文字列の最後（最も右）から，引数 2 で指定した文字数の部分文字列を抽出します。

(b)　形式

```
〔MASTER.〕RIGHTSTR（引数1，引数2）
```

(c)　規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し

- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 引数 2 の値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は，次のどれかを指定してください。
   - 任意の文字集合の文字列型（CHAR，VARCHAR）
   - 各国文字列型（NCHAR，NVARCHAR）
   - 混在文字列型（MCHAR，MVARCHAR）
6. 引数 2 のデータ型は，INTEGER 又は SMALLINT にしてください。
7. 引数 2 に指定できる値の範囲を次の表に示します。

表 2-85　システム定義スカラ関数 RIGHTSTR の引数 2 の値の範囲

引数 1 のデータ型		引数 2 の値の範囲
CHAR(n) 又は VARCHAR(n)	既定文字集合，及び文字集合(UTF16 以外)	0～n
	文字集合(UTF16)	0～n／2
MCHAR(n)又は MVARCHAR(n)		0～n
NCHAR(n)又は NVARCHAR(n)		0～n

8. 結果のデータ型を次の表に示します。

表 2-86　システム定義スカラ関数 RIGHTSTR の結果のデータ型

引数 1 のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

9. 結果のデータ型が文字列型の場合，引数 1 の文字集合が結果の文字集合となります。
10. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
11. 引数 1 の文字列の値の文字数が，引数 2 で指定した文字数よりも少ない場合，結果は引数 1 と同じ値になります。

(d)　使用例

RIGHTSTR の使用例を次に示します。

```
RIGHTSTR('data base system', 6)    ==> 'system'
RIGHTSTR('data system', 0)         ==> ''     (長さ0の文字列)
```

(47)　ROUND

(a)　機能

引数 1 の値の，小数点以下 n けたより後のけた（10 の-n 乗未満のけた）を丸めます。なお，n は引数 2 の値とします。

引数 3 を指定すると，引数 1 の 10 の-n-1 乗の位の値が，引数 3 の値以上ならば切り上げ，そうでなければ切り捨てます。

(b)　形式

```
〔MASTER.〕ROUND（引数1〔，引数2〔，引数3〕〕）
```

(c)　規則

1. 引数 1，引数 2，及び引数 3 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定

- SQL 変数又は SQL パラメタ
- 四則演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数 1 のデータ型は数データ型にしてください。引数 2 及び引数 3 のデータ型は，INTEGER 又は SMALLINT にしてください。
5. 引数 2 を省略すると 0 が仮定されます。引数 2 に指定できる値の範囲を次の表に示します。

表 2-87　システム定義スカラ関数 ROUND の引数 2 に指定できる値の範囲

結果のデータ型	引数 2 に指定できる値の範囲
INTEGER	−9～0
DECIMAL(p, s)	−(p−s−1)～s
FLOAT	−307～323

6. 結果のデータ型を次の表に示します。

表 2-88　システム定義スカラ関数 ROUND の結果のデータ型

引数 1 のデータ型	結果のデータ型
SMALLINT	INTEGER
INTEGER	INTEGER
DECIMAL(p, s)	DECIMAL(p, s)
SMALLFLT	FLOAT
FLOAT	FLOAT

7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1，引数 2，又は引数 3 の値式がナル値であれば，結果もナル値になります。
8. 引数 3 を省略すると，5 が仮定されます。引数 3 に指定できる値の範囲は，1～9 です。引数 1 の 10 の−n−1 乗の位の値が引数 3 の値以上ならば，切り上げ，そうでなければ切り捨てます。

9. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(48) ROUNDMONTH

(a) 機能

引数 1 で指定された日付を，その日付の日の部分を端数として丸めた日付を返します。

20 日締めで年月を求める場合などに利用できます。

(b) 形式

```
〔MASTER.〕ROUNDMONTH（引数1〔，引数2〕）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 の値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数

- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は DATE にしてください。引数 2 のデータ型は，INTEGER 又は SMALLINT にしてください。
6. 結果のデータ型は DATE になります。
7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
8. 引数 2 を省略すると，16 が仮定されます。引数 2 に指定できる値の範囲は，1～32 です。
9. 結果は，次の規則によって求められます。
   - 引数 1 の日＜引数 2 の場合
     結果の年＝引数 1 の年
     結果の月＝引数 1 の月
     結果の日＝ 1
   - 引数 1 の日≧引数 2 の場合
     ・引数 1 の月＜ 12 の場合
     結果の年＝引数 1 の年
     結果の月＝引数 1 の月＋ 1
     結果の日＝ 1
     ・引数 1 の月＝ 12 の場合
     結果の年＝引数 1 の年＋ 1
     結果の月＝ 1
     結果の日＝ 1
10. 関数の結果が，0001 年 01 月 01 日～9999 年 12 月 31 日の範囲でないならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(d) 使用例

ROUNDMONTH の使用例を次に示します。

```
ROUNDMONTH(DATE('1999-08-15'))       ==> '1999-08-01'
ROUNDMONTH(DATE('1999-08-16'))       ==> '1999-09-01'
ROUNDMONTH(DATE('1999-09-20'), 21)   ==> '1999-09-01'
ROUNDMONTH(DATE('1999-09-21'), 21)   ==> '1999-10-01'
ROUNDMONTH(DATE('1999-12-21'), 21)   ==> '2000-01-01'
ROUNDMONTH(DATE('1999-03-31'), 32)   ==> '1999-03-01'
ROUNDMONTH(DATE('2000-02-29'), 29)   ==> '2000-03-01'
ROUNDMONTH(DATE('2000-03-31'), 29)   ==> '2000-04-01'
ROUNDMONTH(DATE('9999-12-31'), 26)   ==> オーバフロー
```

(49) RTRIM

(a) 機能

引数 1 の文字列の右から，引数 2 で指定された文字列を構成する文字を取り除きます。引数 2 で指定された文字列中の文字以外が現れるまで，繰り返し取り除きます。

(b) 形式

```
〔MASTER.〕RTRIM（引数1〔，引数2〕）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。

4. 引数 1 のデータ型は，次のどれかを指定してください。
   - 任意の文字集合の文字列型（CHAR，VARCHAR）
   - 各国文字列型（NCHAR，NVARCHAR）
   - 混在文字列型（MCHAR，MVARCHAR）
5. 引数 2 を指定する場合，指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-89　システム定義スカラ関数 RTRIM の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

6. 引数 1 及び引数 2 のデータ型が両方とも文字列型（CHAR 又は VARCHAR）の場合，引数に指定する値式の文字集合は同じにしてください。
7. 引数 2 の値の長さは，文字列型又は混在文字列型の場合は 30 バイト以下（文字集合が UTF16 の文字データ型の場合は 60 バイト以下），各国文字列型の場合は 30 文字以下にしてください。
8. 結果のデータ型を次の表に示します。

表 2-90　システム定義スカラ関数 RTRIM の結果のデータ型

引数 1（及び引数 2）のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

9. 結果のデータ型が文字列型の場合，引数 1 の文字集合が結果の文字集合となります。
10. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
11. 引数 2 を省略した場合は，次に示す値を仮定します。

表 2-91　引数 2 を省略した場合の仮定値

引数 1 のデータ型	引数 2 を省略した場合の仮定値
CHAR(n)又は VARCHAR(n)	半角空白 1 文字からなる文字列
MCHAR(n)又は MVARCHAR(n)	半角空白と全角空白からなる文字列

引数 1 のデータ型	引数 2 を省略した場合の仮定値
NCHAR(n)又は NVARCHAR(n)	全角空白 1 文字からなる文字列

(d) 使用例

RTRIM の使用例を次に示します。

```
RTRIM('data basebcabcabc', 'abc')   ==> 'data base'
RTRIM('data base    ')              ==> 'data base'
```

(50) RTRIMSTR

(a) 機能

引数 1 の文字列の右から，引数 2 で指定された文字列と一致する部分文字列を，一致しなくなるまで繰り返し取り除きます。

(b) 形式

```
〔MASTER.〕RTRIMSTR（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。

4. 指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-92　システム定義スカラ関数 RTRIMSTR の引数のデータ型の組み合わせ

引数 1 のデータ型	引数 2 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ		×

(凡例)

○：指定できます。

×：指定できません。

5. 引数 1 及び引数 2 のデータ型が両方とも文字列型（CHAR 又は VARCHAR）の場合，引数に指定する値式の文字集合は同じにしてください。

6. 引数 2 の最大長を次の表に示します。

表 2-93　システム定義スカラ関数 RTRIMSTR の引数 2 の最大長

引数 1（及び引数 2）のデータ型		引数 2 の最大長
CHAR 又は VARCHAR	既定文字集合，及び文字集合(UTF16 以外)	255
	文字集合(UTF16)	510
MCHAR 又は MVARCHAR		255
NCHAR 又は NVARCHAR		127

7. 結果のデータ型を次の表に示します。

表 2-94　システム定義スカラ関数 RTRIMSTR の結果のデータ型

引数 1（及び引数 2）のデータ型	引数 2 の最大長
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

8. 結果のデータ型が文字列型の場合，引数 1 の文字集合が結果の文字集合となります。

9. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。

(d) 使用例

RTRIMSTR の使用例を次に示します。

```
RTRIMSTR('data basebcabcabc', 'abc')   ==> 'data basebc'
```

(51) SIGN

(a) 機能

引数の符号（引数が正の値ならば＋ 1，負の値ならば－1，0 ならば 0）を返します。

(b) 形式

```
〔MASTER.〕SIGN（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型を次の表に示します。

表 2-95　システム定義スカラ関数 SIGN の結果のデータ型

引数 1 のデータ型	結果のデータ型
SMALLINT	INTEGER
INTEGER	INTEGER
DECIMAL(p,s)	DECIMAL(1,0)
SMALLFLT	FLOAT
FLOAT	FLOAT

6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(52)　SIN

(a)　機能

ラジアンで角度を指定した引数の，正弦（三角関数 SIN）を返します。

(b)　形式

```
〔MASTER.〕SIN（引数）
```

(c)　規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(53) SINH

(a) 機能

引数の双曲線正弦を返します。

(b) 形式

```
〔MASTER.〕SINH（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。

5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(54) SQRT

(a) 機能

引数の値の平方根を返します。

(b) 形式

```
〔MASTER.〕SQRT（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。

6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 引数に負の値を指定すると，定義域エラー（domain error occurs）になります。

(55) STRTONUM

(a) 機能

引数 1 で指定した数値の文字列表現を，数データ型に変換します。引数 2 のデータ型が INTEGER 又は SMALLINT の場合は INTEGER に，DECIMAL の場合は同じ精度・位取りの DECIMAL に，FLOAT 又は SMALLFLT の場合は FLOAT に変換します。

(b) 形式

```
〔MASTER.〕STRTONUM（引数1，引数2）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - 文字列定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 の値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算

- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型は，文字列型（CHAR 又は VARCHAR）にしてください。また，引数 1 の文字列の長さは，255 バイト以下にしてください。
6. 引数 1 のデータ型の文字集合は UTF16 以外にしてください。引数 1 に文字集合が UTF16 の文字データ型を指定する場合，CAST 指定を用いて既定文字集合の文字データ型に変換してください。
7. 引数 2 のデータ型は，数データ型にしてください。
8. 結果のデータ型を次の表に示します。

表 2-96　システム定義スカラ関数 STRTONUM の結果のデータ型

引数 2 のデータ型	結果のデータ型
SMALLINT	INTEGER
INTEGER	INTEGER
DECIMAL(p,s)	DECIMAL(p,s)
SMALLFLT	FLOAT
FLOAT	FLOAT

9. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 の値式がナル値であれば，結果もナル値になります。引数 2 の値は，結果に影響しません。
10. 引数 1 の文字列の値は，次の形式で指定してください。

- 引数 2 のデータ型が INTEGER の場合
  〔空白文字…〕〔＋｜-〕〔空白文字…〕数字…〔空白文字…〕
- 引数 2 のデータ型が DECIMAL の場合
  〔空白文字…〕〔＋｜-〕〔空白文字…〕
  {数字…〔.〔数字…〕〕｜.数字…}〔空白文字…〕
- 引数 2 のデータ型が FLOAT の場合
  〔空白文字…〕〔＋｜-〕〔空白文字…〕

{数字…〔.〔数字…〕〕｜.数字…}
〔{E｜e}〔+｜-〕数字…〕〔空白文字…〕

11. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

12. 引数 2 のデータ型が DECIMAL 型の場合，データ型の変換で下位のけたがなくなるときは，結果の値は丸められます。

(d) 使用例

STRTONUM の使用例を次に示します。

```
・引数2がINTEGERの場合
 STRTONUM(’ -  1234567  ’, 0)
   ==> -1234567 (INTEGER型)
 STRTONUM(’  + 1234567890123  ’, 0)
   ==> オーバフロー
 STRTONUM(’  1234567.89  ’, 9)
   ==> INTEGERへの変換の，STRTONUM関数の引数1の値が正しくない

・引数2がDECIMALの場合
 STRTONUM(’-1234567  ’, 123456789012. )
   ==> -1234567 (DECIMAL(12,0)型)
 STRTONUM(’  1234567.89  ’, 9999999999.999)
   ==> 1234567.89 (DECIMAL(13,3)型)
 STRTONUM(’  1234567.89  ’, 99999.999)
   ==> オーバフロー
 STRTONUM(’  1.23456789E6 ’, 9999999999.999)
   ==> DECIMALへの変換の，STRTONUM関数の引数1の値が正しくない

・引数2がFLOATの場合
 STRTONUM(’  1234567.89  ’, 1e0)
   ==> 1.23456789E6 (FLOAT型)
 STRTONUM(’  -1234567890123  ’, 0e0)
   ==> -1.234567890123E12 (FLOAT型)
 STRTONUM(’  1.23456789E6 ’, 1E1)
   ==> 1.23456789E6 (FLOAT型)
 STRTONUM(’  1.0E310  ’, 9E9)
   ==> オーバフロー
```

(56) TAN

(a) 機能

ラジアンで角度を指定した引数の，正接（三角関数 TAN）を返します。

(b) 形式

```
〔MASTER.〕TAN（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(57) TANH

(a) 機能

引数の双曲線正接を返します。

(b) 形式

```
〔MASTER.〕TANH（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は数データ型にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。
7. 関数の結果が，結果のデータ型で表現できない値ならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(58) TRANSL（TRANSL_LONG）

(a) 機能

引数 2 で指定した文字列中のどれかの文字が，引数 1 で指定した文字列中に含まれる場合，それらの文字を引数 3 で指定した文字列中の対応する文字に変換した文字列を返します。

引数 2 の文字列中の i 番目の文字が，引数 3 の i 番目の文字に対応します。引数 3 の文字列の文字数が引数 2 の文字列の文字数より少ない場合，引数 3 の文字列として，引数 3 の文字列に引数 4 の文字を繰り返し埋めた文字列が仮定されます。

(b) 形式

```
〔MASTER.〕TRANSL（引数1，引数2，引数3〔，引数4〕）
〔MASTER.〕TRANSL_LONG（引数1，引数2，引数3〔，引数4〕）
```

(c) 規則

1. 引数 1，引数 2，引数 3，及び引数 4 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - 文字列定数，混在文字列定数，又は各国文字列定数（TRANSL_LONG の場合は混在文字列定数だけ指定できます）
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 連結演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数 3 の文字列の文字数が，引数 2 の文字列の文字数よりも少ない場合は，引数 3 の文字列として，引数 3 の文字列に引数 4 の文字を繰り返し埋めた文字列が仮定されます。
5. 引数 4 には，1 文字だけ指定してください。引数 4 を省略すると引数 1 の文字集合の空白文字（引数 1 のデータ型が NCHAR 又は NVARCHAR の場合は全角空白）が仮定されます。
6. TRANSL に指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-97　システム定義スカラ関数 TRANSL の引数のデータ型の組み合わせ（引数 4 を省略する場合）

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	指定可否
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ			×

(凡例)

○：指定できます。

×：指定できません。

表 2-98　システム定義スカラ関数 TRANSL の引数のデータ型の組み合わせ（引数 4 を指定する場合）

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	引数 4 のデータ型	指定可否
CHAR，又は VARCHAR	CHAR，又は VARCHAR	CHAR，又は VARCHAR	CHAR，又は VARCHAR	○
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	NCHAR，又は NVARCHAR	○
上記以外のデータ型の組み合わせ				×

(凡例)

○：指定できます。

×：指定できません。

7. TRANSL_LONG に指定できる引数のデータ型の組み合わせを次の表に示します。

表 2-99　システム定義スカラ関数 TRANSL_LONG の引数のデータ型の組み合わせ（引数 4 を省略する場合）

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	指定可否
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
上記以外のデータ型の組み合わせ			×

(凡例)

○：指定できます。

×：指定できません。

表 2-100　システム定義スカラ関数 TRANSL_LONG の引数のデータ型の組み合わせ（引数 4 を指定する場合）

引数 1 のデータ型	引数 2 のデータ型	引数 3 のデータ型	引数 4 のデータ型	指定可否
MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	MCHAR，又は MVARCHAR	○
上記以外のデータ型の組み合わせ				×

(凡例)

○：指定できます。

×：指定できません。

8. TRANSL の引数 1，引数 2，引数 3，及び引数 4 のすべてのデータ型が文字列型（CHAR 又は VARCHAR）の場合，引数に指定する値式の文字集合は同じにしてください。

9. 引数 2 及び引数 3 の値の最大長を次の表に示します。

表 2-101　システム定義スカラ関数 TRANSL，TRANSL_LONG の引数 2，引数 3 の最大長

引数 1 のデータ型		引数 2 及び引数 3 の最大長
CHAR 又は VARCHAR※	既定文字集合，及び文字集合(UTF16 以外)	255
	文字集合(UTF16)	510
MCHAR 又は MVARCHAR		255
NCHAR 又は NVARCHAR※		127

注※　TRANSL_LONG の場合は指定できません。

10. 結果のデータ型を次に示します。

表 2-102　システム定義スカラ関数 TRANSL の結果のデータ型

引数 1 のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	VARCHAR(n)
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(n)
NCHAR(n)又は NVARCHAR(n)	NVARCHAR(n)

表 2-103　システム定義スカラ関数 TRANSL_LONG の結果のデータ型

引数 1 のデータ型	結果のデータ型
CHAR(n)又は VARCHAR(n)	指定できません
MCHAR(n)又は MVARCHAR(n)	MVARCHAR(32000)
NCHAR(n)又は NVARCHAR(n)	指定できません

11. 結果のデータ型が文字列型の場合，引数 1 の文字集合が結果の文字集合となります。

12. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式のどれかがナル値であれば，結果もナル値になります。

(d) 使用例

TRANSL，TRANSL_LONG の使用例を次に示します。

```
TRANSL('data base system', 'abcdefghijklmnopqrstuvwxyz',
       'ABCDEFGHIJKLMNOPQRSTUVWXYZ')
  ==> 'DATA BASE SYSTEM' （英小文字から英大文字への文字変換）
TRANSL('<data base> system', '[ ] { } ( ) < >', '')
  ==> '  data base  system'
      (引数3は長さ0の文字列で，括弧（'[', ']', '{', '}', '(', ')',
       '<', '>'）から空白への文字変換）
TRANSL('+12345.678', '+-012345678', 'SS', '9')
  ==> 'S99999.999'
TRANSL (M'ｄａｔａ　ｂａｓｅ　ｓｙｓｔｅｍ',
        M'０１２３４５６７８９ａｂｃｄｅｆｇｈｉｊｋｌｍｎ
          ｏｐｑｒｓｔｕｖｗｘｙｚ',
        M'0123456789abcdefghijklmnopqrstuvwxyz')
  ==> M'data base system' （全角文字の半角文字への文字変換）

TRANSL_LONG(M'data base system',
            M'0123456789abcdefghijklmnopqrstuvwxyz',
            M'０１２３４５６７８９ａｂｃｄｅｆｇｈｉｊｋｌｍｎ
              ｏｐｑｒｓｔｕｖｗｘｙｚ')
  ==> M'ｄａｔａ　ｂａｓｅ　ｓｙｓｔｅｍ'
      (半角文字の全角文字への文字変換）
TRANSL('+12345.678', '+012345678', '', 'S')
  ==> 'SSSSSS.SSS'
TRANSL('2000-03-31 12:23:30', '-:', '/.')
  ==> '2000/03/31 12.23.30'
```

(59) TRUNC

(a) 機能

引数 1 の値の，小数点数以下の n けたより後のけた（10 の－n 乗未満のけた）を切り捨てた値を返します。なお，n は引数 2 で指定した値とします。

(b) 形式

```
〔MASTER.〕TRUNC（引数1〔，引数2〕）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。

- 数定数
- 列指定
- SQL 変数又は SQL パラメタ
- 四則演算
- 集合関数
- スカラ関数
- 関数呼出し
- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数 1 のデータ型は数データ型にしてください。引数 2 のデータ型は，INTEGER 又は SMALLINT にしてください。
5. 引数 2 を省略すると，0 が仮定されます。引数 2 に指定できる値の範囲を次の表に示します。

表 2-104　システム定義スカラ関数 TRUNC の引数 2 に指定できる値の範囲

結果のデータ型	引数 2 に指定できる値の範囲
INTEGER	−9 ～ 0
DECIMAL(p, s)	−(p-s-1) ～ s
FLOAT	−307～323

6. 結果のデータ型を次の表に示します。

表 2-105　システム定義スカラ関数 TRUNC の結果のデータ型

引数 1 のデータ型	結果のデータ型
SMALLINT	INTEGER
INTEGER	INTEGER
DECIMAL(p,s)	DECIMAL(p,s)
SMALLFLT	FLOAT
FLOAT	FLOAT

7. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。

(60) TRUNCYEAR

(a) 機能

引数 2 で指定した月及び引数 3 で指定した日を，各年度の開始の月日として，引数 1 で指定された日付を含む年度の最初の日付を返します。

3 月 20 日締めで年度を求める場合，3 月締めで年度を求める場合などに利用できます。

(b) 形式

```
〔MASTER.〕TRUNCYEAR（引数1〔，引数2〔，引数3〕〕）
```

(c) 規則

1. 引数 1，引数 2，及び引数 3 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 引数 1 の値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 引数 2 及び引数 3 の値式に指定できるものを次に示します。
   - 整数定数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 四則演算
   - 集合関数
   - スカラ関数
   - 関数呼出し

- CASE 式
- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

4. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
5. 引数 1 のデータ型を，DATE 型にしてください。
6. 引数 2 及び引数 3 のデータ型は，INTEGER 又は SMALLINT にしてください。
7. 結果のデータ型は DATE になります。
8. 結果の値は，非ナル値制約なし（ナル値を許します）になります。どれかの引数の値式がナル値であれば，結果もナル値になります。
9. 引数 2 を省略すると，1 が仮定されます。引数 2 に指定できる値の範囲は，1～12 です。
10. 引数 3 を省略すると，1 が仮定されます。引数 3 に指定できる値の範囲は，引数 2 の値が 2 の場合は 1～29 です。それ以外の場合は，1～（引数 2 で指定した月の最終日）です。
11. 結果は，次の規則によって求められます。
    - 引数 1 の月＜引数 2，又は引数 1 の月＝引数 2 かつ引数 1 の日＜引数 3 の場合
      結果の年＝引数 1 の年 － 1
    - 引数 1 の月＞引数 2，又は引数 1 の月＝引数 2 かつ引数 1 の日≧引数 3 の場合
      結果の年＝引数 1 の年
    - 結果の年がうるう年でない場合で，引数 2 ＝ 2 かつ引数 3 ＝ 29 のとき
      結果の月＝ 3
      結果の日＝ 1
    - 結果の年がうるう年の場合で，引数 2 が 2 でないとき，又は引数 3 が 29 でないとき
      結果の月＝引数 2
      結果の日＝引数 3
12. 関数の結果が，0001 年 01 月 01 日～9999 年 12 月 31 日の範囲でないならば，オーバフローエラーになります。なお，オーバフローエラー抑止が設定されている場合の結果については，「オーバフローエラー抑止が設定されている場合の演算結果」を参照してください。

(d) 使用例

TRUNCYEAR の使用例を次に示します。

```
TRUNCYEAR(DATE('1999-09-10'))          ==> '1999-01-01'
TRUNCYEAR(DATE('1999-09-11'), 4)       ==> '1999-04-01'
TRUNCYEAR(DATE('1999-03-31'), 4)       ==> '1998-04-01'
TRUNCYEAR(DATE('1999-08-11'), 3, 21)   ==> '1999-03-21'
```

```
TRUNCYEAR(DATE('1999-03-20'), 3, 21)  ==> '1998-03-21'
TRUNCYEAR(DATE('2000-02-28'), 2, 29)  ==> '1999-03-01'
                                      ('1999-02-29'は存在しません)
TRUNCYEAR(DATE('2000-03-20'), 2, 29)  ==> '2000-02-29'
TRUNCYEAR(DATE('0001-03-20'), 4)      ==> オーバフロー
```

(61) WEEK

(a) 機能

引数で指定された日付が，その年の第何週目かを 1～54 の整数で返します。各週は，日曜日から始まるものとします。

(b) 形式

```
〔MASTER.〕WEEK（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ
3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は DATE にしてください。
5. 結果のデータ型は INTEGER になります。

6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d) 使用例

WEEK の使用例を次に示します。

```
WEEK(DATE('2000-01-01'))   ==> 1
WEEK(DATE('2000-01-02'))   ==> 2
WEEK(DATE('2000-02-29'))   ==> 10
WEEK(DATE('2000-12-30'))   ==> 53
WEEK(DATE('2000-12-31'))   ==> 54
  '2000-01-01'は土曜日で，'2000-12-31'は日曜日です。
```

(62) WEEKOFMONTH

(a) 機能

引数で指定された日付が，その月の第何週目かを 1～6 の整数で返します。各週は，日曜日から始まるものとします。

(b) 形式

```
〔MASTER.〕WEEKOFMONTH（引数）
```

(c) 規則

1. 引数に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - 埋込み変数又は？パラメタ
   - スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数のデータ型は DATE にしてください。
5. 結果のデータ型は INTEGER になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数の値式がナル値であれば，結果もナル値になります。

(d) 使用例

WEEKOFMONTH の使用例を次に示します。

```
WEEKOFMONTH(DATE('2000-01-01'))   ==> 1
WEEKOFMONTH(DATE('2000-01-02'))   ==> 2
WEEKOFMONTH(DATE('2000-01-29'))   ==> 5
WEEKOFMONTH(DATE('2000-01-30'))   ==> 6
  '2000-01-01'は土曜日で，'2000-01-30'は日曜日です。
```

(63) YEARS_BETWEEN

(a) 機能

引数で指定した二つの日付間の年数を，実数（FLOAT 型）で返します。

(b) 形式

```
〔MASTER.〕YEARS_BETWEEN（引数1，引数2）
```

(c) 規則

1. 引数 1 及び引数 2 に指定できるものを次に示します。
   - 値式〔AS データ型〕
2. 値式に指定できるものを次に示します。
   - CURRENT_DATE 値関数
   - 列指定
   - SQL 変数又は SQL パラメタ
   - 日付演算
   - 集合関数
   - スカラ関数
   - 関数呼出し
   - CASE 式

- CAST 指定
- 埋込み変数又は？パラメタ
- スカラ副問合せ

3. 値式に埋込み変数又は？パラメタだけを指定する場合は，必ず AS データ型を指定してください。AS データ型を指定した場合，埋込み変数又は？パラメタ以外は指定できません。AS データ型を指定すると，AS 句で指定したデータ型のパラメタを持つ関数が呼び出されます。
4. 引数 1 及び引数 2 のデータ型は，DATE にしてください。
5. 結果のデータ型は FLOAT になります。
6. 結果の値は，非ナル値制約なし（ナル値を許します）になります。引数 1 又は引数 2 の値式がナル値であれば，結果もナル値になります。
7. YEARS_BETWEEN（日付 1，日付 2）の結果の月数は，次の規則に従って計算されます。
   - 日付 1 ＜日付 2 の場合

     結果＝－YEARS_BETWEEN（日付 2，日付 1）
   - 日付 1≧日付 2 の場合

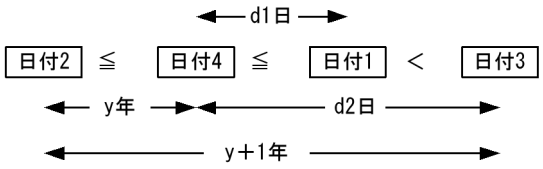
     y を 0 以上の整数とし，日付 4≦日付 1 ＜日付 3 を満たす（日付 2 ＋ y 年）の日付，及び（日付 2 ＋（y ＋ 1）年）の日付を，それぞれ日付 4 及び日付 3 とします。

     日付 4 と日付 1 の間の日数（DAYS（日付 1）－DAYS（日付 4)），及び日付 4 と日付 3 の間の日数（DAYS（日付 3）－DAYS（日付 4））を，それぞれ d1 及び d2 とします。d2 の値は，365 又は 366 のどちらかになります。

     結果の年数は，（y ＋ d1÷d2）年になります。

日付 1，日付 2，日付 3，及び日付 4 の関係を次の図に示します。

図 2-8　日付 1，日付 2，日付 3，及び日付 4 の関係（YEARS_BETWEEN)



(d)　使用例

YEARS_BETWEEN の使用例を次に示します。

```
YEARS_BETWEEN(DATE('2000-06-10'),DATE('1999-06-10'))==>   1
YEARS_BETWEEN(DATE('2000-06-11'),DATE('1999-06-10'))==>   1.002739...
YEARS_BETWEEN(DATE('1999-06-11'),DATE('1998-06-10'))==>   1.002732...
YEARS_BETWEEN(DATE('2014-09-09'),DATE('1999-06-10'))==>  15.249315...
YEARS_BETWEEN(DATE('1999-06-10'),DATE('2014-09-09'))==> -15.249315...
```

2.16.3　プラグイン定義スカラ関数

(1)　XMLQUERY

HiRDB XML Extension と連携することで使用できるスカラ関数です。

詳細は，「XMLQUERY」を参照してください。

(2)　XMLSERIALIZE

HiRDB XML Extension と連携することで使用できるスカラ関数です。

詳細は，「XMLSERIALIZE」を参照してください。

(3)　XMLPARSE

HiRDB XML Extension と連携することで使用できるスカラ関数です。

詳細は，「XMLPARSE」を参照してください。

2.17　CASE 式

2.17.1　CASE 式の形式と規則

(1)　機能

条件付けられた値を指定します。

(2)　形式

```
CASE式：：＝ {探索CASE式 | 単純CASE式 | CASE略式}
探索CASE式：：＝CASE
                  {WHEN 探索条件 THEN {値式 | NULL} } …
                  〔ELSE {値式 | NULL} 〕
                 END
単純CASE式：：＝CASE 値式
                  {WHEN 値式 THEN {値式 | NULL} } …
                  〔ELSE  {値式 | NULL} 〕
                 END
CASE略式：：＝ {NULLIF（値式，値式） | COALESCE（値式〔，値式〕…）}
```

(3)　規則

1. 一つの CASE 式中で WHEN は 255 個まで指定できます。
2. CASE 式の幾つかの探索条件が真であれば，CASE 式の値は CASE 式のデータ型に変換された，探索条件が真となる最初の WHEN の結果の値となります。
3. CASE 式のどの探索条件も真でなければ CASE 式の値は CASE 式のデータ型に変換された，想定されるか，又は指定される ELSE の結果の値となります。
4. 探索条件については「探索条件」を参照してください。
5. CASE 式の結果のデータ型，及びデータ長は，集合演算のデータ型，及びデータ長と同じです。
6. CASE 式の結果のデータ型は，非ナル値制約なし（ナル値を許す）になります。
7. CASE 式の結果のデータ型が文字データ型の場合，結果の文字集合は次の表に示すとおりになります。

表 2-106　CASE 式の結果の文字集合

CASE 式		結果の文字集合
探索 CASE 式		先頭の THEN に指定した値式の文字集合
単純 CASE 式		先頭の THEN に指定した値式の文字集合
CASE 略式	NULLIF	最初の値式の文字集合
	COALESCE	最初の値式の文字集合

8. COALESCE の場合，値式の並びは左から右に順に評価され，ナル値でない最初の値が結果になります。
9. CASE 中の少なくとも一つの THEN は，値式を指定してください。
10. CASE，THEN，及び ELSE の値式に，?パラメタ，及び埋込み変数を単独（単項演算式に指定した場合も含む）で指定できません。
11. ELSE が指定されていない場合は，ELSE NULL が仮定されます。
12. CASE 値式，WHEN 値式，NULLIF 及び COALESCE の値式には，次に示すデータ型の値を指定できません。
    - BLOB
    - 定義長が 32,001 バイト以上の BINARY
    - BOOLEAN
    - 抽象データ型
13. THEN 及び ELSE の値式には，次に示すデータ型の値を指定できません。
    - BLOB
    - 定義長が 32,001 バイト以上の BINARY
    - BOOLEAN
    - 抽象データ型
14. THEN 及び ELSE に指定した値式が文字データ型の場合，値式の文字集合はすべて同じにしてください。ただし，二つ目以降の THEN，又は ELSE に指定した値式が次に示す値式の場合，先頭の THEN に指定した値式の文字集合に変換されます。
    - 文字列定数
15. 単純 CASE 式は，WHEN の値式である V1 と CASE の値式である V2 を，V2 = V1 の型で探索条件に指定した探索 CASE 式と同じです。
16. 単純 CASE 指定中の最初の WHEN の値式，COALESCE の最初の値式，又は NULLIF の両方の値式に，?パラメタ，及び埋込み変数は単独（単項演算式に指定した場合も含む）で指定できません。
17. 単純 CASE 式の一つ以上の WHEN の値式が?パラメタ，又は埋込み変数の場合，?パラメタ，又は埋込み変数のデータ型は，最初の WHEN の値式のデータ型を仮定します。
18. NULLIF の一つの値式が?パラメタ，又は埋込み変数の場合，?パラメタ，又は埋込み変数のデータ型は，もう一方の値式のデータ型を仮定します。
19. COALESCE の一つ以上の値式が?パラメタ，又は埋込み変数の場合，?パラメタ，又は埋込み変数のデータ型は，最初の値式のデータ型を仮定します。
20. COALESCE の値式は 255 個まで指定できます。
21. NULLIF（V1，V2）は，次に示すの CASE 式と同じです。
    CASE WHEN V1 = V2 THEN NULL ELSE V1 END
22. COALESCE（V1，V2）は，次に示すスカラ関数 VALUE，又は CASE 式と同じです。

VALUE（V1，V2）

CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END

23. n が三つ以上の COALESCE（V1，V2，…，Vn）は，次に示すスカラ関数 VALUE，又は CASE 式と同じです。

VALUE（V1，V2，…，Vn）

CASE WHEN V1 IS NOT NULL THEN V1 ELSE COALESCE（V2，…，Vn） END

24. 探索 CASE 式，単純 CASE 式の THEN，及び ELSE の値式のデータ型はそれぞれ比較できるデータ型にしてください。比較できるデータ型については「データ型」を参照してください。ただし，次に示すデータは比較できません。

- 日付データと日付データの文字列表現
- 時刻データと時刻データの文字列表現
- 時刻印データと時刻印データの文字列表現
- 日間隔データと日間隔データの 10 進数表現
- 時間隔データと時間隔データの 10 進数表現
- バイナリデータと 16 進文字列定数

25. 単純 CASE 式の場合，WHEN の値式のデータ型と CASE の値式のデータ型は比較できるデータ型にしてください。比較できるデータ型については「データ型」を参照してください。ただし，次に示すデータは比較できません。

- 日付データと日付データの文字列表現
- 時刻データと時刻データの文字列表現
- 時刻印データと時刻印データの文字列表現
- 日間隔データと日間隔データの 10 進数表現
- 時間隔データと時間隔データの 10 進数表現
- バイナリデータと 16 進文字列定数

26. 単純 CASE 式で CASE 値式と WHEN の値式が文字列型の場合，CASE 値式と WHEN の値式の文字集合は同じにしてください。ただし，WHEN の値式が次に示す値式である場合，CASE 値式の文字集合に変換して比較を行います。

- 文字列定数

27. NULLIF，COALESCE の場合，値式のデータ型はそれぞれ比較できるデータ型にしてください。比較できるデータ型については「データ型」を参照してください。ただし，次に示すデータは比較できません。

- 日付データと日付データの文字列表現
- 時刻データと時刻データの文字列表現
- 時刻印データと時刻印データの文字列表現
- 日間隔データと日間隔データの 10 進数表現

- 時間隔データと時間隔データの 10 進数表現
- バイナリデータと 16 進文字列定数

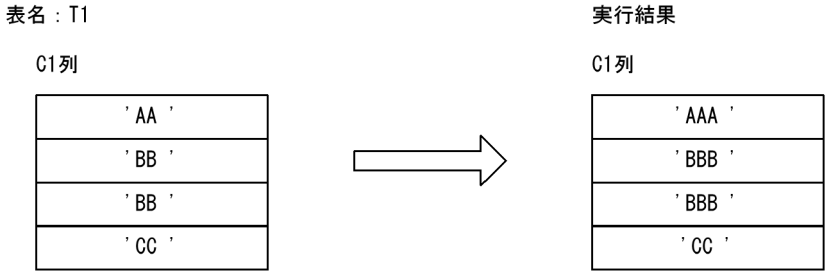
28. NULLIF で，値式が両方とも文字列データ型の場合，二つの値式の文字集合は同じにしてください。ただし，値式のどちらかが次に示す値式である場合，もう一方の値式の文字集合に変換します。
    - 文字列定数
29. COALESCE で，値式が両方とも文字列データ型の場合，二つの値式の文字集合は同じにしてください。ただし，二番目の値式が次に示す値式である場合，一番目の値式の文字集合に変換します。
    - 文字列定数
30. 繰返し列は，添字を指定すれば CASE 式中に指定できます。また，CASE 式の探索条件の繰返し列には添字として ANY を指定でき，IS NULL 述語には添字なし繰返し列を指定できます（指定できる箇所は，探索条件と同じです）。ただし，選択式に指定した CASE 式中の繰返し列には，添字として ANY は指定できません。
31. 予備列は指定できません。
32. ウィンドウ関数は指定できません。

(4) 使用例

(a) 探索 CASE 式の場合

表 T1 のうち，C1 列が'AA'のものを'AAA'，C1 列が'BB'のものを'BBB'に更新します。

```
UPDATE T1 SET C1 =
    CASE WHEN C1='AA' THEN 'AAA' WHEN C1='BB' THEN 'BBB' ELSE C1 END
```



(b) 単純 CASE 式の場合

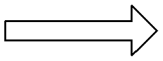
表 T1 のうち，C1 列が'AA'のものを'AAA'，C1 列が'BB'のものを'BBB'に更新します。

```
UPDATE T1 SET C1 =
    CASE C1 WHEN 'AA' THEN 'AAA' WHEN 'BB' THEN 'BBB' ELSE C1 END
```

表名：T1

C1列

'AA '
'BB '
'BB '
'CC '



実行結果

C1列

'AAA '
'BBB '
'BBB '
'CC '

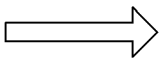
(c) CASE 略式（COALESCE）の場合

表 T1 の列 C1，C2，C3 の順で，ナル値でない列を抽出します。すべてがナル値の場合，結果として 0 が設定されます。

```
SELECT COALESCE(C1,C2,C3,0)FROM T1
```

表名：T1

C1列	C2列	C3列
*	20	*
*	*	*



実行結果

20
0

注　* はナル値を示します。

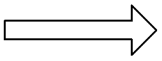
(d) CASE 略式（NULLIF）の場合

表 T1 の列 C1，C2 の値等しい場合，結果としてナル値を返します。列 C1，C2 の値が等しくない場合，列 C1 を抽出します。

```
SELECT NULLIF(C1,C2) FROM T1
```

表名：T1

C1列	C2列
10	20
20	20



実行結果

10
*

注　* はナル値を示します。

2.18　オーバフローエラー抑止が設定されている場合の演算結果

システム定義の pd_overflow_suppress でオーバフローエラー抑止が設定されている場合は，次に示すような場合でもエラーとしないで，オーバフローが発生した演算の結果をナル値として処理を続行します。

- 除算で第 2 演算項の値に 0 を指定した場合
- 計算途中でオーバフローが発生した場合

オーバフローエラー抑止の対象を次に示します。

- 数値型のオーバフロー
- 0 除算エラー
- 日付データ型のオーバフロー
- 時刻データ型のオーバフロー
- 日間隔データ型のオーバフロー
- 時間隔データ型のオーバフロー
- ラベル付き間隔のオーバフロー
- 集合関数 SUM，COUNT，COUNT_FLOAT，AVG のオーバフロー
- ウィンドウ関数のオーバフロー
- スカラ関数のオーバフロー（オーバフローエラー抑止の対象となるスカラ関数を次の表に示します）

表 2-107　オーバフローエラー抑止の対象となるスカラ関数

分類	スカラ関数	スカラ関数の種別
変換関数	DEGREES	システム定義スカラ関数
	NUMEDIT	
	STRTONUM	
数学関数	ABS	システム組込みスカラ関数
	MOD	
	CEIL	システム定義スカラ関数
	COSH	
	EXP	
	FLOOR	
	POWER	
	ROUND	
	SINH	

分類	スカラ関数	スカラ関数の種別
	TAN	
	TANH	
	TRUNCYEAR	
日付操作	ADD_INTERVAL	
	NEXT_DAY	
	ROUNDMONTH	

ただし，オーバフロー抑止が設定されていても，次の場合はエラーになるので注意してください。

- UAP 中で標識変数を指定しないでデータを受け取ろうとした場合
- 非ナル値制約のある列に対して，オーバフロー抑止によって設定されたナル値で，更新，又は挿入しようとした場合

オーバフローが発生した場合，SQL 連絡領域の SQLWARNB に'W'が設定されるので，SQLWARNB を参照してオーバフローの発生の有無を確認できます。

オーバフローエラー抑止が設定されている場合の処理の例を，次に示す管理表（KANRI）を使用して説明します。

表名：管理表（KANRI）

GNO （品番）	TANKA （単価）	SURYO （数量）	GOUKEI （合計）
A-200	20000	1000	ナル値
A-280	28000	500	ナル値
B-300	30000	80000	ナル値
B-350	35000	60000	ナル値
B-380	38000	50000	ナル値

注　単価（TANKA），数量（SURYO），及び合計（GOUKEI）列のデータ属性は，INTEGERにしてください。
また，合計（GOUKEI）列は非ナル値制約なしの列にしてください。

2.18.1　探索条件でオーバフローが発生した場合の例

管理表（KANRI）から単価（TANKA）と数量（SURYO）の積が 15,000,000 より大きい値を検索します。

(1) SQL

```
SELECT GNO,TANKA,SURYO
    FROM KANRI
    WHERE TANKA * SURYO > 15000000
```

表名：管理表（KANRI）

GNO（品番）	TANKA（単価）	SURYO（数量）	
A-200	20000	1000	→ 真
A-280	28000	500	→ 偽
B-300	30000	80000	→ オーバフロー発生
B-350	35000	60000	→ 真
B-380	38000	50000	→ 真

(2) 検索結果

GNO（品番）	TANKA（単価）	SURYO（数量）
A-200	20000	1000
B-350	35000	60000
B-380	38000	50000

オーバフローエラー抑止が設定されていない場合，品番（GNO）が B-300 の行で単価と数量の積（TANKA＊SURYO）でオーバフローが発生し，そこで処理は打ち切られます。

オーバフローエラー抑止が設定されている場合でも，品番（GNO）が B-300 の行で単価と数量の積（TANKA＊SURYO）でオーバフローが発生しますが，そこで処理は続行します。

ただし，オーバフローが発生した演算の結果はナル値になり，条件を満たさない（不定になります）ため，その行は検索されません。

2.18.2 更新値でオーバフローが発生した場合の例

管理表（KANRI）から単価（TANKA）と数量（SURYO）の積（TANKA＊SURYO）の値で合計（GOUKEI）を更新します。

(1) SQL

```
UPDATE KANRI
    SET GOUKEI = TANKA * SURYO
```

(2) 更新結果（オーバフローエラー抑止設定時）

表名：管理表（KANRI）

GNO （品番）	TANKA （単価）	SURYO （数量）	GOUKEI （合計）
A-200	20000	1000	20000000
A-280	28000	500	14000000
B-300	30000	80000	ナル値
B-350	35000	60000	2100000000
B-380	38000	50000	1900000000

オーバフローエラー抑止が設定されていない場合は，品番（GNO）が B-300 の行で単価と数量の積（TANKA＊SURYO）でオーバフローが発生し，そこで処理は打ち切られます。

オーバフローエラー抑止が設定されている場合でも，品番（GNO）が B-300 の行で単価と数量の積（TANKA＊SURYO）でオーバフローは発生するが，そこで処理は続行します。

ただし，オーバフローが発生した演算の結果はナル値になり，合計列（GOUKEI）はナル値に更新されます。オーバフローエラー抑止が設定されているときに，表の更新などをする場合は，ナル値で表が更新されるため注意してください。

2.19 排他オプション

2.19.1 排他オプションの形式と規則

(1) 機能

データ検索時の排他制御について指定します。

排他オプションは，カーソル宣言，FOR 文のカーソル指定，1 行 SELECT 文，及び動的 SELECT 文中に指定します。

(2) 形式

```
排他オプション：：＝〔 {WITH {SHARE｜EXCLUSIVE} LOCK
                          ｜WITHOUT LOCK 〔 {WAIT｜NOWAIT} 〕} 〕
                     〔 {WITH ROLLBACK｜NO WAIT} 〕
```

(3) オペランド

- 〔{WITH {SHARE | EXCLUSIVE} LOCK
  | WITHOUT LOCK 〔{WAIT | NOWAIT}〕}〕

このオペランドを省略すると，WITH SHARE LOCK が仮定されます。

ただし，カーソル宣言，又は動的 SELECT 文で FOR UPDATE 句を指定した場合，又はそのカーソルを使用した行の更新，削除をする場合，WITH EXCLUSIVE LOCK が仮定されます。

WITH SHARE LOCK

一度検索したデータの内容を，トランザクション終了までほかのユーザが参照することは許すが，更新することは許さない（共用モード）場合に指定します。

WITH EXCLUSIVE LOCK

一度検索したデータの内容を，トランザクション終了までほかのユーザが参照する（WITHOUT LOCK NOWAIT による参照は除く）ことも，更新することも許さない（排他モード）場合に指定します。

WITH EXCLUSIVE LOCK を指定した場合，問合せ指定の FROM 句に指定した表に対してだけ有効です。

WITH EXCLUSIVE LOCK を指定しても，インデクスのキー値には，共用モードの排他を参照時だけにかけ，参照終了時に排他を解除するので，ほかのユーザは同じインデクスを使用できます。ただし，検索に用いているインデクスをほかのユーザが更新している場合，待ちが生じることがあります。トランザクションが終了するまでほかのユーザの処理を待たせるためには，LOCK 文を使用します。

WITHOUT LOCK WAIT

一度検索したデータの内容を，トランザクション終了まで排他制御する必要がない場合に指定します。WITHOUT LOCK ［WAIT］を指定した SQL 文の検索では，参照済みの排他資源（行又はページ）の排他は解除されます。

WITHOUT LOCK ［WAIT］を指定すると，トランザクションの終了を待たないで行の検索処理完了を契機に排他を解除するので，排他資源を小さくできます。また，同時実行性も向上します。ただし，同じトランザクション内で同じ行を 2 度検索しても同じデータを受け取れない場合があるので注意してください。

WITHOUT LOCK NOWAIT

ほかのユーザが更新中（又は排他オプションを指定中）のデータでも更新の完了（又はトランザクションの終了）を待たないで参照し，一度検索したデータの内容をトランザクション終了まで排他制御する必要がない場合に指定します。ただし，検索対象表が共用表であり，ほかのユーザによって排他モードで LOCK 文が実行されている場合は，排他解除待ちとなり参照できません。

また，論理ファイルを参照する場合を除き，WITHOUT LOCK NOWAIT を指定した SQL 文の検索では，ほかのトランザクションが表や行に EX モードで排他を掛けているときでも，排他なしと同じ状態として参照できます。

WITHOUT LOCK NOWAIT を指定した場合，同じトランザクション内で同じ行を 2 度検索しても同じデータを受け取れない場合があるので注意してください。さらに，更新したユーザがそのトランザクションを取り消すと，更新中の表を検索したときにデータベースにないデータを受け取ったことになるので注意してください。

カーソル宣言中，又は動的 SELECT 文中で WITHOUT LOCK NOWAIT を指定する場合，そのカーソル宣言，又は動的 SELECT 文で FOR UPDATE 句は指定できません。また，そのカーソルを使用した UPDATE，又は DELETE も実行できません。

- 〔{WITH ROLLBACK ｜ NO WAIT}〕

このオペランドを省略すると，検索するデータがほかのユーザによって使用されているとき，その使用中のユーザのトランザクションが終了するまで待ち状態（WITHOUT LOCK NOWAIT を除く）になります。

WITH ROLLBACK

検索の対象になるデータがほかのユーザに使用されているとき，トランザクションを取り消して無効にする場合に指定します（エラーとなった SQL より前に実行した SQL は，ロールバックされます）。

NO WAIT

検索の対象になるデータがほかのユーザに使用されているとき，検索をエラー（排他エラー）とし，トランザクションを取り消さない場合に指定します（エラーとなった SQL より前に実行した SQL はロールバックされません）。

この場合，指定した SQL で掛けた排他は解除されません。カーソルは閉じられないので，UAP 中で CLOSE 文を実行してカーソルを閉じてください。

ただし，次に示す箇所で排他エラーが発生した場合，NO WAIT を指定した場合でも，トランザクションを取り消して，無効にします。

- 副問合せで検索の対象になる表
- FROM 句の導出表で検索の対象になる表
- GROUP BY 句に値式を指定した問合せで，検索の対象となる表
- 「内部導出表」となる条件のうちの，どれかを満たす問合せで，検索の対象になる表

内部導出表については，「内部導出表」を参照してください。

(4) 留意事項

- WITHOUT LOCK NOWAIT と WITHOUT LOCK NO WAIT とでは，意味が異なるので指定を誤らないように注意してください。

(5) 指定例（SELECT 文中で排他オプションを指定した場合）

```
SELECT SCODE FROM ZAIKO
   WHERE SNAME = N'ブラウス'
   WITH SHARE LOCK
   WITH ROLLBACK
```

2.20　関数呼出し

2.20.1　関数呼出しの形式と規則

(1)　機能

指定した関数を呼び出します。

(2)　形式

```
関数呼出し：：＝〔認可識別子.〕ルーチン識別子（〔引数〔,引数〕…〕）
引数：：＝値式 〔AS データ型〕
```

(3)　オペランド

- 認可識別子

呼び出す関数の認可識別子を指定します。

パブリック関数を呼び出す場合は，認可識別子に PUBLIC を指定します。

- ルーチン識別子

呼び出す関数のルーチン識別子を指定します。

- 引数：：＝値式 〔AS データ型〕

値式

呼び出す関数のパラメタに対する値式を指定します。

AS データ型

呼び出す関数のパラメタに対する？パラメタ，又は埋込み変数の既定義型を指定します。

(4)　共通規則

1. 引数は，指定した順序でパラメタと対応します。
2. 引数のデータ型は，対応するパラメタのデータ型と互換性がなければなりません。互換性があるデータ型については，「変換（代入，比較）できるデータ型」を参照してください。ただし，次の組み合わせの場合は互換性がありません。
   - 文字集合が異なる文字データ
   - 文字データと混在文字データ
   - 日間隔データと日間隔を 10 進数で表現した定数

- 日間隔データと DECIMAL（8，0）に対応する埋込み変数
- 時間隔データと時間隔を 10 進数で表現した定数
- 時間隔データと DECIMAL（6，0）に対応する埋込み変数
- 日付データと日付を文字列で表現した定数
- 日付データと CHAR（10）に対応する埋込み変数
- 日付データと文字集合が UTF16 の CHAR（20）に対応する埋込み変数
- 日付データと，長さが 10 バイト以上の CHAR，又は VARCHAR の埋込み変数
- 日付データと，長さが 20 バイト以上で長さが 2 の倍数の CHAR，又は VARCHAR の文字集合が UTF16 の埋込み変数
- 時刻データと時刻を文字列で表現した定数
- 時刻データと DECIMAL（6，0）に対応する埋込み変数
- 時刻データと，長さが 8 バイト以上の CHAR，又は VARCHAR の埋込み変数
- 時刻データと，長さが 16 バイト以上で長さが 2 の倍数の CHAR，又は VARCHAR の文字集合が UTF16 の埋込み変数
- 時刻印データと時刻印を文字列で表現した定数
- 時刻印データの小数秒精度が 0 の場合は 19 バイト以上，2 以上の場合は 22 バイト以上の CHAR 又は VARCHAR の埋込み変数
- 時刻印データの小数秒精度が 0 の場合は 38 バイト以上で長さが 2 の倍数，2 以上の場合は 44 バイト以上で長さが 2 の倍数の CHAR 又は VARCHAR の文字集合が UTF16 の埋込み変数
- BINARY 型と 16 進文字列定数

さらに，引数のデータ型はパラメタのデータ型と優先度が同じか，又はより優先度が高いデータ型でなければなりません。データ型の優先度については，表「既定義型の優先度」及び表「抽象データ型の優先度」を参照してください。

3. 値式に？パラメタ又は埋込み変数だけを指定する場合は，AS データ型を必ず指定して，値式のデータ型を決定してください。
4. AS データ型を指定した場合，値式には？パラメタ又は埋込み変数以外は指定できません。
5. 値式に？パラメタ，又は埋込み変数を用いた単項演算は指定できません。
6. 引数に繰返し列を指定した場合，添字を指定してください。ただし，添字として ANY は指定できません。
7. 値式に？パラメタ，埋込み変数だけを指定した場合，その？パラメタ，埋込み変数は単純構造にしてください。
8. 値式に予備列は指定できません。
9. 関数呼出しのネスト数は，最大 255 です。関数呼出しのネスト数は，"ルーチン識別子（"の括弧のネスト数です。

10. 引数には，結果のデータ型が BLOB，又は最大長が 32,001 バイト以上の BINARY 型となるスカラ関数 SUBSTR を，単独の値式として指定できません。

11. ウィンドウ関数は指定できません。

(5) 留意事項

1. 認可識別子を省略した場合の仮定値については，「スキーマパス」を参照してください。

(6) 呼び出す関数の決定規則と結果のデータ型

1. 認可識別子，ルーチン識別子，引数の数が一致し，引数のデータ型に抽象データ型を含まないで，かつ引数の順序に対応してパラメタのデータ型が完全一致する場合は，この関数を呼び出します。また，この場合の関数の結果のデータ型は，呼び出す関数の RETURNS 句のデータ型になります。

2. 認可識別子，ルーチン識別子，引数の数のどれかが一致しない関数の場合は，この関数は呼び出しの対象とはなりません。

3. 呼び出す関数の決定規則を，次の表に示します。

表 2-108 関数呼出しの指定内容と，呼び出す関数の決定規則

<table>
<tr><th colspan="3">関数呼出しの指定内容</th><th>呼び出す関数の決定規則</th></tr>
<tr><td rowspan="4">認可識別子，ルーチン識別子，引数の数が一致する</td><td colspan="2">引数の数が 0</td><td rowspan="2">認可識別子，ルーチン識別子，引数のデータ型が一致する関数を呼び出す</td></tr>
<tr><td rowspan="2">引数に抽象データ型を含まない</td><td>引数の順序に対応したパラメタのデータ型が完全に一致※</td></tr>
<tr><td>引数の順序に対応したパラメタのデータ型が一致しない</td><td>下記の「・抽象データ型を含まない場合」の決定規則に従う</td></tr>
<tr><td colspan="2">引数に抽象データ型を含む</td><td>下記の「・抽象データ型を含む場合」の決定規則に従う</td></tr>
<tr><td colspan="3">認可識別子，ルーチン識別子，引数の数が一致しない</td><td>呼び出し可能な関数が存在しないため，SQL 文解析時にエラーとなる</td></tr>
</table>

注※

引数のデータ型が文字列型の場合，「データ型が一致する」とは，文字集合も一致することをいいます。

- 抽象データ型を含まない場合

  左側の引数から順番に各引数の既定義型を基準として，基準と優先度が同じか又はより優先度が低いデータ型の中で最も優先度の高い既定義型をパラメタに持つ関数を呼び出します。既定義型の優先度を次の表に示します。また，この場合，呼び出す関数が SQL 解析時に一意に決まるので，関数の結果のデータ型は呼び出す関数の RETURNS 句のデータ型となります。

表 2-109 既定義型の優先度

各引数のデータ型	優先度
数データ	SMALLINT→INTEGER→DECIMAL→SMALLFLT→FLOAT

各引数のデータ型	優先度
文字データ	CHAR→VARCHAR
各国文字データ	NCHAR→NVARCHAR
混在文字データ	MCHAR→MVARCHAR
長大データ，及びバイナリデータ	BINARY→BLOB

（凡例）

A→B：A が B より優先度が高いことを示します。

- 抽象データ型を含む場合

  抽象データ型を含む場合，次の順番で呼び出す関数を決定します。

4. 基本となる関数の決定

基本となる関数の決定方法は，左側の引数から順番に各引数のデータ型を基準として，基準と優先度が同じか又はより優先度が低いデータ型の中で最も優先度の高いデータ型をパラメタに持つ関数を，基本となる関数とします。データ型が既定義型の場合は，表「既定義型の優先度」の優先度に準じます。データ型が抽象データ型の場合は，次に示す表の優先度に準じます。

表 2-110　抽象データ型の優先度

各引数のデータ型	優先度
抽象データ型	同じデータ型→スーパタイプ※

（凡例）

A→B：A が B より優先度が高いことを示します。

注※

抽象データ型定義中の UNDER 句で直接指定するスーパタイプの方が，そのほかのスーパタイプよりも優先度が高くなります。

5. 候補となる関数の決定

引数が抽象データ型の場合，引数のデータとして取り得る実際の値のデータ型は，引数の定義の抽象データ型と同じデータ型又はサブタイプとなります。そのため，基本となる関数のほかに，引数の抽象データ型と同じデータ型又はサブタイプの抽象データ型を対応するパラメタに持つすべての関数が候補となる関数となります。

候補となる関数が，基本となる関数一つだけの場合，その関数が呼び出す関数となります。関数の結果のデータ型は，呼び出す関数の RETURNS 句のデータ型になります。

6. RETURNS 句のデータ型を用いた候補となる関数の絞り込み

抽象データ型の引数を含む場合，基本となる関数の RETURNS 句のデータ型と，基本となる関数以外の候補となる関数の RETURNS 句のデータ型の互換性のチェックをします。RETURNS 句のデータ型が互換性のない関数の場合，候補となる関数ではなくなります。

互換性のチェックの後，残った候補となる関数の RETURNS 句のデータ型を基に，関数の結果のデータ型を決定します。

互換性のチェックの後，残った候補となる関数の RETURNS 句のデータ型が抽象データ型の場合は，基本となる関数の RETURNS 句の抽象データ型が結果のデータ型になります。

互換性のチェックの後，残った候補となる関数の中に RETURNS 句のデータ型が 32,001 バイト以上の BINARY 型又は BLOB 型の関数を含む場合は，次の規則に従います。

- 結果のデータ長は，一番長いデータ長となります。
- BINARY 型と BLOB 型が混在する場合は，BLOB 型になります。

その他の場合，結果のデータ型及びデータ長は，集合演算（UNION ALL，又は EXCEPT ALL）の結果のデータ型及びデータ長と同じになります。詳細については，「問合せ式」を参照してください。

7. SQL 文実行時の関数の決定

2 及び 3 で関数が一意に決まらない場合，SQL 文実行時に，抽象データ型の引数の実際のデータ型によって，候補となる関数の中から呼び出す関数を一つに決定します。左側の引数より順番に，各引数の実際の値がナル値以外の場合はその値のデータ型を基準として，ナル値の場合はその引数のデータ型を基準とし，その基準のデータ型と同じか又はより優先度が低いデータ型の中で最も優先度の高いデータ型をパラメタとして持つ関数を候補となる関数の中から一つ決定し，呼び出す関数とします。

（例） 抽象データ型を含む場合の呼び出し関数の決定

A，B，C を抽象データ型とし，C を B のスーパタイプ，B を A のスーパタイプとします（抽象データ型の優先度：A→B→C）。

（例 1）

＜前提条件＞

表定義

```
CREATE TABLE T1(C1 C)
```

関数定義

f(A)，f(B)，f(C)

SQL 文

```
SELECT f(C1) FROM T1
```

＜結果＞

基本となる関数

f(C)

関数呼出しが f(C1)の場合の候補となる関数

f(A),f(B),f(C)

呼び出し関数

SQL 文実行時に呼び出す関数を次に示します。

T1.C1 の実際の値	呼び出し関数
A 型	f(A)

T1.C1 の実際の値	呼び出し関数
B 型	f(B)
C 型	f(C)
NULL 値	f(C)

(例 2)

＜前提条件＞

表定義

CREATE TABLE T1(C1 C,C2 B)

関数定義

f(A,A)，f(A,B)，f(A,C)，f(B,A)，f(B,C)，f(C,A)，f(C,B)，f(C,C)

SQL 文

SELECT f(C1,C2) FROM T1

＜結果＞

基本となる関数

f(C,B)

関数呼出しが f(C1,C2)の場合の候補となる関数

f(A,A),f(A,B),f(A,C),f(B,A),f(B,C),f(C,A),f(C,B)

呼び出し関数

SQL 文実行時に呼び出す関数を次に示します。

T1.C1 の実際の値	T1.C2 の実際の値	呼び出し関数
A 型	A 型	f(A,A)
	B 型	f(A,B)
	NULL 値	f(A,B)
B 型	A 型	f(B,A)
	B 型	f(B,C)
	NULL 値	f(B,C)
C 型	A 型	f(C,A)
	B 型	f(C,B)
	NULL 値	f(C,B)
NULL 値	A 型	f(C,A)
	B 型	f(C,B)
	NULL 値	f(C,B)

2.21　内部導出表

ビュー表，又は WITH 句の問合せ名を，名前付きの導出表といいます。

名前付きの導出表を問合せ指定に指定する場合，その名前付きの導出表に対して，内部的に作業表を作成する場合があります。この作業表を作成する名前付きの導出表を内部導出表といいます。

2.21.1　内部導出表の条件

(1)～(10)に示す名前付きの導出表を，問合せ指定の FROM 句に指定した場合，内部導出表を作成する条件は，次のようになります。

(1)　SELECT DISTINCT を指定して導出する名前付きの導出表の場合

名前付きの導出表が副問合せ中に含まれる。又は，名前付きの導出表を FROM 句に指定した問合せ指定が次のどれかを直接含んでいる。

- GROUP BY 句，HAVING 句，又は集合関数
- SELECT DISTINCT
- 表の結合（コンマの結合を指定している場合を含む）
- 選択式に列指定以外の値式を指定している
- 選択式にスカラ副問合せを指定している
- 選択式に FROM 句に指定したビュー表の全列を 1 回ずつ指定していない
- ORDER BY 句
- NEXT VALUE 式
- WRITE 指定
- GET_JAVA_STORED_ROUTINE_SOURCE 指定

(例)

「SELECT DISTINCT C1,C2 FROM T1」によって導出されている名前付きの導出表 V1（ビュー表），Q1（WITH 句の問合せ名）を FROM 句に指定する場合，次に示す記述をすると V1 に対して内部導出表を作成します。

```
例1：
SELECT * FROM T2 WHERE EXISTS(SELECT * FROM V1)

例2：
SELECT VC1,VC2 FROM V1 GROUP BY VC1,VC2

例3：
WITH Q1(QC1,QC2) AS (SELECT DISTINCT C1,C2 FROM T1)
    SELECT DISTINCT * FROM Q1
```

```
例4：
SELECT X.VC1,Y.C1 FROM V1 X,T2 Y WHERE X.VC1=Y.C2

例5：
SELECT V1.VC1,T2.C2 FROM V1 LEFT JOIN T2 ON T2.C2=V1.VC2

例6：
WITH Q1(QC1,QC2) AS (SELECT DISTINCT C1,C2 FROM T1)
    SELECT QC1+100,CURRENT_DATE FROM Q1

例7：
SELECT VC1,(SELECT C1 FROM T2) FROM V1

例8：
SELECT VC1 FROM V1

例9：
SELECT VC1,VC2 FROM V1 ORDER BY 1

例10：
INSERT INTO T2 SELECT NEXT VALUE FOR SEQ1,VC1,VC2 FROM V1
```

(2) GROUP BY 句，HAVING 句，又は集合関数を指定して導出する名前付きの導出表の場合

名前付きの導出表を FROM 句に指定した問合せ指定が，次のどれかを直接含んでいる。

- GROUP BY 句，HAVING 句，又は集合関数
- 表の結合（コンマの結合を指定している場合を含む）
- ウィンドウ関数
- NEXT VALUE 式

（例）

「SELECT C1,C2 FROM T1 GROUP BY C1,C2」によって導出されている名前付きの導出表 V1（ビュー表），Q1（WITH 句の問合せ名）及び「SELECT MAX(C1),C2 FROM T1 GROUP BY C2 HAVING C2<100」によって導出されている名前付きの導出表 V2（ビュー表）を FROM 句に指定する場合，次に示す記述をすると V1，V2，Q2 に対して内部導出表を作成します。

```
例1：
WITH Q1(QC1,QC2) AS (SELECT C1,C2 FROM T1
  GROUP BY C1,C2) SELECT AVG(QC1),QC2 FROM Q1
    GROUP BY QC2

例2：
SELECT V1.VC1,V2.VC1 FROM V1,V2 WHERE V1.VC1=V2.VC1

例3：
WITH Q1(QC1,QC2) AS (SELECT C1,C2 FROM T1
  GROUP BY C1,C2)
  SELECT Q1.QC1,V1.VC1 FROM Q1 INNER JOIN V1 ON V1.VC2=Q1.QC2
```

```
例4：
SELECT COUNT(*) OVER(),C1 FROM V1

例5：
INSERT INTO T2 SELECT NEXT VALUE FOR SEQ1,VC1,VC2 FROM V1
```

(3) 選択式に列指定以外の値式を指定して導出する名前付きの導出表の場合

名前付きの導出表を FROM 句に指定した問合せ指定が，次のどれかを直接含んでいる。

- GROUP BY 句，HAVING 句，又は集合関数※
- ウィンドウ関数
- 外結合，内結合，又は交差結合

注※

グループ分け高速化機能が有効な場合を除きます。ただし，選択式にコンポネント指定を指定して導出する名前付き導出表については，グループ分け高速化機能を適用しません。グループ分け高速化機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

（例）

「SELECT C1+100,C2 || C2 FROM T1」によって導出されている名前付きの導出表 V1（ビュー表），Q1（WITH 句の問合せ名）を FROM 句に指定する場合，次に示す記述をすると V1，Q1 に対して内部導出表を作成します。

```
例1：
SELECT AVG(VC1),VC2 FROM V1 GROUP BY VC2

例2：
SELECT * FROM V1 LEFT JOIN T2 ON T2.C2=V1.VC2

例3：
WITH Q1(QC1,QC2) AS (SELECT C1+100,C2||C2 FROM T1)
  SELECT QC1,QC2 FROM Q1 GROUP BY QC1,QC2 HAVING QC1<=100

例4：
SELECT COUNT(*) OVER(),VC1 FROM V1
```

(4) DISTINCT 指定の集合関数を指定して導出する名前付きの導出表の場合

名前付きの導出表を FROM 句に指定した問合せ指定が，次のどれかを直接含んでいる。

- SELECT DISTINCT
- GROUP BY 句，HAVING 句，又は集合関数
- 表の結合（コンマの結合を指定している場合を含む）
- ウィンドウ関数
- NEXT VALUE 式

（例）

「SELECT AVG(DISTINCT C1) FROM T1」によって導出されている名前付きの導出表 V1（ビュー表）を FROM 句に指定する場合，次に示す記述をすると V1 に対して内部導出表を作成します。

```
SELECT DISTINCT VC1 FROM V1
WITH Q1(C1) AS (SELECT AVG(DISTINCT C1) FROM T1)
  SELECT COUNT(*) OVER(),C1 FROM Q1
```

(5) 表の結合を指定して導出する名前付きの導出表の場合

名前付きの導出表が，外結合，内結合，又は交差結合の表参照に指定されている。

（例）

「SELECT T1.C1,T2.C1 FROM T1,T2」で導出されている名前付きの導出表 V1（ビュー表），Q1（WITH 句の問合せ名）を FROM 句に指定する場合，次に示す記述をすると V1，Q1 に対して内部導出表を作成します。

```
例1：
SELECT V1.* FROM V1 LEFT JOIN T3 ON T3.C1=V1.VC1

例2：
WITH Q1(QC1,QC2) AS (SELECT T1.C1,T2.C1 FROM T1,T2)
  SELECT * FROM Q1 INNER JOIN T3 ON Q1.QC1=T3.C1
```

(6) 外結合，内結合，又は交差結合を指定して導出する名前付きの導出表の場合

名前付きの導出表を FROM 句に指定した問合せ指定が，表の結合を直接含んでいる。

（例）

「SELECT T1.C1,T2.C1 FROM T1 LEFT JOIN T2 ON T1.C2=T2.C2」で導出されている名前付きの導出表 V1（ビュー表），Q1（問合せ名）を FROM 句に指定する場合，次に示す記述をすると V1，Q1 に対して内部導出表を作成します。

```
例1：
SELECT V1.VC1,T3.C1 FROM V1 LEFT JOIN T3 ON T3.C2=V1.VC2

例2：
WITH Q1(QC1,QC2) AS (SELECT T1.C1,T2.C1 FROM T1 LEFT JOIN T2
  ON T1.C2=T2.C2) SELECT Q1.QC1 FROM Q1 INNER JOIN T3 ON T3.C2=Q1.QC2
```

(7) 選択式に副問合せを含む値式を指定して導出する名前付きの導出表の場合

名前付きの導出表を FROM 句に指定した問合せ指定が，次のどれかを直接含んでいる。

- SELECT DISTINCT
- GROUP BY 句，HAVING 句，又は集合関数

- 表の結合（コンマの結合を指定している場合を含む）
- 選択式に列指定以外の値式を指定している
- 選択式にスカラ副問合せを指定している
- 副問合せを含む値式から導出した同じ列を，2 回以上指定している
- 副問合せを含む値式から導出した列を，外への参照をする列として指定している
- バージョン 07-02 より前に定義したビュー表
- WRITE 指定
- GET_JAVA_STORED_ROUTINE_SOURCE 指定

（例）

「SELECT (SELECT C1 FROM T2),C1 FROM T1」によって導出されている名前付きの導出表 V1（ビュー表），Q1（導出問合せ名）を FROM 句に指定する場合，次に示す記述をすると，V1，Q1 に対して内部導出表を作成します。

```
例1：
SELECT VC1,VC2 FROM V1 WHERE VC1>0

例2：
WITH Q1(QC1,QC2) AS (SELECT (SELECT C1 FROM T2),C1 FROM T1)
  SELECT QC1,QC2 FROM Q1 WHERE QC1>0
```

(8) 集合演算によって導出する名前付きの導出表の場合

次のどれかの条件を満たす。

1. 集合演算の演算項のどれかに，次に示すどれかを直接含む。
   - 内部導出表の問合せ
   - 導出表を指定した問合せ
2. 集合演算の演算項のどれかと，名前付きの導出表に対する問合せが，(1)～(7)に示したどれかの条件を満たす。
3. UNION ALL 以外を含む集合演算か，UNION ALL だけを含む集合演算かどうかによって(9)又は(10)の条件を満たす。

（例）

「SELECT (SELECT C1 FROM T2),C2 FROM T1 UNION SELECT C1,C2 FROM T3」によって導出されている名前付きの導出表 V1（ビュー表），Q1（導出問合せ名）を FROM 句に指定する場合，V1，Q1 に対する問合せでは内部導出表を作成します。

```
例1：
SELECT * FROM V1

例2：
WITH Q1(QC1,QC2) AS (
```

```
SELECT (SELECT C1 FROM T2),C2 FROM T1 UNION SELECT C1,C2 FROM T3)
SELECT * FROM Q1
```

(9) UNION ALL 以外を含む集合演算によって導出する名前付きの導出表の場合

次のどれかを満たす。

1. 名前付きの導出表を FROM 句に指定した問合せ指定が，次のどれかを直接含む。
   - GROUP BY 句，HAVING 句，又は集合関数
   - SELECT DISTINCT
   - 表の結合（コンマの結合を指定している場合を含む）
   - WHERE 句
   - 副問合せ
   - 選択式に列指定以外の値式
   - 名前付きの導出表の全列を，選択式に 1 回ずつ指定していない
   - NEXT VALUE 式
   - WRITE 指定
   - GET_JAVA_STORED_ROUTINE_SOURCE 指定
   - 選択項目にない列をソート項目として指定
2. 「集合演算によって導出する名前付きの導出表の場合」の条件を満たす。

（例）

「SELECT C1,C2 FROM T1 UNION SELECT C1,C2 FROM T2」によって導出されている名前付きの導出表 V1（ビュー表），Q1（問合せ名）を FROM 句に指定する場合，次に示す記述をすると，V1，Q1 に対して内部導出表を作成します。

```
例1：
SELECT C1,C2 FROM V1 GROUP BY C1,C2

例2：
WITH Q1(QC1,QC2) AS (SELECT C1,C2 FROM T1 UNION SELECT C1,C2 FROM T2)
  SELECT QC1,QC2 FROM Q1,T3 WHERE QC1=T3.C1

例3：
INSERT INTO T2 SELECT NEXT VALUE FOR SEQ1, VC1, VC2 FROM V1
```

(10) UNION ALL だけの集合演算によって導出する名前付きの導出表の場合

次のどれかを満たす。

1. 名前付きの導出を FROM 句に指定した問合せ指定が，次に示すどれかを直接含む。

- GROUP BY 句，HAVING 句，集合関数
- ウィンドウ関数
- WHERE 句，副問合せ（ただし(1)の問合せが，副問合せ，集合演算の演算項，INSERT 文の問合せ式本体のどれかに含まれる場合だけ）
- 関数呼出し
- システム定義スカラ関数
- コンポネント指定
- WRITE 指定
- GET_JAVA_STORED_ROUTINE_SOURCE 指定
- 選択項目にない列をソート項目として指定
- 集合演算によって導出した名前付きの導出表を，FROM 句に指定した副問合せ
- 導出表を指定した副問合せ
- GROUP BY 句に列指定以外の値式を指定した副問合せ
- 次のどれかを指定した副問合せ
  ・関数呼出し
  ・システム定義スカラ関数
  ・コンポネント指定
- 選択式に指定した SQL 変数，SQL パラメタ，関数呼出しのうち，データ型が次に示すどれかとなるもの
  BLOB 型
  32001 バイト以上の BINARY 型
  抽象データ型
  BOOLEAN 型

2. 名前付きの導出表を表の結合に指定した問合せ指定に，次のどれかを指定している。
   - 名前付きの導出表を指定した表参照に交差結合を指定している
   - 名前付きの導出表を指定した表参照に右外結合，左外結合の両方を指定している
   - 名前付きの導出表を，左外結合の一番左側の外表以外の表参照に指定している
   - 名前付きの導出表を，右外結合の一番右側の外表以外の表参照に指定している
   - 名前付きの導出表を指定した FROM 句に，コンマの結合を指定している（すなわち，その名前付き導出表を指定した結合表以外に，別の表参照を指定している）
   - 副問合せ又は導出表を指定している
   - 問合せ指定が，副問合せ又は集合演算の演算項に含まれる
   - 名前付きの導出表を導出する集合演算項に，次のどれかを含む
     ・表の結合

・GROUP BY 句，HAVING 句，又は集合関数
・SELECT DISTINCT
・選択式に列指定以外の値式
・内部導出表を生成する問合せ
・導出表を指定した問合せ

- 名前付きの導出表のほかに，集合演算を指定して導出した名前付きの導出表を指定している
- 名前付きの導出表を指定した結合表の表参照に，次のどれかを指定している
  ・表の結合を指定して導出した名前付きの導出表
  ・GROUP BY 句，HAVING 句，又は集合関数を指定して導出した名前付きの導出表
  ・SELECT DISTINCT を指定して導出した名前付きの導出表
  ・選択式に列指定以外の値式を指定して導出した名前付きの導出表
  ・内部導出表を生成する問合せを指定して導出した名前付きの導出表
  ・導出表を指定して導出した名前付きの導出表
  ・副問合せを指定して導出した名前付きの導出表
- 次に示す式によって得られる表の総数が，65 を超える
  表の総数＝（名前付き導出表を導出する表の延べ数）
  ＋（名前付き導出表を導出する集合演算の数＋ 1）
  ×（外結合の右側に指定する表の延べ数）
  ＋（名前付き導出表を指定した問合せ以外にも問合せを指定している場合，その問合せに指定した表の延べ数）

3.「集合演算によって導出する名前付きの導出表の場合」の条件を満たす。

（例）

「SELECT C1,C2 FROM T1 UNION ALL SELECT C1,C2 FROM T2」によって導出されている名前付きの導出表 V1（ビュー表），Q1（問合せ名）を FROM 句に指定する場合，次に示す記述をすると，V1，Q1 に対して内部導出表を作成します。

```
例1：
SELECT C1,C2 FROM V1 GROUP BY C1,C2

例2：
WITH Q1(QC1,QC2) AS (SELECT C1,C2 FROM T1 UNION ALL SELECT C1,C2 FROM T2)
  SELECT QC1,QC2 FROM Q1,T3 WHERE QC1=T3.C1

例3：
SELECT * FROM T1 WHERE EXISTS(SELECT * FROM V1 WHERE V1.C1=T1.C1)

例4：
INSERT INTO T3 SELECT * FROM V1 WHERE C1>’C001’
```

2.22 WRITE 指定

2.22.1 WRITE 指定の形式と規則

(1) 機能

BLOB データをシングルサーバ，又はフロントエンドサーバのユニットのファイルに出力し，出力したユニットの IP アドレスとファイル名を返却します。

(2) 形式

```
WRITE（出力BLOB値，ファイル接頭辞，ファイル出力オプション）
```

(3) オペランド

- 出力 BLOB 値

出力 BLOB 値には，次の項目を指定できます。なお，出力 BLOB 値は BLOB 型にしてください。

- 列指定
- コンポネント指定
- 関数呼出し
- 結果のデータ型が BLOB となるスカラ関数 SUBSTR

出力 BLOB 値についての規則を次に示します。

1. 出力 BLOB 値に指定した関数呼出しの引数，又はスカラ関数 SUBSTR の第 2 引数，第 3 引数には，埋込み変数及び？パラメタを指定できます。
2. 出力 BLOB 値に BLOB 列を指定した場合，その BLOB 列を選択式に単独で指定，又はほかの WRITE 指定の出力 BLOB 値に指定できません。

   指定できない例を次に示します。

   ```
   SELECT WRITE(C1,…),C1 FROM …
   SELECT WRITE(C1,…),WRITE(C1,…) … FROM …
   ```

3. FOR READ ONLY 句を指定した場合，出力 BLOB 値には列指定だけ指定できます。
4. 出力 BLOB 値に指定した関数呼出し，又はスカラ関数 SUBSTR の引数中には，副問合せを指定できません。

- ファイル接頭辞

ファイル接頭辞には，HiRDB が組み立てるファイル名の先頭部分を指定します。ファイル接頭辞は VARCHAR 型で，かつ 222 バイト以内で指定してください。

ファイル接頭辞には，次の項目を指定できます。

- 定数（文字列）
- 埋込み変数又は？パラメタ

ファイル接頭辞についての規則を次に示します。

1. ファイル接頭辞は，接続しているシングルサーバ，又はフロントエンドサーバのユニットで有効なディレクトリを含む絶対パス名で指定してください。また，ファイルの利用者は，そのディレクトリに対して，すべての操作ができる権限（アクセス権の種類：フルコントロール）を HiRDB 管理者から与えられていなければなりません。UNIX 版の場合は，読み取り，書き込み，及びディレクトリ内のサーチ権限を HiRDB 管理者から与えられていなければなりません。
2. ファイル接頭辞に指定するファイルセパレータは，HiRDB サーバが UNIX の場合は/を，Windows の場合は¥を指定してください。また，ファイル名に指定できる文字は，HiRDB サーバのプラットフォームの規則に従います。ただし，pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，utf-8_ivs，又は chinese-gb18030 を指定した場合は，ASCII コードの範囲で指定してください。
3. ファイル接頭辞に埋込み変数，又は？パラメタだけを指定する場合，その埋込み変数，又は？パラメタは単純構造にしてください。
4. ファイル接頭辞の文字集合は既定文字集合にしてください。

- **ファイル出力オプション**

ファイル出力オプションには，ファイル出力のモードを指定します。ファイル出力オプションは数データ型で指定してください（DESCRIBE INPUT 文実行時には INTEGER 型を返却します）。

ファイル出力オプションには，次の項目を指定できます。

- 数定数
- 埋込み変数，又は？パラメタ

また，指定できる値を次に示します。

機能	値
再作成（ファイルがある場合は上書きします）	0
追加書き（ファイルがある場合は終端に追加します）	1
上書き禁止（ファイルがある場合はエラーとします）	2
非同期出力（OS に対して非同期出力を要求します）	4

注

非同期出力は，再作成，追加書き，及び上書き禁止と組み合わせて指定できます。組み合わせる場合は，ほかの指定値と論理和をとった値を指定してください。また，非同期出力を指定しない場合は，OS に対して同期出力（即書き）を要求します。

ファイル出力オプションに埋込み変数，又は？パラメタだけを指定する場合，その埋込み変数，？パラメタは単純構造にしてください。

(4) WRITE 指定の結果についての規則

1. WRITE 指定の結果は，非ナル値制約なし（ナル値を許す）の VARCHAR 型，定義長は 255 バイト，文字集合は既定文字集合となります。
2. WRITE 指定の結果は，次のような形式になります。

```
IPアドレス：ファイル接頭辞-列番号-行カウンタ
          ←-----  ファイル名  ----->
```

注　IP アドレスとファイル接頭辞の間には，コロン（：）が付きます。また，ファイル接頭辞，列番号，行カウンタの間には，それぞれハイフン（－）が付きます。

[説明]

IP アドレス：

クライアントの接続先のシングルサーバ，又はフロントエンドサーバのユニットの IP アドレスを返却します。IP アドレスは，XXX.XXX.XXX.XXX の形式で，7～15 バイトの長さになります（XXX は 0～255 の数値文字）。

ファイル接頭辞：

WRITE 指定の第 2 引数に指定したファイル接頭辞を返却します。

列番号：

導出表の指定位置を示す番号を返却します。列番号は，1 から始まり，先頭を 0 パディングした 5 けたの数値文字列となります。

行カウンタ：

検索した行数に対応した，昇順の番号を返却します。行カウンタは，1 から始まり，先頭を 0 パディングした 10 けたの数値文字列となります。2,147,483,646 行を超えると，1 に戻ります。

3. WRITE 指定を使用できないクライアントから検索をした場合，IP アドレスは設定されないで，コロンから始まる文字列を返却します。
4. 出力 BLOB 値，ファイル接頭辞，及びファイル出力オプションのどれかがナル値の場合，結果もナル値になります。また，埋込み変数へのナル値の既定値設定機能を使用している場合は，IP アドレスだけを返却します。
5. 通常の文字データの検索と同様に，WRITE 指定の結果を受け取る埋込み変数が結果の長さよりも短い場合は，あふれた分を切り捨てて，実際の長さを標識変数に設定します。ただし，切り捨てる処理では，埋込み変数の長さは IP アドレスの最大長分を除いた（埋込み変数長－15）バイトとして処理する

ため，IP アドレスを付けた長さでは，埋込み変数に格納できる場合でも切り捨てが発生することがあります。

6. WRITE 指定の結果を受け取る埋込み変数が 15 バイト未満の場合，エラーとなります。

(5) 出力する BLOB データのファイルについての規則

1. 出力 BLOB 値に指定した BLOB データを，シングルサーバ，又はフロントエンドサーバのユニットのファイルに出力します。
2. WRITE 指定の結果がナル値の場合，ファイルを作成しません。
3. WRITE 指定の BLOB データの実長が 0 バイトの場合は，0 バイトの大きさのファイルを作成します。
4. 出力したファイルは，BLOB データだけを含む形式となります。実長の長さ情報は含みません。
5. UNIX 版の場合，作成したファイルの所有者，及びモードは次のようになります。

   所有者　：HiRDB 管理者

   グループ：HiRDB 管理者と同じグループ

   モード　：rw-rw-rw-
6. 検索結果の列数よりその検索結果を受け取る埋込み変数の指定数が少なく，かつ WRITE 指定の結果を受け取る埋込み変数がない場合は，ファイルを作成しません。

(6) 共通規則

1. WRITE 指定は，最も外側の問合せ指定の選択式に単独で指定できます。
2. ORDER BY 指定時のソートのキーになる項目には，WRITE 指定は指定できません。
3. WITH 句中の導出問合せ式では，選択式に WRITE 指定は指定できません。
4. 集合演算をする場合，対象となる導出表の列に WRITE 指定は指定できません。
5. 副問合せ（FROM 句の導出表も含む）中の選択式には，WRITE 指定は指定できません。
6. ビュー定義の導出問合せ式の選択式には，WRITE 指定は指定できません。
7. INSERT 文の問合せ指定の選択式には，WRITE 指定は指定できません。
8. ルーチン中の SQL 手続き文の問合せ指定では，WRITE 指定は指定できません。

(7) 留意事項

1. 作成したファイルの削除は，ユーザが行ってください。検索結果として UAP にファイル名を返却した後，HiRDB は作成したファイルの操作（読み書き）はしませんが，次の点を留意する必要があります。
   - FETCH 直後に削除する場合，同じカーソル検索の直前の FETCH 結果と BLOB 値が同じときは，同じファイル名を返却してファイルを再作成しないことがあります。この場合，直前のファイル名を記憶しておいて，ファイル名が変わったときに削除するような制御が必要となります。
   - カーソルクローズ後に削除する場合，無条件に削除できます。

- トランザクション解決後は，無条件に削除できます。

2. 障害，又はロールバックが発生しても，HiRDB は作成したファイルを削除しません。
3. 通常，SQL エラーが発生した場合，該当する SQL で作成したファイルは削除されます。しかし，HiRDB サーバ内のファイル出力処理が完了した後にエラーが発生した場合は，ファイルを削除しないことがあります。例えば，HiRDB サーバから HiRDB クライアントへの結果返却の通信エラーなどがあります。
4. 配列を使用した FETCH 機能を使用している場合は，1 回の FETCH で配列用要素数分のファイルが作成されるので，ディスク容量に留意してください。
5. ブロック転送機能を使用している場合は，最初の FETCH でブロック転送行数分のファイルを作成し，以降ブロック転送行数分の FETCH 終了後の次の FETCH のたびにブロック転送行数分のファイル作成を繰り返すので，ディスク容量に留意してください。
6. ファイルを削除しないでいると，ディスク容量など OS の資源を圧迫することになるので注意してください。
7. ほかのトランザクションやカーソル検索とファイル名が重複すると，ファイルを互いに破壊する可能性があるので注意してください。例えば，トランザクションごと，カーソルごとにファイル接頭辞のディレクトリ名やファイル名を変えて，名前が重複しないようにすることをお勧めします。
8. WRITE 指定の結果の文字列で切り捨てが発生した場合，完全なファイル名を取得できませんが，ファイルは作成するため，ディスク容量を圧迫しないように注意してください。
9. ファイル出力オプションに非同期出力を指定すると，OS に対して同期出力（即書き）を指定しないで BLOB データファイルへ出力します。そのため，HiRDB 内のファイル出力処理が終了しても，出力デバイスの高負荷などによって OS によるファイル出力処理が完結しないことがあります。その結果，クライアント側にファイル名が返却されても，タイミングによってはファイルが作成されていない場合や，作成が途中の状態になる場合があります。

   ファイル出力オプションに非同期出力を指定しない場合は，上記のような状態は回避できますが，入出力のオーバヘッドが応答時間に大きく影響することに注意してください。

(8) 使用例

BLOB データのファイル出力機能を使用した検索例を次に示します。

(a) BLOB 列を検索する場合

表 T1 から，列 C1，C2 を検索します。このとき，C1 の BLOB データをファイル出力し，そのファイル名を取得します。

表T1

C1	C2
BLOB値1	10
BLOB値2	20
BLOB値3	30
BLOB値4	40

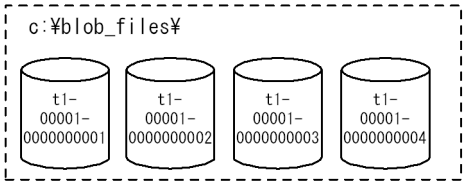
SQL文

```
SELECT WRITE(C1,'c:¥blob_files¥t1',0),C2 FROM T1
```

検索結果

C1	C2
172.16.202.5:c:¥blob_files¥t1-00001-0000000001	10
172.16.202.5:c:¥blob_files¥t1-00001-0000000002	20
172.16.202.5:c:¥blob_files¥t1-00001-0000000003	30
172.16.202.5:c:¥blob_files¥t1-00001-0000000004	40

サーバ側に出力されたBLOBデータ
・IPアドレス：172.16.202.5



(b) BLOB 属性の抽象データ型を検索する場合

表 T2 から，CONTAINS()が真となる ADT1 列を検索します。このとき，列値を EXTRACTS()の引数に渡した結果の BLOB 値をファイルに出力し，ファイル名を取得します。なお，この例は全件ヒットした場合を示しています。

表T2

ADT1
抽象データ型値1
抽象データ型値2
抽象データ型値3
抽象データ型値4

SQL文

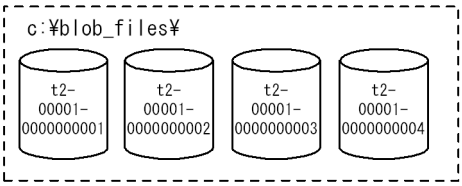
```
SELECT WRITE(EXTRACTS(ADT1,…),'c:¥blob_files¥t2',0) FROM T2
    WHERE CONTAINS(ADT1,…) IS TRUE
```

検索結果

ADT1
172.16.202.5:c:¥blob_files¥t2-00001-0000000001
172.16.202.5:c:¥blob_files¥t2-00001-0000000002
172.16.202.5:c:¥blob_files¥t2-00001-0000000003
172.16.202.5:c:¥blob_files¥t2-00001-0000000004

サーバ側に出力されたBLOBデータ

・IPアドレス：172.16.202.5



2.23 GET_JAVA_STORED_ROUTINE_SOURCE 指定

2.23.1 GET_JAVA_STORED_ROUTINE_SOURCE 指定の形式と規則

(1) 機能

JAR ファイルから，Java クラスのソースファイルを抽出します。

なお，Java ルーチンは AIX，Linux，及び Windows の HiRDB で使用できます。AIX の場合は，POSIX ライブラリ版の HiRDB をセットアップ（pdsetup コマンドを実行）していないとき，又は POSIX ライブラリ版の HiRDB から非 POSIX ライブラリ版の HiRDB に再セットアップしたときは，Java ルーチンを使用できません。pdsetup コマンドについては，マニュアル「HiRDB コマンドリファレンス」を参照してください。

(2) 形式

```
GET_JAVA_STORED_ROUTINE_SOURCE（クラス名，JARファイル名〔，ソースファイル最大長〕）
```

(3) オペランド

- クラス名

クラス名は，255 バイト以下の文字列で指定してください。形式を次に示します。

```
’〔パッケージ名.〕クラス識別子’
```

クラス名には，255 バイト以下の VARCHAR 型の値式を指定できます。

クラス名に，？パラメタ又は埋込み変数だけを指定した場合，その？パラメタ又は埋込み変数は単純構造にしてください。

文字集合は既定文字集合にしてください。

- JAR ファイル名

JAR ファイル名は 255 バイト以下の文字列で指定してください。

また，JAR ファイル名には，255 バイト以下の VARCHAR 型の値式を指定できます。

JAR ファイル名は，パス名で指定しないでください。

JAR ファイル名に？パラメタ又は埋込み変数だけを指定した場合，その？パラメタ又は埋込み変数は単純構造にしてください。

文字集合は既定文字集合にしてください。

- ソースファイル最大長

抽出するソースファイルの最大長（バイト数）を整数定数で指定します。

指定できる範囲は，1～2,147,483,647 です。ナル値は指定できません。また，省略した場合は 2,147,483,647 が仮定されます。

(4) GET_JAVA_STORED_ROUTINE_SOURCE 指定の結果についての規則

1. GET_JAVA_STORED_ROUTINE_SOURCE 指定の結果は，非ナル値制約なし（ナル値を許します）の BLOB 型，定義長はソースファイル最大長で指定した長さとなります。
2. GET_JAVA_STORED_ROUTINE_SOURCE 指定の結果は，JAR ファイルから取り出したファイルの内容（ソースファイル最大長で指定した値を超える内容は切り捨て）となります。
3. 次のどれかの条件を満たす場合，GET_JAVA_STORED_ROUTINE_SOURCE 指定の結果はナル値になります。
   - 引数のうち，どれかがナル値の場合
   - 指定した JAR ファイルがインストールされていない場合
   - JAR ファイル中に指定したクラスに対応するソースファイルがない場合

(5) 共通規則

1. GET_JAVA_STORED_ROUTINE_SOURCE 指定は，次の箇所で指定できます。
   - 最も外側の問合せ指定の選択式に単独で指定
   - 最も外側の問合せ指定の選択式にスカラ関数 LENGTH の引数に指定
2. ORDER BY 句を指定した場合，ソートのキーになる項目に GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
3. WITH 句中の導出問合せ式では，選択式に GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
4. 集合演算をする場合，対象となる導出表の列に GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
5. 副問合せ（FROM 句の導出表も含む）中の選択式に GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
6. ビュー定義の導出問合せ式の選択式に GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
7. INSERT 文の問合せ指定の選択式に GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。

8. GET_JAVA_STORED_ROUTINE_SOURCE 指定は，クラス名からパッケージ名を除いたクラス識別子だけを参照します。したがって，JAR ファイル中に同じクラス識別子のソースファイルが複数ある場合，GET_JAVA_STORED_ROUTINE_SOURCE 指定の結果は複数のソースファイルが連結された形式になります。
9. GET_JAVA_STORED_ROUTINE_SOURCE 指定は，指定したクラス識別子の末尾に'.java'を付けた文字列を Java クラスのソースファイルとみなします。
10. GET_JAVA_STORED_ROUTINE_SOURCE 指定でソースファイルを取り出すためには，JAR ファイル中にソースファイルがなければなりません。JAR ファイルの作成方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(6) 留意事項

1. GET_JAVA_STORED_ROUTINE_SOURCE 指定は，HiRDB に組み込まれた JAR ファイルを対象にします。
2. GET_JAVA_STORED_ROUTINE_SOURCE 指定を含んだ SQL 文を実行するためには，Java 環境がインストールされていなければなりません。
3. クラス名，JAR ファイル名に余分な空白文字を含めないでください。

(7) 使用例

スキーマ（USER1）に登録されている Java ルーチン（JAVAROUTINE）のソースファイルの内容を抽出します。

```
SELECT GET_JAVA_STORED_ROUTINE_SOURCE(CLASS_NAME,JAR_NAME)
  FROM MASTER.SQL_ROUTINES
    WHERE ROUTINE_SCHEMA='USER1' AND ROUTINE_NAME='JAVAROUTINE'
```

2.24　SQL 最適化指定

- 機能

  SQL 文の検索効率を向上させるための最適化を，SQL 文に指定できます。

  SQL 最適化指定は，次の項目に対して指定できます。

  - 使用インデクス
  - 結合方式
  - 副問合せ実行方式

- 共通規則

  1. SQL 最適化指定を指定しても，指定が有効にならない場合があります。SQL 最適化指定が有効かどうかについては，アクセスパス表示ユティリティで確認できます。アクセスパス表示ユティリティについては，マニュアル「HiRDB コマンドリファレンス」を参照してください。
  2. SQL 最適化指定の前後を，「/*>>」と「<<*/」で囲むことができます。囲む場合は，SQL 最適化指定ごとに囲んでください。「/*>>」と「<<*/」で囲んだ場合，注釈にはなりません。「/*>>」と「<<*/」で囲んで指定すると，他 DBMS との共通 AP などでの互換性確保に有効となります。
  3. SQL 最適化指定は，SQL 最適化オプション及び SQL 拡張最適化オプションよりも優先されます。SQL 最適化オプション及び SQL 拡張最適化オプションについては，「ALTER PROCEDURE（手続きの SQL オブジェクトの再作成)」，「ALTER ROUTINE（関数，手続き，及びトリガの SQL オブジェクトの再作成)」，「ALTER TRIGGER（トリガの SQL オブジェクトの再作成)」，「CREATE〔PUBLIC〕PROCEDURE（手続き定義，パブリック手続き定義)」，又は「CREATE TRIGGER（トリガ定義)」を参照してください。

- 留意事項

  1. 結合方式の SQL 最適化指定で BY NEST を指定した場合，その結合の内表に対しての使用インデクスの SQL 最適化指定に，ネストループジョインに使用できないインデクス又はインデクス利用の抑止を指定したときは，使用インデクスの SQL 最適化指定が無効となります。
  2. 「/*>>」と「<<*/」で囲んだ SQL 最適化指定を，更に「/*>>」と「<<*/」で囲むことはできません。

2.24.1　使用インデクスの SQL 最適化指定

(1)　機能

使用インデクスの SQL 最適化指定では，表に対する検索で使用するインデクスの指定，又はインデクス利用の抑止（テーブルスキャン）を指定できます。

(2) 形式

```
使用インデクスのSQL最適化指定：：＝
        {WITH INDEX （インデクス指定〔，インデクス指定〕…） | WITHOUT INDEX}

インデクス指定：：＝｛〔認可識別子.〕インデクス識別子 | PRIMARY KEY
                    | CLUSTER KEY | PRIMARY CLUSTER KEY}
```

(3) オペランド

- WITH INDEX （インデクス指定〔，インデクス指定〕…）

使用するインデクスを指定します。複数指定した場合は，複数インデクス利用となります。

- WITHOUT INDEX

インデクス利用を抑止します（テーブルスキャン）。

- {〔認可識別子.〕インデクス識別子 | PRIMARY KEY | CLUSTER KEY | PRIMARY CLUSTER KEY}

認可識別子
: インデクスを持つユーザの認可識別子を指定します。省略した場合については，「名前の修飾」を参照してください。

インデクス識別子
: 使用するインデクスの名称を指定します。

PRIMARY KEY
: 表定義時に主キー（又は主キーで，かつクラスタキー）を指定して，定義したインデクスを使用する場合に指定します。主キー及びクラスタキーについては，「CREATE TABLE（表定義）」を参照してください。

CLUSTER KEY
: 表定義時にクラスタキー（又は主キーで，かつクラスタキー）を指定して，定義したインデクスを使用する場合に指定します。

PRIMARY CLUSTER KEY
: 表定義時に主キーで，かつクラスタキーを指定して，定義したインデクスを使用する場合に指定します。

(4) 規則

1. 内部導出表となる名前付きの導出表に対しては指定できません。内部導出表については，「内部導出表」を参照してください。
2. 2 表以上の結合，又は集合演算で導出された名前付きの導出表に対しては指定できません。
3. 使用インデクスの SQL 最適化指定を指定して導出された名前付きの導出表に対して，更に使用インデクスの SQL 最適化指定を指定した場合，後の指定が有効となり，導出問合せ式の指定は無効となります。

(5) 留意事項

1. インデクスを利用した検索については，マニュアル「HiRDB UAP 開発ガイド」の検索方式を参照してください。なお，使用するインデクスは，アクセスパス表示ユティリティで確認できます。

2.24.2 結合方式の SQL 最適化指定

(1) 機能

結合方式の SQL 最適化指定では，結合表に対して結合方式を指定できます。

(2) 形式

```
結合方式のSQL最適化指定：：＝BY {NEST | HASH | MERGE}
```

(3) オペランド

- {NEST | HASH | MERGE}

NEST

結合方式をネストループジョインにする場合に指定します。

内表が実表，又は実表を基にした名前付きの導出表の場合はネストループジョインになります。

HASH

結合方式をハッシュジョインにする場合に指定します。

MERGE

結合方式をマージジョインにする場合に指定します。

(4) 規則

1. HASH を指定する場合，SQL 拡張最適化オプションに「ハッシュジョイン，副問合せのハッシュ実行」を指定するときと同様の準備をする必要があります。ハッシュジョイン，副問合せのハッシュ実行を適用する場合の準備については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 留意事項

1. 結合方式については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。なお，結合方式は，アクセスパス表示ユティリティで確認できます。

2.24.3　副問合せ実行方式の SQL 最適化指定

(1)　機能

副問合せ実行方式の SQL 最適化指定では，述語中の副問合せに対して副問合せ実行方式を指定できます。

(2)　形式

```
副問合せ実行方式のSQL最適化指定：：＝ {HASH | NO HASH}
```

(3)　オペランド

- {HASH | NO HASH}

HASH

副問合せの実行方式を，ハッシュ実行にする場合に指定します。

NO HASH

副問合せの実行方式を，ハッシュ実行以外にする場合に指定します。

(4)　規則

1. 副問合せ実行方式の SQL 最適化指定は，述語中の副問合せに指定してください。
2. HASH を指定する場合，SQL 拡張最適化オプションに「ハッシュジョイン，副問合せのハッシュ実行」を指定するときと同様の準備をする必要があります。ハッシュジョイン，副問合せのハッシュ実行を適用する場合の準備については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5)　留意事項

1. 副問合せの実行方式については，マニュアル「HiRDB UAP 開発ガイド」の外への参照のない副問合せの実行方式，又は外への参照のある副問合せの実行方式を参照してください。なお，副問合せの実行方式は，アクセスパス表示ユティリティで確認できます。

2.24.4　SQL 最適化指定の例

SQL 最適化指定の使用例を次に示します。

1. SELECT 文中に，使用インデクスの SQL 最適化指定を指定します。このとき，在庫表（ZAIKO）の検索に，インデクス（IDX1）を使用します。

```
SELECT SNAME FROM ZAIKO WITH INDEX (IDX1)
  WHERE TANKA <= 500
```

2. SELECT 文中に，使用インデクスの SQL 最適化指定を指定します。このとき，在庫表（ZAIKO）の検索に，複数のインデクス（IDX1，IDX2）を使用します。

```
SELECT SNAME FROM ZAIKO WITH INDEX (IDX1,IDX2)
  WHERE TANKA <= 500 OR ZSURYO > 100
```

3. SELECT 文中に，使用インデクスの SQL 最適化指定を指定します。このとき，在庫表（ZAIKO）定義時に，PRIMARY KEY を指定して定義されたインデクスを使用します。

```
SELECT SNAME FROM ZAIKO WITH INDEX (PRIMARY KEY)
  WHERE TANKA <= 500
```

4. SELECT 文中に，使用インデクスの SQL 最適化指定を指定します。このとき，在庫表（ZAIKO）の検索に，インデクス利用を抑止（テーブルスキャン）します。

```
SELECT SNAME FROM ZAIKO WITHOUT INDEX
  WHERE TANKA <= 500
```

5. SELECT 文中に，結合方式の SQL 最適化指定を指定します。このとき，内結合の結合方式を，ネストループジョインにします。

```
SELECT ZAIKO.SCODE,ZAIKO.SNAME,JUTYU.TCODE
  FROM ZAIKO INNER JOIN BY NEST JUTYU
    ON ZAIKO.SCODE = JUTYU.SCODE
```

6. SELECT 文中に，結合方式の SQL 最適化指定を指定します。このとき，外結合の結合方式を，ハッシュジョインにします。

```
SELECT ZAIKO.SCODE,ZAIKO.SNAME,JUTYU.TCODE
  FROM ZAIKO LEFT OUTER JOIN BY HASH JUTYU
    ON ZAIKO.SCODE = JUTYU.SCODE
```

7. SELECT 文中に，結合方式の SQL 最適化指定を指定します。このとき，内結合（INNER 省略）の結合方式を，マージジョインにします。

```
SELECT ZAIKO.SCODE,ZAIKO.SNAME,JUTYU.TCODE
  FROM ZAIKO JOIN BY MERGE JUTYU
    ON ZAIKO.SCODE = JUTYU.SCODE
```

8. SELECT 文中に，副問合せ実行方式の SQL 最適化指定を指定します。このとき，副問合せの実行方式を，ハッシュ実行にします。

```
SELECT SNAME FROM ZAIKO
  WHERE SCODE =ANY
    (HASH SELECT SCODE FROM JUTYU
       WHERE TCODE = '302S')
```

9. SELECT 文中に，副問合せ実行方式の SQL 最適化指定を指定します。このとき，副問合せの実行方式を，ハッシュ実行以外（この例では，作業表実行，又は作業表 ATS 実行となる）にします。

```
SELECT SNAME FROM ZAIKO
  WHERE SCODE =ANY
    (NO HASH SELECT SCODE FROM JUTYU
      WHERE TCODE = '302S')
```

10. 使用インデクスの SQL 最適化指定を，「/*>>」と「<<*/」で囲んで指定します。

```
SELECT SNAME FROM ZAIKO /*>> WITH INDEX (IDX1) <<*/
  WHERE TANKA <= 500
```

11. 結合方式の SQL 最適化指定を，「/*>>」と「<<*/」で囲んで指定します。

```
SELECT ZAIKO.SCODE,ZAIKO.SNAME,JUTYU.TCODE
  FROM ZAIKO INNER JOIN /*>> BY NEST <<*/ JUTYU
    ON ZAIKO.SCODE = JUTYU.SCODE
```

12. 副問合せ実行方式の SQL 最適化指定を，「/*>>」と「<<*/」で囲んで指定します。

```
SELECT SNAME FROM ZAIKO
  WHERE SCODE =ANY
    (/*>> HASH <<*/ SELECT SCODE FROM JUTYU
      WHERE TCODE = '302S')
```

13. SELECT 文中に，結合方式の SQL 最適化指定と使用インデクスの SQL 最適化指定を指定します。このとき，在庫表（ZAIKO）の検索にインデクス（IDX3）を使用し，内結合（INNER 省略）の結合方式をハッシュジョインにします。

```
SELECT ZAIKO.SCODE,ZAIKO.SNAME,JUTYU.TCODE
  FROM ZAIKO WITH INDEX (IDX3) JOIN BY HASH JUTYU
    ON ZAIKO.SCODE = JUTYU.SCODE
```

14. SELECT 文中に，結合方式の SQL 最適化指定と使用インデクスの SQL 最適化指定を，それぞれ「/*>>」と「<<*/」で囲んで指定します。このとき，在庫表（ZAIKO）の検索にインデクス(IDX3)を使用し，内結合（INNER 省略）の結合方式をハッシュジョインにします。

```
SELECT ZAIKO.SCODE,ZAIKO.SNAME,JUTYU.TCODE
  FROM ZAIKO /*>> WITH INDEX (IDX3) <<*/ JOIN /*>> BY HASH <<*/ JUTYU
    ON ZAIKO.SCODE = JUTYU.SCODE
```

2.25 CAST 指定

2.25.1 CAST 指定の形式と規則

(1) 機能

値式のデータを，指定したデータ型に変換します。

(2) 形式

```
CAST指定：：＝CAST（｛値式｜NULL｝ AS データ型）
```

(3) 規則

1. 値式には，次のデータ型は指定できません。
   - BLOB
   - 定義長が 32,001 バイト以上の BINARY
   - 抽象データ型
2. AS データ型には，次のデータ型は指定できません。
   - BLOB
     ただし，CAST(NULL AS BLOB)は指定できます。
   - 長さが 32,001 バイト以上の BINARY
     ただし，CAST(NULL AS BINARY(n))は指定できます。n は 32,001 以上のバイト数です。
   - BOOLEAN
   - 抽象データ型
3. 結果の値は，非ナル値制約なし（ナル値を許します）になります。
4. 値式に NULL を指定するか，又は値式の結果がナル値の場合，結果の値はナル値になります。
5. 実長 0 バイト，又は実長 0 文字のデータを値式に指定した場合，文字型への変換はデータ変換の規則に従います。文字型以外への変換では，エラーになります。
6. 値式に埋込み変数又は?パラメタを単独で指定した場合，埋込み変数又は?パラメタのデータ型は，AS データ型に指定したデータ型が仮定されます。
7. 値式に繰返し列を指定する場合，添字を指定してください。ただし，添字に ANY は指定できません。
8. 値式に予備列は指定できません。
9. AS データ型には，値式に指定するデータ型と型変換できるデータ型を指定してください。データ型の変換可否を次の表に示します。

表 2-111　値式の結果のデータ型と AS データ型との，データ型の変換可否（1/2）

値式の結果のデータ型			AS データ型						
			数データ		文字データ（文字集合）			各国文字データ	混在文字データ
			真数	概数					
			INTEGER, SMALLINT, DECIMAL	FLOAT, SMALLFLT	CHAR, VARCHAR			NCHAR, NVARCHAR	MCHAR, MVARCHAR
					DF	EK	U16		
数データ	真数	INTEGER, SMALLINT, DECIMAL	○	○	○	○	○	×	○
	概数	FLOAT, SMALLFLT	○	○	○	○	○	×	○
文字データ（文字集合）		CHAR, VARCHAR（DF）	○	○	○	○※2	○	×	○
		CHAR, VARCHAR（EK）	○	○	○※2	○	×	×	×
		CHAR, VARCHAR（U16）	○	○	○	×	○	×	○
各国文字データ		NCHAR, NVARCHAR	×	×	×	×	×	○	×
混在文字データ		MCHAR, MVARCHAR	○	○	○	×	○	×	○
論理データ		BOOLEAN	×	×	○	○	○	×	○
日付データ		DATE	×	×	○※1	○※1	○※1	×	○※1
時刻データ		TIME	×	×	○※1	○※1	○※1	×	○※1
時刻印データ		TIMESTAMP	×	×	○※1	○※1	○※1	×	○※1
日間隔データ		INTERVAL YEAR TO DAY	○	×	×	×	×	×	×
時間隔データ		INTERVAL HOUR TO SECOND	○	×	×	×	×	×	×

値式の結果のデータ型		AS データ型						
		数データ		文字データ（文字集合）			各国文字データ	混在文字データ
		真数	概数					
		INTEGER, SMALLINT, DECIMAL	FLOAT, SMALLFLT	CHAR, VARCHAR			NCHAR, NVARCHAR	MCHAR, MVARCHAR
				DF	EK	U16		
バイナリデータ	BINARY	×	×	○	○	○	×	×

（凡例）

○：データ変換ができます。

×：データ変換ができません。

DF：既定文字集合

EK：EBCDIK

U16：UTF16

注※1

AS データ型に指定した長さが，次に示す長さ以上の場合は変換できます。

・値式の結果の文字集合が UTF16 以外の場合

DATE の場合：

10 バイト

TIME の場合：

8 バイト

TIMESTAMP(p)の場合：

p=0 のときは 19 バイト（ピリオドは付きません）

p>0 のときは 20+↓(p+1)/2↓×2 バイト

・値式の結果の文字集合が UTF16 の場合

DATE の場合：

20 バイト

TIME の場合：

16 バイト

TIMESTAMP(p)の場合：

p=0 のときは 38 バイト（ピリオドは付きません）

p>0 のときは 40+↓(p+1)/2↓×4 バイト

AS データ型に指定した長さが，上記の長さより短い場合は変換できません。

固定長の文字データ型に変換する場合で，AS データ型に指定した長さが上記の長さより長い場合は，左詰めにして，末尾をその文字集合の空白で埋めます。

注※2

既定文字集合の SJIS コードを JIS8 コードの 1 バイトコードとみなして，JIS8 コードと EBCDIK コードとの間の変換を行います。

文字コード変換規則については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

表 2-112　値式の結果のデータ型と AS データ型との，データ型の変換可否（2/2）

値式の結果のデータ型			AS データ型					
			日付データ	時刻データ	時刻印データ	日間隔データ	時間隔データ	バイナリデータ
			DATE	TIME	TIMESTAMP	INTERVAL YEAR TO DAY	INTERVAL HOUR TO SECOND	BINARY
数データ	真数	INTEGER, SMALLINT, DECIMAL	×	×	×	○	○	×
	概数	FLOAT, SMALLFLT	×	×	×	×	×	×
文字データ		CHAR, VARCHAR（DF）	○	○	○	×	×	○
		CHAR, VARCHAR（EK）	○	○	○	×	×	○
		CHAR, VARCHAR（U16）	○	○	○	×	×	○
各国文字データ		NCHAR, NVARCHAR	×	×	×	×	×	×
混在文字データ		MCHAR, MVARCHAR	○	○	○	×	×	×
論理データ		BOOLEAN	×	×	×	×	×	×
日付データ		DATE	○	×	○	×	×	×
時刻データ		TIME	×	○	○	×	×	×
時刻印データ		TIMESTAMP	○	○	○	×	×	×
日間隔データ		INTERVAL YEAR TO DAY	×	×	×	○	○	×

値式の結果のデータ型		AS データ型					
		日付データ	時刻データ	時刻印データ	日間隔データ	時間隔データ	バイナリデータ
		DATE	TIME	TIMESTAMP	INTERVAL YEAR TO DAY	INTERVAL HOUR TO SECOND	BINARY
時間隔データ	INTERVAL HOUR TO SECOND	×	×	×	○	○	×
バイナリデータ	BINARY	×	×	×	×	×	○

（凡例）

○：データ変換ができます。

×：データ変換ができません。

DF：既定文字集合

EK：EBCDIK

U16：UTF16

10. ウィンドウ関数は指定できません。

(4) 結果のデータ型ごとの変換規則

(a) 数データ

- 値式の結果が数データの場合

  変換対象のデータ型で，上位有効けたを失ってはなりません。

- 値式の結果が文字列，及び混在文字列の場合

  データの前後の半角空白を取り除いた結果が，数定数の文字列表現でなければなりません。数定数の文字列表現を数値に変換した後，値式の結果が数データの規則に従います。

- 値式の結果が日間隔データ，及び時間隔データの場合

  結果のデータ型（DECIMAL）の精度及び位取りが次の表の場合，日間隔データ，又は時間隔データの10進数形式に変換します。

表 2-113 日間隔データ，及び時間隔データから数値データへの変換規則

値式の結果のデータ型	AS データ型に指定した長さ
INTERVAL YEAR TO DAY	精度 8，位取り 0
INTERVAL HOUR TO SECOND	精度 6，位取り 0

(b) 文字データ，及び混在文字データ

- 文字データ，及び混在文字データに変換する場合の変換規則を次の表に示します。

表 2-114 文字データ，及び混在文字データへの変換規則

値式の結果の長さと，AS データ型の長さの関係	値式の結果のデータ型	
	文字データ，及び混在文字データ	文字データ，及び混在文字データ以外
値式の結果の長さ＜ AS データ型に指定した長さ	AS データ型に指定したデータ型が固定長の場合，左詰めにして末尾を空白（結果のデータ型の文字集合の空白）で埋めます。	
値式の結果の長さ＝ AS データ型に指定した長さ	正常に変換します。	
値式の結果の長さ＞ AS データ型に指定した長さ	左詰めにして末尾を切り捨てます。切り捨てる部分のデータに空白（結果のデータ型の文字集合の空白）以外の文字があれば，SQLWARN1 領域に'W'を設定します。※	エラーになります。

注※

マルチバイト文字の途中で切り捨てが発生する場合，マルチバイト文字の一部分が結果の値として返されます。

- 値式の結果が真数（SMALLINT，INTEGER，及び DECIMAL）の場合

  データを数定数に変換した結果，最短となる（先頭の 0 は取り除く）文字列が結果となります。

  データが 0 未満の場合は，先頭に負符号（-）を付加します。

- 値式の結果が概数（SMALLFLT，及び FLOAT）の場合

  データを数定数に変換した結果，最短となる文字列（先頭は 0 以外です。ただし，データが 0 の場合は 0E0 となります）が結果となります。

  データが 0 未満の場合は，先頭に負符号（-）を付加します。

- 値式の結果が文字列，混在文字列の場合

  値式の結果のデータ長が，AS データ型に指定した長さを超える場合，左詰めにして末尾を切り捨てます。切り捨てる部分のデータ中に空白（結果のデータ型の文字集合の空白）以外の文字が含まれていると，SQLWARN1 領域に'W'が設定されます。

  また，値式の結果が文字集合指定をした文字データの場合，値式のすべての文字をその文字集合の文字の並びとして変換します。

- 値式の結果が論理データの場合

  値式の結果が真の場合は変換結果は'TRUE'，値式の結果が偽の場合は変換結果は'FALSE'，値式の結果が不定の場合は変換結果はナル値となります。

- 値式の結果が日付データ，時刻データ，及び時刻印データの場合

  日付データ，時刻データ，及び時刻印データの規定の文字列表現に変換します。

- 値式の結果がバイナリデータの場合

値式の結果のデータ長が，結果のデータ型に指定したデータ長を超える場合，SQLWARN1 領域に'W'が設定されます。

(c) 各国文字データ

- 値式の結果が各国文字列の場合

  結果のデータ型が固定長で，値式のデータを各国文字列に変換した結果，AS データ型に指定した長さより短い場合は，文字コードに対応した全角空白で埋めます。

  値式のデータを各国文字列に変換した結果，結果のデータ型に指定したデータ長を超える場合は，左詰めにして末尾を切り捨てます。切り捨てる部分のデータ中に文字コードに対応した全角空白以外の文字が含まれていると，SQLWARN1 領域に'W'が設定されます。

(d) 日付データ，時刻データ，及び時刻印データ

- 値式の結果が文字列，及び混在文字列の場合

  日付データ，時刻データ，及び時刻印データの，規定の文字列表現であれば変換できます。ただし，値式のデータの前後の空白（結果のデータ型の文字集合の空白）を取り除き，その結果の小数秒精度が 0，2，4，及び 6 でない場合はエラーとなります。

- 値式の結果が日付データ，時刻データ，及び時刻印データの場合

  次の表に示す組み合わせで変換できます。

表 2-115 日付データ，時刻データ，及び時刻印データの変換規則

値式の結果のデータ型	AS データ型	変換規則
DATE	DATE	変換しません。
	TIMESTAMP(p2)	時刻部分を'00:00:00'として変換します。小数秒部は 0 で埋めます。
TIME	TIME	変換しません。
	TIMESTAMP(p2)	日付部分を CURRENT_DATE に変換します。小数秒部は 0 で埋めます。
TIMESTAMP(p1)	DATE	日付部分を抽出して変換します。
	TIME	時刻部分を抽出して変換します。
	TIMESTAMP(p2)	変換しません。 ただし，p1>p2 の場合は小数秒部を切り捨て，p1<p2 の場合は小数秒部を 0 で埋めます。

(e) 日間隔データ，及び時間隔データ

- 値式の結果が DECIMAL の場合

  精度及び位取りが次の表の場合に，対応する日間隔データ，又は時間隔データに変換できます。

表 2-116　数データから日間隔データ，又は時間隔データへの変換規則

数データの形式	値式の結果のデータ型
精度 8，位取り 0	INTERVAL YEAR TO DAY
精度 6，位取り 0	INTERVAL HOUR TO SECOND

- 値式の結果が日間隔データ，及び時間隔データの場合

  次の表に示す組み合わせで変換できます。

表 2-117　日間隔データ，及び時間隔データの変換規則

値式の結果のデータ型	AS データ型	変換規則
INTERVAL YEAR TO DAY	INTERVAL YEAR TO DAY	変換しません。
	INTERVAL HOUR TO SECOND	日付部分を時刻部分に変換し，INTERVAL HOUR TO SECOND の値の範囲を超えなければ変換できます。超える場合はエラーとなります。
INTERVAL HOUR TO SECOND	INTERVAL YEAR TO DAY	時刻部分が 24 時間以上の場合に，日付部分に繰り上げます。24 時間未満のデータは切り捨てます。
	INTERVAL HOUR TO SECOND	変換しません。

(f)　バイナリデータ

- 値式の結果が文字列の場合

  値式のデータを BINARY に変換した結果，AS データ型に指定した長さを超える場合，SQLWARN1 領域に'W'が設定されます。

2.26　拡張文名

2.26.1　拡張文名の形式と規則

(1)　概要

拡張文名は PREPARE 文に指定することで，PREPARE 文によって前処理された SQL 文を識別します。また，次に示す SQL 文に指定すると拡張文名が識別する SQL 文に対する操作ができます。

- DESCRIBE 文
- DESCRIBE TYPE 文
- ALLOCATE CURSOR 文
- EXECUTE 文
- DEALLOCATE PREPARE 文

(2)　形式

```
拡張文名：：＝有効範囲オプション　：埋込み変数
有効範囲オプション：：＝GLOBAL
```

(3)　説明

- **有効範囲オプション**

  拡張文名の有効範囲を指定します。指定による有効範囲を次の表に示します。

  表 2-118　有効範囲オプションの指定方法

指定	有効範囲
GLOBAL	現行 SQL セッション内（HiRDB 接続から切断まで）で有効になります。

- **：埋込み変数**

  SQL 文識別子を値に持つ可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）の埋込み変数を指定します。

  既定文字集合以外の文字集合は指定できません。

(4)　規則

1. 埋込み変数に設定する値は，SQL 文識別子の指定方法に従ってください。SQL 文識別子の指定方法は，「名前の指定」を参照してください。
2. 同じ値を持つ有効な拡張文名は，同じ拡張文名として判別されます。

3. 拡張文名の値が SQL 文中に直接指定した SQL 文識別子と同じ場合でも，それらは異なる SQL 文を識別するものとして区別されます。

2.27　拡張カーソル名

2.27.1　拡張カーソル名の形式と規則

(1)　概要

拡張カーソル名は ALLOCATE CURSOR 文に指定することで，動的 SELECT 文，又は手続きから返却された結果集合の組に割り当てられたカーソルを識別します。また，次に示す SQL 文に指定することで拡張カーソル名が識別するカーソルに対する操作ができます。

- DESCRIBE CURSOR 文
- OPEN 文
- FETCH 文
- DELETE 文（結果集合カーソルを識別する拡張カーソル名を除く）
- UPDATE 文（結果集合カーソルを識別する拡張カーソル名を除く）
- CLOSE 文

また，拡張カーソル名の値を次に示す SQL 文に直接指定することで拡張カーソル名が識別するカーソルに対する操作ができます。

- 準備可能動的 DELETE 文：位置付け（結果集合カーソルを識別する拡張カーソル名の値を除く）
- 準備可能動的 UPDATE 文：位置付け（結果集合カーソルを識別する拡張カーソル名の値を除く）

(2)　形式

```
拡張カーソル名：：＝ 有効範囲オプション ：埋込み変数
有効範囲オプション：：＝GLOBAL
```

(3)　説明

- **有効範囲オプション**

  拡張文名の有効範囲を指定します。指定による有効範囲を次の表に示します。

  表 2-119　有効範囲オプションの指定方法

指定	有効範囲
GLOBAL	現行 SQL セッション内（HiRDB 接続から切断まで）で有効となります。

- **：埋込み変数**

  カーソル名を値に持つ可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）の埋込み変数を指定します。

既定文字集合以外の文字集合は指定できません。

(4) 規則

1. 埋込み変数に設定する値は，カーソル名の指定方法に従ってください。カーソル名の指定方法は，「名前の指定」を参照してください。
2. 同じ値を持つ拡張カーソル名は，同じ拡張カーソル名として判別されます。
3. ALLOCATE CURSOR 文に指定する場合に，既に同じ値を持つ有効な拡張カーソル名がほかのカーソルを識別している場合はエラーとなります。
4. 拡張カーソル名の値が SQL 文中に直接指定したカーソル名と同じであっても，それらは異なるカーソルを識別するものとして区別します。ただし，準備可能動的 UPDATE 文：位置付け，及び準備可能動的 DELETE 文：位置付けに直接指定したカーソル名の場合だけ，その値を持つ拡張カーソル名と同じカーソルを識別します。

2.28 NEXT VALUE 式

2.28.1 NEXT VALUE 式の形式と規則

(1) 概要

順序数生成子が生成する値を返却します。

(2) 使用権限

FOR PUBLIC USAGE を指定して定義した順序数生成子には，すべてのユーザがアクセスできます。それ以外の順序数生成子には，所有者だけがアクセスできます。

(3) 形式

```
NEXT VALUE式 : := NEXT VALUE FOR 〔認可識別子.〕順序数生成子識別子
```

(4) 説明

- **認可識別子**
  順序数生成子の所有者の認可識別子を指定します。
- **順序数生成子識別子**
  使用する順序数生成子の識別子を指定します。

(5) 構文規則

1. NEXT VALUE 式の結果のデータ型は，指定した順序数生成子が生成する値のデータ型になります。
2. 次の箇所には，副問合せを指定しないで NEXT VALUE 式を指定できます。
   - INSERT 文の問合せ指定の選択式
   - INSERT 文の挿入値
   - UPDATE 文の更新値

   ただし，次に示す箇所には指定できません。
   - CASE 式
   - スカラ関数 VALUE 中
   - GROUP BY 句，HAVING 句，又は集合関数を指定した問合せ指定
   - WINDOW 関数を指定した問合せ指定
   - DISTINCT を含む問合せ指定

- UNION ALL 以外の集合演算の演算項となる問合せ指定

3. NEXT VALUE 式の結果は非ナル値制約なし（ナル値を許す）となります。

(6) 共通規則

1. 順序数生成子を定義した後で，最初に NEXT VALUE 式を実行した場合は開始値が返却されます。
2. 挿入値に指定された場合，INSERT 文で挿入される 1 行ごとに，指定された順序数生成子が生成した値が挿入値となります。
3. 更新値に指定された場合，UPDATE 文で更新される 1 行ごとに，指定された順序数生成子が生成した値が更新値となります。
4. 同一の行に対して同じ順序数生成子に対する NEXT VALUE 式を複数指定した場合，それらの NEXT VALUE 式はすべて同じ値を返却します。

(7) 留意事項

1. ロールバックが発生してトランザクションが無効となった場合でも，順序数生成子の値は元に戻りません。
2. NEXT VALUE 式を指定した SQL の実行結果がエラーとなった場合でも，順序数生成子の値が更新されていることがあります。

(8) 使用例

1. 商品 ID（SID），商品名（SNAME），単価（TANKA）を持つ在庫表（ZAIKO）を定義します。

```
CREATE TABLE ZAIKO(
SID INTEGER,
SNAME NCHAR(8),
TANKA INTEGER)
```

2. 商品 ID を 1000 から 9999 まで発番する順序数生成子 SEQ1 を定義します。

```
CREATE SEQUENCE SEQ1
START WITH 1000
INCREMENT BY 1
MAXVALUE 9999
```

```
NO CYCLE
```

3. 在庫表（ZAIKO）に新たな商品を登録します。

```
INSERT INTO ZAIKO VALUES(NEXT VALUE FOR SEQ1, N’ズボン’, 1200)
INSERT INTO ZAIKO VALUES(NEXT VALUE FOR SEQ1, N’シャツ’, 1000)
INSERT INTO ZAIKO VALUES(NEXT VALUE FOR SEQ1, N’セーター’, 1500)
```

実行結果

在庫表（ZAIKO）

商品ID（SID）	商品名（SNAME）	単価（TANKA）
1000	ズボン	1200
1001	シャツ	1000
1002	セーター	1500

4. 在庫表（ZAIKO）の商品 ID（SID）が”1001”の商品 ID を新しい商品 ID に更新します。

```
UPDATE ZAIKO SET SID = NEXT VALUE FOR SEQ1 WHERE SID = 1001
```

実行結果

在庫表（ZAIKO）

商品ID（SID）	商品名（SNAME）	単価（TANKA）
1000	ズボン	1200
1003	シャツ	1000
1002	セーター	1500

3 定義系 SQL

この章では，定義系 SQL の構文形式，及び文法について説明します。

3.1　全般規定

3.1.1　定義系 SQL の全般規定

(1)　定義系 SQL の種類と機能

定義系 SQL は，スキーマを定義・変更したり，表，インデクス，及び権限を定義・削除したりするときに使用する SQL です。

定義系 SQL の種類と機能を次の表に示します。

表 3-1　定義系 SQL の種類と機能

種　類	機　能
ALTER INDEX（インデクス定義変更）	インデクスの名称を変更します。
ALTER PROCEDURE（手続きの SQL オブジェクトの再作成）	手続きの SQL オブジェクトを再作成します。
ALTER ROUTINE（関数，手続き，及びトリガの SQL オブジェクトの再作成）	関数，手続き，及びトリガの SQL オブジェクトを再作成します。
ALTER TABLE（表定義変更）	表定義を変更します。
ALTER TRIGGER（トリガの SQL オブジェクトの再作成）	トリガの SQL オブジェクトを再作成します。
COMMENT（注釈付加）	表及び列に注釈を付けます。
CREATE AUDIT（監査対象イベントの定義）	監査証跡として記録する監査イベント，及びその対象を定義します。
CREATE CONNECTION SECURITY（CONNECT 関連セキュリティ機能の定義）	CONNECT 関連セキュリティ機能に関するセキュリティ項目を定義します。
CREATE FUNCTION（関数定義）	関数を定義します。
CREATE PUBLIC FUNCTION（パブリック関数定義）	パブリック関数を定義します。
CREATE INDEX　形式 1（インデクス定義）	実表の列にインデクス（昇順，降順）を定義します。
CREATE INDEX　形式 2（インデクス定義）	
CREATE INDEX　形式 3（部分構造インデクス定義）	
CREATE PROCEDURE（手続き定義）	手続きを定義します。
CREATE PUBLIC PROCEDURE（パブリック手続き定義）	パブリック手続きを定義します。
CREATE SCHEMA（スキーマ定義）	スキーマを定義します。
CREATE SEQUENCE（順序数生成子定義）	順序数生成子を定義します。
CREATE TABLE（表定義）	実表を定義します。
CREATE TRIGGER（トリガ定義）	トリガを定義します。

種　類	機　能
CREATE TYPE（型定義）	抽象データ型を定義します。
CREATE VIEW（ビュー定義）	ビュー表を定義します。
CREATE PUBLIC VIEW（パブリックビュー定義）	パブリックビューを定義します。
DROP AUDIT（監査対象イベントの削除）	CREATE AUDIT で定義した監査対象と内容が一致する定義を，監査対象から削除します。
DROP CONNECTION SECURITY（CONNECT 関連セキュリティ機能の削除）	CONNECT 関連セキュリティ機能に関するセキュリティ項目を削除します。
DROP DATA TYPE（ユーザ定義型削除）	ユーザ定義型を削除します。
DROP FUNCTION（関数削除）	関数を削除します。
DROP PUBLIC FUNCTION（パブリック関数削除）	パブリック関数を削除します
DROP INDEX（インデクス削除）	インデクスを削除します。
DROP PROCEDURE（手続き削除）	手続きを削除します。
DROP PUBLIC PROCEDURE（パブリック手続き削除）	パブリック手続きを削除します。
DROP SCHEMA（スキーマ削除）	スキーマを削除します。
DROP SEQUENCE（順序数生成子削除）	順序数生成子を削除します。
DROP TABLE（表削除）	実表を削除します。さらに，その実表に対するインデクス，注釈，アクセス権限及びビュー表も削除します。
DROP TRIGGER（トリガ削除）	トリガを削除します。
DROP VIEW（ビュー表の削除）	ビュー表を削除します。
DROP PUBLIC VIEW（パブリックビューの削除）	パブリックビューを削除します。
GRANT CONNECT（CONNECT 権限定義）	ユーザに CONNECT 権限を与えます。
GRANT DBA（DBA 権限定義）	ユーザに DBA 権限を与えます。
GRANT RDAREA（RD エリア利用権限定義）	ユーザに RD エリアの利用権限を与えます。
GRANT SCHEMA（スキーマ定義権限定義）	ユーザにスキーマ定義権限を与えます。
GRANT SCHEMA OPERATION（スキーマ操作権限定義）	ユーザにスキーマ操作権限を与えます。
GRANT アクセス権限（アクセス権限定義）	ユーザにアクセス権限を与えます。
GRANT　形式 2（監査人のパスワード変更）	監査人のパスワードを変更します。
REVOKE CONNECT（CONNECT 権限削除）	ユーザに与えた CONNECT 権限を取り消します。
REVOKE DBA（DBA 権限削除）	ユーザに与えた DBA 権限を取り消します。
REVOKE RDAREA（RD エリア利用権限削除）	ユーザに与えた RD エリアの利用権限を取り消します。
REVOKE SCHEMA（スキーマ定義権限削除）	ユーザに与えたスキーマ定義権限を取り消します。
REVOKE SCHEMA OPERATION（スキーマ操作権限削除）	ユーザに与えたスキーマ操作権限を取り消します。

種　類	機　能
REVOKE アクセス権限（アクセス権限削除）	ユーザに与えたアクセス権限を取り消します。

(2)　共通規則

定義系 SQL が正常に実行された場合，処理完了と同時に COMMIT されます。

(3)　留意事項

定義系 SQL は，OLTP 下の X/Open に従った UAP から指定できません。

リードレプリカ機能を使用する場合，定義系 SQL は更新 DB で実行します。参照 DB では定義系 SQL を実行できません。リードレプリカ機能の詳細は，マニュアル「HiRDB システム運用ガイド」の「リードレプリカ機能の運用」を参照してください。

(4)　定義系 SQL 実行時に必要な権限について

スキーマ所有者は自分のリソースを定義できます。ただし，スキーマ操作権限を持つユーザは，スキーマ操作ができる他人のスキーマ内リソースを定義することができます。このため，スキーマ操作権限を利用して定義系 SQL を実行する場合は，各定義系 SQL の規定の説明にある「自分」及び「実行ユーザ」の記載を「スキーマ操作ができるスキーマ」と読み替えてください。

スキーマ操作権限の使用例については，マニュアル「HiRDB システム運用ガイド」の「他ユーザにスキーマ操作権限を与えます」及び「スキーマ操作権限を取り消すには」を参照してください。スキーマ操作権限を使用して操作できるリソースの一覧は，マニュアル「HiRDB システム運用ガイド」の「ユーザ権限の種類」の「スキーマ操作権限」を参照してください。

また，上記以外の権限によって実行できる定義系 SQL については，各定義系 SQL の使用権限を確認してください。

3.2 ALTER INDEX（インデクス定義変更）

3.2.1 ALTER INDEX の形式と規則

(1) 機能

CREATE INDEX で定義したインデクスの名称を変更します。

(2) 使用権限

インデクスの所有者

自分が所有するインデクスだけ指定できます。

(3) 形式

```
ALTER INDEX 〔認可識別子.〕インデクス識別子 インデクス定義変更動作

  インデクス定義変更動作：：＝ インデクス名変更定義

  インデクス名変更定義：：＝RENAME INDEX TO インデクス識別子 〔WITH PROGRAM〕
```

(4) オペランド

(a) 〔認可識別子.〕インデクス識別子

認可識別子

定義を変更するインデクスの所有者の認可識別子を指定します。

インデクス識別子

定義を変更するインデクスの名称を指定します。

(b) インデクス名変更定義：：＝ RENAME INDEX TO インデクス識別子 〔WITH PROGRAM〕

インデクスの名称を変更する場合に指定します。

インデクス識別子

変更後のインデクスの名称を指定します。

インデクス識別子についての規則を次に示します。

1. CREATE INDEX で定義した B-tree インデクス，インデクス型を指定したインデクス，及び部分構造インデクスの名称を変更できます。

2. スキーマ内にあるインデクス，インデクス型を指定したインデクス，及び部分構造インデクスと同じ名称には変更できません。

3. 一時インデクスの名称は指定できません。

WITH PROGRAM

インデクスの名称を変更するときに，そのインデクスを使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

WITH PROGRAM を省略した場合，そのインデクスを使用する手続き，及びトリガの有効な SQL オブジェクトがあると，そのインデクスの定義は変更できません。

(5) 留意事項

1. Java 手続き中では定義系 SQL が実行できますが，Java 手続きを呼び出す手続きが，インデクスを使用する SQL を発行している場合，呼び出された Java 手続き内でそのインデクスに対し ALTER INDEX を発行していると ALTER INDEX はエラーになります。

2. ALTER INDEX は，OLTP 下の X/Open に従った UAP から指定できません。

3. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して，手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。

4. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，無効となった手続き，及びトリガに関する SQL_ROUTINE_RESOURCES 表，SQL_TRIGGER_USAGE 表，SQL_TRIGGER_COLUMNS 表，及び SQL_ROUTINE_PARAMS 表の行は削除します。

5. ALTER INDEX 対象のインデクスを使用しないで，そのインデクスを定義している表をアクセスしている手続き，又はトリガが存在する場合，ALTER INDEX を実行すると，該当する手続き，又はトリガの SQL オブジェクト中のインデクス情報は無効になります。この場合，インデクス情報が無効な手続きの呼び出しは実行できますが，実行するたびにリコンパイルされるため性能が劣化します。また，インデクス情報が無効となった手続きを呼び出す手続き，又はトリガは実行できなくなります。そのため，インデクス情報が無効となった手続き，及びトリガに対して ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して，SQL オブジェクトを再作成する必要があります。なお，インデクス情報が無効かどうかはディクショナリ表 SQL_ROUTINES の INDEX_VALID 列を参照して確認できます。

6. データベース作成ユティリティ（pdload），及びデータベース再編成ユティリティ（pdrorg）を実行して，インデクス情報出力モードでインデクスを作成した場合，インデクスが作成される前に ALTER INDEX を使用してインデクスの名称を変更すると，出力済みのインデクス情報ファイルではインデクスの一括作成ができなくなります。なお，誤って先にインデクスの名称を変更した場合は，ALTER INDEX でインデクスの名称を元に戻してからインデクスの一括作成を行い，その後で再度インデクスの名称を変更してください。

7. プラグインインデクスの遅延一括作成を行う場合，インデクスが作成される前に ALTER INDEX を使用してプラグインインデクスの名称を変更すると，出力済みのインデクス情報ファイルではプラグインインデクスの一括作成ができなくなります。なお，誤って先にプラグインインデクスの名称を変更した

場合は，プラグインインデクスの名称を元に戻してからプラグインインデクスの一括作成を行い，その後で再度プラグインインデクスの名称を変更してください。

8. 未完状態のインデクス（EMPTY 指定で CREATE INDEX した直後のインデクス）の名称を ALTER INDEX で変更できます。

(6) 使用例

インデクス（IDX1）の名称を（IDX2）に変更します。

```
ALTER INDEX IDX1 RENAME INDEX TO IDX2
```

3.3 ALTER PROCEDURE（手続きの SQL オブジェクトの再作成）

3.3.1 ALTER PROCEDURE の形式と規則

(1) 機能

SQL 手続きの SQL オブジェクトを再作成します。また，Java 手続きのコンパイルオプションを変更します。

(2) 使用権限

手続きの所有者

自分が所有する手続き（自分が定義したパブリック手続きを含む）の SQL オブジェクトを再作成できます。

- AUTHORIZATION 句には自分の認可識別子だけ指定できます。
- ルーチン識別子には自分が所有する手続きだけ指定できます。
- AUTHORIZATION 句，及びルーチン識別子を両方省略した場合，エラーとなります。

DBA 権限を持つユーザ

自分が所有する手続き（自分が定義したパブリック手続きを含む）と，他ユーザが所有する手続き（他ユーザが定義したパブリック手続きを含む）の SQL オブジェクトを再作成できます。

- AUTHORIZATION 句に自分，及び他ユーザの認可識別子を指定できます。
- ルーチン識別子に自分，及び他ユーザが所有する手続きのルーチン識別子を指定できます。
- AUTHORIZATION 句，及びルーチン識別子を両方省略することでシステム内の全手続きを再作成できます。

(3) 形式

```
  ALTER PROCEDURE
       〔 { 〔認可識別子.〕ルーチン識別子
          | 〔AUTHORIZATION 認可識別子〕
            〔ALL | INDEX USING〔認可識別子.〕表識別子〕 } 〕
       〔SQLコンパイルオプション〔SQLコンパイルオプション〕…〕

  SQLコンパイルオプション：：＝ {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPAT
IBLE} 〕
                               | OPTIMIZE LEVEL SQL最適化オプション
                                          〔, SQL最適化オプション〕…
                               | ADD OPTIMIZE LEVEL SQL拡張最適化オプション
                                          〔, SQL拡張最適化オプション〕…
                               | SUBSTR LENGTH 文字の最大長 }
```

(4) オペランド

(a) 〔認可識別子.〕ルーチン識別子

特定の手続きの SQL オブジェクトだけを再作成する場合に指定します。

指定された手続きのインデクス情報が有効か無効か，又は SQL オブジェクトが有効か無効かにかかわらず，必ず SQL オブジェクトを再作成します。

SQL コンパイルオプションの変更のために使用します。

認可識別子

SQL オブジェクトを再作成する手続きの所有者の認可識別子を指定します。

パブリック手続きの SQL オブジェクトを再作成する場合は，認可識別子に PUBLIC を指定します。

ルーチン識別子

SQL オブジェクトを再作成する手続き名を指定します。

(b) 〔AUTHORIZATION 認可識別子〕

〔ALL ｜ INDEX USING〔認可識別子.〕表識別子〕

再作成する手続きを，手続きの所有者の認可識別子と，手続きの状態で指定します。

〔AUTHORIZATION 認可識別子〕

手続きの所有者の認可識別子を指定して，そのユーザが所有するすべての手続き（そのユーザが定義したパブリック手続きを含む）の SQL オブジェクトを再作成します。

このオペランドを省略した場合，システム内のすべての手続きの SQL オブジェクトを再作成します。

ただし，実際に SQL オブジェクトを再作成するかどうかは，ALL 句，INDEX USING 句の指定との組み合わせによって決まります。

認可識別子

SQL オブジェクトを再作成する手続きの所有者の認可識別子を指定します。

〔ALL ｜ INDEX USING〔認可識別子.〕表識別子〕

どの状態の手続きの，SQL オブジェクトを再作成するかを指定します。

ALL 句，INDEX USING 句のどちらも指定しなかった場合，SQL オブジェクトが無効な手続きだけを再作成します。

ALL

指定した手続きの SQL オブジェクトが有効か無効か，また，インデクス情報が有効か無効かにかかわらず，すべて再作成する場合に指定します。

INDEX USING〔認可識別子.〕表識別子

インデクス情報が無効な手続きの SQL オブジェクトだけを再作成します。

インデクスの追加又は削除をした場合に，手続きの SQL オブジェクト中のインデクス情報が無効となるため，インデクスを追加又は削除した実表の表識別子を指定して，その表を使用する SQL オブジェクトでインデクス情報が無効なすべての手続きの SQL オブジェクトを再作成します。

なお，手続きの SQL オブジェクト中のインデクス情報だけが無効な場合でも，その手続きを実行できますが，インデクス情報が有効な場合の方が性能は良くなります。

指定した実表又はビュー表を基表として定義したビュー表を使用している手続きについても，インデクス情報が無効ならば SQL オブジェクトを再作成します。

INDEX USING 句を指定すると，インデクス情報だけが無効な手続きの SQL オブジェクトを再作成しますが，SQL オブジェクトが無効な手続きについては再作成しません。SQL オブジェクトが無効な手続きの再作成が必要な場合は，別途，INDEX USING 句を省略するか，ALL 指定で ALTER PROCEDURE を発行してください。

〔認可識別子.〕表識別子

SQL オブジェクトを再作成する手続きが使用する実表又はビュー表の所有者の認可識別子と表識別子を指定します。

認可識別子を省略した場合，実行ユーザの認可識別子が仮定されます。

表識別子にパブリックビューを指定する場合は，認可識別子に PUBLIC を指定してください。

(c) SQL コンパイルオプション

```
: := {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE} 〕
      | OPTIMIZE LEVEL SQL最適化オプション〔, SQL最適化オプション〕…
      | ADD OPTIMIZE LEVEL SQL拡張最適化オプション〔, SQL拡張最適化オプション〕…
      | SUBSTR LENGTH 文字の最大長 }
```

SQL コンパイルオプションには，ISOLATION，OPTIMIZE LEVEL，ADD OPTIMIZE LEVEL，SUBSTR LENGTH をそれぞれ 1 回しか指定できません。

〔ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE}〕〕

SQL のデータ保証レベルを指定します。

データ保証レベル

データ保証レベルとは，トランザクションのどの時点までデータの内容を保証するかのレベルです。次に示すデータ保証レベルを指定できます。

- 0

  データの内容を保証しない場合に指定します。

  0 レベルを指定すると，ほかのユーザが更新中のデータでも，更新完了を待たないで参照できます。ただし，参照する表が共用表の場合，ほかのユーザが排他モードで LOCK 文を実行しているときには，排他解除待ちとなります。

- 1

  検索処理の終了までデータの内容を保証したい場合に指定します。

  1 レベルを指定すると，検索処理が終了するまで（HiRDB がページ，又は行を見終わるまで）一度検索したデータをほかのユーザは更新できません。

- 2

  トランザクションの終了まで一度検索したデータの内容を保証したい場合に指定します。

  2 レベルを指定すると，トランザクションが終了するまで一度検索したデータをほかのユーザは更新できません。

このオペランドを省略した場合，前回の SQL オブジェクト作成時（CREATE PROCEDURE，ALTER PROCEDURE，又は ALTER ROUTINE の実行時）に指定した値が仮定されます。

データ保証レベルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

〔FOR UPDATE　{EXCLUSIVE ｜ COMPATIBLE}〕

手続き中で，FOR UPDATE 句を指定した（FOR UPDATE が仮定される場合を含む）カーソル又は問合せに対して，SQL コンパイルオプションで指定したデータ保証レベルに関係なく，常に WITH EXCLUSIVE LOCK を仮定する場合に指定します。

このオペランドを省略した場合，EXCLUSIVE を仮定します。ただし，0904 互換モードを適用している場合は COMPATIBLE を仮定します。

バージョン 09-50 より前の HiRDB で FOR UPDATE EXCLUSIVE を省略した手続きと同じ動作をさせたい場合は，COMPATIBLE を指定します。

このオペランドと，ISOLATION データ保証レベルの関係から決まる FOR UPDATE の排他オプションを次に示します。

ISOLATION データ保証レベル	FOR UPDATE EXCLUSVIE	FOR UPDATE COMPATIBLE
0	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
1	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
2	WITH EXCLUSIVE LOCK	WITH EXCLUSIVE LOCK

《クライアント環境定義との関係》

ALTER PROCEDURE に対して，PDISLLVL，PDFORUPDATEEXLOCK の指定は無効となります。

《SQL との関係》

手続き中の SQL 文に排他オプションを指定している場合は SQL コンパイルオプションで指定するデータ保証レベル，FOR UPDATE EXCLUSIVE，及び FOR UPDATE COMPATIBLE から仮定する排他オプションより SQL 文に指定した排他オプションが優先されます。

《システム定義との関係》

ALTER PROCEDURE に対して，pd_isolation_level オペランドの指定は無効となります。

〔OPTIMIZE LEVEL SQL 最適化オプション〔，SQL 最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDSQLOPTLVL」を参照してください。

SQL 最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法がありますが，通常時は識別子で指定する方法をお勧めします。

省略した場合，前回の SQL オブジェクト作成時（CREATE PROCEDURE，ALTER PROCEDURE，又は ALTER ROUTINE）に採用した値が仮定されます。

識別子で指定する場合：

```
OPTIMIZE LEVEL ”識別子”〔，”識別子”〕…
```

＜指定例＞

- ネストループジョイン優先とグループ分け高速化処理を適用する場合
  OPTIMIZE LEVEL "PRIOR_NEST_JOIN","RAPID_GROUPING"
- すべての最適化を適用しない場合
  OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。ただし，同時に"NONE"以外の識別子を指定すると，"NONE"は無効になります。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
OPTIMIZE LEVEL 符号なし整数〔，符号なし整数〕…
```

＜指定例＞

- 複数の SQL オブジェクト作成，AND の複数インデクス利用の抑止，及び複数インデクス利用の強制を適用する場合
  符号なし整数をコンマで区切って指定する場合：
  OPTIMIZE LEVEL 4,10,16
  符号なし整数の和を指定する場合：
  OPTIMIZE LEVEL 30
- 既に 14（4+10）を指定していて，新たに 16 を追加する場合
  OPTIMIZE LEVEL 14,16
- すべての最適化を適用しない場合
  OPTIMIZE LEVEL 0

＜規則＞

1. バージョン 06-00 より前の HiRDB から，バージョン 06-00 以降の HiRDB にバージョンアップする場合，バージョン 06-00 より前の合計値指定も有効となります。最適化オプションを変更する必要がない場合は，バージョン 06-00 以降の HiRDB にバージョンアップしたときにこのオペランドの指定値を変更する必要はありません。
2. 符号なし整数は一つ以上指定してください。
3. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
4. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。ただし，同時に 0 以外の識別子を指定すると，0 は無効になります。
5. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。
6. 複数の最適化方法を指定する場合，その符号なし整数の和を指定することもできます。ただし，同じ最適化方法の値は二つ以上足さないでください（足した結果が別の最適化方法とみなされることもあるため）。
7. 複数の最適化方法の値を足して指定する場合，どの最適方法を指定しているのか分かりにくくなるため，コンマで区切って指定する方法をお勧めします。また，既に複数の最適化方法の値を足して指定している場合で，新たに別の最適化方法が必要になったときは，追加する値をコンマで区切って後ろに指定できます。

《システム定義との関係》

1. ALTER PROCEDURE に対して，システム定義の pd_optimize_level オペランドの指定は無効となります。
2. システム定義の pd_floatable_bes オペランド，又は pd_non_floatable_bes オペランドを指定している場合，「フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）」及び「フロータブルサーバ対象限定（データ取り出しバックエンドサーバ）」の指定は無効となります。
3. システム定義の pd_indexlock_mode オペランドに KEY を指定している場合（インデクスキー値排他の場合），「更新 SQL の作業表作成抑止」の指定は無効になります。

《クライアント環境定義との関係》

ALTER PROCEDURE に対して，PDSQLOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔ADD OPTIMIZE LEVEL SQL 拡張最適化オプション〔，SQL 拡張最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 拡張最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDADDITIONALOPTLVL」を参照してください。

SQL 拡張最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法があります。

省略した場合，前回の SQL オブジェクト作成時（CREATE PROCEDURE，ALTER PROCEDURE，又は ALTER ROUTINE）に採用した値が仮定されます。

識別子で指定する場合：

```
ADD OPTIMIZE LEVEL ”識別子”〔,”識別子”〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL "COST_BASE_2","APPLY_HASH_JOIN"

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
ADD OPTIMIZE LEVEL 符号なし整数〔,符号なし整数〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL 1,2

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL 0

＜規則＞

1. 符号なし整数は一つ以上指定してください。
2. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。
4. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。

《システム定義との関係》

ALTER PROCEDURE に対して，システム定義の pd_optimize_level オペランドの指定は無効となります。

《クライアント環境定義との関係》

ALTER PROCEDURE に対して，PDADDITIONALOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔SUBSTR LENGTH 文字の最大長〕

1 文字を表現する最大バイト数を指定します。

文字の最大長に指定できる値は，3～6（pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8_ivs を指定した場合は 3～10）です。

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，又は utf-8_ivs を指定した場合にだけ有効となり，スカラ関数 SUBSTR の結果の長さに影響します。SUBSTR については，「SUBSTR」を参照してください。

＜規則＞

バージョン 08-00 より前の HiRDB からバージョン 08-00 以降の HiRDB にバージョンアップする場合は，3 を仮定します。文字の最大長を変更する必要がない場合は，バージョン 08-00 以降の HiRDB にバージョンアップしたときにこのオペランドを指定する必要はありません。

《システム定義との関係》

ALTER PROCEDURE に対して，システム定義の pd_substr_length オペランドの指定は無効となります。pd_substr_length オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

《クライアント環境定義との関係》

ALTER PROCEDURE に対して，PDSUBSTRLEN の指定は無効となります。PDSUBSTRLEN については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

《pdntenv コマンド又は pdsetup コマンドで指定した文字コード種別との関係》

文字コード種別に utf-8，又は utf-8_ivs を指定した場合だけ有効になります。

そのほかの文字コード種別の場合は，構文チェックだけ行い，指定を無視します。

このオペランドを省略した場合，前回の SQL オブジェクト作成時（CREATE PROCEDURE，ALTER PROCEDURE，又は ALTER ROUTINE の実行時）に指定した値が仮定されます。

(5) 共通規則

1. ALTER PROCEDURE で SQL コンパイルオプションを指定する場合，再作成する手続きの元の CREATE PROCEDURE に SQL コンパイルオプションを反映してできる SQL 文は，SQL 文の最大長を超えないようにしてください。
2. 次の条件では，Java 手続きから ALTER PROCEDURE を実行できません。
   - 実行中の SQL オブジェクトが再作成になる場合

(6) 留意事項

1. ALTER PROCEDURE は，OLTP 下の X/Open に従った UAP から指定できません。
2. 複数の手続きの SQL オブジェクトを再作成する場合，手続きごとに自動的に COMMIT 文，又は ROLLBACK 文が実行されます。
3. ALTER PROCEDURE 実行直後に GET DIAGNOSTICS 文を実行すると，ALTER PROCEDURE の診断情報が取得できます。このとき，再作成が正常終了した手続きの SQL コードは 0 となります。
4. 手続き中の SQL 文のデータ保証レベル，SQL 最適化オプション，SQL 拡張最適化オプション，及び文字の最大長は，ルーチンの定義時，又は再作成時の指定で決まり，手続き呼出し時のシステム定義やクライアント環境定義の影響を受けません。
5. Java 手続き及び Java 関数は SQL オブジェクトが作成されないため，SQL オブジェクトの再作成はしません。SQL コンパイルオプションの更新だけできます。
6. ルーチン識別子には，トリガ動作手続きの識別子は指定できません。トリガの SQL オブジェクトを再作成するには，ALTER ROUTINE 又は ALTER TRIGGER を使用してください。
7. 他ユーザが定義したパブリック手続きの SQL オブジェクトを再作成した場合も，その SQL オブジェクトの定義者は変わりません。

(7) 使用例

1. ユーザ（USER1）の手続き（PROC1）をデータ保証レベル 1 で再作成します。

```
ALTER PROCEDURE
    USER1.PROC1 ISOLATION 1
```

2. ユーザ（USER1）の表（T1）を参照するユーザ（USER1）の有効な手続きの SQL オブジェクトの中で，インデクス情報が無効となっている手続きを再作成します。

```
ALTER PROCEDURE
    AUTHORIZATION USER1 INDEX USING USER1.T1
```

3. すべての手続きの中で，SQL オブジェクトが無効となっている手続きを再作成します。

```
ALTER PROCEDURE
```

3.4 ALTER ROUTINE（関数，手続き，及びトリガの SQL オブジェクトの再作成）

3.4.1 ALTER ROUTINE の形式と規則

(1) 機能

SQL 関数，SQL 手続き，及びトリガの SQL オブジェクトを再作成します。また，Java 関数及び Java 手続きのコンパイルオプションを変更します。

(2) 使用権限

関数，手続き，及びトリガの所有者

自分が所有する関数，手続き（自分が定義したパブリック関数，パブリック手続きを含む），及びトリガの SQL オブジェクトを再作成できます。

- AUTHORIZATION 句には，自分の認可識別子だけ指定できます。
- AUTHORIZATION 句を省略した場合，エラーとなります。

DBA 権限を持つユーザ

自分が所有する関数，手続き（自分が定義したパブリック関数，パブリック手続きを含む），及びトリガ，並びに他ユーザが所有する関数，手続き（他ユーザが定義したパブリック関数，パブリック手続きを含む），及びトリガの SQL オブジェクトを再作成できます。

- AUTHORIZATION 句には，自分及び他ユーザの認可識別子を指定できます。
- AUTHORIZATION 句を省略することで，システム内のすべての関数及び手続きの SQL オブジェクトを再作成できます。

(3) 形式

```
  ALTER ROUTINE 〔〔AUTHORIZATION 認可識別子〕〔ALL〕〕
      〔SQLコンパイルオプション〔SQLコンパイルオプション〕…〕

  SQLコンパイルオプション：：＝ {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE｜COMPAT
IBLE}〕
                               ｜OPTIMIZE LEVEL SQL最適化オプション
                                          〔, SQL最適化オプション〕…
                               ｜ADD OPTIMIZE LEVEL SQL拡張最適化オプション
                                          〔, SQL拡張最適化オプション〕…
                               ｜SUBSTR LENGTH 文字の最大長 }
```

(4) オペランド

(a) 〔AUTHORIZATION 認可識別子〕 〔ALL〕

再作成する関数，手続き，及びトリガを，所有者の認可識別子と，関数，手続き，及びトリガの状態で指定します。

AUTHORIZATION 認可識別子

関数，手続き，及びトリガの所有者（パブリック関数，パブリック手続きの定義者を含む）の認可識別子を指定して，そのユーザが所有するすべての関数，手続き，及びトリガの SQL オブジェクトを再作成します。このオペランドを省略した場合，システム内のすべての関数，手続き，及びトリガの SQL オブジェクトを再作成します。ただし，実際に SQL オブジェクトを再作成するかどうかは，ALL 句の指定との組み合わせによって決まります。

認可識別子

再作成する関数，手続き，及びトリガの，所有者の認可識別子を指定します。

ALL

指定した関数，手続き，及びトリガの SQL オブジェクトが有効か無効かに関係なく，すべて再作成する場合に指定します。

このオペランドを省略した場合，SQL オブジェクトが無効な関数，手続き，及びトリガだけ再作成します。

(b) SQL コンパイルオプション

```
: := {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE} 〕
       | OPTIMIZE LEVEL SQL最適化オプション〔, SQL最適化オプション〕…
       | ADD OPTIMIZE LEVEL SQL拡張最適化オプション〔, SQL拡張最適化オプション〕…
       | SUBSTR LENGTH 文字の最大長 }
```

SQL コンパイルオプションには ISOLATION，OPTIMIZE LEVEL，ADD OPTIMIZE LEVEL，SUBSTR LENGTH をそれぞれ 1 回しか指定できません。

〔ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE}〕〕

SQL のデータ保証レベルを指定します。

データ保証レベル

データ保証レベルとは，トランザクションのどの時点までデータの内容を保証するかのレベルです。次に示すデータ保証レベルを指定できます。

- 0

  データの内容を保証しない場合に指定します。

  0 レベルを指定すると，ほかのユーザが更新中のデータでも，更新完了を待たないで参照できます。ただし，参照する表が共用表の場合，ほかのユーザが排他モードで LOCK 文を実行しているときには，排他解除待ちとなります。

- 1

検索処理の終了までデータの内容を保証したい場合に指定します。

1 レベルを指定すると，検索処理が終了するまで（HiRDB がページ，又は行を見終わるまで）一度検索したデータをほかのユーザは更新できません。

- 2

トランザクションの終了まで一度検索したデータの内容を保証したい場合に指定します。

2 レベルを指定すると，トランザクションが終了するまで一度検索したデータをほかのユーザは更新できません。

〔FOR UPDATE　{EXCLUSIVE ｜ COMPATIBLE}〕

手続き中で，FOR UPDATE 句を指定した（FOR UPDATE が仮定される場合を含む）カーソル又は問合せに対して，SQL コンパイルオプションで指定したデータ保証レベルに関係なく，常に WITH EXCLUSIVE LOCK を仮定する場合に指定します。

このオペランドを省略した場合，EXCLUSIVE を仮定します。ただし，0904 互換モードを適用している場合は COMPATIBLE を仮定します。

09-50 より前の HiRDB で FOR UPDATE EXCLUSIVE を省略した関数，手続き，及びトリガと同じ動作をさせたい場合は COMPATIBLE を指定します。

このオペランドと，ISOLATION データ保証レベルの関係から決まる FOR UPDATE の排他オプションを次に示します。

ISOLATION データ保証レベル	FOR UPDATE EXCLUSVIE	FOR UPDATE COMPATIBLE
0	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
1	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
2	WITH EXCLUSIVE LOCK	WITH EXCLUSIVE LOCK

《クライアント環境定義との関係》

ALTER ROUTINE に対して，PDISLLVL，PDFORUPDATEEXLOCK の指定は無効となります。

《SQL との関係》

手続き中の SQL 文に排他オプションを指定している場合は SQL コンパイルオプションで指定したデータ保証レベル，FOR UPDATE EXCLUSIVE，及び FOR UPDATE COMPATIBLE から仮定する排他オプションより SQL 文に指定した排他オプションが優先されます。

《システム定義との関係》

ALTER ROUTINE に対して，pd_isolation_level オペランドの指定は無効となります。

このオペランドを省略した場合，前回の SQL オブジェクト作成時（CREATE PROCEDURE，ALTER PROCEDURE，CREATE TYPE，ALTER ROUTINE，CREATE TRIGGER，又は ALTER TRIGGER の実行時）に指定した値が仮定されます。

データ保証レベルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

〔OPTIMIZE LEVEL SQL 最適化オプション〔，SQL 最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDSQLOPTLVL」を参照してください。

SQL 最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法がありますが，通常時は識別子で指定する方法をお勧めします。

省略した場合，前回の SQL オブジェクト作成時（CREATE PROCEDURE，ALTER PROCEDURE，CREATE TYPE，ALTER ROUTINE，CREATE TRIGGER，又は ALTER TRIGGER）に採用した値が仮定されます。

識別子で指定する場合：

```
OPTIMIZE LEVEL ”識別子”〔，”識別子”〕…
```

＜指定例＞

- ネストループジョイン優先とグループ分け高速化処理を適用する場合

  OPTIMIZE LEVEL "PRIOR_NEST_JOIN","RAPID_GROUPING"

- すべての最適化を適用しない場合

  OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。ただし，同時に"NONE"以外の識別子を指定すると，"NONE"は無効になります。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
OPTIMIZE LEVEL 符号なし整数〔，符号なし整数〕…
```

＜指定例＞

- 複数の SQL オブジェクト作成，AND の複数インデクス利用の抑止，及び複数インデクス利用の強制を適用する場合

  符号なし整数をコンマで区切って指定する場合：

  OPTIMIZE LEVEL 4,10,16

  符号なし整数の和を指定する場合：

  OPTIMIZE LEVEL 30

- 既に 14（4+10）を指定していて，新たに 16 を追加する場合

  OPTIMIZE LEVEL 14,16

- すべての最適化を適用しない場合

OPTIMIZE LEVEL 0

＜規則＞

1. バージョン 06-00 より前の HiRDB から，バージョン 06-00 以降の HiRDB にバージョンアップする場合，バージョン 06-00 より前の合計値指定も有効となります。最適化オプションを変更する必要がない場合は，バージョン 06-00 以降の HiRDB にバージョンアップしたときにこのオペランドの指定値を変更する必要はありません。
2. 符号なし整数は一つ以上指定してください。
3. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
4. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。ただし，同時に 0 以外の識別子を指定すると，0 は無効になります。
5. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。
6. 複数の最適化方法を指定する場合，その符号なし整数の和を指定することもできます。ただし，同じ最適化方法の値は二つ以上足さないでください（足した結果が別の最適化方法とみなされることもあるため）。
7. 複数の最適化方法の値を足して指定する場合，どの最適方法を指定しているのか分かりにくくなるため，コンマで区切って指定する方法をお勧めします。また，既に複数の最適化方法の値を足して指定している場合で，新たに別の最適化方法が必要になったときは，追加する値をコンマで区切って後ろに指定できます。

《システム定義との関係》

1. ALTER ROUTINE に対して，システム定義の pd_optimize_level オペランドの指定は無効となります。
2. システム定義の pd_floatable_bes オペランド，又は pd_non_floatable_bes オペランドを指定している場合，「フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）」及び「フロータブルサーバ対象限定（データ取り出しバックエンドサーバ）」の指定は無効となります。
3. システム定義の pd_indexlock_mode オペランドに KEY を指定している場合（インデクスキー値排他の場合），「更新 SQL の作業表作成抑止」の指定は無効になります。

《クライアント環境定義との関係》

ALTER ROUTINE に対して，PDSQLOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔ADD OPTIMIZE LEVEL SQL 拡張最適化オプション〔，SQL 拡張最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 拡張最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDADDITIONALOPTLVL」を参照してください。

SQL 拡張最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法があります。省略した場合，前回の SQL オブジェクト作成時（CREATE PROCEDURE，ALTER PROCEDURE，CREATE TYPE，ALTER ROUTINE，CREATE TRIGGER，又は ALTER TRIGGER）に採用した値が仮定されます。

識別子で指定する場合：

```
ADD OPTIMIZE LEVEL ”識別子”〔,”識別子”〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL "COST_BASE_2","APPLY_HASH_JOIN"

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
ADD OPTIMIZE LEVEL 符号なし整数〔,符号なし整数〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL 1,2

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL 0

＜規則＞

1. 符号なし整数は一つ以上指定してください。
2. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。
4. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。

《システム定義との関係》

ALTER ROUTINE に対して，システム定義の pd_optimize_level オペランドの指定は無効となります。

《クライアント環境定義との関係》

ALTER PROCEDURE に対して，PDADDITIONALOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔SUBSTR LENGTH 文字の最大長〕

1 文字を表現する最大バイト数を指定します。

文字の最大長に指定できる値は，3～6（pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8_ivs を指定した場合は 3～10）です。

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，又は utf-8_ivs を指定した場合にだけ有効となり，スカラ関数 SUBSTR の結果の長さに影響します。SUBSTR については，「SUBSTR」を参照してください。

＜規則＞

バージョン 08-00 より前の HiRDB からバージョン 08-00 以降の HiRDB にバージョンアップする場合は，3 を仮定します。文字の最大長を変更する必要がない場合は，バージョン 08-00 以降の HiRDB にバージョンアップしたときにこのオペランドを指定する必要はありません。

《システム定義との関係》

ALTER ROUTINE に対して，システム定義の pd_substr_length オペランドの指定は無効となります。pd_substr_length オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

《クライアント環境定義との関係》

ALTER ROUTINE に対して，PDSUBSTRLEN の指定は無効となります。PDSUBSTRLEN については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

《pdntenv コマンド又は pdsetup コマンドで指定した文字コード種別との関係》

文字コード種別に utf-8，又は utf-8_ivs を指定した場合だけ有効になります。

そのほかの文字コード種別の場合は，構文チェックだけ行い，指定を無視します。

このオペランドを省略した場合，前回の SQL オブジェクト作成時（CREATE PROCEDURE，ALTER PROCEDURE，CREATE FUNCTION，CREATE TYPE，ALTER ROUTINE，CREATE TRIGGER，又は ALTER TRIGGER の実行時）に指定した値が仮定されます。

(5) 共通規則

1. ALTER ROUTINE で SQL コンパイルオプションを指定する場合，再作成するルーチンの元の CREATE PROCEDURE，CREATE FUNCTION，CREATE TYPE，又は CREATE TRIGGER に

SQL コンパイルオプションを反映してできる SQL 文は，SQL 文の最大長を超えないようにしてください。

2. SQL コンパイルオプションの指定は手続き及びトリガに対してだけ有効となり，関数に対しては無効となります。
3. 次の条件では，Java 手続きから ALTER ROUTINE を実行できません。
   - 実行中の SQL オブジェクトが再作成になる場合

(6) 留意事項

1. ALTER ROUTINE は，OLTP 下の X/Open に従った UAP から指定できません。
2. 複数の関数，手続き，及びトリガの SQL オブジェクトを再作成する場合，関数，手続き，及びトリガごとに自動的に COMMIT 文又は ROLLBACK 文が実行されます。
3. ALTER ROUTINE 実行直後に GET DIAGNOSTICS 文を実行すると，ALTER ROUTINE の診断情報を取得できます。このとき，再作成が正常終了した関数，手続き，及びトリガの SQL コードは 0 となります。
4. データ保証レベル，SQL 最適化オプション，SQL 拡張最適化オプション，及び文字の最大長は，ルーチン及びトリガの定義時，又は再作成時の指定で決まり，手続き呼出し時，関数呼出し時，及びトリガ動作実行時のシステム定義やクライアント環境定義の影響を受けません。
5. 手続き及びトリガで使用している表（そのトリガを定義した表を除く）に対してインデクスの追加又は削除をした場合，手続き及びトリガの SQL オブジェクト中のインデクス情報が無効となります。この場合，このトリガは実行できません。また，手続き又はトリガからこの手続きを実行できないため，ALL 指定で SQL オブジェクトを再作成してください。
6. Java 手続き及び Java 関数は SQL オブジェクトが作成されないため，SQL オブジェクトの再作成はしません。SQL コンパイルオプションの更新だけできます。
7. ネストするトリガがある場合，次のことに注意してください。
   - ネストするトリガの一部，又はすべてが無効状態となっている場合，1 回の ALTER ROUTINE 実行ではすべてのトリガが有効にならないことがあります（ALTER ROUTINE は KFPA11528-E エラーとなります）。この場合，ALTER ROUTINE が正常終了するまで，ALTER ROUTINE を繰り返し実行してください。
8. ネストするトリガがある場合，かつトリガ動作中の探索条件に関数を指定している場合は，次のことに注意してください。
   - この関数を削除した場合，ALTER ROUTINE が正常終了しても，トリガ実行時に KFPA11529-E エラーとなることがあります。この場合，実行時エラーとなったトリガ又はルーチンを再作成（ALTER TRIGGER 又は ALTER PROCEDURE）して再実行してください。実行時エラーとなるトリガ又はルーチンは，関数を削除したトリガの呼び出し元のトリガ又はルーチンです。
9. ループするトリガがある場合，次のことに注意してください。

- ループするトリガがすべて無効状態になっている場合，ALTER ROUTINE での再作成はできません。この場合，ループするトリガ及びこれらのトリガを定義している表にあるすべてのトリガを削除し，再定義してください。

10. 他ユーザが定義したパブリック関数，パブリック手続きの SQL オブジェクトを再作成した場合も，その SQL オブジェクトの定義者は変わりません。

(7) 使用例

1. すべての関数，手続き，及びトリガの中で，SQL オブジェクトが無効となっている関数及び手続きを再作成します。

```
ALTER ROUTINE
```

2. ユーザ（USER1）の関数，手続き，及びトリガをすべて再作成します。

```
ALTER ROUTINE
    AUTHORIZATION USER1 ALL
```

3.5 ALTER TABLE（表定義変更）

3.5.1 ALTER TABLE の形式と規則

(1) 機能

表定義変更には，次に示す機能があります。

- 実表の最後に新しい列を追加します。
- 予備列から列を切り出し，新たな列を追加します。
- ハッシュ関数によって横分割している実表に対して，表格納用 RD エリアを追加します。
- 実表又は列の属性を変更します。
  - 可変長データ型の最大長を大きくします。
  - 文字データを混在文字データに変更します。
  - 繰返し列の最大要素数を大きくします。
  - 可変長文字データの格納方式を変更します。
  - 列回復制約を変更します。
  - 列の既定値を設定，変更，又は削除します。
  - 既定値のない非ナル値制約の列を，既定値のある非ナル値制約の列に変更します。
  - 更新可能列属性を変更します。
  - データの格納されていない実表のクラスタキーの一意性制約を変更します。
  - 実表の最小排他リソース単位を変更します。
  - ハッシュ関数によって横分割している実表に対して，ハッシュ関数を変更します。
  - 空き領域の再利用機能を適用又は解除します。
  - 空き領域の再利用機能でのセグメント数上限を変更します。
  - 改竄防止表に変更します。
- データの格納されていない実表の列を削除します。
- 実表又は列の名称を変更します。
- 横分割している実表やマトリクス分割している実表の分割格納条件を変更します。
- 主キーを追加，又は削除します。

(2) 使用権限

実表の所有者

自分が所有する実表にだけ指定できます。

(3) 形式

```
ALTER TABLE 〔認可識別子.〕表識別子 表定義変更動作
表定義変更動作：：＝ {  列追加定義
                      ｜表制約定義追加
                      ｜RDエリア追加定義
                      ｜列属性変更定義
                      ｜列削除定義
                      ｜表制約定義削除
                      ｜表名変更定義
                      ｜列名変更定義
                      ｜分割格納条件変更定義 }
```

≪各項目の詳細≫

```
列追加定義：：＝
    ADD 列名 データ型〔ARRAY ［最大要素数］〕
                〔〔列回復制約1〕
                  {LOB列格納用RDエリア指定
                    ｜マトリクス分割LOB列格納用RDエリア指定
                    ｜抽象データ型定義内LOB格納用RDエリア指定
                      〔プラグイン指定〕
                    ｜マトリクス分割LOB属性格納用RDエリア指定
                      〔プラグイン指定〕} 〕
                〔圧縮指定〕
                〔DEFAULT句〕
                〔非ナル値制約〕
                〔予備列定義〕
                〔更新可能列属性〕
                〔INTO 予備列名〕
                〔WITH PROGRAM〕
  列回復制約1：：＝RECOVERY〔 {ALL｜PARTIAL｜NO} 〕
  LOB列格納用RDエリア指定：：＝
                 IN  {LOB列格納用RDエリア名
                       ｜（LOB列格納用RDエリア名）
                       ｜（（LOB列格納用RDエリア名）
                          〔,  （LOB列格納用RDエリア名）〕…）}
  マトリクス分割LOB列格納用RDエリア指定：：＝2次元格納用RDエリア指定
  マトリクス分割LOB属性格納用RDエリア指定：：＝2次元格納用RDエリア指定
  2次元格納用RDエリア指定：：＝（マトリクス分割用RDエリアリスト
                                 〔, マトリクス分割用RDエリアリスト〕…）
  マトリクス分割用RDエリアリスト：：＝（RDエリア名〔, RDエリア名〕…）
  抽象データ型定義内LOB格納用RDエリア指定：：＝
        ALLOCATE（属性名〔..属性名〕…
            IN  {LOB属性格納用RDエリア名
                  ｜（LOB属性格納用RDエリア名）
                   ｜（（LOB属性格納用RDエリア名）
                       〔,  （LOB属性格納用RDエリア名）〕…）}
             〔, 属性名〔..属性名〕…
                  IN  {LOB属性格納用RDエリア名
```

```
                        | (LOB属性格納用RDエリア名)
                        | ( (LOB属性格納用RDエリア名)
                          〔, (LOB属性格納用RDエリア名)〕…) } 〕…)
  圧縮指定：：= COMPRESSED 〔BY 圧縮分割サイズ〕
  圧縮分割サイズ：：= n〔 {K|M|G} 〕
  DEFAULT句：：= {DEFAULT 〔既定値〕 |DEFAULT 既定値 ON ROW EXISTS}
  既定値：：= {定数|USER|CURRENT_USER|CURRENT_DATE|CURRENT DATE
              |CURRENT_TIME|CURRENT TIME
              |CURRENT_TIMESTAMP〔(小数秒精度)〕 〔USING BES〕
              |CURRENT TIMESTAMP〔(小数秒精度)〕〔USING BES〕
              |NULL}
  非ナル値制約：：= { {NULL|NOT NULL〔WITH DEFAULT〕} ※1
                  | 〔NOT NULL〕   WITH DEFAULT ※2}
  予備列定義：：=FOR RESERVED
  更新可能列属性：：=UPDATE 〔ONLY FROM NULL〕
  予備列名：：=列名
表制約定義追加：：=
      ADD PRIMARY KEY (列名〔 {ASC|DESC} 〕
                          〔, 列名〔 {ASC|DESC} 〕…〕 )
            〔IN {インデクス格納用RDエリア名
                | (インデクス格納用RDエリア名)
                | ( (インデクス格納用RDエリア名)
                    〔, (インデクス格納用RDエリア名)〕…)
                |マトリクス分割インデクス格納用RDエリア指定} 〕
            〔インデクスオプション〔インデクスオプション〕…〕
  マトリクス分割インデクス格納用RDエリア指定：：=2次元格納用RDエリア指定
  2次元格納用RDエリア指定：：= (マトリクス分割用RDエリアリスト
                                〔, マトリクス分割用RDエリアリスト〕…)
  マトリクス分割用RDエリアリスト：：= (RDエリア名〔, RDエリア名〕…)
  インデクスオプション：：= {PCTFREE=未使用領域の比率
                            |UNBALANCED SPLIT
                            |EMPTY}
RDエリア追加定義：：=
      ADD RDAREA 表格納用RDエリア名
            〔FOR COLUMN 列名
              {LOB列格納用RDエリア指定
               |抽象データ型定義内LOB格納用RDエリア指定}
              〔, 列名 {LOB列格納用RDエリア指定
                       |抽象データ型定義内LOB格納用RDエリア指定} 〕…〕
            〔FOR INDEX インデクス識別子 インデクス格納用RDエリア指定
              〔, インデクス識別子 インデクス格納用RDエリア指定〕…〕
            〔FOR 〔PRIMARY〕 CLUSTER KEY インデクス格納用RDエリア指定〕
            〔FOR PRIMARY KEY インデクス格納用RDエリア指定〕
            〔WITH PROGRAM〕
  LOB列格納用RDエリア指定：：=
                IN {LOB列格納用RDエリア名
                    | (LOB列格納用RDエリア名)
                    | ( (LOB列格納用RDエリア名)
                      〔, (LOB列格納用RDエリア名)〕…) }
  抽象データ型定義内LOB格納用RDエリア指定：：=
       ALLOCATE (属性名〔..属性名〕…
           IN {LOB属性格納用RDエリア名
               | (LOB属性格納用RDエリア名)
               | ( (LOB属性格納用RDエリア名)
                 〔, (LOB属性格納用RDエリア名)〕…) }
            〔, 属性名〔..属性名〕…
                IN {LOB属性格納用RDエリア名
                    | (LOB属性格納用RDエリア名)
```

```
                        | （（LOB属性格納用RDエリア名）
                          〔,（LOB属性格納用RDエリア名）〕…）｝〕…）
  インデクス格納用RDエリア指定：：＝
                IN｛インデクス格納用RDエリア名
                    |（インデクス格納用RDエリア名）
                    |（（インデクス格納用RDエリア名）
                      〔,（インデクス格納用RDエリア名）〕…）｝
列属性変更定義：：＝
      CHANGE｛列名｛〔｛VARCHAR（データ長）
                          |NVARCHAR（データ長）
                          |MCHAR（｛＊|データ長｝）
                          |MVARCHAR（｛＊|データ長｝）｝
                              〔ARRAY ［｛＊|最大要素数｝］〕〕
                      |〔ARRAY ［最大要素数］〕
                      |BINARY（データ長）｝
                      〔｛NO SPLIT|SPLIT｝〕
                      〔列回復制約２〕
                      〔｛SET DEFAULT句|DROP DEFAULT｝〕
                      〔WITH DEFAULT〕
                      〔更新可能列属性〕
                |CLUSTER KEY〔UNIQUE〕
                |LOCK｛ROW|PAGE｝
                |HASHハッシュ関数名
                |SEGMENT REUSE
                        ｛〔セグメント数〔｛K|M｝〕〕|NO｝
                |SEGMENT REUSE OPTION 再利用オプション値
                |INSERT ONLY
                  〔WHILE ｛日間隔データ|ラベル付き間隔｝ BY 列名〕｝
              〔WITH PROGRAM〕
  列回復制約２：：＝ RECOVERY｛ALL|PARTIAL|NO｝
列削除定義：：＝
      DROP 列名 〔WITH PROGRAM〕
表制約定義削除：：＝
      DROP PRIMARY KEY 〔WITH PROGRAM〕
表名変更定義：：＝
      RENAME TABLE TO 表識別子
                〔WITH PROGRAM〕
列名変更定義：：＝
      RENAME COLUMN FROM 変更前列名 TO 変更後列名
                〔WITH PROGRAM〕
分割格納条件変更定義：：＝
      CHANGE RDAREA ｛横分割表変更指定|マトリクス分割表変更指定｝
           〔LOB列格納用RDエリア変更指定〕
           〔インデクス格納用RDエリア変更指定
             〔インデクス格納用RDエリア変更指定〕…〕
           〔クラスタキー格納用RDエリア変更指定〕
           〔主キー格納用RDエリア変更指定〕
           〔WITHOUT PURGE〕
           〔WITH PROGRAM〕
  横分割表変更指定：：＝
     ｛〔PARTITIONED〕変更前境界値リスト INTO 変更後境界値分割指定
      |PARTITIONED CONDITION 変更前RDエリア情報リスト
         INTO 変更後格納条件分割指定｝
  マトリクス分割表変更指定：：＝
           MULTIDIM（変更対象列名 変更前境界値リスト）
           AT 変更後境界値リスト
           INTO マトリクス分割表格納用RDエリア変更指定
  変更前境界値リスト：：＝境界値リスト
```

```
変更後境界値リスト：：＝境界値リスト
境界値リスト：：＝（（｛境界値｜MAX｝）〔，（｛境界値｜MAX｝）〕…）
変更後境界値分割指定：：＝｛表格納用RDエリア名
                              ｜（表格納用RDエリア名）
                              ｜（〔（表格納用RDエリア名）境界値，〕
                                 …（表格納用RDエリア名））｝
変更前RDエリア情報リスト：：＝
        ｛表格納用RDエリア名
           ｜（表格納用RDエリア名）
           ｜（（表格納用RDエリア名）
             〔，（表格納用RDエリア名）〕…〔，OTHERS〕）
           ｜OTHERS｝
変更後格納条件分割指定：：＝｛表格納用RDエリア名
                                ｜（表格納用RDエリア名）
                                ｜（（表格納用RDエリア名）格納条件
                                   ｛，（表格納用RDエリア名）格納条件
                                      〔〔，（表格納用RDエリア名）格納条件〕
                                       …〕
                                      〔｛，（表格納用RDエリア名）
                                          ｜，OTHERS｝〕）
                                    ｜，（表格納用RDエリア名））
                                    ｜，OTHERS）｝
                              ｜OTHERS｝
格納条件：：＝列名＝｛定数｜（定数〔，定数…〕）｝
インデクス格納用RDエリア変更指定：：＝
         FOR INDEX インデクス名
             INTO 変更後インデクス格納用RDエリア名リスト
変更後インデクス格納用RDエリア名リスト：：＝RDエリア名リスト
RDエリア名リスト：：＝
        ｛インデクス格納用RDエリア名
          ｜（インデクス格納用RDエリア名）
          ｜（（インデクス格納用RDエリア名）
             〔，（インデクス格納用RDエリア名）〕…〔，OTHERS〕）
          ｜２次元格納用RDエリア指定
          ｜OTHERS｝
主キー格納用RDエリア変更指定：：＝
         FOR PRIMARY KEY
             INTO 変更後インデクス格納用RDエリア名リスト
クラスタキー格納用RDエリア変更指定：：＝
         FOR 〔PRIMARY〕CLUSTER KEY
                INTO 変更後インデクス格納用RDエリア名リスト
LOB列格納用RDエリア変更指定：：＝
    FOR COLUMN 列名
       ｛LOB列格納用RDエリア変更リスト
        ｜INTO マトリクス分割LOB列格納用RDエリア変更指定｝
          〔，列名 ｛LOB列格納用RDエリア変更リスト
                   ｜INTOマトリクス分割LOB列格納用RDエリア変更指定｝〕
          〕…
LOB列格納用RDエリア変更リスト：：＝
               INTO ｛LOB列格納用RDエリア名
                      ｜（LOB列格納用RDエリア名）
                      ｜（（LOB列格納用RDエリア名）
                         〔，（LOB列格納用RDエリア名）〕…〔，OTHERS〕）
                      ｜OTHERS｝
マトリクス分割表格納用RDエリア変更指定：：＝２次元格納用RDエリア指定
マトリクス分割LOB列格納用RDエリア変更指定：：＝２次元格納用RDエリア指定
    ２次元格納用RDエリア指定：：＝（マトリクス分割格納用RDエリアリスト
```

```
                                        〔，マトリクス分割格納用RDエリアリスト〕…）
    マトリクス分割格納用RDエリアリスト：：＝（RDエリア名〔，RDエリア名〕…）
```

注※1　FIX 表以外の表の列の場合

注※2　FIX 表の列の場合

(4)　オペランド

(a)　〔認可識別子.〕表識別子

認可識別子

定義変更する実表の所有者の認可識別子を指定します。

表識別子

定義変更する実表の名称を指定します。

(b)　列追加定義

```
：：＝ADD 列名 データ型 〔ARRAY ［最大要素数］〕
  〔〔列回復制約１〕
    {LOB列格納用RDエリア指定
      ｜マトリクス分割LOB列格納用RDエリア指定
      ｜抽象データ型定義内LOB格納用RDエリア指定〔プラグイン指定〕
      ｜マトリクス分割LOB属性格納用RDエリア指定〔プラグイン指定〕｝〕
  〔圧縮指定〕〔DEFAULT句〕〔非ナル値制約〕〔予備列定義〕
  〔更新可能列属性〕〔INTO 予備列名〕〔WITH PROGRAM〕
```

列名

次のどちらかの名称を指定します。

- 実表に追加する列の名称
- 予備列から切り出す列の名称

列名についての規則を次に示します。

＜実表に追加する列の規則＞

1. 列を追加しようとする実表内で同じ列名は指定できません。
2. 既に列が 30,000 個ある実表には，列は追加できません。
3. データが格納されている FIX 表には，列は追加できません。
4. 列を追加すると，既存の行の追加した列にはナル値が設定されます。ナル値の値の設定は，「UPDATE 文　形式 1（データ更新）」を参照してください。
5. 予備列が存在する表に対して，列は追加できません。

＜予備列から切り出す列の規則＞

1. 列を追加しようとする実表内で同じ列名は指定できません。

2. 切り出した結果，列数が 30,001 個以上になる場合は，切り出し列を定義できません。

3. 予備列すべてを切り出す場合を除き，予備列と同じ名称を指定できません。

データ型

次のどちらかのデータ型を指定します。

- 実表に追加する列のデータ型
- 予備列から切り出す列のデータ型

データ型についての規則を次に示します。

＜実表に追加する列のデータ型の規則＞

1. データ型に BLOB を指定する場合，LOB 列格納用 RD エリアを指定してください。
2. データ型に抽象データ型を指定する場合，LOB 属性格納用 RD エリア名を指定してください。ただし，スーパタイプで BLOB が定義されている抽象データ型は指定できません。
3. 抽象データ型の認可識別子を省略した場合で，かつ仮定された認可識別子の抽象データ型がないときは，'MASTER'のスキーマ中に同一名称の抽象データ型があると，その抽象データ型を指定したものとします。

＜予備列から切り出す列のデータ型の規則＞

1. 予備列の定義長を超える長さのデータ型を指定できません。
   データ長については，「既定義型データ長一覧」を参照してください。
2. 予備列の定義長と等しいデータ長のデータ型を指定した場合，予備列すべてが切り出され，予備列が存在しない表となります。その際，予備列に付けていた注釈も削除されます。

ARRAY ［最大要素数］

実表に追加する繰返し列の最大要素数を指定します。

ARRAY 最大要素数についての規則を次に示します。

1. 最大要素数には，2～30,000 の符号なし整数を指定します。
2. ARRAY 最大要素数を省略した場合，繰返し列でないことを示します。
3. 繰返し列は，次のデータ型には指定できません。
   ・BLOB
   ・BINARY
   ・抽象データ型
4. FIX 表には指定できません。
5. 繰返し列は，圧縮列に指定できません。

列回復制約 1：：＝ RECOVERY〔{ALL ｜ PARTIAL ｜ NO}〕

データ型が BLOB の列，又は BLOB 属性の抽象データ型の列を表に追加する場合，列を格納する LOB 列格納用 RD エリア，又は抽象データ型定義内 LOB 格納用 RD エリアのデータベースの更新ログ取得方式を指定します。

なお，列回復制約は，データ型が BLOB 又は BLOB 属性の抽象データ型以外の列には指定できません。省略すると，更新前ログ取得モード(PARTIAL)が仮定されます。仮定値は，システム共通定義の pd_ddl_tbl_recovery オペランドで変更できます。pd_ddl_tbl_recovery オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

ALL

ログ取得モードでユーザ LOB 用 RD エリアを運用するときに指定します。ログ取得モードで運用すると，ロールバック，及びロールフォワードに必要なデータベースの更新ログを取得します。

PARTIAL

更新前ログ取得モードでユーザ LOB 用 RD エリアを運用するときに指定します。更新前ログ取得モードで運用すると，ロールバックに必要なデータベースの更新ログを取得します。

NO

ログレスモードでユーザ LOB 用 RD エリアを運用するときに指定します。ログレスモードで運用すると，データベースの更新ログを取得しません。

指定するデータベースの更新ログ取得方式によって，UAP を実行するときの運用方法や，障害が発生したときのユーザ LOB 用 RD エリアの回復方法が異なります。ログレスモードの運用については，マニュアル「HiRDB システム運用ガイド」を参照してください。

LOB 列格納用 RD エリア指定：：＝

IN {LOB 列格納用 RD エリア名
| (LOB 列格納用 RD エリア名)
| ((LOB 列格納用 RD エリア名)
〔, (LOB 列格納用 RD エリア名)〕…)}

BLOB 型の列を追加する場合，BLOB 列のデータを格納するユーザ LOB 用 RD エリアの名前を指定します。

LOB 列格納用 RD エリアについての規則を次に示します。

1. データ型に BLOB を指定した列は，LOB 列格納用 RD エリア名を必ず指定してください。BLOB 以外のデータ型を指定した列には指定できません。
2. 指定する RD エリア名は，表格納用 RD エリアと同じサーバに定義しているユーザ LOB 用 RD エリアを指定してください。
3. 横分割表に列を追加する場合は，表定義で指定したユーザ用 RD エリアと同じ数のユーザ LOB 用 RD エリアを指定してください。その場合，同じサーバのユーザ用 RD エリアとユーザ LOB 用 RD エリアが，同じ順番になるように指定してください。
4. 境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表に LOB 列を追加する場合は，表定義で指定した表格納用 RD エリアと 1 対 1 に対応するように LOB 列格納用 RD エリアを指定してください。また，表定義で指定している表格納用 RD エリアに重複した RD エリアがある場合は，LOB 列格納用 RD エリアも同一位置が重複するように指定してください。

   上記以外の場合は，LOB 列格納用 RD エリアは重複して指定できません。

マトリクス分割 LOB 列格納用 RD エリア指定：：＝ 2 次元格納用 RD エリア指定

2 次元格納用 RD エリア指定：：＝（マトリクス分割用 RD エリアリスト

〔，マトリクス分割用 RD エリアリスト〕…）

マトリクス分割用 RD エリアリスト：：＝

（RD エリア名〔，RD エリア名〕…）

マトリクス分割した表に BLOB 型の列を追加する場合に，BLOB 列を格納する RD エリア名を指定します。

マトリクス分割した表に BLOB 型の列を追加する場合，マトリクス分割 LOB 列格納用 RD エリア指定を指定してください。

RD エリア名については，LOB 列格納用 RD エリア指定の説明を参照してください。

抽象データ型定義内 LOB 格納用 RD エリア指定：：＝

ALLOCATE（属性名〔..属性名〕…

IN {LOB 属性格納用 RD エリア名

| (LOB 属性格納用 RD エリア名)

| ((LOB 属性格納用 RD エリア名)

〔，(LOB 属性格納用 RD エリア名)〕…)}

〔，属性名〔..属性名〕…

IN {LOB 属性格納用 RD エリア名

| (LOB 属性格納用 RD エリア名)

| ((LOB 属性格納用 RD エリア名)

〔，(LOB 属性格納用 RD エリア名)〕…)}〕…）

LOB 属性を含む抽象データ型の列を追加する場合に指定します。

属性名〔..属性名〕

抽象データ型を構成する属性名を指定します。抽象データ型の属性が抽象データ型であり，その入れ子になっている抽象データ型の属性に LOB 型の属性がある場合は，その LOB 型の属性名を指定してください。

属性名は，次のような場合に指定してください。

- 抽象データ型定義の属性

  抽象データ型定義の属性のデータ型が LOB 型であれば指定します。

- 抽象データ型定義の入れ子

  抽象データ型の属性が抽象データ型であり，その入れ子になっている抽象データ型の属性に LOB 型の属性であれば，その LOB 型の属性名を指定してください。

LOB 属性格納用 RD エリア名

抽象データ型の任意の階層にある，BLOB 属性のデータを格納するユーザ LOB 用 RD エリアの名前を指定します。

LOB 属性格納用 RD エリアについての規則を次に示します。

1. データ型に BLOB を含む抽象データ型を指定した場合，各 BLOB 属性に対してユーザ LOB 用 RD エリアの名前を必ず指定してください。BLOB 以外のデータ型を指定した属性には指定できません。
2. 指定する RD エリア名は，表格納用 RD エリアと同じサーバに定義しているユーザ LOB 用 RD エリアを指定してください。
3. 横分割表に列を追加する場合は，表定義で指定したユーザ用 RD エリアと同じ数のユーザ LOB 用 RD エリアを指定してください。その場合，同じサーバのユーザ用 RD エリアとユーザ LOB 用 RD エリアが，同じ順番になるようにしてください。
4. 境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表に LOB 属性を含む抽象データ型の列を追加する場合は，表定義で指定した表格納用 RD エリアと 1 対 1 に対応するように LOB 属性格納用 RD エリアを指定してください。また，表定義で指定している表格納用 RD エリアに重複した RD エリアがある場合は，LOB 属性格納用 RD エリアも同一位置が重複するように指定してください。

   上記以外の場合は，LOB 属性格納用 RD エリアは重複して指定できません。

プラグイン指定：：＝ PLUGIN プラグインオプション

プラグイン機能が実装されている抽象データ型として定義された列に対して，プラグイン機能に渡すためのパラメタ情報を文字列定数（最大 255 バイト）で記述します。パラメタ情報には，16 進文字列定数は指定できません。

パラメタ情報の詳細については，各種プラグインマニュアルを参照してください。

マトリクス分割 LOB 属性格納用 RD エリア指定：：＝ 2 次元格納用 RD エリア指定

2 次元格納用 RD エリア指定：：＝（マトリクス分割用 RD エリアリスト

〔，マトリクス分割用 RD エリアリスト〕…）

マトリクス分割用 RD エリアリスト：：＝

（RD エリア名〔，RD エリア名〕…）

マトリクス分割した表に LOB 属性を含む抽象データ型列を追加する場合に，LOB 属性を格納する RD エリア名を指定します。

マトリクス分割した表に LOB 属性を含む抽象データ型列を追加する場合，マトリクス分割 LOB 属性格納用 RD エリア指定を指定してください。

RD エリア名については，抽象データ型定義内 LOB 格納用 RD エリア指定の LOB 属性格納用 RD エリア名の説明を参照してください。

圧縮指定：：＝

COMPRESSED 〔BY 圧縮分割サイズ〕

圧縮指定は，列データを圧縮する場合に指定します。BINARY 型の列に圧縮を指定する場合は，定義長が 256 バイト以上である必要があります。255 バイト以下の場合，エラーになります。

圧縮分割サイズ：：＝ n〔{K ｜ M ｜ G}〕

圧縮処理や伸張処理のオーバヘッドを削減したい場合，又は圧縮効率を向上させたい場合に指定します。ただし，圧縮分割サイズを大きくすると，メモリ所要量が増加します。詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

データを n バイト単位で分割して圧縮します。n は符号なし整数で指定し，単位として K（キロバイト），M（メガバイト）及び G（ギガバイト）が指定できます（指定省略時はバイト単位）。圧縮分割サイズに指定できる値の範囲を次の表に示します。

表 3-2　圧縮分割サイズに指定できる値の範囲

単位	n に指定できる値の範囲		圧縮分割サイズの値
	BINARY 型	抽象データ型（XML 型）	
指定しない（バイト）	32,000≦n≦2,147,483,647	1,048,576≦n≦2,147,483,647	n
K	32≦n≦2,097,152	1024≦n≦2,097,152	n×1,024
M	1≦n≦2,048	1≦n≦2,048	n×1,048,576
G	1≦n≦2	1≦n≦2	n×1,073,741,824

圧縮分割サイズの注意事項を次に示します。

- 圧縮分割サイズに指定できる値の範囲外の値を指定した場合，エラーになります。ただし，実際の最大長の計算結果が 2,147,483,648 の場合は，2,147,483,647 になります。
- 圧縮分割サイズに定義長より大きい値を指定した場合，分割しないで圧縮します。
- 圧縮分割サイズを省略すると，BINARY 型の場合は 32,000 バイトを仮定し，抽象データ型（XML 型）の場合は 1 メガバイトを仮定します。ALTER TABLE 文で列の定義長を大きくしても，圧縮列の定義追加時に仮定した圧縮分割サイズは変わりません。

圧縮指定についての規則を次に示します。

1. 圧縮指定が指定できる列は次のデータ型の列だけです。

   ・BINARY 型

   ・抽象データ型（XML 型）※

   注※

   抽象データ型（XML 型）の列のデータを圧縮するためには，バージョン 09-03 以降の HiRDB XML Extension が必要です。バージョン 09-02 以前の HiRDB XML Extension を使用している場合，抽象データ型（XML 型）の列に圧縮指定はできません。

DEFAULT 句：：＝ {DEFAULT 〔既定値〕 | DEFAULT 既定値 ON ROW EXISTS}

追加する列に既定値を設定する場合に指定します。

DEFAULT 句についての規則を次に示します。

1. 一つの ALTER TABLE で，WITH DEFAULT と同時には指定できません。
2. データが格納されている実表に対して指定する場合，DEFAULT 既定値 ON ROW EXISTS を指定してください。
3. BLOB 型，抽象データ型，及び定義長が 32,001 バイト以上の BINARY 型の列には指定できません。
4. 繰返し列には指定できません。
5. 予備列には指定できません。

6. 追加する列が既定の文字集合以外の場合，次に示す項目は DEFAULT 句に指定できません。
   ・CURRENT_DATE(CURRENT DATE)
   ・CURRENT_TIME(CURRENT TIME)
   ・CURRENT TIMESTAMP(CURRENT_TIMESTAMP)

ON ROW EXISTS

データが格納されている実表に DEFAULT 句を指定した列を追加する場合に指定します。

ON ROW EXISTS を指定する場合の規則を次に示します。

1. FIX 表の列には指定できません。
2. DEFAULT 句の既定値には定数又は NULL を指定してください。
3. 列追加前に格納済みの行には，DEFAULT 句に指定した既定値が設定されます。
4. 次のデータ型には指定できません。
   ・BLOB 型
   ・抽象データ型
   ・定義長 256 バイト以上の VARCHAR 型
   ・定義長 128 文字以上の NVARCHAR 型
   ・定義長 256 バイト以上の MVARCHAR 型
   ・定義長 256 バイト以上の BINARY 型
5. 繰返し列には指定できません。
6. ON ROW EXISTS を指定した列は，列追加後に既定値の変更及び削除はできません。

《注意事項》

ON ROW EXISTS 指定時の注意事項を次に示します。

1. 追加した列のデータ分行長が伸びた結果，ページ内に格納できなくなることがあるため，RD エリアのページ長を列追加後の行長で再度見積もり，ページ内に行が格納できる場合に ON ROW EXISTS を指定してください。
2. 列追加時に，列追加前に格納済みの行は更新されません。次の契機で行を更新し，追加した列のデータが格納されます。
   ・追加した列を更新する。
   ・表の再編成を実行する。

非ナル値制約：：＝ {{NULL ｜ NOT NULL〔WITH DEFAULT〕} ※1
　　｜ 〔NOT NULL〕 WITH DEFAULT ※2}

注※1　FIX 表以外の表の列の場合

注※2　FIX 表の列の場合

NULL

列名で指定した列にナル値を許す場合に指定できます。

FIX 表の列には指定できません。

NOT NULL

列名で指定した列にナル値を許さないように制約（非ナル値制約）が必要な場合に指定します。

NOT NULL についての規則を次に示します。

1. データが格納されている実表に対して指定する場合は，DEFAULT 既定値 ON ROW EXISTS を指定してください。
2. NOT NULL を省略すると，追加した列にナル値が許され，既にデータが入っている行に追加した列にはナル値が設定されます。
3. NOT NULL は，繰返し列及び抽象データ型の列に指定できません。

WITH DEFAULT

既定値のある非ナル値制約の列を追加する場合に指定します。

WITH DEFAULT についての規則を次に示します。

1. FIX 表以外の表で WITH DEFAULT を指定する場合，NOT NULL を指定してください。
2. 一つの ALTER TABLE で，DEFAULT 句と同時には指定できません。
3. WITH DEFAULT は，繰返し列には指定できません。
4. WITH DEFAULT は，追加する予備列には指定できません。
5. WITH DEFAULT は，抽象データ型の列には指定できません。

予備列定義：：= FOR RESERVED

列を予備列として追加する場合に指定します。予備列は，将来，FIX 表に列を追加する見込みがある場合に定義します。予備列を定義すると，FIX 表にデータが格納されている状態でも，予備列から切り出して列を追加できます。

予備列には，定義長分の 0x00 が格納されます。0x00 以外の値は格納できません。

次の列に対しては，FOR RESERVED を指定できません。

1. 非 FIX 表の列
2. 予備列を定義済みの表の列
3. CHAR 型以外の列又は文字集合を指定した CHAR 型の列
4. 改竄防止表の列

更新可能列属性：：= UPDATE 〔ONLY FROM NULL〕

改竄防止表に更新可能な列を追加する場合，又は改竄防止表に変更する予定のある表に更新可能な列を追加する場合に指定します。

更新可能列属性は改竄防止表の場合だけ有効となります。

改竄防止表への変更については，「CHANGE」の「INSERT ONLY」オプションを参照してください。

更新可能列属性についての規則を次に示します。

1. 次に示す更新できない列には指定できません。
   ・予備列

UPDATE

改竄防止表で，更新可能な列を追加する場合に指定します。

UPDATE ONLY FROM NULL

改竄防止表で，行の値がナル値から非ナル値へ一度だけ更新可能な列を追加する場合に指定します。

改竄防止表で，UPDATE ONLY FROM NULL 指定した列の値の更新可否を次に示します。

更新前の列の値	更新後の列の値	更新可否
ナル値	ナル値	○
ナル値	非ナル値	○
非ナル値	ナル値	×
非ナル値	非ナル値※	×

（凡例）

○：更新できます。

×：更新できません。

注

繰返し列の場合は，ナル値（要素数が 0 の値）から添え字指定なしでの列単位更新だけが実行できます。

注※

更新前の値と同値を含みます。

UPDATE ONLY FROM NULL 指定についての規則を次に示します。

1. NOT NULL を指定した列には指定できません。
2. FIX 表の場合指定できません。
3. BLOB 型及び定義長が 32,001 バイト以上の BINARY 型の列には指定できません。

更新可能列属性の指定可否と指定時の列の値の更新可否は次のようになります。

表種別	UPDATE 指定		UPDATE ONLY FROM NULL 指定		左記の指定なし	
	指定可否	列の値の更新可否	指定可否	列の値の更新可否	指定可否	列の値の更新可否
非改竄防止表	○	○	○	○	－	○
改竄防止表	○	○	○	○※	－	×

（凡例）

○：更新できます。

×：更新できません。

－：該当しません。

注※

ナル値から非ナル値へ一度だけ更新できます。

INTO 予備列名

新規に追加する列を，予備列から切り出したい場合に指定します。切り出した列は，定義長分の 0x00 で初期化されます。列名には，予備列以外の列名を指定できません。

WITH PROGRAM

次のどれかの操作をする場合，その表を使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

- DEFAULT 句指定の列を表に追加する場合
- NOT NULL 指定の列を表に追加する場合
- BLOB 型，又は抽象データ型の列を表に追加する場合
- 予備列から列を切り出す場合
- 定義長が 256 バイト以上である可変長文字列型の列を表に追加する場合

(c) 表制約定義追加

```
: : =ADD PRIMARY KEY (列名 〔 {ASC | DESC} 〕
                      〔, 列名 〔 {ASC | DESC} 〕 …〕 )
          〔IN {インデクス格納用RDエリア名
              | (インデクス格納用RDエリア名)
              | ( (インデクス格納用RDエリア名)
                  〔, (インデクス格納用RDエリア名) 〕…)
              | マトリクス分割インデクス格納用RDエリア指定} 〕
          〔インデクスオプション〔インデクスオプション〕…〕
```

主キーを追加する場合に指定します。

主キーを追加する場合の規則を次に示します。

1. 表に主キーが定義されている場合は指定できません。
2. 表にインデクスが 255 個定義されている場合は指定できません。
3. 同じ列構成のインデクス（クラスタキーも含む）が既に定義されている場合，主キーは追加できません。同じ列構成とは，次の条件をすべて満たすものをいいます。
   - 列名の並び，及び列数が一致している
   - 昇順，降順の指定がすべて一致している，又はすべて逆になっている
4. 主キーを構成する列に非ナル値制約が定義されている必要があります。
5. 重複したデータがある場合，主キーは追加できません。
6. 主キーを定義しようとしている表を使用する手続き及びトリガが定義されている場合，その SQL オブジェクト中のインデクス情報は無効になります。この場合，トリガは実行できなくなります。また，手

続きからこの手続き又はトリガが実行できなくなるため，SQL オブジェクトを再作成する必要があります。

7. 主キーを追加すると，指定した列に対してインデクスを定義します。
8. 主キーのインデクス識別子は，(PRIMARYttttttttttt)のような名称となります。ttttttttttt は表 ID（10 進数）です。なお，長さは 19 バイトです。
9. その他の規則については，「CREATE TABLE（表定義）」の「複数列一意性制約定義」を参照してください。

〔IN {インデクス格納用 RD エリア名
| (インデクス格納用 RD エリア名)
| ((インデクス格納用 RD エリア名)
〔, (インデクス格納用 RD エリア名)〕…)
| マトリクス分割インデクス格納用 RD エリア指定}〕

インデクス格納用 RD エリア名，マトリクス分割インデクス格納用 RD エリア指定については，「CREATE INDEX　形式 1（インデクス定義）」の「オペランド」を参照してください。

インデクスオプション：：＝

{PCTFREE ＝未使用領域の比率
| UNBALANCED SPLIT
| EMPTY}

インデクスオプションの規則を次に示します。

1. 同一のインデクスオプションは，繰り返して指定できません。
2. 各インデクスオプションの規則については，「CREATE INDEX　形式 1（インデクス定義）」の「オペランド」を参照してください。

(d)　RD エリア追加定義

```
：：＝ADD RDAREA 表格納用RDエリア名
  〔FOR COLUMN 列名
    {LOB列格納用RDエリア指定
      | 抽象データ型定義内LOB格納用RDエリア指定}
    〔, 列名　{LOB列格納用RDエリア指定
              | 抽象データ型定義内LOB格納用RDエリア指定} 〕…〕
  〔FOR INDEX インデクス識別子 インデクス格納用RDエリア指定
    〔, インデクス識別子 インデクス格納用RDエリア指定〕…〕
  〔FOR 〔PRIMARY〕 CLUSTER KEY インデクス格納用RDエリア指定〕
  〔FOR PRIMARY KEY インデクス格納用RDエリア指定〕
  〔WITH PROGRAM〕
```

表格納用 RD エリア名

ハッシュ関数によって横分割している表に対して，ユーザ用 RD エリアを追加する場合に指定します。

表格納用 RD エリア名についての規則を次に示します。

1. 表にユニーク指定のインデクスが定義されている場合，ユーザ用 RD エリアは追加できないことがあります。詳細については，「CREATE INDEX　形式 1（インデクス定義）」の UNIQUE の説明を参照してください。
2. リバランス機能を使用しない場合，データが格納されている FIX ハッシュ分割の表に対してユーザ用 RD エリアは追加できません。リバランス機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。
3. リバランス機能を使用している表を格納する RD エリアに，ほかの表及びインデクスは格納できません。
4. リバランス機能を使用する FIX ハッシュ分割表に対して，RD エリアを追加した場合，リバランスユティリティを実行し正常終了するまでは，その表を検索，更新する SQL の性能が劣化することがあります。
5. リバランス機能を使用する FIX ハッシュ分割表にユニーク指定のインデクス，ユニーククラスタキー，又は主キーを定義している場合，その表に対して RD エリアを追加すると，リバランスユティリティを実行し正常終了するまでは，その表へのデータの追加，更新はできません。
6. 空き領域の再利用機能を使用した表に RD エリアを追加する場合，追加した RD エリアにも空き領域の再利用機能が適用されます。
7. 表格納用 RD エリアに共用 RD エリアは指定できません。
8. 表定義変更後の総分割 RD エリア数の最大数は重複を含んで 4096 個です。

   ただし，リバランス機能を使用した場合の総分割 RD エリア数の最大数は重複を除いて 1,024 個です。

列名

RD エリアを追加する表に，BLOB 又は BLOB 属性を含む抽象データ型の列がある場合に指定します。

列名には BLOB 型の列，又は BLOB 属性を含む抽象データ型で定義した列を指定してください。

表に定義している BLOB 型の列，又は BLOB 属性を含む抽象データ型の列はすべて指定する必要があります。

LOB 列格納用 RD エリア指定：：＝

```
IN　{LOB 列格納用 RD エリア名
      | (LOB 列格納用 RD エリア名)
      | ((LOB 列格納用 RD エリア名)
        〔, (LOB 列格納用 RD エリア名)〕…)}
```

BLOB 列のデータを格納するユーザ LOB 用 RD エリアの名前を指定します。

指定する RD エリア名は，表格納用 RD エリア名に指定した RD エリアと同じサーバにあるユーザ LOB 用 RD エリアを指定してください。LOB 列格納用 RD エリアは，表定義で指定した表格納用 RD エリアと 1 対 1 に対応するように指定してください。また，表格納用 RD エリアに重複した RD エリアがある場合は，LOB 列格納用 RD エリアも同一位置が重複するように指定してください。上記以外の場合は，LOB 列格納用 RD エリアは重複して指定できません。

抽象データ型定義内 LOB 格納用 RD エリア指定：：＝

ALLOCATE（属性名〔..属性名〕…

IN {LOB 属性格納用 RD エリア名

| (LOB 属性格納用 RD エリア名)

| ((LOB 属性格納用 RD エリア名)

〔,（LOB 属性格納用 RD エリア名)〕…)}

〔,属性名〔..属性名〕…

IN {LOB 属性格納用 RD エリア名

| (LOB 属性格納用 RD エリア名)

| ((LOB 属性格納用 RD エリア名)

〔,（LOB 属性格納用 RD エリア名)〕…)}〕…)

属性名〔..属性名〕

抽象データ型を構成する属性名を指定します。抽象データ型の属性が抽象データ型であり，その入れ子になっている抽象データ型の属性に LOB 型の属性がある場合は，その LOB 型の属性名を指定してください。

属性名は，次のような場合に指定してください。

- 抽象データ型定義の属性

  抽象データ型定義の属性のデータ型が LOB 型であれば指定してください。

- 抽象データ型定義の入れ子

  抽象データ型の属性が抽象データ型であり，その入れ子になっている抽象データ型の属性に LOB 型の属性であれば，その LOB 型の属性名を指定してください。

LOB 属性格納用 RD エリア名

抽象データ型の任意の階層にある，BLOB 属性のデータを格納するユーザ LOB 用 RD エリアの名前を指定します。

LOB 属性格納用 RD エリアについての規則を次に示します。

1. 抽象データ型内にあるすべての BLOB 属性に対して，ユーザ LOB 用 RD エリアを指定してください。
2. 指定する RD エリア名は，表格納用 RD エリア名に指定した RD エリアと同じサーバに定義しているユーザ LOB 用 RD エリアを指定してください。
3. LOB 属性格納用 RD エリアは，表定義で指定した表格納用 RD エリアと 1 対 1 に対応するように指定してください。また，表格納用 RD エリアに重複した RD エリアがある場合は，LOB 属性格納用 RD エリアも同一位置が重複するように指定してください。

   上記以外の場合は，LOB 属性格納用 RD エリアは重複して指定できません。

FOR INDEX インデクス識別子 インデクス格納用 RD エリア指定

〔,インデクス識別子 インデクス格納用 RD エリア指定〕…〕

RD エリアを追加する表にインデクスが定義されている場合，追加する表格納用 RD エリアに対応するインデクス格納用の RD エリアを指定します。

インデクス識別子

RD エリアを追加する表に定義されているインデクスの，インデクス識別子を指定します。

インデクス格納用 RD エリア指定

インデクス格納用 RD エリア指定については，ADD RDAREA のインデクス格納用 RD エリア指定の説明を参照してください。

FOR 〔PRIMARY〕 CLUSTER KEY インデクス格納用 RD エリア指定

RD エリアを追加する表にクラスタキーが定義されている場合，追加する表格納用 RD エリアに対応するクラスタキー格納用の RD エリアを指定します。

PRIMARY

クラスタキーを主キーとして定義している場合に指定します。

インデクス格納用 RD エリア指定

インデクス格納用 RD エリア指定については，ADD RDAREA のインデクス格納用 RD エリア指定の説明を参照してください。

FOR PRIMARY KEY インデクス格納用 RD エリア指定

RD エリアを追加する表に主キーが定義されている場合，追加する表格納用 RD エリアに対応する主キー格納用の RD エリアを指定します。

クラスタキーを構成する列を主キーとして定義している場合，FOR PRIMARY CLUSTER KEY を指定してください。

インデクス格納用 RD エリア指定については，ADD RDAREA のインデクス格納用 RD エリア指定の説明を参照してください。

インデクス格納用 RD エリア指定：：＝

IN｛インデクス格納用 RD エリア名

｜（インデクス格納用 RD エリア名）

｜（(インデクス格納用 RD エリア名)

〔,（インデクス格納用 RD エリア名)〕…)｝

インデクス，クラスタキー，又は主キーを格納する RD エリアの名称を指定します。

CREATE INDEX 形式 1 で定義したインデクス（インデクス構成列のデータ型が抽象データ型以外），CREATE INDEX 形式 3 で定義したインデクス（部分構造インデクス），クラスタキー，及び主キーの場合は，一時表用 RD エリア以外のユーザ用 RD エリアを指定します。ただし，HiRDB/パラレルサーバでは，共用表の場合は共用 RD エリアを指定します。

CREATE INDEX 形式 2 で定義したインデクス（インデクス構成列のデータ型が抽象データ型）の場合は，ユーザ LOB 用 RD エリアを指定します。

インデクス格納用 RD エリアを指定する場合の規則を次に示します。

1. HiRDB/パラレルサーバの場合，追加する表を格納する RD エリアと，それに対応するインデクスを格納する RD エリアは，同じバックエンドサーバになければなりません。
2. サーバ内で横分割しているインデクスは，表を格納する RD エリアとインデクスを格納する RD エリアの数を同じにする必要があります。

3. サーバ内で横分割していないインデクスは，表を格納するバックエンドサーバとインデクスを格納する RD エリアの数を同じにする必要があります。

4. ユーザ用 RD エリアを追加する場合，次の計算式を満たすような RD エリアを指定する必要があります。

   キー長≦(インデクス格納用 RD エリアのページサイズ÷2)－1242

5. インデクス格納用 RD エリアは，表定義で指定した表格納用 RD エリアと 1 対 1 に対応するように指定してください。また，表格納用 RD エリアに重複した RD エリアがある場合は，インデクス格納用 RD エリアも同一位置が重複するように指定してください。

6. インデクス種別によるインデクス格納用 RD エリアの指定可否を次の表に示します。インデクス格納用 RD エリアの指定が必要となっているインデクスについては，必ずインデクス格納用 RD エリアを指定してください。

表 3-3　インデクス種別によるインデクス格納用 RD エリアの指定可否

インデクスの定義方法	インデクスの分割方法		インデクス格納用 RD エリアの指定要否
CREATE INDEX 形式 1，又は形式 3 で定義したインデクス，及び主キー	1 サーバ内だけの横分割※1	サーバ内分割キーインデクス	○
		サーバ内非分割キーインデクス	
	サーバ間だけの横分割※2	サーバ内分割キーインデクス	
		サーバ内非分割キーインデクス	×※5※6
	サーバ内，かつサーバ間の横分割※3	サーバ内分割キーインデクス	○
		サーバ内非分割キーインデクス	
	非分割※4	非分割キーインデクス	×※5
CREATE INDEX 形式 2 で定義したインデクス，及びクラスタキー	1 サーバ内だけの横分割※1		○
	サーバ間だけの横分割※2		
	サーバ内，かつサーバ間の横分割※3		

（凡例）

○：インデクス格納用 RD エリアを指定する必要があります。

×：インデクス格納用 RD エリアを指定する必要はありません。

注※1

HiRDB/シングルサーバでの横分割，又は HiRDB/パラレルサーバで一つのバックエンドサーバに閉じた横分割のことをいいます。

注※2

HiRDB/パラレルサーバで複数のバックエンドサーバにわたった横分割で，かつバックエンドサーバ内ではインデクスを分割していない横分割のことをいいます。

注※3

HiRDB/パラレルサーバで複数のバックエンドサーバにわたった横分割で，かつバックエンドサーバ内でもインデクスを分割している横分割のことをいいます。

注※4

横分割していないインデクスのことをいいます。

注※5

RD エリアの追加によって，表を格納するバックエンドサーバの数が増える場合は，インデクス格納用 RD エリアも指定する必要があります。

注※6

RD エリアを追加しても表を格納するバックエンドサーバの数が変わらない場合，インデクス格納用 RD エリアを指定することもできます。インデクス格納用 RD エリアを指定すると，インデクスはサーバ内の横分割をしたインデクスとなります。

WITH PROGRAM

ハッシュ分割表に RD エリアを追加する場合，その表を使用する手続き，及びトリガの有効な SQL オブジェクトがあれば，その SQL オブジェクトを無効にするときに指定します。

(e) 列属性変更定義

```
: :=CHANGE {列名 { 〔 {VARCHAR (データ長)
                        |NVARCHAR (データ長)
                        |MCHAR ( {*|データ長} )
                        |MVARCHAR ( {*|データ長} ) }
                        〔ARRAY 〔 {*|最大要素数} 〕〕〕
                    | 〔ARRAY 〔最大要素数〕〕
                    |BINARY (データ長) }
                    〔 {NO SPLIT|SPLIT} 〕 〔列回復制約２〕
                    〔 {SET DEFAULT句|DROP DEFAULT} 〕
                    〔WITH DEFAULT〕 〔更新可能列属性〕
                |CLUSTER KEY 〔UNIQUE〕
                |LOCK {ROW|PAGE}
                |HASH ハッシュ関数名
                |SEGMENT REUSE
                        {〔セグメント数〔 {K|M} 〕〕 |NO}
                |SEGMENT REUSE OPTION 再利用オプション値
                |INSERT ONLY
                  〔WHILE 〔日間隔データ|ラベル付き間隔〕 BY 列名〕}
                〔WITH PROGRAM〕
```

列名

定義を変更する場合，列の名称を指定します。

列名についての規則を次に示します。

1. ビュー表定義時にビュー定義文中で次に示す指定をした列は，ALTER TABLE で定義内容を変更できません。

・選択式や探索条件中でスカラ関数の引数として指定した列

・選択式や探索条件中で関数呼出しの引数として指定した列

・選択式や探索条件中で連結演算の演算項として指定した列
・選択式や探索条件中で CASE 式の値式中に指定した列
・選択式や探索条件中で CAST 指定の値式中に指定した列

2. ビュー表定義時にビュー定義文中で次に示す指定をした場合，そのビュー表で指定した表の列は，ALTER TABLE で定義内容を変更できません。
・行副問合せを含む場合
・集合演算を用いたスカラ副問合せ，表副問合せを含む場合
・FROM 句の導出表，又は EXISTS 述語以外で結果の列数が 2 以上の表副問合せを含む場合
・選択式中にスカラ副問合せを含む場合
・比較述語，IN 述語，及び限定述語の左側の行値構成子が，スカラ副問合せを含む場合
・比較述語の右側の行値構成子が，スカラ副問合せを含む場合でかつ行値構成子要素数が 2 以上の場合
・IN 述語の右側の行値構成子の並び中に，スカラ副問合せを含む場合
・BETWEEN 述語，LIKE 述語，XLIKE 述語，SIMILAR 述語，NULL 述語及び論理述語中にスカラ副問合せを含む場合
・値式中にスカラ副問合せを含む場合

3. 列名は，表識別子で指定した実表中の列名を指定してください。

4. 列名の後に，列定義の変更内容を指定するオペランドを一つ以上指定してください。

{VARCHAR（データ長）
| NVARCHAR（データ長）
| MCHAR（{＊｜データ長}）
| MVARCHAR（{＊｜データ長}）
| BINARY（データ長）}

可変長データのデータ長を変更する場合に指定します。又はデータ型を，CHAR から MCHAR に，VARCHAR から MVARCHAR に変更する場合に指定します。

規則を次に示します。

1. 可変長データ型の列の最大長は小さくできません。

2. インデクスの定義された可変長データ型の列の場合，次の計算式を満たさないような変更はできません。
キー長≦
MIN((インデクス格納用 RD エリアのページサイズ÷2)－1242，4036)

3. 最大長を変更しない場合，データ型の指定は省略してください。

4. ビュー定義文中の導出問合せ式に指定した列のうち，次に示す指定をした列は，32,001 バイト以上の BINARY に変更できません。
・比較述語，限定述語，及び IN 述語の副問合せ中に指定した列
・重複排除に指定した列
・グループ分け，又は集合関数に指定した列

・集合演算の対象となる問合せ指定の選択式に指定した列

・内部導出表を作成する条件の，どれかを満たす問合せ指定中で，内部導出表として展開されるビュー表を定義するときに指定した列

内部導出表となる条件については，「内部導出表」を参照してください。

5. CHAR を MCHAR に変更できます。その場合，データ長は変更できません。データ長には，変更前のデータ長，又は＊を指定してください。
6. VARCHAR を MVARCHAR に変更できます。その場合，列の最大長を大きくできます。最大長を変更しない場合，データ長には変更前の最大長，又は＊を指定してください。
7. 文字集合を指定した列は，データ型を変更できません。
8. 次の BINARY 型の列は 32,001 バイト以上に変更できません。
   ・更新可能列属性の UPDATE ONLY FROM NULL を指定している列
   ・データ長の変更と同時に更新可能列属性の UPDATE ONLY FROM NULL を付与する列
9. 予備列のデータ型を CHAR から MCHAR に変更することはできません。

ARRAY ［{＊｜最大要素数}］

繰返し列の最大要素数を大きくする場合に指定します。

規則を次に示します。

1. 最大要素数には，2～30,000 の符号なし整数を指定します。
2. 省略した場合は，繰返し列でないことを示します。
3. 最大要素数を変更しない場合は，＊を指定してください。
4. 最大要素数は小さくできません。

ARRAY ［最大要素数］

繰返し列の最大要素数を大きくする場合に指定します。

最大要素数には，2～30,000 の符号なし整数を指定します。

最大要素数は小さくできません。

{NO SPLIT ｜ SPLIT}

可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）のデータ格納時に，データベース容量を削減したい場合に指定します。

可変長文字列型の定義長を 255 バイト以下から 256 バイト以上に変更する場合で，データベース容量を削減したい場合は，NO SPLIT を指定してください。

NO SPLIT，及び SPLIT の詳細（ノースプリットオプション）については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

NO SPLIT

実データ長が 256 バイト以上の場合に指定します。

- バージョン 09-50 より前の HiRDB で，CREATE TABLE 又は ALTER TABLE ADD によって定義した，定義長が 256 バイト以上である可変長文字列型の列のうち，定義時に NO SPLIT を指定していない列に対して指定します。

- 実表にデータが格納されている場合，指定できません。

SPLIT

実データ長が 255 バイト以下の場合に指定します。

- バージョン 09-50 より前の HiRDB で，CREATE TABLE 又は ALTER TABLE ADD によって定義した，定義長が 255 バイト以下である可変長文字列型の列のうち，定義時に NO SPLIT を指定した列に対して指定します。
- 実表にデータが格納されている場合，指定できません。

列回復制約 2：：＝ RECOVERY {ALL | PARTIAL | NO}

データ型が BLOB の列又は BLOB 属性を含む抽象データ型の列の場合，列に対するデータベースの更新ログ取得方式を変更するときに指定します。ALL，PARTIAL，及び NO の内容は ALTER TABLE ADD の列回復制約 1 と同じです。

{SET DEFAULT 句 | DROP DEFAULT}

SET DEFAULT 句

DEFAULT 句：：＝ DEFAULT 〔既定値〕

既定値を設定又は変更する場合に指定します。規則を次に示します。

1. 指定できる列のデータ型は ALTER TABLE ADD のデフォルト句と同じです。
2. WITH DEFAULT を定義している列に対して SET DEFAULT 句を指定した場合，WITH DEFAULT の指定は無効となり，SET DEFAULT 句の指定が有効となります。
3. 一つの ALTER TABLE で，WITH DEFAULT と同時には指定できません。
4. 最大長を変更する列が既定の文字集合以外の場合，DEFAULT 句に次の項目は指定できません。
   ・CURRENT_DATE(CURRENT DATE)
   ・CURRENT_TIME(CURRENT TIME)
   ・CURRENT_TIMESTAMP(CURRENT TIMESTAMP)
5. 改竄防止表の行削除禁止期間を指定した挿入履歴保持列には指定できません。
6. 予備列に対して SET DEFAULT 句は指定できません。
7. ON ROW EXISTS を指定した既定値は変更できません。

DROP DEFAULT

既定値を削除する場合に指定します。

DROP DEFAULT は，DEFAULT 句を指定している列に対して指定してください。ON ROW EXISTS を指定した既定値は削除できません。

WITH DEFAULT

既定値のない非ナル値制約のある列を，既定値のある非ナル値制約の列に変更する場合に指定します。

WITH DEFAULT についての規則を次に示します。

1. 非ナル値制約のない列には指定できません。

2. 既に既定値のある非ナル値制約の列に対して，WITH DEFAULT は指定できません。
3. DEFAULT 句を定義している列に対して，WITH DEFAULT は指定できません。
4. 一つの ALTER TABLE で，SET DEFAULT 句，又は DROP DEFAULT 句と同時には指定できません。
5. 既定値を持つ列を既定値のない列に変更できません。
6. 繰返し列には指定できません。
7. 抽象データ型の列には指定できません。
8. 予備列には指定できません。

更新可能列属性：：= UPDATE 〔ONLY FROM NULL〕

改竄防止表に変更する前に更新可能列属性を変更する場合に指定します。

更新可能列属性は改竄防止表の場合だけ有効になります。

改竄防止表への変更については，「INSERT ONLY」オプションを参照してください。

更新可能列属性についての規則を次に示します。

1. 改竄防止表に対しては指定できません。
2. SYSTEM GENERATED を指定した列には指定できません。
3. 既に更新可能列属性を指定している列には指定できません。
4. 次に示す更新できない列には指定できません。
   - クラスタキー構成列
   - 分割キー構成列（フレキシブルハッシュ分割表の分割キー構成列を除く）
   - 予備列

UPDATE

改竄防止表に変更後も更新可能な列になります。

UPDATE ONLY FROM NULL

改竄防止表に変更後ナル値から非ナル値へ一度だけ更新できます。

改竄防止表で UPDATE ONLY FROM NULL 指定した列の値の更新可否を次に示します。

更新前の列の値	更新後の列の値	更新可否
ナル値	ナル値	○
ナル値	非ナル値	○
非ナル値	ナル値	×
非ナル値	非ナル値※	×

（凡例）

○：更新できます。

×：更新できません。

注

繰返し列の場合は，ナル値（要素数が 0 の値）から添え字指定なしでの列単位更新だけ実行できます。

注※　更新前の値と同値含む

UPDATE ONLY FROM NULL 指定についての規則を次に示します。

- NOT NULL を指定した列には指定できません。
- FIX 表の場合指定できません。
- 主キー又はクラスタキー構成列には指定できません。
- 分割キー構成列には指定できません。
- BLOB 型及び定義長が 32,001 バイト以上の BINARY 型の列には指定できません。

更新可能列属性の指定可否と指定時の列の値の更新可否は次のようになります。

表種別	UPDATE 指定		UPDATE ONLY FROM NULL 指定		左記の指定なし	
	指定可否	列の値の更新可否	指定可否	列の値の更新可否	指定可否	列の値の更新可否
非改竄防止表	○	○	○	○	－	○
改竄防止表	×	○	×	○※	－	×

（凡例）

○：更新できます。

×：更新できません。

－：該当しません。

注※

ナル値から非ナル値へ一度だけ更新できます。

CLUSTER KEY UNIQUE

一意性制約のないクラスタキーを一意性制約のあるクラスタキーに変更する場合に指定します。

CLUSTER KEY UNIQUE についての規則を次に示します。

1. 実表にデータが入っている場合，クラスタキーの属性は変更できません。
2. 表定義で一意性制約のあるクラスタキーを定義している場合は指定できません。
3. フレキシブルハッシュ分割の表に対しては指定できません。

CLUSTER KEY

一意性制約のあるクラスタキーを一意性制約のないクラスタキーに変更する場合に指定します。

実表にデータが入っている場合，クラスタキーの属性は変更できません。

表定義で一意性制約のないクラスタキーを定義している場合は指定できません。

LOCK {ROW | PAGE}

表の最小排他資源単位を変更する場合に指定します。

行単位に変更する場合は LOCK ROW を，ページ単位に変更する場合は LOCK PAGE を指定してください。

HASH ハッシュ関数名

表をハッシュ関数によって横分割している場合に，ハッシュ関数を変更するときに指定します。データが格納されている FIX ハッシュ分割の表に対しては変更できません。

リバランス機能を使用する表のハッシュ関数に，HASH1～HASH6，HASHZ，HASH0 は指定できません。また，リバランス機能を使用しない表のハッシュ関数に，HASHA～HASHF は指定できません。

SEGMENT REUSE {〔セグメント数〔{K | M}〕〕 | NO}

セグメント数を変更します。空き領域を再利用する処理はオーバヘッドが掛かるため，性能よりも格納効率を優先するシステムにだけ適用してください。

空き領域の再利用機能については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

セグメント数〔{K | M}〕

空き領域の再利用機能を使用し，かつ該当する表のセグメント数に上限を設定する場合，その上限のセグメント数を指定します。セグメント数は，1～268,435,440 の符号なし整数で指定できます。また，単位として，K（キロ），又は M（メガ）を指定できますが，268,435,440 を超える値は指定できません。

行の挿入，削除を頻繁に行う表に対して，行の挿入性能の向上，及び指定したセグメント内での格納効率の向上が期待できます。

セグメント数の指定なし

空き領域の再利用機能を使用し，かつ該当する表のセグメント数に上限を設定しない場合，セグメント数を省略します。

行の挿入，削除を頻繁に行う表で，かつ RD エリアに格納される表が該当する表だけの場合に適用してください。行の挿入性能の向上，及び該当する表を格納する RD エリア内での，空き領域の格納効率の向上が期待できます。

NO

空き領域の再利用機能を使用しない場合に指定します。

行の挿入，削除を頻繁に行わない表に対しては NO を指定してください。

SEGMENT REUSE についての規則を次に示します。

1. 空き領域の再利用機能は，LOB 列，LOB 属性の抽象データ型列，及びインデクスに対しては無効となります。
2. 空き領域の再利用機能は，リバランス表に対しては指定できません。

SEGMENT REUSE OPTION 再利用オプション値

再利用オプション機能の適用範囲を変更する場合に指定します。この機能を適用している場合，空き領域の再利用機能を使用したときのデータ格納効率が向上します。空き領域の再利用機能が非適用の表に対しては指定できません。

再利用オプション値には0～3を値として指定できます。0を指定した場合は，再利用オプション機能を適用しません。なお，このオプションの推奨値は3です。

それぞれを指定すると，次の場合に格納効率の向上が期待できます。

各機能の再利用オプション値	機能	格納効率の向上が期待できるケース
1	UPDATE対応	• 固定長データ，ADT列のNULL値からのUPDATEがある • BINARY列のUPDATEがある • VARCHAR列，NVARCHAR列，MVARCHAR列（NO SPLIT指定）のUPDATEがある • 確保済みセグメント数に比べてサーチモードの切り替え回数が少ない（pddbstコマンドで確認できる）
2	分岐行が多発する表の格納効率向上	分岐行を作成する表を運用するケース
3	UPDATE対応及び分岐行が多発する表の格納効率向上	• 固定長データ，ADT列のNULL値からのUPDATEがある • BINARY列のUPDATEがある • VARCHAR列，NVARCHAR列，MVARCHAR列（NO SPLIT指定）のUPDATEがある • 確保済みセグメント数に比べてサーチモードの切り替え回数が少ない（pddbstコマンドで確認できる） • 分岐行を作成する表を運用するケース

INSERT ONLY 〔WHILE {日間隔データ｜ラベル付き間隔} BY 列名〕

改竄防止表に変更する場合に指定します。改竄防止表については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

改竄防止表とした場合，その表の値は更新できません。ただし，更新可能列は更新できます。

改竄防止表には，行の削除を禁止する期間（行削除禁止期間）を指定できます。行削除禁止期間を指定する場合，WHILEで期間を指定し，列名には挿入履歴保持列（DATE型の列で，かつSYSTEM GENERATEDの列）を指定してください。削除禁止期間を指定しない場合，永久的に行の削除はできません。

日間隔データ

行削除禁止期間を，日間隔データの10進数表現で指定します。日間隔データの10進数表現については，「日間隔データの10進数表現」を参照してください。

なお，日間隔データは正の値だけ指定できます。

ラベル付き間隔

行削除禁止期間を，ラベル付き間隔で指定します。ラベル付き間隔については，「日付演算」を参照してください。

ラベル付き間隔の値式には，括弧で囲まれていない正の整数定数だけ指定できます。

列名

DATE 型で，かつ SYSTEM GENERATED の列を指定してください。

行削除禁止期間には行を挿入した日を含み，行削除禁止期間の計算は，「日付演算」の「日付データと日間隔データとを加減算する場合の規則」に従います。削除禁止最終日付と削除可能日付は，それぞれ次の演算結果となります。

- 削除禁止最終日付＝行挿入日付＋削除禁止期間－1 日
- 削除可能日付＝行挿入日付＋削除禁止期間

行挿入日付と削除禁止期間の指定値での，削除禁止最終日付と削除可能日付の関係については，「削除禁止最終日付と削除可能日付の関係」を参照してください。

表を改竄防止表に変更する場合，次に示す制限があります。

1. 既に改竄防止表となっている表は指定できません。
2. 外部キーを定義した表は改竄防止表に変更できません。
3. 検査制約を定義した表は改竄防止表に変更できません。
4. すべての列に更新可能列属性を指定している表は改竄防止表に変更できません。
5. インナレプリカ機能を適用した RD エリアに表を定義している場合，その表を改竄防止表に変更できません。
6. 指定した表に行がある場合，その表を改竄防止表に変更できません。
7. 予備列を定義した表を，改竄防止表とすることはできません。

WITH PROGRAM

次の場合に指定します。

- 列名の変更，DEFAULT 句の既定値の変更，表の排他制御モード変更，ハッシュ分割した表のハッシュ関数の変更，SEGMENT REUSE の変更，又は改竄防止表に変更をする場合に，その表を使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいとき
- クラスタキーの一意性制約属性の変更をする場合に，そのクラスタキーを使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいとき

(f)　列削除定義：：＝ DROP 列名〔WITH PROGRAM〕

列名

列を削除する場合に指定します。

列名についての規則を次に示します。

1. 列名は，表識別子で指定した実表中の列名にしてください。
2. 実表にデータが格納されている場合，列は削除できません。
3. 列が 1 個の場合は，削除できません。
4. 列を削除すると，その列に対するインデクス，注釈，及び削除する列を使用しているビュー表（パブリックビュー表を含む）も削除されます。

5. クラスタキー又は主キーを定義したインデクスを構成する列は削除できません。
6. 列のデータ型が BLOB の場合は削除できません。
7. 抽象データ型列がある表の列は指定できません。
8. 改竄防止表の挿入履歴保持列の場合，その列は削除できません。
9. 改竄防止表の列を削除した結果，すべての列が更新可能列となる場合，その列は削除できません。
10. 列を削除した結果，予備列だけの表になる場合，その列は削除できません。
11. 予備列を削除すると，その予備列を持つ表を使用しているビュー表（パブリックビュー表を含む）も削除されることがあります。

WITH PROGRAM

表中の列を削除するときに，その表を使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

(g) 表制約定義削除：：＝ DROP PRIMARY KEY〔WITH PROGRAM〕

PRIMARY KEY

主キーを削除する場合に指定します。

主キーを削除する場合の規則を次に示します。

1. 表に主キーが定義されていない場合は指定できません。
2. 参照している外部キーが存在する場合，主キーは削除できません。
3. クラスタキーとして定義した主キー（PRIMARY CLUSTER KEY）は削除できません。

WITH PROGRAM

主キーを削除するときに，その主キーを使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

WITH PROGRAM を省略した場合，その主キーを使用する手続き及びトリガの有効な SQL オブジェクトがあると，その主キーは削除できません。

(h) 表名変更定義：：＝ RENAME TABLE TO 表識別子〔WITH PROGRAM〕

表の名称を変更する場合に指定します。

表識別子

変更後の実表の名称を指定します。

表識別子についての規則を次に示します。

1. インデクスが付いている表の名称も変更できます。
2. ビュー表の名称は変更できません。
3. 次に示す表の名称は変更できません。
   - ビュー表の基になる表
   - CREATE PROCEDURE の SQL 手続き文中に指定した表

・CREATE TRIGGER の SQL 手続き文中に指定した表，及びトリガを定義した表

4. スキーマ内にある実表及びビュー表と同じ名称には，変更できません。

5. 改竄防止表の名称は変更できません。

WITH PROGRAM

表識別子の名称を変更する場合に，その表を使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。ただし，WITH PROGRAM を指定しても，トリガを定義した表の表名は変更できません。

(i) 列名変更定義：：＝ RENAME COLUMN FROM 変更前列名 TO 変更後列名〔WITH PROGRAM〕

列の名称を変更する場合に指定します。

変更前列名，変更後列名

名称を変更する列の，変更前と変更後の名称を指定します。

変更前列名，変更後列名についての規則を次に示します。

1. インデクスのキー列の名称も変更できます。

2. ビュー表の列名は変更できません。

3. 次に示す列の名称は変更できません。

・ビュー表の基になる表の列

・CREATE PROCEDURE の SQL 手続き文中に指定した表の列

・CREATE TRIGGER の SQL 手続き文中に指定した表の列，トリガ契機列，及びトリガ動作条件中に指定した列

4. 既に表にある列名には変更できません。

5. 変更前列名が表にない場合は，変更できません。

6. 改竄防止表の列名は変更できません。

WITH PROGRAM

列の名称を変更するときに，その列を含む表を使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。ただし，WITH PROGRAM を指定しても，次の列の列名は変更できません。

- トリガ契機列
- トリガ動作条件中で新旧値相関名を使用して参照している列
- トリガ SQL 文中で新旧値相関名を使用して参照している列

(j) 分割格納条件変更定義

```
：：＝CHANGE RDAREA {横分割表変更指定｜マトリクス分割表変更指定}
  〔LOB列格納用RDエリア変更指定〕
  〔インデクス格納用RDエリア変更指定
```

```
    〔インデクス格納用RDエリア変更指定〕…〕
  〔クラスタキー格納用RDエリア変更指定〕
  〔主キー格納用RDエリア変更指定〕
  〔WITHOUT PURGE〕〔WITHOUT PROGRAM〕
```

横分割された表やマトリクス分割された表の分割格納条件を変更する場合に指定します。

次に示す表は，分割格納条件を変更できません。

- 抽象データ型の列を持つ表
  本文中では，データ型に BLOB 型を指定した列を”LOB 列”と記述します。
  一つの ALTER TABLE 文での，表の分割格納条件の分割，統合の単位は次のとおりです。

表種別	表の分割方法		実行種別	
			分割	統合
横分割表	キーレンジ分割	境界値指定	任意の 1 個の境界値を 2～16 個に分割します。	任意の 2～16 個の境界値を 1 個に統合します。
		格納条件指定	任意の 1 個の RD エリアを 2～16 個に分割します。	任意の 2～16 個の RD エリアを 1 個に統合します。
マトリクス分割表	キーレンジ分割	境界値指定	一つの次元の 1 個の境界値を 2～16 個に分割します。結果として，n 個の境界値を 2n～16n 個の境界値に分割します。	一つの次元の 2～16 個の境界値を 1 個に統合します。結果として，2n～16n 個の境界値を n 個の境界値に統合します。

（凡例）
n：分割又は統合する表の分割格納条件に該当する別次元の分割数

表の分割格納条件を分割・統合する場合の SQL 例については，「使用例」の 9.（境界値指定の横分割表の場合），10.（格納条件指定の横分割表の場合），又は 11.（マトリクス分割表の場合）を参照してください。また，分割格納条件を変更した場合，変更対象の RD エリアに格納されているデータは，ALTER TABLE 実行時にシステムが削除します（「WITHOUT PURGE」オプションを指定した場合，変更対象のデータを，削除する範囲が変わる場合があります。詳細は「WITHOUT PURGE」オプションを参照してください)。このため，ALTER TABLE 実行後も使用するデータは，データを格納し直す必要があります。分割格納条件の変更を行う前にデータをアンロードし，分割格納条件の変更後にデータのロードを行うなどの操作が必要です。削除するデータを格納し直す方法については，マニュアル「HiRDB システム運用ガイド」の「表の分割格納条件を変更するときの運用」を参照してください。

表の分割格納条件の変更時の規則を次に示します。

《表の分割格納条件の変更前の確認事項》
表の分割格納条件の変更前の確認事項を次に示します。

1. 分割格納条件を変更する場合，HiRDB Advanced High Availability が必要です。
2. 表の種別と分割方法によって，分割格納条件を変更できない場合があります。表の種別と分割方法による変更の可否を次に示します。

表種別	分割方法			変更の可否
横分割表	キーレンジ分割	格納条件指定	格納条件の比較演算子に=だけを用いた表	○
			格納条件の比較演算子に=以外を用いた表	×
		境界値指定		○
	ハッシュ分割			×
マトリクス分割表	第 1 次元：境界値指定 第 2 次元：境界値指定			○
	第 1 次元：境界値指定 第 2 次元：ハッシュ分割			○※2
非分割表※1	−			×

（凡例）

○：変更できます。

×：変更できません。

−：該当しません。

注※1　分割格納条件を指定していない表です。

注※2　境界値指定の次元だけ変更できます。ハッシュ分割の次元は変更できません。

3. 表格納用 RD エリア，インデクス格納用 RD エリア及び LOB 列格納用 RD エリアが，1 対 1 の関係にない場合，分割格納条件を変更できません。

4. 格納条件指定の横分割表，又は境界値指定のキーレンジ分割の組み合わせのマトリクス分割表の場合，次の条件をすべて満たすときは分割格納条件を変更できません。

・変更対象の表にインデクス，主キー，又はクラスタキーが定義されている

・変更対象の表又は変更した結果の表の表格納用 RD エリアが一つである

5. 改竄防止表の分割格納条件は，変更できません。

6. インナレプリカ機能を使用している場合の確認事項を次に示します。

・表，インデクス，主キー，クラスタキー，LOB 列を格納する RD エリアの世代数をすべて同じにする必要があります。

・表に参照制約を定義している場合，参照表及び被参照表を格納する RD エリアの世代数を，すべて同じにする必要があります。

《表の分割格納条件の変更時の規則》

表の分割格納条件の変更時の規則を次に示します。

1. 1 回の ALTER TABLE で分割格納条件の分割と統合を同時に実行できません。2 回の ALTER TABLE に分けて実行する必要があります。

2. 変更対象の表の RD エリア（横分割表変更指定，又はマトリクス分割表変更指定の変更前境界値リストで特定する RD エリア，又は変更前 RD エリア情報リストで指定する RD エリア），インデクス格納用 RD エリア及び LOB 列格納用 RD エリアは，1 対 1 の関係にする必要があります。
3. 表，インデクス，主キー，クラスタキー，及び LOB 列は，表を基準に分割数や RD エリアの重複の対応関係を維持する必要があります（例えば，表の変更前後で，システムが境界値を一つにまとめる場合，及び指定する RD エリアが変更対象の境界値以外のデータを格納している場合，変更前後で同じ RD エリアを使用し，同じ RD エリアを指定する位置などをインデクス，主キー，クラスタキー，及び LOB 列も同じように指定する必要があります。詳細については，マニュアル「HiRDB システム運用ガイド」の「表の分割格納条件の変更」を参照してください)。
4. 格納条件指定の横分割表の場合，表定義変更後の表格納用 RD エリア名は重複しないようにしてください。
5. 表定義変更後の総分割 RD エリア数の最大数は重複を含んで 4,096 個です。
6. 表定義変更後の総格納条件指定数（格納条件を指定しない RD エリアも含む）の最大数は 15000 個です。
7. 変更対象として指定した RD エリアの表のデータは，ALTER TABLE 実行時にシステムが削除します（その RD エリアに対応するインデクス格納用 RD エリア，及び LOB 列格納用 RD エリアのデータもシステムが削除します)。また，インナレプリカ機能を使用している場合，変更対象の表，インデクス，主キー，クラスタキー，及び LOB 列（LOB 型属性列含む）を格納するすべての世代のレプリカ RD エリアのデータも ALTER TABLE 実行時にシステムが削除します（変更時の注意事項の 4 を参照してください)。
8. システム定義の pd_check_pending オペランドに USE を指定した場合，変更対象の表が被参照表のとき，被参照表を参照する参照表に対し，ディクショナリ表と RD エリア中の表情報を検査保留状態に設定します。また，インナレプリカ機能を使用している場合は，すべての世代で RD エリア中の表情報を検査保留状態に設定します。
9. 検査保留状態の表の分割格納条件を変更した場合の，検査保留状態の解除に関する規則を次に示します。

   <<RD エリアの表情報中に設定された検査保留状態>>

   データ削除対象の RD エリアだけ，変更対象表の検査保留状態を解除します。

   <<ディクショナリ表中に設定された検査保留状態>>

   システム定義の pd_check_pending オペランドに USE を指定している場合に，検査保留状態を解除するディクショナリ表を次の表に示します。NOUSE を指定している場合は，検査保留状態を解除するディクショナリ表はありません。

表 3-4　システム定義の pd_check_pending オペランドに USE を指定している場合に，検査保留状態を解除するディクショナリ表

変更対象表に定義されている制約	表情報に検査保留状態が設定された RD エリアの有無※	検査保留状態を解除するディクショナリ表	
参照制約	なし	SQL_REFERENTIAL_CONSTRAINTS 表	CHECK_PEND 列

変更対象表に定義されている制約	表情報に検査保留状態が設定された RD エリアの有無※	検査保留状態を解除するディクショナリ表	
		SQL_TABLES 表	CHECK_PEND 列
検査制約	なし	SQL_CHECKS 表	CHECK_PEND2 列
		SQL_TABLES 表	CHECK_PEND2 列
参照制約及び検査制約	参照制約：なし 検査制約：なし	SQL_REFERENTIAL_CONSTRAINTS 表	CHECK_PEND 列
		SQL_CHECKS 表	CHECK_PEND2 列
		SQL_TABLES 表	CHECK_PEND 列, CHECK_PEND2 列
	参照制約：なし 検査制約：あり	SQL_REFERENTIAL_CONSTRAINTS 表	CHECK_PEND 列
		SQL_TABLES 表	CHECK_PEND 列
	参照制約：あり 検査制約：なし	SQL_CHECKS 表	CHECK_PEND2 列
		SQL_TABLES 表	CHECK_PEND2 列
制約が定義されていない	なし	－	－

（凡例）

－：該当しません。

注※　インナレプリカ機能を使用している場合は，すべての世代の変更対象表の表格納用 RD エリアを示します。

1. 変更対象のデータを削除する範囲を次に示します（「WITHOUT PURGE」オプションを指定した場合，変更対象のデータを，削除する範囲が変わることがあります。詳細については，「WITHOUT PURGE」オプションを参照してください）。

   境界値指定の横分割表の場合：

   ・変更前境界値リストに指定した境界値で特定する RD エリアの表，インデクス，主キー，クラスタキー，及び LOB 列のデータ。

   ・変更前境界値リストに指定した境界値で特定する RD エリアが，変更対象以外の境界値のデータを格納する RD エリアと重複していた場合，変更対象以外の境界値のデータ。

   ・インナレプリカ機能を使用している場合，変更前境界値リストに指定した境界値で特定する RD エリアの全世代のレプリカ RD エリアの表，インデクス，主キー，クラスタキー，及び LOB 列のデータ。

   格納条件指定の横分割表の場合：

   ・変更前 RD エリア情報リストに指定した RD エリアの表，インデクス，主キー，クラスタキー，及び LOB 列のデータ。

   ・インナレプリカ機能を使用している場合，変更前 RD エリア情報リストに指定した RD エリアの全世代のレプリカ RD エリアの表，インデクス，主キー，クラスタキー，及び LOB 列のデータ。

マトリクス分割表の場合：

・変更対象列名と変更前境界値リストに指定した境界値で特定する複数の RD エリアの表，インデクス，主キー，クラスタキー，及び LOB 列のデータ。

・変更前境界値リストに指定した境界値で特定する RD エリアが，変更対象以外の境界値のデータを格納する RD エリアと重複していた場合，変更対象以外の境界値のデータ。

・インナレプリカ機能を使用している場合，変更前境界値リストに指定した境界値で特定する RD エリアの全世代のレプリカ RD エリアの表，インデクス，主キー，クラスタキー，及び LOB 列のデータ。

2. 境界値指定の横分割表又はマトリクス分割表の場合，RD エリアが重複（隣り合っていない境界値を格納する RD エリアが同じ RD エリアの場合）又は連続（隣り合った境界値を格納する RD エリアが同じ RD エリアの場合）するとき，境界値指定の横分割表とマトリクス分割表で次のようになります。詳細については，マニュアル「HiRDB システム運用ガイド」を参照してください。

   境界値指定の横分割表の場合を次に示します。

項目		境界値指定の横分割表
ALTER TABLE に指定する変更後の RD エリア※	重複	重複してもかまいません。
	連続	連続しないようにしてください。
分割格納条件を変更した結果の表全体の RD エリア	重複	重複してもかまいません。
	連続	分割時は，連続した RD エリアをシステムが一つにまとめます。
		統合時は指定できません。

   マトリクス分割表の場合を次に示します。

項目		境界値指定の横分割表
ALTER TABLE に指定する変更後の RD エリア※	重複	重複してもかまいません。
	連続	連続してもかまいません。
分割格納条件を変更した結果の表全体の RD エリア	重複	重複してもかまいません。
	連続	連続した RD エリアをシステムは一つにまとめません。

   注※

   変更後境界値分割指定，又はマトリクス分割表格納用 RD エリア変更指定に指定した RD エリアです。

3. 統合した結果，表格納用 RD エリアが次の状態になる指定はできません。

   ・表の分割数が 1 個になる場合

   ・一つの次元の分割数が 1 個になる場合

4. 変更後の RD エリアに共用 RD エリアは指定できません。

5. 分割格納条件を変更する表を使用する手続き，及びトリガを定義している場合，「WITH PROGRAM」を指定して手続き，及びトリガを無効化する必要があります。

   分割格納条件を変更する場合の注意を次に示します。

《変更時の注意事項》

1. 分割格納条件を変更する表を使用する手続き，及びトリガが定義され，「WITH PROGRAM」を指定して手続き，及びトリガを無効化した場合，定義無効化になった手続き，及びトリガは ALTER ROUTINE で再作成する必要があります（手続きの場合，ALTER PROCEDURE，トリガの場合，ALTER TRIGGER で再作成することもできます）。
2. 変更対象の境界値で特定する RD エリアが，変更対象以外の境界値のデータを格納する RD エリアと重複していた場合，システムは変更対象の RD エリアの表のデータをすべて削除するため，変更対象以外の境界値のデータも削除します。
3. 変更対象の表のデータを変更した結果の表に格納し直す場合，及び障害に備えてデータのバックアップを取得する場合，変更する前のデータをデータベース再編成ユティリティ（pdrorg）でアンロードし，変更した結果の表にロードする運用方法や，データベース複写ユティリティ(pdcopy)で RD エリアのバックアップを取得する運用方法などが必要です。詳細については，マニュアル「HiRDB システム運用ガイド」の「表の分割格納条件を変更するときの運用」を参照してください。
4. 変更する表が被参照表で，表のデータを削除する場合，参照表，及び被参照表の整合性を保てなくなります。この場合，表の分割格納条件の変更を行い，表のデータを変更した結果の表に格納し直した後に，参照表と被参照表の整合性を確認してください。変更後の表に格納し直す方法については，変更時の注意事項の項番 3 を，確認方法については，マニュアル「HiRDB システム運用ガイド」の「表の分割格納条件を変更するときの運用」を参照してください。
5. インナレプリカ機能を使用している場合，参照表，及び被参照表を格納する RD エリアの世代数をすべて同じにしていなかったとき，参照表，及び被参照表の整合性を保てなくなります。参照表，及び被参照表の整合性が保てなくなった場合，RD エリアのバックアップからデータを回復するなどの運用が必要です。運用方法については，マニュアル「HiRDB システム運用ガイド」の「表の分割格納条件を変更するときの運用」を参照してください。
6. システム定義の pd_check_pending オペランドに USE を指定した場合，分割格納条件を変更する表を被参照表とする参照表がほかのユーザによって使用されていたとき，そのトランザクションが終了してから参照表を検査保留状態に設定します。

(k) 横分割表変更指定

```
::= { 〔PARTITIONED〕変更前境界値リスト INTO 変更後境界値分割指定
    | PARTITIONED CONDITION 変更前RDエリア情報リスト
      INTO 変更後格納条件分割指定}
```

境界値指定又は格納条件指定の横分割表の，分割格納条件を変更する場合に指定します。

PARTITIONED

境界値指定の横分割表の分割格納条件を変更する場合に指定します。横分割表変更指定の規則を次に示します。

＜分割時＞

1. 一つの SQL で，一つの境界値，及び境界値に対応した RD エリアを分割できます。
2. 変更前境界値リストに指定した境界値を，変更後境界値分割指定に従って分割します。

＜統合時＞

1. 一つの SQL で，連続した複数の境界値，及び境界値に対応した RD エリアを統合できます。
2. 変更前境界値リストに指定した境界値を，変更前境界値リストに指定した最大の境界値，及び変更後境界値分割指定に指定した RD エリアに統合します。

PARTITIONED CONDITION

格納条件指定の横分割表の分割格納条件を変更する場合に指定します。横分割表変更指定の規則を次に示します。

＜分割時＞

1. 一つの SQL で，格納条件に対応した一つの RD エリアを分割できます。
2. 変更前 RD エリア情報リストに指定した RD エリアを，変更後格納条件分割指定に従って分割します。

＜統合時＞

1. 一つの SQL で，格納条件に対応した複数の RD エリアを統合できます。
2. 変更前 RD エリア情報リストに指定した RD エリアを，変更後格納条件分割指定に指定した RD エリアに統合します。

マトリクス分割表変更指定：：＝

MULTIDIM（変更対象列名　変更前境界値リスト）
AT 変更後境界値リスト
INTO マトリクス分割表格納用 RD エリア変更指定

マトリクス分割表の分割格納条件を変更する場合に指定します。

マトリクス分割表格納用 RD エリア変更指定についての規則を次に示します。

＜分割時＞

1. 一つの SQL で，任意の一つの次元の境界値，及び境界値に対応した RD エリア群を分割できます。
2. 変更対象列名で特定した次元の，変更前境界値リストで特定した RD エリア群を，変更後境界値リストに指定した境界値，及びマトリクス分割表格納用 RD エリア変更指定に指定した RD エリア群に分割します。

＜統合時＞

1. 一つの SQL で，任意の一つの次元の連続した複数の境界値，及び複数の RD エリア群を統合できます。
2. 変更対象列名で特定した次元の，変更前境界値リストで特定した RD エリア群を，変更後境界値リストに指定した境界値で，マトリクス分割表格納用 RD エリア変更指定に指定した RD エリア群に統合します。

MULTIDIM（変更対象列名 変更前境界値リスト）

マトリクス分割表の変更対象の分割列名，及び境界値を指定します。

変更対象列名

変更対象の次元の分割列名を指定します。

変更前境界値リスト：：＝（（{境界値｜ MAX}）〔,（{境界値｜ MAX}）〕…）

変更対象の境界値を指定します。

AT 変更後境界値リスト

INTO マトリクス分割表格納用 RD エリア変更指定

変更後の境界値及び表を格納用する RD エリアを指定します。

(l) 変更前境界値リスト：：＝境界値リスト，変更後境界値リスト：：＝境界値リスト

変更前境界値リスト：：＝境界値リスト

境界値リスト：：＝（（{境界値｜ MAX}）〔,（{境界値｜ MAX}）〕…）

変更前の境界値を指定します。指定した境界値から変更対象の RD エリアを特定します。

変更前境界値リストについての規則を次に示します。

＜分割時＞

1. 境界値又は MAX を 1 個指定できます。
2. 表に定義していない境界値は指定できません。
3. 表に定義した最大の境界値より大きい分割キー値に対しては，境界値の代わりに MAX を指定してください。

＜統合時＞

1. 境界値及び MAX を 2～16 個指定できます。
2. MAX は 1 個指定できます。
3. 表に定義していない境界値は指定できません。
4. 表に定義した最大の境界値より大きい分割キー値に対しては，境界値の代わりに MAX を指定してください。
5. 連続した境界値を，昇順に指定してください。
6. 同じ境界値を複数指定できません。

境界値

変更する前の表に定義してある境界値を指定します。変更対象の境界値及び RD エリアを特定するために指定します。

MAX

変更する前の表に定義してある最大の境界値を超えた範囲を変更対象にする場合に指定します。

変更後境界値リスト：：＝境界値リスト

境界値リスト：：＝（（{境界値｜ MAX}）〔,（{境界値｜ MAX}）〕…）

変更後の境界値を指定します。

変更後境界値リストの規則を次に示します。

＜分割時＞

1. 境界値及び MAX を 1～16 個指定できます。
2. MAX は 1 個指定できます。
3. 境界値には定数を指定してください。
4. 連続した境界値を昇順に指定してください。
5. 最後に指定する境界値は，変更前境界値リストで指定した値を指定してください。
6. 変更前境界値リストで指定した境界値の範囲を超える境界値は指定できません。
7. 境界値に指定できる値については，「CREATE TABLE（表定義）」を参照してください。

＜統合時＞

1. 境界値又は MAX を 1 個指定できます。
2. 指定する境界値は，変更前境界値リストで最後に指定した値を指定する必要があります。
3. 変更した結果の表で，一つの次元の分割数が一つになる指定はできません。

境界値

変更後の境界値を指定します。

MAX

変更した結果の表の，最後の境界値の範囲を指定します。

(m)　変更後境界値分割指定

```
：：＝｛表格納用RDエリア名
　　｜（表格納用RDエリア名）
　　｜（〔（表格納用RDエリア名）境界値，〕…（表格納用RDエリア名））｝
```

変更前境界値リスト（(j)）で指定した境界値及び変更対象の RD エリアを，指定した境界値及び RD エリアに変更します。指定した境界値を分割キーに持つデータは，直前に指定した RD エリアに格納します。

＜分割時＞

1. 表格納用 RD エリア名は 2～16 個指定できます。
2. 最後の RD エリアに対しては，境界値を指定できません。
3. 境界値は 1～15 個指定できます。
4. 境界値には定数を指定してください。
5. 境界値は昇順に指定してください。
6. 変更前境界値リストで指定した，変更対象の境界値を分割した最後の範囲を格納する RD エリアの場合，RD エリアの直後に境界値を指定しない形式で RD エリア名を記述してください。
7. 変更前境界値リストで指定した変更対象の範囲を超える境界値は指定できません。

8. 表格納用 RD エリア名には，同一の RD エリアを複数指定できます。ただし，同一の RD エリアを連続して指定できません。
9. 境界値に指定できる値については，「CREATE TABLE（表定義）」を参照してください。

＜統合時＞

1. 表格納用 RD エリア名は 1 個指定できます。
2. 境界値は指定できません。
3. 変更した結果の表で，格納する RD エリアが一つになる指定はできません。

＜注意事項＞

分割した結果，隣り合った境界値を格納する RD エリアが，同じ RD エリアだった場合，システムが境界値を最大の境界値にまとめます。

表格納用 RD エリア名

変更後にデータを格納する RD エリアを指定します。ただし，一時表用 RD エリアは指定できません。

境界値

分割時に変更後の境界値を指定します。統合時は指定できません。

(n) 変更前 RD エリア情報リスト

```
：：＝｛表格納用RDエリア名
        ｜（表格納用RDエリア名）
        ｜（（表格納用RDエリア名）
            〔，（表格納用RDエリア名）〕…〔，OTHERS〕）
        ｜OTHERS｝
```

変更前の表格納用 RD エリア名を指定します。指定した表格納用 RD エリア名から変更対象の格納条件を特定します。

変更前 RD エリア情報リストについての規則を次に示します。

＜分割時＞

1. 表格納用 RD エリア名又は OTHERS を 1 個指定できます。
2. 表に指定していない表格納用 RD エリア名は指定できません。
3. 格納条件に指定している定数が一つだけの表格納用 RD エリア名は指定できません。ただし，格納条件を指定していない表格納用 RD エリア名は指定できます。
4. 格納条件を指定している表格納用 RD エリア名を指定した場合，RD エリア内の格納条件をほかの RD エリアに分割し，新たに格納条件を追加することはできません。
5. 格納条件を指定していない表格納用 RD エリア名を指定した場合，新たに格納条件を追加できます。
6. 表定義を変更する表に格納条件を指定していない表格納用 RD エリアが存在しない場合，格納条件を追加するときは，表格納用 RD エリア名の代わりに OTHERS を指定してください。
7. 格納条件を指定していない表格納用 RD エリアが存在する場合，OTHERS は指定できません。

＜統合時＞

1. 表格納用 RD エリア名又は OTHERS を 2～16 個指定できます。
2. OTHERS は 1 個指定できます。
3. 表に指定していない表格納用 RD エリア名は指定できません。
4. 同じ表格納用 RD エリア名は複数指定できません。
5. 格納条件を指定していない表格納用 RD エリア名を含んだ指定をした場合，格納条件を削除できます。
6. 格納条件を指定していない表格納用 RD エリアが存在しない場合，格納条件を削除するときは，表格納用 RD エリア名の代わりに OTHERS を指定してください。
7. 格納条件を指定していない表格納用 RD エリアが存在する場合，OTHERS は指定できません。

表格納用 RD エリア名

変更する前の表に指定してある表格納用 RD エリア名を指定します。変更対象の格納条件を特定するために指定します。

OTHERS

変更する表に格納条件を指定していない表格納用 RD エリアが存在しない場合に，格納条件を追加又は削除するときに指定します。

(o) 変更後格納条件分割指定

```
：：＝｛表格納用RDエリア名
        ｜（表格納用RDエリア名）
        ｜（（表格納用RDエリア名）格納条件
             ｛，（表格納用RDエリア名）格納条件
               〔〔，（表格納用RDエリア名）格納条件〕…〕
               〔｛，（表格納用RDエリア名）
                   ｜，OTHERS｝〕）
              ｜，（表格納用RDエリア名））
              ｜，OTHERS）｝
        ｜OTHERS｝
  格納条件：：＝列名＝｛定数｜（定数〔，定数…〕）｝
```

変更前 RD エリア情報リストで指定した変更対象の RD エリアを，指定した格納条件及び RD エリアに変更します。

変更後格納条件分割指定についての規則を次に示します。

＜分割時＞

1. 表格納用 RD エリア名又は OTHERS を 2～16 個指定できます。
2. OTHERS は 1 個指定できます。
3. 格納条件を指定しない表格納用 RD エリア名の指定は 1 個指定できます。

＜変更前 RD エリアリストに格納条件を指定している RD エリア名を指定した場合＞

1. 格納条件を指定しない RD エリア名及び OTHERS を指定できません。

＜変更前 RD エリアリストに格納条件を指定していない RD エリア名を指定した場合＞

1. 格納条件を指定しない RD エリア名又は OTHERS を含めて指定してください。
2. 格納条件を指定していない RD エリアを削除する場合，OTHERS を指定します。

＜変更前 RD エリアリストに OTHERS を指定した場合＞

1. 格納条件を指定しない RD エリア名又は OTHERS を含めて指定してください。
2. 格納条件を追加する場合，OTHERS を指定してください。

＜統合時＞

1. 表格納用 RD エリア名又は OTHERS は 1 個指定できます。
2. 格納条件は指定できません。
3. 格納条件を削除する場合，OTHERS を指定してください。
4. 統合を行った結果の表で格納条件がすべてなくなる指定はできません。

＜変更前 RD エリアリストに格納条件を指定している RD エリア名だけを指定した場合＞

1. 格納条件を指定しない RD エリア名を指定できません。

＜変更前 RD エリアリストに格納条件を指定していない RD エリア名を含んで指定した場合＞

1. 格納条件を指定していない RD エリアを削除する場合，OTHERS を指定してください。

＜変更前 RD エリアリストに OTHERS を含んで指定した場合＞

1. 格納条件を指定しない RD エリアを追加できます。

表格納用 RD エリア名

変更後にデータを格納する RD エリア名を指定します。ただし，一時表用 RD エリアは指定できません。

格納条件

分割時に変更後の格納条件を指定します。統合時は指定できません。

列名には分割キーに指定している列名を指定してください。

格納条件には定数を指定してください。格納条件に指定できる値については，「CREATE TABLE（表定義）」を参照してください。

＜変更前 RD エリア情報リストに格納条件を指定している表格納用 RD エリア名を指定した場合＞

1. 変更対象の表格納用 RD エリアに指定している格納条件はすべて指定してください。
2. 存在しない格納条件は指定できません。
3. 格納条件を重複して指定できません。

＜変更前 RD エリア情報リストに格納条件を指定していない表格納用 RD エリア名又は OTHERS を指定した場合＞

1. 表定義に存在しない格納条件を指定してください。
2. 表定義に存在する格納条件は指定できません。

OTHERS

表定義変更した結果，表に定義した格納条件を満たさないデータを格納する RD エリアが必要ない場合に指定します。

(p) インデクス格納用 RD エリア変更指定：：＝ FOR INDEX インデクス名 INTO 変更後インデクス格納用 RD エリア名リスト

分割格納条件を変更する表にインデクスを定義している場合に指定します。

インデクス格納用 RD エリア変更指定の規則を次に示します。

1. インデクス格納用 RD エリアだけの変更はできません。
2. インデクス格納用 RD エリアは横分割表変更指定又はマトリクス分割表変更指定で指定した境界値と同じ範囲又は格納条件の RD エリアを変更できます。
3. 表に定義したインデクスをすべて指定する必要があります。

インデクス名

分割格納条件を変更する表にインデクスを定義している場合，定義しているインデクスの識別子を指定します。

変更後インデクス格納用 RD エリア名リスト：：＝

{インデクス格納用 RD エリア名
| （インデクス格納用 RD エリア名）
| （（インデクス格納用 RD エリア名）
〔，（インデクス格納用 RD エリア名）〕…〔，OTHERS〕）
| ２次元格納用 RD エリア指定}
| OTHERS}

２次元格納用 RD エリア指定：：＝（マトリクス分割用 RD エリアリスト
〔，マトリクス分割用 RD エリアリスト〕…）

マトリクス分割用 RD エリアリスト：：＝

（インデクス格納用 RD エリア名
〔，インデクス格納用 RD エリア名〕…）

インデクス，主キー，又はクラスタキーを格納する RD エリア名を指定します。

変更後インデクス格納用 RD エリア名リストの規則を次に示します。

1. 変更後インデクス格納用 RD エリア名リストに指定する RD エリアの数は，次の箇所に指定した RD エリアの数と同じにする必要があります。
   - 変更後境界値分割指定（境界値指定の横分割表の場合）
   - 変更後格納条件分割指定（格納条件指定の横分割表の場合）
   - マトリクス分割表格納用 RD エリア変更指定（マトリクス分割表の場合）

2. 変更後境界値分割指定で重複した RD エリアを指定した場合，変更後インデクス格納用 RD エリア名リストに指定する RD エリアも，同じように重複して指定する必要があります。

3. 表，インデクス，主キー，及びクラスタキーは，表を基準に，分割数や RD エリアの重複の対応関係を維持する必要があります（例えば，表の変更前後で，システムが境界値を一つにまとめる場合，及び指定する RD エリアが変更対象の境界値以外のデータを格納している場合，変更前後で同じ RD エリアを使用し，同じ RD エリアを指定する位置などをインデクス，主キー，及びクラスタキーも同じように指定する必要があります)。

4. 変更後格納条件指定で OTHERS を指定した場合，変更後格納条件指定に対応する変更後インデクス格納用 RD エリア名リストに OTHERS を指定する必要があります。

2 次元格納用 RD エリア指定

マトリクス分割表の格納用 RD エリアを指定します。

マトリクス分割用 RD エリアリスト

マトリクス分割表の一つの次元用の RD エリアを指定します。

インデクス格納用 RD エリア名

変更後のデータを格納する RD エリアの名称を指定します。ただし，一時表用 RD エリアは指定できません。

OTHERS

変更後格納条件分割指定で OTHERS を指定した場合に指定します。

変更後格納条件分割指定で OTHERS を指定していない場合は OTHERS を指定できません。

(q) 主キー格納用 RD エリア変更指定：：＝ FOR PRIMARY KEY INTO 変更後インデクス格納用 RD エリア名リスト

分割格納条件を変更する表に主キーを定義している場合に指定します。

主キー格納用 RD エリア変更指定の規則を次に示します。

1. 主キー格納用 RD エリア変更指定は，1 個だけ指定できます。

2. 主キーを格納する RD エリアだけの変更はできません。

FOR PRIMARY KEY

表に主キーを定義している場合に指定します。

変更後インデクス格納用 RD エリア名リスト

変更後の主キーを格納する RD エリアを指定します。詳細は変更後インデクス格納用 RD エリア名リスト（(p)）を参照してください。

(r) クラスタキー格納用 RD エリア変更指定：：＝ FOR 〔PRIMARY〕 CLUSTER KEY INTO 変更後インデクス格納用 RD エリア名リスト

分割格納条件を変更する表にクラスタキーを定義している場合に指定します。

クラスタキー格納用 RD エリア変更指定の規則を次に示します。

1. クラスタキーを格納する RD エリアだけの変更はできません。
2. クラスタキー格納用 RD エリア変更指定は，1 個だけ指定できます。

FOR 〔PRIMARY〕 CLUSTER KEY

クラスタキーを格納するインデクス格納用 RD エリアを変更する場合に指定します。

PRIMARY

クラスタキーを主キーとして定義している場合に指定します。

変更後インデクス格納用 RD エリア名リスト

変更後のクラスタキーを格納する RD エリアの名称を指定します。詳細は変更後インデクス格納用 RD エリア名リスト（(n)）を参照してください。

(s) LOB 列格納用 RD エリア変更指定

```
: :＝FOR COLUMN  列名
  {LOB列格納用RDエリア変更リスト
    |INTO マトリクス分割LOB列格納用RDエリア変更指定}
  〔, 列名 {LOB列格納用RDエリア名指定
           |INTO マトリクス分割LOB列格納用RDエリア変更指定} 〕…
```

分割格納条件を変更する表に LOB 列を定義していた場合に指定します。

LOB 列格納用 RD エリア変更指定の規則を次に示します。

1. LOB 列格納用 RD エリアだけの変更はできません。
2. 列名には LOB 列を指定してください。
3. 表に定義している LOB 列はすべて指定する必要があります。

列名

表に定義している LOB 列の名称を指定します。

(t) LOB 列格納用 RD エリア変更リスト

```
: :＝INTO
  {LOB列格納用RDエリア名
    | (LOB列格納用RDエリア名)
    | ( (LOB列格納用RDエリア名)
        〔, (LOB列格納用RDエリア名) 〕…〔, OTHERS〕 )
    |OTHERS}
```

横分割表の LOB 列のデータを格納するユーザ LOB 用 RD エリアの名称を指定します。

LOB 列格納用 RD エリアについての規則を次に示します。

1. データ型に BLOB を指定した列は，LOB 列格納用 RD エリア名を必ず指定してください。BLOB 以外のデータ型を指定した列には指定できません。
2. 変更後境界値分割指定で重複した RD エリアを指定した場合，LOB 列格納用 RD エリア名も同じように重複した RD エリアを指定する必要があります。
3. 表，及び LOB 列は，表を基準に分割数や RD エリアの重複の対応関係を維持する必要があります（例えば，表の変更前後で，システムが境界値を一つにまとめる場合，及び指定する RD エリアが変更対象の境界値以外のデータを格納している場合，変更前後で同じ RD エリアを使用し，同じ RD エリアを指定する位置などを LOB 列も同じように指定する必要があります）。
4. 変更後格納条件指定で OTHERS を指定した場合，変更後格納条件指定に対応する LOB 列格納用 RD エリア変更リストに OTHERS を指定する必要があります。

LOB 列格納用 RD エリア名

LOB 列を格納する RD エリア名を指定します。

OTHERS

変更後格納条件分割指定で OTHERS を指定した場合に指定します。

変更後格納条件分割指定で OTHERS を指定していない場合は OTHERS を指定できません。

(u) マトリクス分割表格納用 RD エリア変更指定：：＝ 2 次元格納用 RD エリア指定

```
マトリクス分割LOB列格納用RDエリア変更指定：：＝2次元格納用RDエリア指定
```

マトリクス分割表格納用 RD エリア変更指定：：＝ 2 次元格納用 RD エリア指定

2 次元格納用 RD エリア指定：：＝（マトリクス分割用 RD エリアリスト〔，マトリクス分割用 RD エリアリスト〕…）

マトリクス分割用 RD エリアリスト：：＝（RD エリア名〔，RD エリア名〕…）

マトリクス分割された表の，変更後の RD エリアを指定します。

マトリクス分割表格納用 RD エリア変更指定の規則を次に示します。

＜分割時（第 1 次元分割列で分割する場合）＞

1. マトリクス分割用 RD エリアリストの数は，変更後境界値リストで指定した境界値の数と同じです。
2. 一つのマトリクス分割用 RD エリアリストに指定する RD エリアの数は，第 2 次元の分割数と同じです。
3. SQL 例については，「使用例」の 10.の＜例 1 ＞を参照してください。

＜分割時（第 2 次元分割列で分割する場合）＞

1. マトリクス分割用 RD エリアリストの数は，第 1 次元の分割数と同じです。
2. 一つのマトリクス分割用 RD エリアリストに指定する RD エリアの数は，変更後境界値リストで指定した境界値の数と同じです。
3. SQL 例については，「使用例」の 10.の＜例 3 ＞を参照してください。

＜統合時（第 1 次元分割列で統合する場合）＞

1. マトリクス分割用 RD エリアリストの数は，1 個です。
2. 一つのマトリクス分割用 RD エリアリストに指定する RD エリアの数は，第 2 次元の分割数と同じです。
3. SQL 例については，「使用例」の 10.の＜例 2 ＞を参照してください。

＜統合時（第 2 次元分割列で統合する場合）＞

1. マトリクス分割用 RD エリアリストの数は，第 1 次元の分割数と同じです。
2. 一つのマトリクス分割用 RD エリアリストに指定する RD エリアの数は，1 個です。
3. SQL 例については，「使用例」の 10.の＜例 4 ＞を参照してください。

＜注意事項＞

1. マトリクス分割表格納用 RD エリア変更指定を指定する場合の注意事項を次に示します。
2. 変更した結果の表の隣り合った境界値を格納する RD エリアが，同じ RD エリアとなった場合でも，システムは RD エリアを一つにまとめません（境界値指定の横分割表の変更ではシステムが RD エリアを一つにまとめます）。

2 次元格納用 RD エリア指定

マトリクス分割表の格納用 RD エリアを指定します。

マトリクス分割用 RD エリアリスト

マトリクス分割表の一つの次元用の RD エリアを指定します。

RD エリア名

変更後のデータを格納する RD エリアの名称を指定します。ただし，一時表用 RD エリアは指定できません。

マトリクス分割 LOB 列格納用 RD エリア変更指定：：＝ 2 次元格納用 RD エリア指定

2 次元格納用 RD エリア指定：：＝（マトリクス分割用 RD エリアリスト〔，マトリクス分割用 RD エリアリスト〕…）

マトリクス分割用 RD エリアリスト：：＝（RD エリア名〔，RD エリア名〕…）

マトリクス分割した表に LOB 列を定義している場合に，格納する RD エリア名を指定します。

マトリクス分割 LOB 列格納用 RD エリア変更指定についての規則を次に示します。

1. データ型に BLOB を指定した列は，LOB 列格納用 RD エリア名を必ず指定してください。BLOB 以外のデータ型を指定した列には指定できません。
2. マトリクス分割表変更指定で重複した RD エリアを指定した場合，LOB 列格納用 RD エリア名も，同じように重複して指定する必要があります。
3. 表，及び LOB 列は，表を基準に分割数や RD エリアの重複の対応関係を維持する必要があります（例えば，表の変更前後で，システムが境界値を一つにまとめる場合，及び指定する RD エリアが変更対象の境界値以外のデータを格納している場合，変更前後で同じ RD エリアを使用し，同じ RD エリアを指定する位置などを LOB 列も同じように指定する必要があります）。

2 次元格納用 RD エリア指定

マトリクス分割表の格納用 RD エリアを記述する場合に指定します。

マトリクス分割用 RD エリアリスト

マトリクス分割表の一つの次元用の RD エリアを指定します。

RD エリア名

変更後のデータを格納する RD エリアの名称を記述します。

WITHOUT PURGE

変更後の表のデータが，分割格納条件の境界値の範囲又は格納条件指定に一致しない場合，ALTER TABLE 実行時に HiRDB がそのデータを削除します。しかし，変更前後で同じ RD エリアを指定している場合に（変更対象の RD エリアを，変更後境界値分割指定，変更後格納条件分割指定，又はマトリクス分割表変更指定に指定している場合），同じ RD エリアの表のデータを ALTER TABLE 実行時に HiRDB に削除させないようにするときに指定します。この指定は，分割格納条件の変更前の表のデータを変更後も使用し，表のデータのアンロード，及びデータロードの運用を削減したい場合に有効です。

ただし，WITHOUT PURGE を指定しても，変更後に使用されない RD エリアの表のデータは，変更対象の表と関連を持たなくなり，整合性を保てなくなるため，システムが削除します。

WITHOUT PURGE を指定できるのは，次に示す条件を満たす場合（マトリクス分割表の場合は，RD エリアを分割・統合する各次元で次の条件を満たす場合）です。これ以外の場合に WITHOUT PURGE を指定すると，エラーとなります。詳細については，マニュアル「HiRDB システム運用ガイド」の「表の分割格納条件の変更」を参照してください。

《境界値指定の横分割表の場合》

変更対象の RD エリアを，変更後の RD エリアに含んでいること

《格納条件指定の横分割表の場合》

変更対象の RD エリアを，変更後の RD エリアに含んでいること

《マトリクス分割表の場合》

1. 変更対象の RD エリア群を，変更後の RD エリア群に含んでいること
2. 1.の RD エリア群の指定順序が同じ場合

《注意事項》

WITHOUT PURGE 指定時の注意事項を次に示します。

1. 分割格納条件の分割時は，変更対象の RD エリアのデータが，分割後に割り当てる境界値の範囲のデータ又は分割後に割り当てる格納条件の値とすべて一致していることを確認する必要があります。詳細については，マニュアル「HiRDB システム運用ガイド」の「表の分割格納条件の変更」を参照してください。
2. 分割格納条件の統合時は，削除する RD エリアのデータを登録し直す必要がある場合，分割格納条件を変更する前にアンロードし，分割格納条件を変更した結果の表にロードする必要があります。ただし，WITHOUT PURGE によって削除しないデータを，分割格納条件を変更した後にロードした場合，二重にデータを登録することになるため，アンロード，及びデータロードを行

う RD エリアに対する注意が必要です。詳細については，マニュアル「HiRDB システム運用ガイド」の「表の分割格納条件の変更」を参照してください。

3. 格納条件指定の横分割表の場合，分割格納条件の分割時に変更前 RD エリア情報リストで OTHERS を指定したとき，WITHOUT PURGE は指定できません。

WITH PROGRAM

変更対象の表を使用する手続き，及びトリガの有効な SQL オブジェクトを無効にする場合に指定します。

(5) 共通規則

1. CHANGE 句で実表の列を変更すると，ビュー表の列も変更されます。
2. 格納条件を指定した列に対して ALTER TABLE は指定できません。
3. WITH PROGRAM を省略した場合，定義変更する表，又はその表を参照して定義したビュー表を使用する手続き，及びトリガの有効な SQL オブジェクトがあると，表定義を変更できません。
4. 繰返し列でない列を繰返し列に変更できません。また，繰返し列を繰返し列でない列に変更できません。
5. 境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表に，BLOB 列又は BLOB 属性を含む抽象データ型の列を追加する場合，表定義で指定した表格納用 RD エリア名に対応するように，それぞれ LOB 列格納用 RD エリア名又は LOB 属性格納用 RD エリア名を指定しなければなりません。そのため，表定義で指定している表格納用 RD エリア名に重複した RD エリア名があるときは，LOB 列格納用 RD エリア名又は LOB 属性格納用 RD エリア名を重複して指定することになります。
6. ADD COLUMN，ADD RDAREA，又は CHANGE RDAREA で指定する表格納用 RD エリア，LOB 列格納用 RD エリア，LOB 属性格納用 RD エリア，及びインデクス格納用 RD エリアは，事前にデータベース初期設定ユティリティで作成しておくか，データベース構成変更ユティリティで追加しておく必要があります。
7. LOB 列格納用 RD エリア，LOB 属性格納用 RD エリア，及びユーザ LOB 用 RD エリアを指定したインデクス格納用 RD エリアは重複して指定できません。それぞれ，異なるユーザ LOB 用 RD エリアを指定してください。
8. ほかの BLOB 列，BLOB 属性，又はインデクスに割り当てられているユーザ LOB 用 RD エリアは指定できません。
9. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から ALTER TABLE は実行できません。
10. ADD COLUMN，ADD RDAREA，又は CHANGE RDAREA で指定する，表格納用 RD エリア，LOB 列格納用 RD エリア，LOB 属性格納用 RD エリア，及びインデクス格納用 RD エリアに，インナレプリカ機能を適用している RD エリアと，適用していない RD エリアとを混在して指定できません。インナレプリカ機能を適用している RD エリアを指定する場合，オリジナル RD エリアの名称を指定します。
11. インナレプリカ機能を使用している場合の ALTER TABLE の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
12. 監査証跡表の場合，ALTER TABLE は実行できません。
13. 被参照表，参照表に対して，DROP 句を用いた表定義変更はできません。

14. 被参照表，参照表に対して，RENAME 句を用いた表名変更はできません。
15. 被参照表の主キー構成列，外部キー構成列に対して定義変更を行う場合，次に示す制限があります。
    - CHANGE 句を用いたデータ型，データ長の変更はできません。
    - RENAME 句を用いた列名変更はできません。
16. 検査制約を定義した表に対して，DROP 句を用いた表定義変更はできません。
17. 検査制約を定義した表に対して，RENAME 句を用いた表名変更はできません。
18. 検査制約を定義した列に対して定義変更を行う場合，次に示す制限があります。
    - CHANGE 句を用いたデータ型，データ長の変更はできません。
    - CHANGE 句を用いた SPLIT の変更はできません。
    - CHANGE 句を用いた既定値の設定，解除はできません。
    - CHANGE 句を用いた WITH DEFAULT の設定はできません。
    - RENAME 句を用いた列名変更はできません。
19. 一時表の場合，ALTER TABLE は実行できません。

(6) 留意事項

1. 表定義時に FIX を指定した実表にデータが格納されている場合，ALTER TABLE では次に示す項目が指定できます。
    - 表格納用 RD エリアの追加
    - ハッシュ関数の変更
    - CHAR から MCHAR への列属性の変更
    - 表名，及び列名の変更
    - 最小排他資源単位の変更
    - 分割格納条件の変更
    - 更新可能列属性の付与（UPDATE 指定だけ）
    - 改竄防止表への変更
    - 予備列からの列の切り出し
2. WITH DEFAULT を指定した場合，データが格納されている表に列を追加できません。
3. 行単位インタフェースを使用して，日付データを CHAR（10）で受け渡しする場合，日付データ型を使用しないで，CHAR（10）で列を指定してください。
4. 行単位インタフェースを使用して，時刻データを CHAR（8）で受け渡しする場合，時刻データ型を使用しないで CHAR（8）で列を指定してください。

5. 行単位インタフェースを使用して，時刻印データを 19，22，24，又は 26 バイトの CHAR で受け渡しをする場合，時刻印データ型を使用しないで 19，22，24，又は 26 バイトの CHAR で列を指定してください。
6. ALTER TABLE は，OLTP 下の X/Open に従った UAP から指定できません。
7. 表又は列の名称を変更した場合，ユティリティや運用コマンドでは変更後の名称を指定してください。
8. 表又は列の名称を変更した場合，名称変更前に作成した次のファイルは使用できません。
   - データベース再編成ユティリティのアンロードデータファイル
   - データベース作成ユティリティ及びデータベース再編成ユティリティのインデクス情報ファイル
   - ディクショナリ搬出入ユティリティの搬出ファイル
9. ハッシュ関数を変更，又は表格納用 RD エリアを追加する場合，データを再ロードする必要はありません。ただし，追加された表格納用 RD エリアには，INSERT 文実行時に初めてデータが格納されるため，表格納用 RD エリア追加時にはデータが格納されません。
10. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった手続き，及びトリガの行は削除されます。
11. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して，手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
12. ALTER の ADD 指定で追加する列については，列データ抑制指定はできません。
13. WITHOUT ROLLBACK オプションを指定した表には，CHANGE LOCK PAGE を指定できません。
14. トリガを定義した表には，列を追加できます。トリガ契機列を省略したトリガの場合には，追加した列もトリガの実行対象になります。トリガ契機列を指定している場合には，トリガの実行対象にはなりません。なお，列を追加してもトリガの SQL オブジェクトは無効になりません。
15. トリガ契機列は，列定義変更，又は列削除ができます。ただし，トリガ契機列をすべて削除した場合は，そのトリガは削除されます。なお，一度削除した列と同名の列を追加しても，その列はトリガの実行対象にはなりません。また，トリガ契機列の列定義変更又は列削除をしても，トリガの SQL オブジェクトは無効になりません。
16. トリガ契機列の列削除をした後，ほかの操作でそのトリガの SQL オブジェクトを無効にした場合，無効にした SQL オブジェクトを実行するためには，次のどちらかの操作をする必要があります。
   - 列定義を元に戻し，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成します。
   - 無効にしたトリガを DROP TRIGGER で削除してから，列削除した列を使用しないように CREATE TRIGGER でトリガを再定義します。ただし，次のすべての条件を満たすトリガがある場合は，それらもすべて DROP TRIGGER で削除し，定義していた順に CREATE TRIGGER で再定義しないと，トリガ動作の実行順序が変わります。

     （条件）

     ・無効にしたトリガよりも定義したのが後である。

・無効にしたトリガと定義した表が同じである。

・無効にしたトリガとトリガ契機（INSERT，UPDATE，又は DELETE）が同じである（ただし，UPDATE の場合は，トリガ契機列の指定の有無，及び内容に関係なく同じとみなされます）。

・無効にしたトリガとトリガ動作時期（BEFORE 又は AFTER）が同じである。

・無効にしたトリガとトリガの動作する単位（行単位又は文単位）が同じである。

17. WITH PROGRAM を指定して，トリガ SQL 文中で参照している表に対して NOT NULL 列の追加をすると，そのトリガの SQL オブジェクトは無効になります。無効にしたトリガの SQL オブジェクトを実行するためには，ALTER TRIGGER 又は ALTER ROUTINE を実行して，トリガの SQL オブジェクトを再作成しておく必要があります。

18. WITH PROGRAM を指定して次の操作をすると，そのトリガの SQL オブジェクトは無効になります。

- トリガ SQL 文中で参照している表に対して，列定義変更，列削除，列名変更，又は表名変更をした場合
- トリガ SQL 文中で新旧値相関名を使用して参照している列を，列定義変更，又は列削除した場合

無効にしたトリガの SQL オブジェクトを実行するためには，次のどちらかの操作をする必要があります。

- 列定義，列名，又は表名を元に戻し，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成します。
- 無効にしたトリガを DROP TRIGGER で削除してから，列定義変更，若しくは列削除した列，又は名称変更前の列名，表名を使用しないように CREATE TRIGGER でトリガを再定義します。ただし，次のすべての条件を満たすトリガがある場合は，それらもすべて DROP TRIGGER で削除し，定義していた順に CREATE TRIGGER で再定義しないと，トリガ動作の実行順序が変わります。

  （条件）

  ・無効にしたトリガよりも定義したのが後である。

  ・無効にしたトリガと定義した表が同じである。

  ・無効にしたトリガとトリガ契機（INSERT，UPDATE，又は DELETE）が同じである（ただし，UPDATE の場合は，トリガ契機列の指定の有無，及び内容に関係なく同じとみなされます）。

  ・無効にしたトリガとトリガ動作時期（BEFORE 又は AFTER）が同じである。

  ・無効にしたトリガとトリガの動作する単位（行単位又は文単位）が同じである。

19. WITH PROGRAM を指定して，トリガ動作条件中で新旧値相関名を使用して参照している列の列定義変更，又は列削除をした場合，そのトリガの SQL オブジェクトは無効になります。また，トリガを引き起こす SQL を前処理するときにもエラーになります。無効にしたトリガの SQL オブジェクトを実行したり，トリガを引き起こす SQL を前処理したりするためには，次のどちらかの操作をする必要があります。

- 列定義を元に戻し，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成します。
- 無効にしたトリガを DROP TRIGGER で削除してから，列定義変更，又は列削除した列を使用しないように CREATE TRIGGER でトリガを再定義します。ただし，次のすべての条件を満たすトリガがある場合は，それらもすべて DROP TRIGGER で削除し，定義していた順に CREATE TRIGGER で再定義しないと，トリガ動作の実行順序が変わります。

（条件）

・無効にしたトリガよりも定義したのが後である。

・無効にしたトリガと定義した表が同じである。

・無効にしたトリガとトリガ契機（INSERT，UPDATE，又は DELETE）が同じである（ただし，UPDATE の場合は，トリガ契機列の指定の有無，及び内容に関係なく同じとみなされます）。

・無効にしたトリガとトリガ動作時期（BEFORE 又は AFTER）が同じである。

・無効にしたトリガとトリガの動作する単位（行単位又は文単位）が同じである。

20. トリガを定義した表の表名，及び次の列の列名は変更できません。

- トリガ契機列
- トリガ動作条件中で新旧値相関名を使用して参照している列
- トリガ SQL 文中で新旧値相関名を使用して参照している列

(7) 使用例

1. 在庫表（ZAIKO）に，倉庫住所（SADRS）列を追加します。

```
ALTER TABLE ZAIKO
    ADD SADRS VARCHAR(40)
```

2. 在庫表（ZAIKO）に，既定値のある非ナル値制約の列の倉庫住所（SADRS）を追加します。

```
ALTER TABLE ZAIKO
    ADD SADRS VARCHAR(40)
    NOT NULL WITH DEFAULT
```

3. 在庫表（ZAIKO）の，可変長データ型である倉庫住所（SADRS）列の最大長を 60 に変更します。

```
ALTER TABLE ZAIKO
    CHANGE SADRS VARCHAR(60)
```

4. 在庫表（ZAIKO）の商品コード（SCODE）列に設定されているクラスタキーの属性を，一意性制約のないものから一意性制約のあるものへ変更します。

```
ALTER TABLE ZAIKO
    CHANGE CLUSTER KEY UNIQUE
```

5. 在庫表（ZAIKO）の倉庫住所（SADRS）列を削除します。

```
ALTER TABLE ZAIKO
    DROP SADRS
```

6. 在庫表（ZAIKO）の倉庫住所（SADRS）の列に対する手続きの有効なオブジェクトを無効にして，倉庫住所（SADRS）の列を削除します。

```
ALTER TABLE ZAIKO
    DROP SADRS WITH PROGRAM
```

7. 在庫表（ZAIKO）に倉庫地図（SMAP）列を追加します。

```
ALTER TABLE ZAIKO
    ADD SMAP BLOB(1M) IN (RMAPLOB1)
```

8. ハッシュ分割の新在庫表（NZAIKO）を格納するRDエリアを追加します。このとき，インデクス（ISCODE）とBLOB型の倉庫地図（SMAP）列を格納するRDエリアも追加します。

```
ALTER TABLE NZAIKO
    ADD RDAREA RDA3
      FOR COLUMN SMAP IN (RMAPLOB3)
      FOR INDEX ISCODE IN (RDA4)
```

9. 境界値指定の横分割表の境界値の変更例を次に示します。

＜例1＞境界値の分割，統合

変更前

```
CREATE FIX TABLE "T1"("C1" INT,"C2" INT) PARTITIONED BY "C1"
  IN(("TA1")100,("TA2")200,("TA3")400,("TA4")500,("TA5")600,("TA6"))
CREATE INDEX "I1" ON "T1"("C1")
  IN(("IA1"),("IA2"),("IA3"),("IA4"),("IA5"),("IA6"))
```

<<状態1>>

境界値	100	200	400	500	600	
RDエリア	TA1	TA2	TA3	TA4	TA5	TA6
	IA1	IA2	IA3	IA4	IA5	IA6

1.統合《状態1→状態2》

```
ALTER TABLE "T1" CHANGE RDAREA
  ((100),(200)) INTO "TA11"
  FOR INDEX "I1" INTO "IA11"
```

<<状態2>>

境界値	200	400	500	600	
RDエリア	TA11	TA3	TA4	TA5	TA6
	IA11	IA3	IA4	IA5	IA6

2.分割《状態2→状態3》

```
ALTER TABLE "T1" CHANGE RDAREA
  ((400)) INTO (("TA12")300,("TA13"))
  FOR INDEX "I1" INTO (("IA12"),("IA13"))
```

<<状態3>>

境界値	200	300	400	500	600	
RDエリア	TA11	TA12	TA13	TA4	TA5	TA6
	IA11	IA12	IA13	IA4	IA5	IA6

3.統合《状態3→状態4》

```
ALTER TABLE ”T1” CHANGE RDAREA
  ((600),(MAX)) INTO ”TA11”
  FOR INDEX ”I1” INTO ”IA11”
```

<<状態4>>

境界値	200	300	400	500	
RDエリア	TA11	TA12	TA13	TA4	TA11
	IA11	IA12	IA13	IA4	IA11

A※

注※　Aの状態にするには，1.，2.，及び 3. が必要です。

＜例 2 ＞システムが隣り合った境界値を統合する場合(分割時の特殊例)

変更前

```
CREATE FIX TABLE ”T1”(”C1” INT,”C2” INT) PARTITIONED BY ”C1”
  IN((”TA1”)100,(”TA2”)200,(”TA3”)400,(”TA4”)500,(”TA5”)600,(”TA6”))
CREATE INDEX ”I1” ON ”T1”(”C1”)
  IN((”IA1”),(”IA2”),(”IA3”),(”IA4”),(”IA5”),(”IA6”))
```

境界値	100	200	400	500	600	
RDエリア	TA1	TA2	TA3	TA4	TA5	TA6
	IA1	IA2	IA3	IA4	IA5	IA6

境界値の変更

```
ALTER TABLE ”T1” CHANGE RDAREA
  ((400)) INTO ((”TA3”)300,(”TA4”))
FOR INDEX ”I1” INTO ((”IA3”),(”IA4”))
```

境界値	100	200	300	500	600	
RDエリア	TA1	TA2	TA3	TA4	TA5	TA6
	IA1	IA2	IA3	IA4	IA5	IA6

10. 格納条件指定の横分割表の RD エリアの変更例を次に示します。

変更前《状態 1》

```
CREATE FIX TABLE ”T1”(”C1” CHAR(3),”C2” INT)
  IN((”TA1”)”C1”=’001’,(”TA2”)”C1”=’002’,
  (”TA3”)”C1”=’003’,(”TA4”)”C1”=(’004’,’005’),(”TA5”))
CREATE INDEX ”I1” ON ”T1”(”C1”)
  IN((”IA1”),(”IA2”),(”IA3”),(”IA4”),(”IA5”))
```

<<状態1>>

格納条件	=’001’	=’002’	=’003’	=(’004’,’005’)	指定なし
RDエリア	TA1	TA2	TA3	TA4	TA5
	IA1	IA2	IA3	IA4	IA5

＜例 1 ＞格納条件の追加

1.分割《状態 1 →状態 2》

```
ALTER TABLE "T1" CHANGE RDAREA PARTITIONED CONDITION
  (("TA5")) INTO (("TA6")"C1"='006',("TA5"))
   FOR INDEX "I1" INTO (("IA6"),("IA5"))
```

<<状態2>>

格納条件	='001'	='002'	='003'	=('004','005')	='006'	指定なし
RDエリア	TA1	TA2	TA3	TA4	TA6	TA5
	IA1	IA2	IA3	IA4	IA6	IA5

＜例２＞格納条件の削除

2.統合《状態 1 →状態 3》

```
ALTER TABLE "T1" CHANGE RDAREA PARTITIONED CONDITION
  (("TA1"),("TA5")) INTO "TA5"
   FOR INDEX "I1" INTO "IA5"
```

<<状態3>>

格納条件	='002'	='003'	=('004','005')	指定なし
RDエリア	TA2	TA3	TA4	TA5
	IA2	IA3	IA4	IA5

＜例３＞ RD エリアの分割

3.分割《状態 1 →状態 4》

```
ALTER TABLE "T1" CHANGE RDAREA PARTITIONED CONDITION
  (("TA4")) INTO (("TA4")"C1"='004',("TA7")"C1"='005')
   FOR INDEX "I1" INTO (("IA4"),("IA7"))
```

<<状態4>>

格納条件	='001'	='002'	='003'	='004'	='005'	指定なし
RDエリア	TA1	TA2	TA3	TA4	TA7	TA5
	IA1	IA2	IA3	IA4	IA7	IA5

＜例４＞ RD エリアの統合

4.統合《状態 1 →状態 5》

```
ALTER TABLE "T1" CHANGE RDAREA PARTITIONED CONDITION
  (("TA1"),("TA2")) INTO "TA2"
   FOR INDEX "I1" INTO "IA2"
```

<<状態5>>

格納条件	=('001','002')	='003'	=('004','005')	指定なし
RDエリア	TA2	TA3	TA4	TA5
	IA2	IA3	IA4	IA5

11. マトリクス分割表の分割格納条件の分割，及び統合の例を次に示します。

前提

在庫表（ZAIKO）を定義します。このとき，商品番号（GNO）と入庫日（NYUUKOBI）のそれぞれに境界値を設定し，それぞれのデータが該当するRDエリアに格納されるようにします。この状態を《状態1》とします。

```
CREATE FIX TABLE "USERA"."ZAIKO"
("GNO" CHAR(5),"GNAME" CHAR(8),"KIKAKU" CHAR(3),"TANKA" INTEGER,"NYUUKOBI" CHAR(10))
  PARTITIONED BY MULTIDIM
    ("GNO"      (('20000')),
     "NYUUKOBI" (('2010-12-31'),('2011-12-31')))
    IN (("RDAT01","RDAT02","RDAT03"),
        ("RDAT01","RDAT04","RDAT05"))
  CLUSTER KEY ("GNO","NYUUKOBI")
    IN (("RDAI01","RDAI02","RDAI03"),
        ("RDAI01","RDAI04","RDAI05"))
```

<<状態1>> 初期状態

次元1＼次元2		NYUUKOBI ～2010	～2011	その他
GNO	20000以下	RDAT01 RDAI01	RDAT02 RDAI02	RDAT03 RDAI03
	その他	RDAT01 RDAI01	RDAT04 RDAI04	RDAT05 RDAI05

<<状態2>> 第1次元(GNO)の「その他」を2分割した状態

60000以下	RDAT01 RDAI01	RDAT04 RDAI04	RDAT05 RDAI05
その他	RDAT11 RDAI11	RDAT14 RDAI14	RDAT15 RDAI15

～2012	その他
RDAT03 RDAI03	RDAT13 RDAI13
RDAT05 RDAI05	RDAT15 RDAI15

<<状態3>>
第2次元(NYUUKOBI)の「その他」を2分割した状態

（凡例）
RDATnn:表用RDエリア
RDAImm:インデクス用RDエリア

＜例1＞第1次元列でのRDエリアの分割（《状態1》→《状態2》）

```
ALTER TABLE "USERA"."ZAIKO" CHANGE RDAREA
MULTIDIM ("GNO" ((MAX))) AT (('60000'),(MAX))
  INTO (("RDAT01","RDAT04","RDAT05"),
        ("RDAT11","RDAT14","RDAT15"))
FOR CLUSTER KEY
  INTO (("RDAI01","RDAI04","RDAI05"),
        ("RDAI11","RDAI14","RDAI15"))
```

注

RDAT01，RDAT04，RDAT05のデータ，及びRDAI01，RDAI04，RDAI05のキーは削除されます。

＜例2＞第1次元列でのRDエリアの統合（《状態2》→《状態1》）

```
ALTER TABLE "USERA"."ZAIKO" CHANGE RDAREA
MULTIDIM ("GNO" (('60000'),(MAX))) AT ((MAX))
  INTO (("RDAT01","RDAT04","RDAT05"))
```

```
FOR CLUSTER KEY
  INTO (("RDAI01","RDAI04","RDAI05"))
```

注

RDAT01，RDAT11，RDAT04，RDAT14，RDAT05，RDAT15 のデータ，及び RDAI01，RDAI11，RDAI04，RDAI14，RDAI05，RDAI15 のキーは削除されます。

＜例 3＞第 2 次元列での RD エリアの分割（《状態 1》→《状態 3》）

```
ALTER TABLE "USERA"."ZAIKO" CHANGE RDAREA
MULTIDIM ("NYUUKOBI" ((MAX))) AT (('2012-12-31'),(MAX))
  INTO (("RDAT03","RDAT13"),
        ("RDAT05","RDAT15"))
FOR CLUSTER KEY
  INTO (("RDAI03","RDAI13"),
        ("RDAI05","RDAI15"))
```

注

RDAT03，RDAT05 のデータ，及び RDAI03，RDAI05 のキーは削除されます。

＜例 4＞第 2 次元列での RD エリアの統合（《状態 3》→《状態 1》）

```
ALTER TABLE "USERA"."ZAIKO" CHANGE RDAREA
MULTIDIM ("NYUUKOBI" (('2012-12-31'),(MAX))) AT ((MAX))
  INTO (("RDAT03"),
        ("RDAT05"))
FOR CLUSTER KEY
  INTO (("RDAI03"),
        ("RDAI05"))
```

注

RDAT03，RDAT13，RDAT05，RDAT15 のデータ，及び RDAI03，RDAI13，RDAI05，RDAI15 のキーは削除されます。

12. 非改竄防止表の受注表（JUTYU）を，次の条件で改竄防止表に変更します。

《表定義時の条件》

- あらかじめ挿入履歴保持列として，JINSERTDATE 列を定義する。
- 改竄防止表に変更後も更新可能な列として BIKOU 列を定義する。

```
  CREATE TABLE JUTYU
      (DNO CHAR(6),TCODE CHAR(5),SCODE CHAR(4),
       BIKOU CHAR(60) UPDATE,
       JSURYO INTEGER,JDATE DATE,JTIME TIME,
       JINSERTDATE DATE NOT NULL WITH DEFAULT SYSTEM GENERATED)
```

《表定義変更時の条件》

- 行削除禁止期間を 10 年とする。

```
  ALTER TABLE JUTYU
      CHANGE INSERT ONLY WHILE 10 YEARS BY JINSERTDATE
```

3.6 ALTER TRIGGER（トリガの SQL オブジェクトの再作成）

3.6.1 ALTER TRIGGER の形式と規則

(1) 機能

トリガの SQL オブジェクトを再作成します。

(2) 使用権限

トリガの所有者

自分が所有するトリガの SQL オブジェクトを再作成できます。

DBA 権限を持つユーザ

自分が所有するトリガと，他ユーザが所有するトリガの SQL オブジェクトを再作成できます。

(3) 形式

```
  ALTER TRIGGER 〔認可識別子.〕トリガ識別子
       {CHANGE｜ALTER}  ROUTINE OBJECT
       〔SQLコンパイルオプション〔SQLコンパイルオプション〕…〕

  SQLコンパイルオプション：：＝ {ISOLATION データ保証レベル 〔FOR UPDATE  {EXCLUSIVE｜COMPAT
IBLE}〕
                                ｜OPTIMIZE LEVEL SQL最適化オプション
                                        〔, SQL最適化オプション〕…
                                ｜ADD OPTIMIZE LEVEL SQL拡張最適化オプション
                                        〔, SQL拡張最適化オプション〕…
                                ｜SUBSTR LENGTH 文字の最大長 }
```

(4) オペランド

(a) 〔認可識別子.〕トリガ識別子

認可識別子

SQL オブジェクトを再作成するトリガの所有者の認可識別子を指定します。省略した場合，ALTER TRIGGER を実行するユーザの認可識別子が仮定されます。

トリガ識別子

SQL オブジェクトを再作成するトリガの識別子を指定します。

(b) {CHANGE ｜ ALTER} ROUTINE OBJECT

トリガを再作成することを示します。CHANGE 及び ALTER は，どちらを指定しても意味は変わりません。

(c) SQL コンパイルオプション

```
::= {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE|COMPATIBLE}〕
      |OPTIMIZE LEVEL SQL最適化オプション〔, SQL最適化オプション〕…
      |ADD OPTIMIZE LEVEL SQL拡張最適化オプション〔, SQL拡張最適化オプション〕…
      |SUBSTR LENGTH 文字の最大長 }
```

SQL コンパイルオプションには ISOLATION，OPTIMIZE LEVEL，ADD OPTIMIZE LEVEL，SUBSTR LENGTH をそれぞれ 1 回しか指定できません。

〔ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE}〕〕

SQL のデータ保証レベルを指定します。

データ保証レベル

データ保証レベルとは，トランザクションのどの時点までデータの内容を保証するかのレベルです。次に示すデータ保証レベルを指定できます。

- 0

  データの内容を保証しない場合に指定します。

  0 レベルを指定すると，ほかのユーザが更新中のデータでも，更新完了を待たないで参照できます。ただし，参照する表が共用表の場合，ほかのユーザが排他モードで LOCK 文を実行しているときには，排他解除待ちとなります。

- 1

  検索処理の終了までデータの内容を保証したい場合に指定します。

  1 レベルを指定すると，検索処理が終了するまで（HiRDB がページ，又は行を見終わるまで）一度検索したデータをほかのユーザは更新できません。

- 2

  トランザクションの終了まで一度検索したデータの内容を保証したい場合に指定します。

  2 レベルを指定すると，トランザクションが終了するまで一度検索したデータをほかのユーザは更新できません。

〔FOR UPDATE {EXCLUSIVE | COMPATIBLE}〕

トリガ中で，FOR UPDATE 句を指定した（FOR UPDATE が仮定される場合を含む）カーソル又は問合せに対して，SQL コンパイルオプションで指定したデータ保証レベルに関係なく，常に WITH EXCLUSIVE LOCK を仮定する場合に指定します。

このオペランドを省略した場合，EXCLUSIVE を仮定します。ただし，0904 互換モードを適用している場合は COMPATIBLE を仮定します。

バージョン 09-50 より前の HiRDB で FOR UPDATE EXCLUSIVE を省略したトリガと同じ動作をさせたい場合は COMPATIBLE を指定します。

このオペランドと，ISOLATION データ保証レベルの関係から決まる FOR UPDATE の排他オプションを次に示します。

ISOLATION データ保証レベル	FOR UPDATE EXCLUSVIE	FOR UPDATE COMPATIBLE
0	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
1	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
2	WITH EXCLUSIVE LOCK	WITH EXCLUSIVE LOCK

《クライアント環境定義との関係》

ALTER TRIGGER に対して，PDISLLVL，PDFORUPDATEEXLOCK の指定は無効となります。

《SQL との関係》

トリガ中の SQL 文に排他オプションを指定している場合は，SQL コンパイルオプションで指定するデータ保証レベル，FOR UPDATE EXCLUSIVE，及び FOR UPDATE COMPATIBLE から仮定する排他オプションより SQL 文に指定した排他オプションが優先されます。

《システム定義との関係》

ALTER TRIGGER に対して，pd_isolation_level オペランドの指定は無効となります。

このオペランドを省略した場合，前回の SQL オブジェクト作成時（CREATE TRIGGER，ALTER TRIGGER，又は ALTER ROUTINE の実行時）に指定した値が仮定されます。

データ保証レベルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

〔OPTIMIZE LEVEL SQL 最適化オプション〔，SQL 最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDSQLOPTLVL」を参照してください。

SQL 最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法がありますが，通常時は識別子で指定する方法をお勧めします。

省略した場合，前回の SQL オブジェクト作成時（CREATE TRIGGER，ALTER TRIGGER，又は ALTER ROUTINE）に採用した値が仮定されます。

識別子で指定する場合：

```
OPTIMIZE LEVEL ”識別子”〔，”識別子”〕…
```

＜指定例＞

- ネストループジョイン優先とグループ分け高速化処理を適用する場合
  OPTIMIZE LEVEL "PRIOR_NEST_JOIN","RAPID_GROUPING"
- すべての最適化を適用しない場合
  OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。

3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。ただし，同時に"NONE"以外の識別子を指定すると，"NONE"は無効になります。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
OPTIMIZE LEVEL 符号なし整数〔，符号なし整数〕…
```

＜指定例＞

- 複数の SQL オブジェクト作成，AND の複数インデクス利用の抑止，及び複数インデクス利用の強制を適用する場合

  符号なし整数をコンマで区切って指定する場合：

  OPTIMIZE LEVEL 4,10,16

  符号なし整数の和を指定する場合：

  OPTIMIZE LEVEL 30
- 既に 14（4+10）を指定していて，新たに 16 を追加する場合

  OPTIMIZE LEVEL 14,16
- すべての最適化を適用しない場合

  OPTIMIZE LEVEL 0

＜規則＞

1. バージョン 06-00 より前の HiRDB から，バージョン 06-00 以降の HiRDB にバージョンアップする場合，バージョン 06-00 より前の合計値指定も有効となります。最適化オプションを変更する必要がない場合は，バージョン 06-00 以降の HiRDB にバージョンアップしたときにこのオペランドの指定値を変更する必要はありません。
2. 符号なし整数は一つ以上指定してください。
3. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
4. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。ただし，同時に 0 以外の識別子を指定すると，0 は無効になります。
5. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。
6. 複数の最適化方法を指定する場合，その符号なし整数の和を指定することもできます。ただし，同じ最適化方法の値は二つ以上足さないでください（足した結果が別の最適化方法とみなされることもあるため）。
7. 複数の最適化方法の値を足して指定する場合，どの最適方法を指定しているのか分かりにくくなるため，コンマで区切って指定する方法をお勧めします。また，既に複数の最適化方法の値を足して指定している場合で，新たに別の最適化方法が必要になったときは，追加する値をコンマで区切って後ろに指定できます。

《システム定義との関係》

1. ALTER TRIGGER に対して，システム定義の pd_optimize_level オペランドの指定は無効となります。
2. システム定義の pd_floatable_bes オペランド，又は pd_non_floatable_bes オペランドを指定している場合，「フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）」及び「フロータブルサーバ対象限定（データ取り出しバックエンドサーバ）」の指定は無効となります。
3. システム定義の pd_indexlock_mode オペランドに KEY を指定している場合（インデクスキー値排他の場合），「更新 SQL の作業表作成抑止」の指定は無効になります。

《クライアント環境定義との関係》

ALTER TRIGGER に対して，PDSQLOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔ADD OPTIMIZE LEVEL SQL 拡張最適化オプション〔，SQL 拡張最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 拡張最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDADDITIONALOPTLVL」を参照してください。

SQL 拡張最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法があります。

省略した場合，前回の SQL オブジェクト作成時（CREATE TRIGGER，ALTER TRIGGER，又は ALTER ROUTINE）に採用した値が仮定されます。

識別子で指定する場合：

```
ADD OPTIMIZE LEVEL ”識別子”〔,”識別子”〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL "COST_BASE_2","APPLY_HASH_JOIN"

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。
4. 識別子は大文字及び小文字で指定できます。

5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
ADD OPTIMIZE LEVEL 符号なし整数〔,符号なし整数〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL 1,2

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL 0

＜規則＞

1. 符号なし整数は一つ以上指定してください。
2. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。
4. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。

《システム定義との関係》

ALTER TRIGGER に対して，システム定義の pd_optimize_level オペランドの指定は無効となります。

《クライアント環境定義との関係》

ALTER TRIGGER に対して，PDADDITIONALOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔SUBSTR LENGTH 文字の最大長〕

1 文字を表現する最大バイト数を指定します。

文字の最大長に指定できる値は，3～6（pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8_ivs を指定した場合は 3～10）です。

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，又は utf-8_ivs を指定した場合にだけ有効となり，スカラ関数 SUBSTR の結果の長さに影響します。SUBSTR については，「SUBSTR」を参照してください。

＜規則＞

バージョン 08-00 より前の HiRDB からバージョン 08-00 以降の HiRDB にバージョンアップする場合，3 を仮定します。文字の最大長を変更する必要がない場合は，バージョン 08-00 以降の HiRDB にバージョンアップしたときにこのオペランドを指定する必要はありません。

《システム定義との関係》

ALTER TRIGGER に対して，システム定義の pd_substr_length オペランドの指定は無効となります。pd_substr_length オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

《クライアント環境定義との関係》

ALTER TRIGGER に対して，PDSUBSTRLEN の指定は無効となります。PDSUBSTRLEN については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

《pdntenv コマンド又は pdsetup コマンドで指定した文字コード種別との関係》

文字コード種別に utf-8，又は utf-8_ivs を指定した場合だけ有効となります。

そのほかの文字コード種別の場合は，構文チェックだけ行い，指定を無視します。

このオペランドを省略した場合，前回の SQL オブジェクト作成時（CREATE TRIGGER，ALTER TRIGGER，又は ALTER ROUTINE の実行時）に指定した値が仮定されます。

(5) 共通規則

1. SQL オブジェクトが無効なトリガに対して，ALTER TRIGGER を実行して正常に終了すると，そのトリガの SQL オブジェクトは有効になります。
2. SQL オブジェクトがインデクス無効状態になっているトリガに対して，ALTER TRIGGER を実行して正常に終了すると，トリガの SQL オブジェクトのインデクス無効状態は解除されます。SQL オブジェクトがインデクス無効状態のトリガは，実行時にエラーになります。
3. ALTER TRIGGER で SQL コンパイルオプションを指定する場合，再作成するトリガの元の CREATE TRIGGER に SQL コンパイルオプションを反映してできる SQL 文は，SQL 文の最大長を超えないようにしてください。
4. 次の条件では，Java 手続きから ALTER TRIGGER を実行できません。
   - 実行中の SQL オブジェクトが再作成になる場合

(6) 留意事項

1. ALTER TRIGGER は，OLTP 下の X/Open に従った UAP から指定できません。
2. ALTER TRIGGER 実行直後に GET DIAGNOSTICS 文を実行すると，ALTER TRIGGER の診断情報が取得できます。このとき，再作成が正常終了したトリガの SQL コードは 0 となります。
3. トリガ中のトリガ SQL 文のデータ保証レベル，SQL 最適化オプション，SQL 拡張最適化オプション，及び文字の最大長は，トリガの定義時，又は変更時の指定で決まり，トリガ動作実行時のシステム定義やクライアント環境定義の影響を受けません。

(7) 使用例

1. 無効になったトリガ（TRIG1）の SQL オブジェクトを再作成します。

```
ALTER TRIGGER TRIG1
    CHANGE ROUTINE OBJECT
```

3.7 COMMENT（注釈付加）

3.7.1 COMMENT の形式と規則

(1) 機能

表，及び列に注釈を付けます。既に注釈が付いている場合，その注釈を新しい注釈に変更します。

(2) 使用権限

表の所有者

自分が所有する表に注釈を付けることができます。

(3) 形式

```
COMMENT ON {TABLE 〔認可識別子.〕表識別子
               |COLUMN 〔認可識別子.〕表識別子.列名}
              IS '文字列'
```

(4) オペランド

(a) COMMENT ON

自分が所有する表，及び列に対して注釈を付けることができます。

(b) {TABLE〔認可識別子.〕 表識別子 | COLUMN〔認可識別子.〕 表識別子.列名}

表に注釈を付ける場合には TABLE を指定します。

列に注釈を付ける場合には COLUMN を指定します。

表識別子にパブリックビューを指定する場合は，認可識別子に PUBLIC を指定してください。

(c) '文字列'

注釈を文字列で指定します。指定できる文字列の長さは 0～255 バイトです。

なお，文字列として各国文字列定数を指定する場合に'N'は必要ありません。

また，次に示す文字列は指定できません。

- 16 進文字列定数

(5) 留意事項

1. 付けた注釈はデータディクショナリ表の SQL_TABLES 表，SQL_COLUMNS 表を検索すると参照できます。
2. 既に注釈がある場合，前の注釈を削除して新しい注釈に変更します。
3. データディクショナリ表に注釈を付けることはできません。
4. COMMENT は，OLTP 下の X/Open に従った UAP からは指定できません。

(6) 使用例

1. 在庫表（ZAIKO）に注釈を付けます。

```
COMMENT ON TABLE ZAIKO IS '1995年7月作成'
```

2. 在庫表（ZAIKO）の単価（TANKA）列に注釈を付けます。

```
COMMENT ON COLUMN ZAIKO.TANKA
    IS '1995.7 カイテイ'
```

3.8 CREATE AUDIT（監査対象イベントの定義）

3.8.1 CREATE AUDIT の形式と規則

(1) 機能

監査証跡として記録する監査対象イベント，及びその対象を定義します。

(2) 使用権限

監査権限を持つユーザ

監査権限を持つユーザが，CREATE AUDIT の各定義文を実行できます。

(3) 形式

```
CREATE AUDIT
       〔AUDITTYPE  {PRIVILEGE | EVENT | ANY}〕
      FOR  操作種別
       〔選択オプション〕
       〔WHENEVER  {SUCCESSFUL | UNSUCCESSFUL | ANY} 〕
```

≪各項目の詳細≫

```
操作種別：：＝ { ANY
              | SESSION   〔 {セッション種別 | ANY} 〕
              | PRIVILEGE   〔 {権限操作種別 | ANY} 〕
              | DEFINITION  〔 {オブジェクト定義イベント種別 | ANY} 〕
              | ACCESS      〔 {オブジェクト操作イベント種別 | ANY} 〕
              | UTILITY       〔 {ユティリティイベント種別 | ANY} 〕 }
選択オプション：：＝ {ON オブジェクト名 | BY AUTHORIZATION 認可識別子}
    オブジェクト名：：＝
               {FUNCTION  認可識別子.ルーチン識別子
             | INDEX  認可識別子.インデクス識別子
             | LIST  認可識別子.表識別子
             | PROCEDURE  認可識別子.ルーチン識別子
             | RDAREA  RDエリア名
             | SCHEMA  認可識別子
             | TABLE  〔認可識別子.〕表識別子
             | TRIGGER  認可識別子.トリガ識別子
             | TYPE  認可識別子.データ型識別子
             | VIEW  認可識別子.表識別子
             | SEQUENCE  認可識別子.順序数生成子識別子}
    セッション種別：：＝
          {CONNECT | DISCONNECT | AUTHORIZATION}
    権限操作種別：：＝
          {GRANT | REVOKE}
    オブジェクト定義イベント種別：：＝
          {CREATE | DROP | ALTER}
    オブジェクト操作イベント種別：：＝
```

```
        {SELECT | INSERT | UPDATE | DELETE | PURGE | ASSIGN | CALL | LOCK
          | NEXT VALUE}
    ユティリティイベント種別 : : =
        {PDLOAD | PDRORG | PDEXP | PDCONSTCK}
```

(4) オペランド

(a) 〔AUDITTYPE {PRIVILEGE | EVENT | ANY}〕

権限チェック時の監査証跡を取得するか，イベントの最終結果の監査証跡を取得するかを指定します。

PRIVILEGE

権限チェック時の監査証跡を取得します。

EVENT

イベントの最終結果の監査証跡を取得します。

ANY

上記のすべての種別についての監査証跡を取得します。

PRIVILEGE，EVENT，及び ANY は，個別で定義，及び削除をします。例えば，同じ監査イベントに対して，PRIVILEGE，EVENT，及び ANY をすべて定義した状態で，ANY だけを DROP AUDIT で削除しても，PRIVILEGE，及び EVENT の定義は残ります（監査対象のままとなります）。

(b) 操作種別

```
: : = {ANY
       | SESSION   〔 {セッション種別 | ANY} 〕
       | PRIVILEGE   〔 {権限操作種別 | ANY} 〕
       | DEFINITION  〔 {オブジェクト定義イベント種別 | ANY} 〕
       | ACCESS   〔 {オブジェクト操作イベント種別 | ANY} 〕
       | UTILITY  〔 {ユティリティイベント種別 | ANY} 〕 }
```

監査対象とする操作種別を指定します。個々の操作種別と ANY は，個別に定義，及び削除をします。例えば，SESSION，PRIVILEGE，及び ANY を定義した状態で，ANY だけを DROP AUDIT で削除しても，SESSION，及び PRIVILEGE の定義は残ります（監査対象のままとなります）。

ANY

すべての操作種別を監査対象とします。

SESSION 〔{セッション種別 | ANY}〕

セッションセキュリティイベントを監査対象にする場合に指定します。

ANY はすべてのセッションセキュリティイベントを監査対象とします。個々のセッション種別と ANY は，個別に定義，及び削除をします。例えば，CONNECT，AUTHORIZATION，及び ANY を定義した状態で，ANY だけを DROP AUDIT で削除しても，CONNECT，及び AUTHORIZATION の定義は残ります（監査対象のままとなります）。

PRIVILEGE 〔{権限操作種別｜ANY}〕

権限管理イベントを監査対象にする場合に指定します。ANY はすべての権限管理イベントを監査対象とします。個々の権限操作種別と ANY は，個別で定義，及び削除をします。例えば，GRANT，REVOKE，及び ANY を定義した状態で，ANY だけを DROP AUDIT で削除しても，GRANT，及び REVOKE の定義は残ります（監査対象のままとなります）。

DEFINITION 〔{オブジェクト定義イベント種別｜ANY}〕

定義系 SQL イベントを監査対象にする場合に指定します。ANY はすべての定義系 SQL イベントを監査対象とします。個々のオブジェクト定義イベント種別と ANY は，個別で定義，及び削除をします。例えば，CREATE，DROP，及び ANY を定義した状態で，ANY だけを DROP AUDIT で削除しても，CREATE，及び DROP の定義は残ります（監査対象のままとなります）。

ACCESS 〔{オブジェクト操作イベント種別｜ANY}〕

操作系 SQL イベントを監査対象にする場合に指定します。ANY はすべての操作系 SQL イベントを監査対象とします。個々のオブジェクト操作種別と ANY は，個別で定義，及び削除をします。例えば，SELECT，INSERT，及び ANY を定義した状態で，ANY だけを DROP AUDIT で削除しても，SELECT，及び INSERT の定義は残ります（監査対象のままとなります）。

UTILITY 〔{ユティリティイベント種別｜ANY}〕

ユティリティイベントを監査対象に定義する場合に指定します。ANY はすべてのユティリティイベントを監査対象とします。個々のユティリティイベント種別と ANY は，個別で定義，及び削除をします。例えば，PDLOAD，PDRORG，及び ANY を定義した状態で，ANY だけを DROP AUDIT で削除しても，PDLOAD，及び PDRORG の定義は残ります（監査対象のままとなります）。

(c) 〔WHENEVER {SUCCESSFUL | UNSUCCESSFUL | ANY}〕

監査イベントの最終結果の成否又は権限チェックの成否によって，その監査イベントを監査対象とするかどうかを指定します。WHENEVER の指定に対して取得する監査証跡を次の表に示します。

表 3-5 WHENEVER の指定に対して取得する監査証跡

WHENEVER の指定	AUDITTYPE に PRIVILEGE 又は ANY を指定した場合	AUDITTYPE に EVENT 又は ANY を指定した場合
SUCCESSFUL	権限チェックが成功した場合だけ，権限チェック時の監査証跡を取得します。	監査イベントが成功した場合だけ，監査イベントの最終結果の監査証跡を取得します。
UNSUCCESSFUL	権限チェックが失敗した場合だけ，権限チェック時の監査証跡を取得します。	監査イベントが失敗した場合だけ，監査イベントの最終結果の監査証跡を取得します。
ANY	権限チェックの成否に関係なく権限チェック時の監査証跡を取得します。	監査イベントの成否に関係なく監査イベントの最終結果の監査証跡を取得します。

イベントの最終結果によっては一部失敗という結果があります。一部失敗時の監査証跡は SUCCESSFUL，UNSUCCESSFUL，ANY のどれを指定しても出力します。

SUCCESSFUL，UNSUCCESSFUL，及び ANY は，個別で定義，及び削除をします。例えば，同じ監査イベントに対して，SUCCESSFUL，UNSUCCESSFUL，及び ANY をすべて定義した状態で，ANY だ

けを DROP AUDIT で削除しても，SUCCESSFUL，及び UNSUCCESSFUL の定義は残ります（監査対象のままとなります）。

(d) セッション種別：：＝ {CONNECT | DISCONNECT | AUTHORIZATION}

監査対象とする HiRDB に対する接続，接続中のユーザ変更，又は切断操作を指定します。セッション種別とその監査イベントが発生する操作を次の表に示します。

表 3-6 セッション種別とその監査イベントが発生する操作

セッション種別	権限チェックの監査イベントが発生する操作（AUDITTYPE に PRIVILEGE を指定した場合）	イベントの最終結果の監査証跡を取得する監査イベントが発生する操作（AUDITTYPE に EVENT を指定した場合）
CONNECT	HiRDB への接続	同左
DISCONNECT	なし	HiRDB からの切断
AUTHORIZATION	SET SESSION AUTHORIZATION 文の実行	同左

(e) 権限操作種別：：＝ {GRANT | REVOKE}

権限に関する操作を監査対象とする場合に指定します。権限操作種別とその監査イベントが発生する操作を次の表に示します。

表 3-7 権限操作種別とその監査イベントが発生する操作

権限操作種別	権限チェックの監査イベントが発生する操作（AUDITTYPE に PRIVILEGE を指定した場合）	イベントの最終結果の監査証跡を取得する監査イベントが発生する操作（AUDITTYPE に EVENT を指定した場合）
GRANT	GRANT の実行	同左
REVOKE	REVOKE の実行	同左

(f) オブジェクト定義イベント種別：：＝ {CREATE | DROP | ALTER}

監査対象とするオブジェクトの作成，削除，又は定義変更の操作を指定します。オブジェクト定義イベント種別とその監査イベントが発生する操作を次の表に示します。

表 3-8 オブジェクト定義イベント種別とその監査イベントが発生する操作

オブジェクト定義イベント種別	権限チェックの監査イベントが発生する操作（AUDITTYPE に PRIVILEGE を指定した場合）	イベントの最終結果の監査証跡を取得する監査イベントが発生する操作（AUDITTYPE に EVENT を指定した場合
CREATE	次の SQL の実行 • ALTER PROCEDURE※1 • ALTER ROUTINE※1 • ALTER TRIGGER※1	同左

オブジェクト定義イベント種別	権限チェックの監査イベントが発生する操作（AUDITTYPE に PRIVILEGE を指定した場合）	イベントの最終結果の監査証跡を取得する監査イベントが発生する操作（AUDITTYPE に EVENT を指定した場合
	• ASSIGN LIST 文 • CREATE CONNECTION SECURITY • CREATE FUNCTION • CREATE INDEX • CREATE PROCEDURE • CREATE PUBLIC FUNCTION • CREATE PUBLIC PROCEDURE • CREATE SCHEMA • CREATE SEQUENCE • CREATE TABLE • CREATE TRIGGER • CREATE TYPE • CREATE VIEW • CREATE PUBLIC VIEW • UAP からの CALL 文※2	
DROP	次の SQL の実行 • DROP CONNECTION SECURITY • DROP DATA TYPE • DROP FUNCTION • DROP INDEX • DROP LIST 文 • DROP PROCEDURE • DROP PUBLIC FUNCTION • DROP PUBLIC PROCEDURE • DROP SCHEMA • DROP SEQUENCE • DROP TABLE • DROP TRIGGER • DROP VIEW • DROP PUBLIC VIEW • REVOKE※3	同左
ALTER	次の SQL の実行 • ALTER INDEX • ALTER PROCEDURE • ALTER ROUTINE • ALTER TABLE • ALTER TRIGGER • COMMENT	同左

注※1

内部的に CREATE PROCEDURE が実行されます。

注※2

呼び出し対象となるプロシジャのインデクス情報が無効な場合，呼び出しごとに内部的に CREATE PROCEDURE が実行されます。このような場合，ALTER PROCEDURE 又は ALTER ROUTINE で呼び出し対象となるプロシジャの SQL オブジェクトを再作成することで，呼び出しごとに内部的に実行される CREATE PROCEDURE が実行されなくなります。

注※3

ビューの基表からの SELECT 権限が削除された場合，ビュー表を削除するために内部的に DROP VIEW が実行されます。

オブジェクト定義イベントで行われる権限チェックには，以下のようなものがあります。

- 監査イベントでのスキーマ定義権限チェック
- ストアドプロシジャ定義時，又はストアドプロシジャの SQL オブジェクト再作成時での，SQL 手続き文中の操作系 SQL 及び制御系 SQL のアクセス権限チェック
- メンバに手続きを含むユーザ定義型の定義時，又はユーザ定義型のメンバの SQL オブジェクト再作成時での，SQL 手続き文中の操作系 SQL 及び制御系 SQL のアクセス権限チェック
- トリガ定義時，又はトリガの SQL オブジェクト再作成時での，トリガ SQL 文中の操作系 SQL 及び制御系 SQL のアクセス権限チェック
- ビュー定義時の基表へのアクセス権限チェック

(g) オブジェクト操作イベント種別：：＝ {SELECT | INSERT | UPDATE | DELETE | PURGE | ASSIGN | CALL | LOCK | NEXT VALUE}

監査対象とするオブジェクトに対する操作を指定します。オブジェクト操作イベントの監査対象定義を指定した場合，手続き及びトリガ SQL 文中のオブジェクト操作も監査対象となります。オブジェクト操作イベント種別とその監査イベントが発生する操作を次の表に示します。

表 3-9　オブジェクト操作イベント種別とその監査イベントが発生する操作

オブジェクト操作イベント種別	権限チェックの監査イベントが発生する操作（AUDITTYPE に PRIVILEGE を指定した場合）	イベントの最終結果の監査証跡を取得する監査イベントが発生する操作（AUDITTYPE に EVENT を指定した場合）
SELECT	• 1 行 SELECT 文の実行※ • 問合せ指定を指定した INSERT 文の実行※ • 探索条件に副問合せを指定した UPDATE 文の実行※ • 探索条件に副問合せを指定した DELETE 文の実行※ • リストに対する問合せの実行※	• 同左
INSERT	• INSERT 文の実行※	• 同左

オブジェクト操作イベント種別	権限チェックの監査イベントが発生する操作（AUDITTYPE に PRIVILEGE を指定した場合）	イベントの最終結果の監査証跡を取得する監査イベントが発生する操作（AUDITTYPE に EVENT を指定した場合）
UPDATE	• UPDATE 文の実行※	• 同左
DELETE	• DELETE 文の実行※	• 同左
PURGE	• PURGE TABLE 文の実行※	• 同左
CALL	• 権限チェックイベントはありません	• CALL 文による手続の実行※
LOCK	• LOCK 文の実行※	• 同左
ASSIGN	• ASSIGN LIST 文形式 1 の実行	• ASSIGN LIST 文形式 1 の実行 • ASSIGN LIST 文形式 2 の実行
NEXT VALUE	• NEXT VALUE 式の実行	• 同左

注※

動的 SELECT 文を含みます。

次の SQL 文中での問合せは，オブジェクト操作イベント種別が SELECT で定義されている場合に，監査証跡を出力します。

- INSERT 文で指定した問合せ指定
- UPDATE 文及び DELETE 文の探索条件中に指定した副問合せ
- リストに対する問合せ

(h) ユティリティイベント種別：：＝ {PDLOAD | PDRORG | PDEXP | PDCONSTCK}

ユティリティイベントを監査対象として定義します。ユティリティイベント種別とその監査イベントが発生する操作を次の表に示します。

表 3-10 ユティリティイベント種別とその監査イベントが発生する操作

ユティリティイベント種別	権限チェックの監査イベントが発生する操作（AUDITTYPE に PRIVILEGE を指定した場合）	イベントの最終結果の監査証跡を取得する監査イベントが発生する操作（AUDITTYPE に EVENT を指定した場合）
PDLOAD	pdload の実行	同左
PDRORG	pdrorg の実行	同左
PDEXP	pdexp，pddefrev の実行	同左
PDCONSTCK	pdconstck の実行	同左

(i)　選択オプション：：＝ {ON オブジェクト名｜ BY AUTHORIZATION 認可識別子}

ON オブジェクト名

監査証跡を取得するオブジェクトを絞り込む場合に指定します。監査証跡の取得対象とするオブジェクトを指定します。各オブジェクト名の名称規則については，「名前の指定」を参照してください。

オブジェクトにパブリックビュー，パブリック手続き，又はパブリック関数を指定した場合は，認可識別子を PUBLIC と指定してください。

ディクショナリ表を指定する場合は，オブジェクト名に TABLE を指定し，認可識別子を省略して表識別子だけを指定してください。その場合，ディクショナリ表 SQL_AUDITS のオブジェクト所有者の列には’(Data dictionary)’を格納します。

BY AUTHORIZATION 認可識別子

監査対象とするイベント実行者の認可識別子を指定します。

(5)　規則

1. セキュリティ監査機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。
2. 実際に監査証跡を記録するには，システム定義の pd_audit オペランドの設定，又は pdaudbegin コマンドを実行する必要があります。
3. CREATE AUDIT，及び DROP AUDIT を実行した際の監査証跡は，セキュリティ監査機能が有効な場合は必ず記録します。

(6)　留意事項

1. CREATE AUDIT は，OLTP 下の X/Open に従った UAP からは指定できません。
2. 監査対象イベントの定義では，操作種別とそのほかのオペランドとの組み合わせによって，証跡が出力されない場合があります。そのような定義の指定をした場合は，KFPA19680-E メッセージが出力されます。次の表に指定可否の組み合わせの詳細を示します。

表 3-11　イベントタイプ，イベントサブタイプと AUDITTYPE の指定可否

イベントタイプ	イベント サブタイプ	AUDITTYPE 指定可否		
		PRIVILEGE	EVENT	ANY
ANY	－	△※1※2	○	△※1※2
SESSION	DISCONNECT	×	○	△※2
	ANY	△※2	○	△※2
	上記以外のすべて	○	○	○
PRIVILEGE	すべて	○	○	○
DEFINITION	すべて	○	○	○
ACCESS	CALL	×	○	△※1

イベントタイプ	イベント サブタイプ	AUDITTYPE 指定可否		
		PRIVILEGE	EVENT	ANY
	ANY	△※1	○	△※1
	上記以外のすべて	○	○	○
UTILITY	すべて	○	○	○

（凡例）

－：該当しません。

○：指定できます。

×：指定できません（KFPA19680-E メッセージが出力されます）。

△：指定できますが，証跡が出力されない場合があります。

注※1

イベント CALL の権限チェックの証跡は出力されません。

注※2

イベント DISCONNECT の権限チェックの証跡は出力されません。

表 3-12　イベントタイプ，イベントサブタイプとオブジェクト名の指定可否（1/2）

イベント タイプ	イベント サブタイプ	FUNCTION	INDEX	LIST	PROCEDURE
ANY	－	△	△	△	△
SESSION	すべて	×	×	×	×
PRIVILEGE	GRANT	×	×	×	×
	REVOKE	×	×	×	×
	ANY	×	×	×	×
DEFINITION	CREATE	○	○	×	○
	DROP	○	○	×	○
	ALTER	○	○	×	○
	ANY	○	○	×	○
ACCESS	SELECT	×	×	○	×
	INSERT	×	×	×	×
	UPDATE	×	×	×	×
	DELETE	×	×	×	×
	PURGE	×	×	×	×
	ASSIGN	×	×	○	×
	CALL	×	×	×	○
	LOCK	×	×	×	×

イベントタイプ	イベントサブタイプ	FUNCTION	INDEX	LIST	PROCEDURE
	NEXT VALUE	×	×	×	×
	ANY	×	×	△	△
UTILITY	PDLOAD	×	×	×	×
	PDRORG	×	×	×	×
	PDEXP	×	×	×	○
	PDCONSTCK	×	×	×	×
	ANY	×	×	×	△

（凡例）

－：該当しません。

○：指定できます。

×：指定できません（KFPA19680-E メッセージが出力されます）。

△：指定できますが，証跡が出力されない場合があります。

表 3-13　イベントタイプ，イベントサブタイプとオブジェクト名の指定可否（2/2）

イベントタイプ	イベントサブタイプ	RDAREA	SCHEMA	SERVER	TABLE	TRIGGER	TYPE	VIEW	SEQUENCE
ANY	－	△	△	△	△	△	△	△	△
SESSION	すべて	×	×	×	×	×	×	×	×
PRIVILEGE	GRANT	×	×	×	○	×	×	○	×
	REVOKE	×	×	×	○	×	×	○	×
	ANY	×	×	×	○	×	×	○	×
DEFINITION	CREATE	○	○	○	○	○	○	○	○
	DROP	×	○	○	○	○	○	○	○
	ALTER	○	×	×	○	○	×	○	×
	ANY	△	△	△	○	○	△	○	△
ACCESS	SELECT	×	×	×	○	×	×	○	×
	INSERT	×	×	×	○	×	×	○	×
	UPDATE	×	×	×	○	×	×	○	×
	DELETE	×	×	×	○	×	×	○	×
	PURGE	×	×	×	○	×	×	×	×
	ASSIGN	×	×	×	○	×	×	×	×
	CALL	×	×	×	×	×	×	×	×

イベントタイプ	イベントサブタイプ	RDAREA	SCHEMA	SERVER	TABLE	TRIGGER	TYPE	VIEW	SEQUENCE
	LOCK	×	×	×	○	×	×	○	×
	NEXT VALUE	×	×	×	×	×	×	×	○
	ANY	×	×	×	△	×	×	△	△
UTILITY	PDLOAD	×	×	×	○	×	×	×	○
	PDRORG	×	○	×	○	×	×	×	×
	PDEXP	×	×	×	○	○	×	○	×
	PDCONSTCK	×	×	×	○	×	×	×	×
	ANY	×	△	×	○	△	×	△	△

（凡例）

－：該当しません。

○：指定できます。

×：指定できません（KFPA19680-E メッセージが出力されます）。

△：指定できますが，証跡が出力されない場合があります。

3. AUDITTYPE 句，FOR ＜操作種別＞及び各種別，WHENEVER 句の中にある ANY 指定を定義している状態でバージョンアップする場合，個々の種別が増えたときはすべて監査対象に含まれます。

   例えば，CREATE AUDIT FOR ANY を定義している状態で，バージョンアップをして操作種別が新たに増えた場合，その操作種別も自動的に監査対象になります。

4. 既に定義されている監査対象イベントの定義を CREATE AUDIT 文で実行することはできません。KFPA11908-E メッセージが出力されます。

(7) 使用例

1. すべての監査イベントの権限チェック時を監査対象として定義します。

```
CREATE AUDIT FOR ANY WHENEVER ANY
```

2. HiRDB への接続の権限チェック時を監査対象として定義します。

```
CREATE AUDIT FOR SESSION CONNECT
```

3. GRANT 文実行の権限チェック時を監査対象として定義します。

```
CREATE AUDIT FOR PRIVILEGE GRANT
```

4. オブジェクト作成の権限チェック時を監査対象として定義します。

```
CREATE AUDIT FOR DEFINITION CREATE
```

5. INSERT の権限チェック時を監査対象として定義します。

```
CREATE AUDIT FOR ACCESS INSERT
```

6. すべての監査イベントを監査対象として定義します。

```
CREATE AUDIT AUDITTYPE ANY FOR ANY
```

7. すべての監査イベントのイベント終了時を監査対象として定義します。

```
CREATE AUDIT AUDITTYPE EVENT FOR ANY
```

8. 監査証跡の取得対象オブジェクトを表 USER1.T1 に絞り込みます。

```
CREATE AUDIT AUDITTYPE EVENT FOR ANY ON TABLE “USER1”.”T1”
```

9. 監査証跡の取得対象を USER1 に絞り込みます。

```
CREATE AUDIT AUDITTYPE ANY FOR ANY BY AUTHORIZATION ”USER1”
```

3.9 CREATE CONNECTION SECURITY（CONNECT 関連セキュリティ機能の定義）

3.9.1 CREATE CONNECTION SECURITY の形式と規則

(1) 機能

CONNECT 関連セキュリティ機能に関するセキュリティ項目を定義します。

(2) 使用権限

DBA 権限を持つユーザ

DBA 権限を持つユーザが，CREATE CONNECTION SECURITY の各定義文を実行できます。

(3) 形式

```
CREATE CONNECTION SECURITY FOR {セキュリティ対象〔, セキュリティ対象 〕
                               |パスワード有効期間
                               |接続制約}

セキュリティ対象：：＝ {CONNECT〔 PERMISSION COUNT 定数
                                 〔 LOCK { 定数 DAY〔S〕 |定数 HOUR〔S〕
                                         |定数 MINUTE〔S〕 |UNLIMITED } 〕〕
                     |PASSWORD〔 TEST 〕 パスワード文字制限定義
                                      〔 REUSE MAX 定数 〕}
パスワード有効期間：：＝INTERVAL PASSWORD TO 認可識別子 パスワード有効期間定義
接続制約：：＝CONSTRAINT 接続制約名 FOR 〔EXCEPT〕 'IPアドレス' 〔TO 認可識別子〕
パスワード文字制限定義：：＝〔 MIN LENGTH 定数 〕
                           〔 USER IDENTIFIER { RESTRICT|UNRESTRICT } 〕
                           〔 SIMILAR { RESTRICT|UNRESTRICT } 〕
                           〔 FORCE CHARACTER
                              ( { 文字種〔,文字種〕・・・|ALL } ) 〕
文字種：：＝UPPER|LOWER|NUMERIC
パスワード有効期間定義：：＝(パスワード有効期限間隔)
パスワード有効期限間隔：：＝定数 DAY〔S〕
```

(4) オペランド

(a) セキュリティ対象

```
：：＝ {CONNECT 〔 PERMISSION COUNT 定数
          〔 LOCK { 定数 DAY〔S〕 |定数 HOUR〔S〕
                  |定数 MINUTE〔S〕 |UNLIMITED } 〕〕
       |PASSWORD 〔TEST〕 パスワード文字制限定義
                      〔 REUSE MAX 定数 〕}
```

セキュリティ対象には CONNECT，PASSWORD をそれぞれ 1 回しか指定できません。

セキュリティ対象を省略した場合，省略したセキュリティ対象は定義されません。CONNECT，PASSWORD 両方のセキュリティ対象を省略することはできません。

セキュリティ対象に CONNECT，又は PASSWORD だけを指定し，CONNECT，又は PASSWORD 以降の各オペランドをすべて省略した場合，省略した各オペランドのデフォルト値が設定されます。

CONNECT 〔 PERMISSION COUNT 定数
〔 LOCK { 定数 DAY〔S〕 | 定数 HOUR〔S〕 | 定数 MINUTE〔S〕
| UNLIMITED }〕〕

連続認証失敗回数制限に関する設定値を指定します。

PERMISSION COUNT 定数

連続認証失敗アカウントロック状態に陥るまでの連続認証失敗回数の許容回数を指定します。連続認証失敗回数が指定値を超えると連続認証失敗アカウントロック状態になります。

PERMISSION COUNT 指定を省略した場合は 2 が設定されます。また，PERMISSION COUNT 指定を省略した場合は LOCK 指定はできません。

定数

連続認証失敗アカウントロック状態に陥るまでの連続認証失敗回数の許容回数を指定します。

最小値は 1（回），最大値は 10（回）です。

定数には符号なし整数を指定してください。

LOCK { 定数 DAY〔S〕 | 定数 HOUR〔S〕 | 定数 MINUTE〔S〕 | UNLIMITED }

連続認証失敗アカウントロック状態を継続する期間を指定します。

LOCK 指定を省略した場合，LOCK 1440 MINUTE（LOCK 1 DAY,LOCK 24 HOUR）が設定されます。

定数には符号なし整数を指定してください。

定数 DAY〔S〕

連続認証失敗アカウントロック状態を継続する期間を日単位で指定します。

最小値は 1（日），最大値は 31（日）です。

定数 HOUR〔S〕

連続認証失敗アカウントロック状態を継続する期間を時間単位で指定します。

最小値は 1（時間），最大値は 744（時間）です。

定数 MINUTE〔S〕

連続認証失敗アカウントロック状態を継続する期間を分単位で指定します。

最小値は 10（分），最大値は 44640（分）です。

UNLIMITED

連続認証失敗アカウントロック状態を継続する期間を無期限にする場合に指定します。

PASSWORD 〔TEST〕 パスワード文字制限定義〔 REUSE MAX 定数 〕

パスワード文字制限強化に関する設定値を指定します。

TEST

パスワード文字制限を定義する前に，あらかじめ変更する制限内容に不適合なパスワードを持つ認可識別子を確認する場合に指定します。

TEST を指定した場合，現状のパスワード文字列がパスワード文字制限定義で指定した文字制限に合っているか否かのチェックを行います。

TEST を指定した場合は，パスワード文字制限定義の設定は定義されません。

REUSE MAX 定数

過去に使用したパスワードを，新規パスワードとして再利用することを禁止する場合に，再利用禁止対象として記録するパスワード履歴の個数を指定します。

最小値は 1（件），最大値は 36（件）です。

定数には符号なし整数を指定してください。

パスワード履歴には，再利用禁止を設定した後に変更したパスワード以降の履歴を記録します。再利用禁止を設定した後にユーザを作成した場合は，ユーザ作成時のパスワード以降のパスワード履歴を記録します。

- パスワード有効期間：：= INTERVAL PASSWORD TO 認可識別子 パスワード有効期間定義
  指定した認可識別子に対して，パスワードの有効期間を設定します。
- パスワード有効期間定義：：=(パスワード有効期限間隔)
- パスワード有効期限間隔：：=定数 DAY〔S〕
  指定したユーザに対して，パスワードを使用できる期間（有効期限間隔）を日数で指定します。
  最小値は 1（日），最大値は 999（日）です。
  定数には符号なし整数を指定してください。

有効期限間隔を設定した日から有効期限間隔の指定日数後が，パスワードの有効期限となります。パスワードを更新すると，更新日から有効期限間隔の指定日数後が，パスワードの有効期限となります。

有効期限を過ぎると現在のパスワードは使用禁止となり，それ以降の接続はエラーになります。

OS 認証ユーザ，及びパスワードのないユーザには対して，パスワード有効期間を指定できません。指定した場合はエラーになります。

パスワード有効期限間隔を設定したユーザを，パスワードなしに変更することはできません。

(b) 接続制約

：：= CONSTRAINT 接続制約名 FOR 〔EXCEPT〕 ‘IP アドレス’ 〔TO 認可識別子〕

IP アドレスによる接続許可・拒否に関する制約を指定します。

接続拒否・許可に関係なく，接続制約を一つ以上定義した場合，許可の制約を定義していない IP アドレスからは接続できません。この機能についての詳細は，マニュアル「HiRDB システム運用ガイド」の「IP アドレスによる接続制限」を参照してください。

CONSTRAINT 接続制約名

指定する制約を特定するための接続制約名を指定します。

FOR 〔EXCEPT〕 ‘IP アドレス‘

接続を許可・拒否するクライアントの IP アドレスを文字列定数で指定します。

接続を拒否する場合は EXCEPT を指定してください。EXCEPT を省略すると，接続を許可する設定になります。

IP アドレス

IPv4 アドレスを指定します。以下の形式で指定できます。

表 3-14　IPv4 アドレス指定の形式

形式		説明
個別設定		1 オクテットごとにピリオドで区切られた 10 進数で指定します。
範囲指定	ネットワークアドレス／アドレスプリフィックス表記	ネットワークアドレスは 1 オクテットごとにピリオドで区切られた 10 進数で指定します。 アドレスプレフィックスは 0 以上 32 以下のネットワーク部のビット数を指定します。
	ワイルドカードを使用した形式	1 オクテットごとに 10 進数の代わりにアスタリスク（*）を指定できます。 アスタリスクを指定すると，0～255 の範囲を一括で示します。

〔TO 認可識別子〕

接続制限の対象とするユーザの認可識別子を指定します。

指定する認可識別子は，存在するユーザの認可識別子である必要があります。

省略した場合，すべてのユーザに適用される接続制限を定義します。

(c)　パスワード文字制限定義

```
: : = 〔 MIN LENGTH 定数 〕
      〔 USER IDENTIFIER { RESTRICT | UNRESTRICT } 〕
      〔 SIMILAR { RESTRICT | UNRESTRICT } 〕
      〔 FORCE CHARACTER ( { 文字種 〔, 文字種〕 ・・・ | ALL } ) 〕
```

MIN LENGTH 定数

パスワードの最小許容バイト数を指定します。

指定した定数未満のバイト数のパスワードを禁止します。

MIN LENGTH 指定を省略した場合，8 が設定されます。

最小値は 6，最大値は 15 です。

定数には符号なし整数を指定してください。

USER IDENTIFIER {RESTRICT | UNRESTRICT}

認可識別子を含むパスワードを禁止するかどうかを指定します。USER IDENTIFIER 指定を省略した場合，RESTRICT が設定されます。

RESTRICT

認可識別子を含むパスワードを禁止する場合に指定します。

大文字，小文字は区別しません。

（例）

認可識別子が USER1 の場合，次のような文字列がパスワードに含まれることを禁止します。

- USER1（英大文字だけ）
- user1（英小文字だけ）
- UseR1（英大文字と英小文字を含む）

UNRESTRICT

認可識別子を含むパスワードを禁止しない場合に指定します。

SIMILAR { RESTRICT | UNRESTRICT }

パスワードを構成する文字のすべてを単一文字種とすることを禁止するかどうかを指定します。

SIMILAR 指定を省略した場合，RESTRICT が設定されます。

以下の例のようなパスワードを禁止するかどうかを指定します。

例）

FDBGLAOT（英大文字だけ）

24681357（数字だけ）

RESTRICT

パスワードを構成する文字のすべてを単一文字種にすることを禁止する場合に指定します。

UNRESTRICT

パスワードを構成する文字のすべてを単一文字種にすることを禁止しない場合に指定します。ただし，FORCE CHARACTER を指定した場合は，FORCE CHARACTER の指定に従います。

FORCE CHARACTER ({文字種〔, 文字種〕・・・ | ALL }

パスワードを構成する文字に使用する文字種を指定します。パスワードは，各文字種をそれぞれ 1 文字以上含む構成にしてください。

ALL

すべての文字種を含むパスワードを使用します。

- 文字種：：＝ UPPER | LOWER | NUMERIC

UPPER

英大文字（A～Z，#，@，¥）を含むパスワードを使用します。

LOWER

英小文字（a～z）を含むパスワードを使用します。

NUMERIC

数字（0～9）を含むパスワードを使用します。

(5) 共通規則

1. 指定したセキュリティ対象が既に定義されている場合，同じセキュリティ対象を重複して定義できません。
2. CONNECT 関連セキュリティ機能に関する項目の定義内容を変更したい場合，一度 CONNECT 関連セキュリティ機能に関する項目の定義を削除してから，再度 CONNECT 関連セキュリティ機能に関する項目を定義してください。
3. DBA 権限保持者，及び監査人が一人でも指定したパスワード文字制限定義に違反している場合，CREATE CONNECTION SECURITY FOR PASSWORD はエラーとなります。ただし，TEST オプションを指定している場合はエラーにはなりません。

(6) 留意事項

1. 単一文字種の制限に関する指定を省略した場合は RESTRICT が設定され，制限が有効になります。制限をしない場合は必ず UNRESTRICT を指定してください。
2. パスワード文字制限が定義されている状態でも，TEST オペランド指定によるパスワードの事前チェックはできます。

(7) 使用例

1. 以下の設定内容で CONNECT 関連セキュリティ機能に関するセキュリティ項目を定義します。

   連続認証失敗回数制限に関する定義

   ・連続認証失敗許容回数 5 回

   ・ロック継続期間 7 日

   パスワード文字制限に関する定義

   ・パスワード最小許容バイト数 10 文字

   ・認可識別子を含むパスワードを禁止

   ・単一文字種を禁止

```
CREATE CONNECTION SECURITY FOR
  CONNECT PERMISSION COUNT 5
          LOCK 7 DAY,
  PASSWORD MIN LENGTH 10
           USER IDENTIFIER RESTRICT
           SIMILAR RESTRICT
```

2. 以下の設定内容で CONNECT 関連セキュリティ機能に関するセキュリティ項目を定義します。

   連続認証失敗回数制限に関する定義

   ・連続認証失敗許容回数 5 回

   ・ロック継続期間 15 時間

   パスワード文字制限に関する定義

   ・デフォルト値を設定

```
CREATE CONNECTION SECURITY FOR
  CONNECT PERMISSION COUNT 5
          LOCK 15 HOUR,
  PASSWORD
```

3. 以下の設定内容で CONNECT 関連セキュリティ機能に関するセキュリティ項目を定義します。

   連続認証失敗回数制限に関する定義

   ・定義しない

   パスワード文字制限に関する定義

   ・デフォルト値を設定

```
CREATE CONNECTION SECURITY FOR PASSWORD
```

4. 以下の設定内容で CONNECT 関連セキュリティ機能に関するパスワード有効期間を定義します。

   パスワード有効期間に関する定義

   ・設定する対象のユーザは USER1

   ・パスワード有効期限間隔 90 日

```
CREATE CONNECTION SECURITY FOR
  INTERVAL PASSWORD TO USER1 (90 DAY)
```

5. IP アドレスによる接続制限の CONNECT 関連セキュリティ機能に関するセキュリティ項目を定義する場合は，マニュアル「HiRDB システム運用ガイド」の「IP アドレスによる接続制限」を参照してください。

3.10 CREATE 〔PUBLIC〕FUNCTION（関数定義，パブリック関数定義）

3.10.1 CREATE FUNCTION（関数定義）

(1) 機能

関数を定義します。

(2) 使用権限

スキーマを所有するユーザ

自分が所有する関数を定義できます。

(3) 形式

```
CREATE 関数本体

関数本体：：＝FUNCTION 〔認可識別子.〕ルーチン識別子
                      （〔SQLパラメタ名 データ型
                            〔，SQLパラメタ名 データ型〕…〕）
                     RETURNS データ型
                      〔LANGUAGE {SQL｜JAVA｜C}〕
                      〔SQLコンパイルオプション〕
                      {SQL手続き文｜外部ルーチン指定}
SQLコンパイルオプション：：＝SUBSTR LENGTH 文字の最大長
外部ルーチン指定：：＝EXTERNAL NAME {外部Javaルーチン名｜外部Cストアドルーチン名}
                      PARAMETER STYLE パラメタスタイル
パラメタスタイル：：＝{JAVA｜RDSQL}
```

(4) オペランド

(a) 〔認可識別子.〕ルーチン識別子

認可識別子

定義する関数の所有者の認可識別子を指定します。

ルーチン識別子

定義する関数のルーチン識別子を指定します。ルーチン識別子は，所有者のルーチン中で同じ識別子を使用できます。

(b) （〔SQL パラメタ名 データ型〔，SQL パラメタ名 データ型〕…〕）

SQL パラメタ名

関数のパラメタの名称を指定します。一つの関数中で，SQL パラメタ名は重複して指定できません。

データ型

関数のパラメタのデータ型を指定します。

BOOLEAN は指定できません。

指定するデータ型が抽象データ型で，認可識別子を省略した場合，省略時に仮定される認可識別子に同一名称の抽象データ型がないときは，認可識別子'MASTER'に同一名称の抽象データ型があれば，その抽象データ型を指定したものとします。

次のデータ型は指定できません。

- LANGUAGE 句で JAVA 又は C を指定した場合，抽象データ型
- LANGUAGE 句で C を指定した場合，BINARY 型及び BLOB 型

指定できるデータ型については，「型マッピング」又は「SQL パラメタのデータ型と，C 関数に渡すパラメタのデータ型の対応関係」を参照してください。

(c) RETURNS データ型

データ型

関数の戻り値のデータ型を指定します。

次のデータ型は指定できません。

- LANGUAGE 句で C 以外を指定した場合，BOOLEAN
- LANGUAGE 句で JAVA 又は C を指定した場合の抽象データ型
- LANGUAGE 句で C を指定した場合，BINARY 型及び BLOB 型

指定できるデータ型については，「型マッピング」又は「SQL パラメタのデータ型と，C 関数に渡すパラメタのデータ型の対応関係」を参照してください。

指定するデータ型が抽象データ型で，認可識別子を省略した場合，省略時に仮定される認可識別子に同一名称の抽象データ型がないときは，認可識別子'MASTER'に同一名称の抽象データ型があれば，その抽象データ型を指定したものとします。

(d) LANGUAGE {SQL | JAVA | C}

定義する関数の記述言語を指定します。

外部ルーチン指定を指定する場合，記述言語は JAVA 又は C にしてください。

SQL

関数の処理部分を SQL 手続き文にする場合に指定します。

JAVA

関数の処理部分を外部ルーチン指定にする場合で，関数を Java クラスのメソッドで実装するときに指定します。

C

関数の処理部分を外部ルーチン指定にする場合で，関数を C 言語で実装するときに指定します。C を指定した場合，SQL パラメタの最大数が 128 に制限されます。

このオペランドの指定によって，他オペランドの指定要否が変わります。LANGUAGE 句指定による他オペランドの指定要否を次の表に示します。

表 3-15　CREATE FUNCTION の LANGUAGE 句指定による他オペランドの指定要否

他オペランド	LANGUAGE 句		
	SQL	JAVA	C
EXTERNAL NAME	×	○	○
PARAMETER STYLE	×	JAVA	RDSQL
SQL 手続き文	○	×	×

（凡例）

○：指定してください。

×：指定しないでください。

JAVA：JAVA を指定してください。

RDSQL：RDSQL を指定してください。

(e)　SQL コンパイルオプション：：＝ SUBSTR LENGTH 文字の最大長

〔SUBSTR LENGTH 文字の最大長〕

1 文字を表現する最大バイト数を指定します。

文字の最大長に指定できる値は，3～6（pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8_ivs を指定した場合は 3～10）です。

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，又は utf-8_ivs を指定した場合にだけ有効となり，スカラ関数 SUBSTR の結果の長さに影響します。SUBSTR については，「SUBSTR」を参照してください。

《システム定義との関係》

SUBSTR LENGTH を省略すると，システム定義の pd_substr_length オペランドの指定値が仮定されます。pd_substr_length オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

《クライアント環境定義との関係》

CREATE FUNCTION に対して，PDSUBSTRLEN の指定は無効となります。PDSUBSTRLEN については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

《pdntenv コマンド又は pdsetup コマンドで指定した文字コード種別との関係》

文字コード種別に utf-8，又は utf-8_ivs を指定した場合だけ有効となります。

そのほかの文字コード種別の場合は，構文チェックだけ行い，指定を無視します。

(f)　SQL 手続き文

SQL 関数で実行する SQL 手続き文を指定します。

SQL 手続き文については，「ルーチン制御 SQL」の「全般規定」を参照してください。

指定できるのは複合文だけです。また，SQL 関数の中で最後に実行する SQL 手続き文は，RETURN 文でなければなりません。

(g)　EXTERNAL NAME　{外部 Java ルーチン名｜外部 C ストアドルーチン名}

外部 Java ルーチン名

Java 言語で記述した Java メソッドを外部ルーチンとして指定します。外部 Java ルーチン名の指定方法については，「外部 Java ルーチン名」を参照してください。

外部 C ストアドルーチン名

C 言語で記述した C 関数を外部ルーチンとして指定します。外部 C ストアドルーチン名の指定方法については，「外部 C ストアドルーチン名」を参照してください。

LANGUAGE 句で C を指定する場合は，「SQL パラメタのデータ型と，C 関数に渡すパラメタのデータ型の対応関係」を参照してください。

(h)　PARAMETER STYLE パラメタスタイル

外部ルーチンを呼び出す場合，パラメタとして引き渡される内容を指定します。

パラメタスタイルが JAVA の場合

SQL データ型で定義している外部 Java 関数のパラメタは，SQL データ型に対応した Java データ型の Java メソッドのパラメタとして渡されます。

Java データ型で定義している Java メソッドの戻り値は，Java データ型に対応した SQL データ型の外部 Java 関数の戻り値として返されます。

パラメタスタイルが RDSQL の場合

SQL パラメタ数を n とすると，外部 C 関数の SQL パラメタは，次の表に示すように，C 関数のパラメタとして渡されます。

表 3-16　C 関数のパラメタとして渡される内容

パラメタ番号 (n は SQL パラメタ数)	内容	内容の詳細	データ型
1 番目～n 番目	SQL パラメタのデータ部	SQL パラメタのデータ部に対応する，入力パラメタです。	SQL パラメタのデータ型に対応するデータ型※1
n+1 番目	戻り値のデータ	外部 C ストアドルーチンを実装した C 関数が，戻り値を設定するための出力パラメタです。	RETURNS で指定した SQL データ型

パラメタ番号 (n は SQL パラメタ数)	内容	内容の詳細	データ型
			に対応するデータ型 ※1
n+2 番目～ 2n+1 番目	SQL パラメタの標識部	SQL パラメタの標識部に対応する，入力パラメタです。データがナル値の場合，負の値を設定して C 関数に渡します。	short*
2n+2 番目	戻り値の標識部	外部 C ストアドルーチンの戻り値の標識部を設定するための出力パラメタです。初期値として，0 が設定されています。 外部 C ストアドルーチンを実装した C 関数内では，次の説明に従って，標識部を設定してください。 • 出力値がナル値の場合 設定値：－1 • 出力値が非ナル値の場合 設定値：0	short*
2n+3 番目	SQLSTATE	外部 C ストアドルーチンを実装した C 関数が，SQLSTATE の値を設定するための出力パラメタです。領域長は 6 バイトです。領域の先頭から 5 バイト目までに SQLSTATE の値を設定してください。外部 C ストアドルーチンを実装した C 関数内では，次の説明に従って，SQLSTATE の値を設定してください。 • C 関数が正常終了したとき 設定値：‘00000’ SQL 実行結果：正常終了します。 • C 関数が異常終了したとき 設定値：形式’38XYY’の値 X，Y は，それぞれ次の範囲の値です。 X：’I’～’Z’ Y：’0’～’9’又は’A’～’Z’ (例)’38I01’,’38ZCD’ SQL 実行結果：SQL エラーになります。 • C 関数が任意の状態で終了したとき 設定値：‘00000’，形式’38XYY’の値以外 SQL 実行結果：SQL エラーになります。	char*
2n+4 番目	ルーチン名	ルーチン名を表す入力パラメタです。	struct{ short 変数名 1; char 変数名 2[30]; }*※2
2n+5 番目	特定名	関数を特定するための特定名を表す入力パラメタです。	struct{ short 変数名 1; char 変数名 2[30]; }*※2

パラメタ番号 (n は SQL パラメタ数)	内容	内容の詳細	データ型
2n+6 番目	メッセージテキスト	外部 C ストアドルーチンを実装した C 関数内でエラーが発生した場合，エラーが発生した詳細な理由を設定するための出力パラメタです。 外部 C ストアドルーチンの実行終了時，SQLSTATE にクラス 38 の値が設定されていると，メッセージテキストを埋め込んだエラーメッセージが出力されます。 なお，設定できるメッセージテキストの長さは，最大 80 バイトです。	struct{ short 変数名 1; char 変数名 2[80]; }*※2

注※1

外部 C 関数定義時に指定できる SQL パラメタのデータ型と，外部 C ストアドルーチンを実装した C 関数に渡すパラメタのデータ型の対応関係については，「SQL パラメタのデータ型と，C 関数に渡すパラメタのデータ型の対応関係」を参照してください。

注※2

変数名 1 に文字列長（バイト数），変数名 2 に内容を表す文字列を設定します。

表 3-17　SQL パラメタのデータ型と，C 関数に渡すパラメタのデータ型の対応関係

SQL パラメタの データ型	C 関数に渡すパラメタのデータ型	領域サイズ(byte)	備考
INT〔EGER〕	int*	4	−
SMALLINT	short*	2	−
〔LARGE〕 DEC〔IMAL〕 〔(p〔,s〕)〕, 〔LARGE〕 NUMERIC 〔(p〔,s〕)〕	char* ※2	↓p/2↓ + 1	1≦p≦38，0≦s≦p
FLOAT, DOUBLE PRECISION	double*	8	−
SMALLFLT, REAL	float*	4	−
CHAR〔ACTER〕〔(n)〕	char* ※1	n + 1	1≦n≦30,000
CHAR〔ACTER〕〔(n)〕 CHARACTER SET 〔MASTER.〕EBCDIK			
CHAR〔ACTER〕〔(n)〕 CHARACTER SET 〔MASTER.〕UTF16	char* ※1	n + 2	2≦n≦30,000
VARCHAR(n)	char* ※1	n + 1	1≦n≦32,000

SQL パラメタのデータ型	C 関数に渡すパラメタのデータ型	領域サイズ(byte)	備考
VARCHAR(n) CHARACTER SET〔MASTER.〕EBCDIK			
CHAR〔ACTER〕VARYING(n)	char* ※1	n + 1	1≦n≦32,000
CHAR〔ACTER〕VARYING(n) CHARACTER SET〔MASTER.〕EBCDIK			
VARCHAR(n) CHARACTER SET〔MASTER.〕UTF16	char* ※1	n + 2	2≦n≦32,000
CHAR〔ACTER〕VARYING(n) CHARACTER SET〔MASTER.〕UTF16	char* ※1	n + 2	2≦n≦32,000
NCHAR〔(n)〕, NATIONAL CHAR〔ACTER〕〔(n)〕	char* ※1	2n + 1	1≦n≦15,000
NVARCHAR(n), NATIONAL CHAR〔ACTER〕VARYING(n), NCHAR VARYING(n)	char* ※1	2n + 1	1≦n≦16,000
MCHAR〔(n)〕	char* ※1	n + 1	1≦n≦30,000
MVARCHAR(n)	char* ※1	n + 1	1≦n≦32,000
DATE	char*※4	4	−
TIME	char*※4	3	−
INTERVAL YEAR TO DAY	char* ※3	5	−
INTERVAL HOUR TO SECOND	char* ※3	4	−
TIMESTAMP〔(p)〕	char* ※4	n	領域サイズ n は，p の値に従って，次のように決定します。 • p=0 のとき，n=7 • p=2 のとき，n=8 • p=4 のとき，n=9 • p=6 のとき，n=10

SQL パラメタのデータ型	C 関数に渡すパラメタのデータ型	領域サイズ(byte)	備考
BOOLEAN	int* ※5	4	0:偽 1:真
標識変数	short*	2	－

（凡例）

－：備考はありません。

注

int 型が 4 バイトの 2 進形式データを表現する処理系以外では，C 関数に渡すパラメタのデータ型の int 型を，4 バイト 2 進形式データを表すデータ型に読み替えてください。

注※ 1

文字集合指定 UTF16 が指定されていない SQL のデータ型（CHAR(n)，VARCHAR(n)，NCHAR(n)，NVARCHAR(n)，MCHAR(n)，MVARCHAR(n)）と，C 言語のデータ型（char[n+1]，char[n+1]，char[2n+1]，char[2n+1]，char[n+1]，char[n+1]）との間での変換規則を示します。

SQL のデータ型から C 言語のデータ型への変換（C ストアドプロシジャの場合は，入力パラメタ及び入出力パラメタ，C ストアドファンクションの場合は，入力パラメタ）

- CHAR(n)から char[n+1]への変換
- VARCHAR(n)から char[n+1]への変換
- NCHAR(n)から char[2n+1]への変換
- NVARCHAR(n)から char[2n+1]への変換
- MCHAR(n)から char[n+1]への変換
- MVARCHAR(n)から char[n+1]への変換

文字列の終端にナル文字を付け加えます。

C 言語のデータ型から SQL のデータ型への変換（C ストアドプロシジャの場合は，出力パラメタ及び入出力パラメタ，C ストアドファンクションの場合は，戻り値）

- char[n+1]から CHAR(n)への変換
- char[n+1]から VARCHAR(n)への変換
- char[2n+1]から NCHAR(n)への変換
- char[2n+1]から NVARCHAR(n)への変換
- char[n+1]から MCHAR(n)への変換
- char[n+1]から MVARCHAR(n)への変換

C 言語の文字列から HiRDB が受け取る文字列の長さは，先頭からナル文字の一つ前までの長さとします。SQL のデータ型が CHAR(n)，NCHAR(n)又は MCHAR(n)であり，HiRDB が受け取った文字列の長さが SQL のデータ型の定義長に満たない場合は，文字列の終わりに空白を追加して定義長にします。なお，n+1 個の配列要素の中にナル文字がない場合，異常終了します。

文字集合指定 UTF16 が指定されている SQL のデータ型（CHAR(n)，VARCHAR(n)）と，C 言語のデータ型（char[n+2]，char[n+2]）との間での変換規則を示します。

SQL のデータ型から C 言語のデータ型への変換（C ストアドプロシジャの場合は，入力パラメタ及び入出力パラメタ，C ストアドファンクションの場合は，入力パラメタ）

- CHAR(n)から char[n+2]への変換
- VARCHAR(n)から char[n+2]への変換

文字列の終端に 2 バイトのナル文字を付け加えます。

C 言語のデータ型から SQL のデータ型への変換（C ストアドプロシジャの場合は，出力パラメタ及び入出力パラメタ，C ストアドファンクションの場合は，戻り値）

- char[n+2]から CHAR(n)への変換
- char[n+2]から VARCHAR(n)への変換

C 言語の文字列から HiRDB が受け取る文字列の長さは，先頭からナル文字の一つ前までの長さとします。SQL のデータ型が CHAR(n)であり，HiRDB が受け取った文字列の長さが SQL のデータ型の定義長に満たない場合は，文字列の終わりに空白を追加して定義長にします。なお，n+2 個の配列要素の中に 2 バイトのナル文字がない場合，異常終了します。

注※2

DECIMAL 型は，パック 10 進形式でデータを表現します。DECIMAL 型については，「データ型」を参照してください。

C 言語によって DECIMAL 型の値を設定する記述例を次に示します。

- 123.4567（奇数けた）の場合

  unsigned char ex1[4]={0x12,0x34,0x56,0x7c};
- -123.456（偶数けた）の場合

  unsigned char ex2[4]={0x01,0x23,0x45,0x6d};
- 0（奇数けた）の場合

  unsigned char ex3[1]={0x0c};

注※3

INTERVAL YEAR TO DAY 型，INTERVAL HOUR TO SECOND 型は，パック 10 進形式でデータを表現します。各データ型については，「データ型」を参照してください。

注※4

DATE 型，TIME 型，TIMESTAMP 型は，符号なしパック 10 進形式でデータを表現します。各データ型については，「データ型」を参照してください。

注※5

BOOLEAN 型は，C ストアドファンクションの戻り値としてだけ使用できます。

(5) 共通規則

1. SQL パラメタは，代入文の代入先には指定できません。
2. 関数のパラメタは，30,000 個（LANGUAGE 句が C の場合は 128 個）以下でなければなりません。ただし，LANGUAGE 句に SQL 以外を指定した場合，30,000 個以下でも，外部ルーチンの言語仕様の制限で実行時にエラーになるときがあります。
3. SQL パラメタは，入力用の SQL パラメタとなります。
4. SQL 手続き文中で指定する関数は，既に定義してある関数だけ指定できます。
5. 関数を定義すると，システムが各関数を一意に特定するための特定名を定義します。特定名の名称規則を次に示します。

   特定名：：＝F 関数名 オブジェクト ID

   - 先頭 1 バイト'F'固定
   - 2 バイト目から関数名（関数名が 19 バイトを超える場合は，19 文字まで使用します）
   - 関数名に続いてオブジェクト ID が 10 バイト（右詰めで先頭から 0 が埋め込まれた値となります）
6. システムが提供する関数と同じ関数は定義できません。次に示す条件 1 又は条件 2 のどちらかの条件をすべて満たす場合，その関数は定義できません。

   条件 1

   - 関数の SQL パラメタ数が二つ
   - 1 番目の SQL パラメタのデータ型が抽象データ型
   - 関数名と同じ名称の属性が，1 番目の SQL パラメタで指定した抽象データ型中の属性として定義されている
   - 2 番目の SQL パラメタのデータ型が，関数名と同じ名称の属性のデータ型と同じ

   条件 2

   - 関数の SQL パラメタ数が一つ
   - SQL パラメタのデータ型が抽象データ型
   - 関数名と同じ名称の属性が，SQL パラメタで指定した抽象データ型中の属性として定義されている
7. LANGUAGE 句を省略，又は LANGUAGE 句に SQL を指定した場合，外部ルーチン指定は指定できません。
8. LANGUAGE 句に SQL 以外を指定した場合，SQL 手続き文は指定できません。
9. 関数本体中に指定した関数呼出しの関数名が認可識別子で修飾されている場合，現在定義しようとしている関数と，認可識別子，ルーチン識別子，引数の数が一致する関数は指定できません。

   関数本体中に指定した関数呼出しの関数名が認可識別子で修飾されていない場合，現在定義しようとしている関数と，ルーチン識別子，引数の数が一致する関数は指定できません。
10. 次の条件をすべて満たす関数が定義されている場合，CREATE FUNCTION は実行できません。
    - 認可識別子が同じ

- ルーチン識別子（関数名）が同じ
- パラメタの数が同じ
- パラメタのデータ型が同じ※

注※

固定長文字データ型又は可変長文字データ型の文字集合指定が異なる場合は，異なるデータ型として扱います。

11. ALTER ROUTINE で SQL コンパイルオプションを指定する場合，再作成する関数の元の CREATE FUNCTION に SQL コンパイルオプションを反映してできる SQL 文は，SQL 文の最大長を超えないようにしてください。
12. 実行中の SQL オブジェクトが無効になる場合，外部 Java 手続き中から CREATE FUNCTION は実行できません。

(6) 留意事項

1. CREATE FUNCTION は，OLTP 下の X/Open に従った UAP から指定できません。
2. SQL パラメタは，ナル値を持てます。
3. SQL 手続き文中で複数の SQL 文を実行するためには，複合文などのルーチン制御 SQL を使用します。
4. SQL 関数を定義すると，それを実行するためのアクセス手順を記述した SQL オブジェクトが作成されます。
5. 関数定義に伴う SQL オブジェクトの無効化について示します。
   - 定義する関数と，所有者，ルーチン識別子，及び SQL パラメタ数が同じ関数が既にある場合，既にある関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクトがあれば，その SQL オブジェクトは無効となります。
   - 定義する関数と，ルーチン識別子及び SQL パラメタ数が同じで，かつ認可識別子が'MASTER'の関数がある場合，認可識別子'MASTER'の関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクトのうち，定義する関数の認可識別子が所有する SQL オブジェクトは無効となります。また，有効な SQL オブジェクトがパブリック関数及びパブリック手続きの場合，定義する関数の認可識別子と，パブリック関数及びパブリック手続きを定義した認可識別子が同じならば，そのパブリック関数及びパブリック手続きの SQL オブジェクトは無効となります。
   - 定義する関数と，ルーチン識別子及び SQL パラメタ数が同じパブリック関数がある場合，そのパブリック関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクトのうち，定義する関数の認可識別子が所有する SQL オブジェクトは無効となります。また，有効な SQL オブジェクトがパブリック関数及びパブリック手続きの場合，定義する関数の認可識別子と，パブリック関数及びパブリック手続きを定義した認可識別子とが同じならば，そのパブリック関数及びパブリック手続きの SQL オブジェクトは無効となります。

ストアドファンクションを定義，又は削除するときの注意事項については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

6. 5 に示した無効になる関数のうち，次のどちらかの条件を満たす関数をビュー定義で使用している場合，関数定義がエラーになります。
   - 引数のデータ型に抽象データ型を使用している
   - 戻り値のデータ型に抽象データ型を使用している
7. 関数，手続き，及びトリガの有効な SQL オブジェクトが無効になった場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった関数，手続き，及びトリガの行は削除されます。
8. 無効となった関数，手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して，関数，手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
9. 関数定義に伴って無効となった関数を使用したビュー表がある場合，そのビュー表を操作するには ALTER ROUTINE を実行して関数の SQL オブジェクトを再作成しておく必要があります。
10. ルーチン内でのルーチンの呼び出しを多数又は無限に繰り返す場合，OS のスタックがオーバフローすることがあります。
11. SQL コンパイルオプションの SUBSTR LENGTH は，関数の定義時，又は変更時の指定で決まり，関数呼出し時のシステム定義やクライアント環境定義の影響を受けません。

3.10.2 CREATE PUBLIC FUNCTION（パブリック関数定義）

(1) 機能

すべてのユーザから，認可識別子でルーチン識別子を修飾しなくても使用できる関数（パブリック関数）を定義します。

(2) 使用権限

スキーマを所有するユーザ

自分が所有するパブリック関数を定義できます。

(3) 形式

```
CREATE PUBLIC 関数本体
```

(4) オペランド

(a) PUBLIC

関数をパブリック関数として定義する場合に指定します。

パブリック関数にすると，同じ内容の関数をユーザごとに定義しなくても，一つの関数をルーチン識別子だけを指定して複数のユーザで使用できます。

(b)　関数本体

「〔認可識別子.〕ルーチン識別子」以外の説明については，「CREATE FUNCTION（関数定義）」と同じです。

〔認可識別子.〕ルーチン識別子

認可識別子

パブリック関数のため，認可識別子は指定できません。

ルーチン識別子

定義するパブリック関数のルーチン識別子を指定します。ルーチン識別子は，パブリック関数中で同じルーチン識別子を使用できます。

(5)　共通規則

1. 関数本体の「〔認可識別子.〕ルーチン識別子」の認可識別子は指定できません。
2. 関数本体中に指定した関数呼出しの関数名が PUBLIC で修飾されている場合，現在定義しようとしているパブリック関数と，ルーチン識別子，引数の数が一致するパブリック関数は指定できません。

   関数本体中に指定した関数呼出しの関数名が認可識別子で修飾されていない場合，現在定義しようとしているパブリック関数と，ルーチン識別子，引数の数が一致する関数は指定できません。
3. 次の条件をすべて満たすパブリック関数が定義されている場合，CREATE PUBLIC FUNCTION は実行できません。
   - ルーチン識別子（関数名）が同じ
   - パラメタの数が同じ
   - パラメタのデータ型が同じ※

   注※

   固定長文字データ型又は可変長文字データ型の文字集合指定が異なる場合は，異なるデータ型として扱います。
4. ALTER ROUTINE で SQL コンパイルオプションを指定する場合，再作成する関数の元の CREATE PUBLIC FUNCTION に SQL コンパイルオプションを反映してできる SQL 文は，SQL 文の最大長を超えないようにしてください。
5. 実行中の SQL オブジェクトが無効になる場合，外部 Java 手続き中から CREATE PUBLIC FUNCTION は実行できません。
6. その他の規則は，「CREATE FUNCTION（関数定義）」と同じです。

(6) 留意事項

1. CREATE PUBLIC FUNCTION は，OLTP 下の X/Open に従った UAP から指定できません。
2. パブリック関数定義に伴う SQL オブジェクトの無効化について，次に示します。
   - 定義するパブリック関数と，ルーチン識別子及び SQL パラメタ数が同じパブリック関数が既にある場合，既にあるパブリック関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクトがあれば，その SQL オブジェクトは無効となります。
   - 定義するパブリック関数と，ルーチン識別子及び SQL パラメタ数が同じで，かつ認可識別子が'MASTER'の関数がある場合，認可識別子'MASTER'の関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクトがあれば，その SQL オブジェクトは無効となります。

   ストアドファンクションを定義，又は削除するときの注意事項については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。
3. 2 に示した無効になる関数のうち，次のどちらかの条件を満たす関数をビュー定義で使用している場合，パブリック関数定義がエラーになります。
   - 引数のデータ型に抽象データ型を使用している
   - 戻り値のデータ型に抽象データ型を使用している
4. 無効となった関数を使用したビュー表がある場合，そのビュー表を操作するには ALTER ROUTINE を実行して関数の SQL オブジェクトを再作成しておく必要があります。
5. ディクショナリ表の所有者を格納する列（SQL_ROUTINES 表の ROUTINE_SCHEMA 列など）には PUBLIC が設定されます。また，パブリック関数を定義した認可識別子は SQL_ROUTINES 表の ROUTINE_CREATOR 列に格納されます。
6. パブリック関数の削除は，DROP PUBLIC FUNCTION で行います。
7. その他の留意事項は，「CREATE FUNCTION（関数定義）」と同じです。

3.11　CREATE INDEX　形式 1（インデクス定義）

3.11.1　CREATE INDEX 形式 1 の形式と規則

(1)　機能

実表の一つ，又は複数の列にインデクスを定義します。

(2)　使用権限

表の所有者

公用 RD エリアに自分が所有する表のインデクスを定義できます。

私用 RD エリアの利用権限を持つ表の所有者

利用権限を持つ私用 RD エリアに自分が所有する表のインデクスを定義できます。

(3)　形式 1 ＜インデクスの定義＞

```
CREATE 〔UNIQUE〕 INDEX 〔認可識別子.〕インデクス識別子
      ON 〔認可識別子.〕表識別子（列名〔 {ASC｜DESC} 〕
                                〔, 列名〔 {ASC｜DESC} 〕〕…）
       〔IN {RDエリア名
            |（RDエリア名）
            |（（RDエリア名）〔,（RDエリア名）〕…）
            |マトリクス分割インデクス格納用RDエリア指定} 〕
       〔インデクスオプション〕…

マトリクス分割インデクス格納用RDエリア指定：：＝2次元格納用RDエリア指定
2次元格納用RDエリア指定：：＝
            （マトリクス分割用RDエリアリスト〔, マトリクス分割用RDエリアリスト〕…）
マトリクス分割用RDエリアリスト：：＝（RDエリア名〔, RDエリア名〕…）
インデクスオプション：：＝ {PCTFREE=未使用領域の比率
                          |UNBALANCED SPLIT
                          |EMPTY
                          |除外値指定}
除外値指定：：＝EXCEPT VALUES （NULL〔, NULL〕…）
```

(4)　オペランド

(a)　〔UNIQUE〕

インデクスを定義するキー値（インデクスとして定義する一つ，又は複数の列の全体の値）が，すべての行で異なっている場合に指定します。

UNIQUE を指定して，インデクス作成，又は更新時に重複キー値を検出した場合，HiRDB がエラーを返します。ただし，ナル値を含むキー値の場合，重複した値があっても重複キーにはなりません。

UNIQUE を指定する場合は次の点に注意してください。

1. 表を横分割する場合

表を横分割する場合の UNIQUE 指定可否を次の表に示します。

表 3-18　表を横分割する場合の UNIQUE 指定可否

表の分割方法※1		UNIQUE を指定しようとするインデクスの構成列※2		インデクスの分割方法※4	UNIQUE 指定可否
サーバ内の横分割	キーレンジ分割（マトリクス分割以外），及び FIX ハッシュ分割	分割キーインデクス		表の分割数と一致	○
				表の分割数と不一致	－
		非分割キーインデクス	分割キーを含む（順不同）	表の分割数と一致	○
				表の分割数と不一致（横分割しない）	○
			分割キーを含まない（順不同）	表の分割数と一致	×
				表の分割数と不一致（横分割しない）	○※3
	フレキシブルハッシュ分割	該当しない		該当しない	×
	マトリクス分割	分割キーインデクス		表の分割数と一致	○
				表の分割数と不一致	－
		非分割キーインデクス	分割キーをすべて含む場合（順不同）	表の分割数と一致	○
				表の分割数と不一致	－
			分割キーをすべて含まない場合（順不同）	表の分割数と一致	×
				表の分割数と不一致（横分割しない）	－
サーバ間の横分割（サーバ内分割なし）	キーレンジ分割（マトリクス分割以外），及び FIX ハッシュ分割	分割キーインデクス		表の分割数と一致	○
				表の分割数と不一致	－
		非分割キーインデクス	分割キーをすべて含む場合（順不同）	表の分割数と一致	○
				表の分割数と不一致	－
			分割キーをすべて含まない場合（順不同）	表の分割数と一致	×
				表の分割数と不一致	－
	フレキシブルハッシュ分割	該当しない		該当しない	×
	マトリクス分割	分割キーインデクス		表の分割数と一致	○
				表の分割数と不一致	－

表の分割方法※1		UNIQUE を指定しようとするインデクスの構成列※2		インデクスの分割方法※4	UNIQUE 指定可否
		非分割キーインデクス	分割キーをすべて含む場合（順不同）	表の分割数と一致	○
				表の分割数と不一致	−
			分割キーをすべて含まない場合（順不同）	表の分割数と一致	×
				表の分割数と不一致	−
サーバ間の横分割（サーバ内分割あり）	キーレンジ分割（マトリクス分割以外），及び FIX ハッシュ分割	分割キーインデクス		表の分割数と一致	○
				表の分割数と不一致	−
		非分割キーインデクス	分割キーをすべて含む場合（順不同）	表の分割数と一致（サーバ内でも分割）	○
				表の分割数と不一致（サーバ内は非分割）	○
			分割キーをすべて含まない場合（順不同）	表の分割数と一致（サーバ内でも分割）	×
				表の分割数と不一致（サーバ内は非分割）	×
	フレキシブルハッシュ分割	該当しない		該当しない	×
	マトリクス分割	分割キーインデクス		表の分割数と一致	○
				表の分割数と不一致	−
		非分割キーインデクス	分割キーをすべて含む場合（順不同）	表の分割数と一致	○
				表の分割数と不一致	−
			分割キーをすべて含まない場合（順不同）	表の分割数と一致	×
				表の分割数と不一致	−

（凡例）

○：UNIQUE を指定できます。

×：UNIQUE を指定できません。

−：インデクスを定義できません。

注※1

サーバ内の横分割とは，HiRDB/シングルサーバでの横分割，又は HiRDB/パラレルサーバで一つのバックエンドサーバに閉じた横分割のことをいいます。

サーバ間の横分割とは，HiRDB/パラレルサーバで複数のバックエンドサーバにわたった横分割のことをいいます。なお，サーバ内の横分割とサーバ間の横分割が混在する場合は，サーバ間の横分割に分類します。表の横分割については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

注※2

分割キーインデクスとは，表を横分割するときに格納条件を指定した列（分割キー）を第 1 構成列とするインデクスのことで，次のインデクスを指します。

単一列分割の場合：

分割キーに作成した単一列インデクス，又は分割キーを先頭とする複数の列に作成した複数列インデクス

複数列分割の場合：

分割キーを先頭とし，かつ分割に指定した列を先頭から同順にすべて含んでいて，複数の列に作成した複数列インデクス

分割キーインデクス以外のインデクスを，**非分割キーインデクス**といいます。

注※3

ハッシュ分割表のリバランス機能使用時は×となります（RD エリアを追加してサーバ間分割に移行した場合に，UNIQUE が指定できなくなるため）。

注※4

インデクスの分割方法は，インデクス格納用 RD エリアの指定方法に依存します。IN オペランドを省略した場合は，フレキシブルハッシュ分割を除き，インデクスの分割方法が原因で UNIQUE を指定できなくなることはありません。IN オペランドで明示的にインデクス格納用 RD エリアを指定する場合には，UNIQUE を指定できないときがあるので注意してください。

2. 繰返し列を使用している場合

UNIQUE を指定したインデクスの構成列に，繰返し列を含むことはできません。

(b) 〔認可識別子.〕インデクス識別子

認可識別子には，作成したインデクスを所有するユーザの認可識別子を指定します。

インデクス識別子には，定義するインデクスの名称を指定します。

表所有者が定義した自分の所有するインデクス内に同じ名称は指定できません。

(c) 〔認可識別子.〕表識別子

認可識別子には，表を所有するユーザの認可識別子を指定します。

表識別子には，インデクスを作成する実表の名称を指定します。

(d) (列名〔{ASC | DESC}〕〔, 列名〔{ASC | DESC}〕〕…)

列名

インデクスを定義する列の名称を指定します。

列名は最大 64 個指定できます。

列名を複数個指定する場合，同じ名称は指定できません。

ASC

インデクスを昇順に定義する場合に指定します。

DESC

インデクスを降順に定義する場合に指定します。

(e) IN

```
{RDエリア名
  |（RDエリア名）
  |（（RDエリア名）〔,（RDエリア名）〕…）
  |マトリクス分割インデクス格納用RDエリア指定}
```

マトリクス分割インデクス格納用 RD エリア指定：：＝ 2 次元格納用 RD エリア指定

2 次元格納用 RD エリア指定：：＝（マトリクス分割用 RD エリアリスト〔，マトリクス分割用 RD エリアリスト〕…）

マトリクス分割用 RD エリアリスト：：＝（RD エリア名〔，RD エリア名〕…）

インデクスを格納する RD エリアの名称を指定します。

マトリクス分割インデクス格納用 RD エリア指定は，マトリクス分割表にインデクスを定義する場合に指定します。RD エリア名の規則については，マトリクス分割以外の表と同じです。

インデクスを格納する RD エリアについての規則を次に示します。

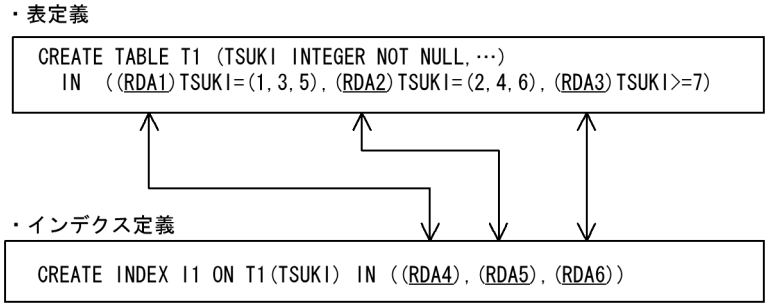
1. RD エリア名は一時表用 RD エリア以外のユーザ用 RD エリアである必要があります。また，HiRDB/パラレルサーバでは次の制限があります。表識別子で指定した表が共用表の場合，RD エリア名は共用 RD エリアでなければいけません。表識別子で指定した表が非共用表の場合，RD エリア名に共用 RD エリアは指定できません。
2. RD エリアは，あらかじめデータベース初期設定ユティリティで作成，又はデータベース構成変更ユティリティで追加されていなければなりません。
3. RD エリア名を省略すると，表識別子で指定した表が格納されている RD エリアに格納されます。ただし，HiRDB/シングルサーバで表を分割した場合，又は HiRDB/パラレルサーバで同じバックエンドサーバ内で表を分割した場合，分割キー以外の列を先頭に指定したインデクスは，最初に分割条件を指定した表格納先 RD エリアに格納されます。マトリクス分割表の場合，分割キーに指定したすべての列を，インデクス構成列の先頭から同順に指定しないときは，2 次元格納用 RD エリア指定を省略できません。
4. RD エリア名を複数個指定する場合，同じ RD エリア名は指定できません。ただし，境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表で，表格納用 RD エリア名が重複している場合は指定できます。
5. 表が複数の RD エリアに分割して格納されている場合，インデクス格納用 RD エリアの指定は次のようになります。

・分割キーインデクスの場合，RD エリア名は表を格納している RD エリアと同じ数だけ指定します。このとき，インデクスの格納先は CREATE TABLE で指定した表格納用 RD エリアの指定順に対応します。

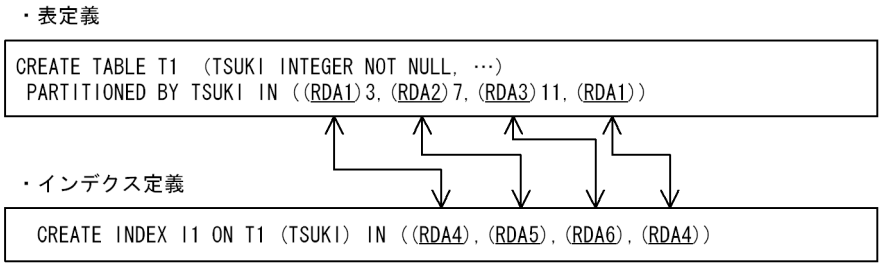
・境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表で，表格納用 RD エリア名が重複している場合は，それに対応するようにインデクス格納用 RD エリア名を指定します。

定義例を次に示します。

1．格納条件による横分割の場合



2．境界値による横分割の場合

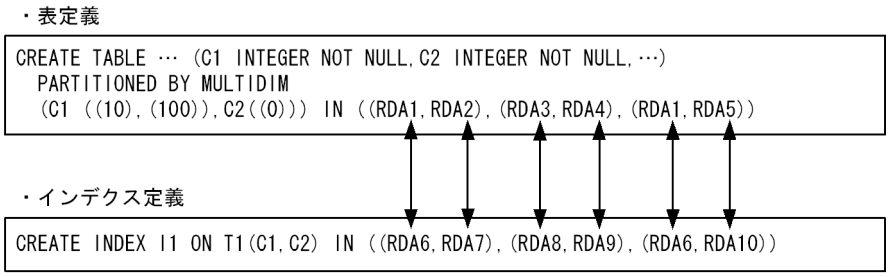


（凡例）

RDA1～6：RDエリア名を示します。

：表のRDエリアRDA1，RDA2，及びRDA3が，インデクスのRDエリアRDA4，RDA5，及びRDA6とそれぞれ対応していることを示します。

3．マトリクス分割の場合



（凡例）

RDA1～RDA10：RDエリア名を示します。

：表のRDエリアRDA1，RDA2，RDA3，RDA4，及びRDA5が，インデクスのRDエリアRDA6，RDA7，RDA8，RDA9，及びRDA10とそれぞれ対応していることを示します。

6. HiRDB/パラレルサーバの場合，表を格納する RD エリアと対応するインデクスを格納する RD エリアは同じバックエンドサーバ内の必要があります。

7. 非分割キーインデクスの場合の RD エリアの指定は次のようにしてください。

   ・インデクスをサーバ内横分割しない場合，RD エリア名は表を分割して格納しているサーバの数だけ指定します。HiRDB/パラレルサーバの場合，表があるバックエンドサーバごとに一つの RD エリアを指定します。HiRDB/シングルサーバの場合，一つ指定できます。ただし，マトリクス分割表の場合，分割キーインデクスのときと同じです。

   ・インデクスをサーバ内横分割する場合，RD エリア名は表を格納している RD エリアと同じ数だけ指定します。このとき，インデクスの格納先は CREATE TABLE で指定した表格納用 RD エリアの指定順に対応します。また，境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表で，表格納用 RD エリア名が重複している場合は，それに対応するようにインデクス格納用 RD エリア名を指定します。

   なお，指定順は任意であり，同一サーバ上の表格納 RD エリアに対応するインデクスが各々格納されます。

8. リバランス表を格納しているユーザ用 RD エリアは指定できません。

9. リバランス表に対してインデクスを定義する場合は，RD エリア名を必ず指定してください。

(f) PCTFREE=未使用領域の比率

インデクス作成時のインデクスページ内の未使用領域の比率を指定します。未使用領域の比率は，0～99（単位：%）を指定できます。省略すると，30%が仮定されます。仮定値は，システム共通定義の pd_ddl_idx_pctfree オペランドで変更できます。pd_ddl_idx_pctfree オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

データベース作成ユティリティ及びデータベース再編成ユティリティでのインデクスの一括作成時に，未使用領域の比率でインデクスが作成されます。INSERT 文や UPDATE 文など，ほかの追加，及び更新では PCTFREE=0 が仮定されます。

インデクスを定義した後，行の追加が頻繁に発生する場合，未使用領域の比率を高く設定してください。

(g) UNBALANCED SPLIT

ページスプリットする場合，両ページに対するキーの振り分けを不均衡にします。

スプリット対象ページへのキーの追加位置が，ページの前半部の場合はスプリット後の左側のページの空きを多く確保し，後半部の場合は右側のページの空きを多く確保するように分割します。このようなインデクスをアンバランスインデクススプリットといいます。

アンバランスインデクススプリットについては，マニュアル「HiRDB システム運用ガイド」を参照してください。

(h) EMPTY

未完状態のインデクスを作成する場合に指定します。

EMPTY を指定すると，インデクスを定義するときの同時実行性を向上させることができます。また，表の格納データが多く，複数のインデクスの定義を同時に実行する場合に効果があります。表にデータが格納されていない場合は効果がありません。

EMPTY オプションの使用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(i) 除外値指定

ナル値だけで構成されるキー値を除いて，インデクスを定義します。

非ナル値制約の列を含む場合には，除外値指定は指定できません。また，除外値指定を指定した場合，データベース再編成ユティリティでインデクス順にアンロードできません。

除外値を指定したインデクスの構成列に，繰返し列を含むことはできません。

(5) 共通規則

1. インデクスは，一つの表に最大 255 個定義できます。
2. インデクスは，ナル値を含む列，又は行のない列に対しても定義できます。
3. インデクスは，次に示すデータ型の列では定義できません。
   - BLOB
   - BINARY
   - 抽象データ型
4. インデクスを複数の列で構成する場合，項番 3 に加え，次に示すデータ型の列は指定できません。
   - FLOAT
   - SMALLFLT
5. インデクスを構成する列の長さの合計は，次の計算式を満たすようにしてください。

   列の長さの合計≦

   MIN((インデクス格納用 RD エリアのページサイズ÷2)－1242，4036)

   定義するインデクスが一時インデクスの場合は，一時表を実体化する際に上記を満たすかどうかをチェックします。

   一時表の実体化については，マニュアル「HiRDB システム導入・設計ガイド」の「一時表」を参照してください。

   インデクスを構成する各列の長さを次の表に示します。

表 3-19　インデクスを構成する列の長さ

データ型	各列の長さの合計が 255 バイト以下の場合			各列の長さの合計が 256 バイト以上の場合		
	単一列インデクスを構成する列	複数列インデクスを構成する列		単一列インデクスを構成する列	複数列インデクスを構成する列	
		構成列が固定長だけの場合	構成列に可変長を含む		構成列が固定長だけの場合	構成列に可変長を含む
INTEGER	4	5	6	−	5	7
SMALLINT	2	3	4	−	3	5
DECIMAL〔(m〔, n〕)〕	↓m÷2↓＋1	↓m÷2↓＋2	↓m÷2↓＋3	−	↓m÷2↓＋2	↓m÷2↓＋4
FLOAT	8	−	−	−	−	−
SMALLFLT	4	−	−	−	−	−
CHAR（n），MCHAR（n）	n1	n1＋1	n1＋2	n1	n1＋1	n1＋3
NCHAR（n）	2×n2	2×n2＋1	2×n2＋2	2×n2	2×n2＋1	2×n2＋3
DATE	4	5	6	−	5	7
TIME	3	4	5	−	4	6
TIMESTAMP	7＋p÷2	8＋p÷2	9＋p÷2	−	8＋p÷2	10＋p÷2
INTERVAL YEAR TO DAY	5	6	7	−	6	8
INTERVAL HOUR TO SECOND	4	5	6	−	5	7
VARCHAR, MVARCHAR	n1+1	−	n1＋2	n1+2	−	n1＋3
NVARCHAR	2×n2+1	−	2×n2＋2	2×n2+2	−	2×n2＋3

（凡例）

m，n：正の整数

n1：実際のデータ長

n2：文字数

p：小数秒精度

−：該当しません。

6. 同じ列構成のインデクス（クラスタキー，主キーも含む）は，一つだけ定義できます。同じ列構成とは，次の条件をすべて満たすものをいいます。
   - 列名の並び，及び列数が一致している
   - 昇順，降順の指定がすべて一致している，又はすべて逆になっている
7. インデクスを定義しようとしている表に対して手続き及びトリガが定義されている場合，その SQL オブジェクト中のインデクス情報は無効になります。この場合，トリガは実行できなくなります。また，

手続きからこの手続き又はトリガは実行できなくなるため，SQL オブジェクトを再作成する必要があります。

8. 同一のインデクスオプションは，繰り返して指定できません。
9. 複数の繰返し列から構成されるインデクスを定義した場合，それらの繰返し列の現在要素数は同じにしてください。
10. 実行中の SQL オブジェクト中のインデクス情報が無効になる場合，Java 手続き中から CREATE INDEX は実行できません。
11. インデクスを格納する RD エリアに，インナレプリカ機能を適用している RD エリアと適用していない RD エリアは混在して指定できません。インナレプリカ機能を適用している RD エリアを指定する場合，RD エリア名にはオリジナル RD エリア名を指定します。
12. インナレプリカ機能を使用している場合の CREATE INDEX の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
13. 一つの RD エリアに格納できるインデクスは，最大 500 個です。
14. インデクスの構成列に，予備列は指定できません。
15. 一時インデクスを定義する場合，次の指定はできません。
    - 格納用 RD エリア名
    - EMPTY
16. SQL セッション固有一時表を使用している SQL セッション中で，使用中の SQL セッション固有一時表に対してインデクスは定義できません。

(6) 留意事項

1. インデクスが付いている列の値をユーザが更新すると，インデクスも更新されます。
2. 複数列インデクスを定義すると，各列の指定順位はキー値作成の指定順位になります。
3. 単一列インデクスが定義されている列にも，複数列インデクスを定義できます。
4. CREATE INDEX は，OLTP 下の X/Open に従った UAP からは指定できません。
5. EMPTY を指定してインデクスを定義した場合，データベース再編成ユティリティでインデクスを再作成する必要があります。インデクスの再作成については，マニュアル「HiRDB コマンドリファレンス」を参照してください。
6. WITHOUT ROLLBACK を指定している表にインデクスを定義した場合の規則については，「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
7. 除外値指定を指定したインデクスは，除外値の行が選択される可能性がある SQL 文では使用されないので注意してください。
8. 大量のデータが登録されている表に対して CREATE INDEX 文でインデクスを作成する場合，処理時間が長くなるおそれがあります。CREATE INDEX 文を実行する前に，タイマ監視に関する指定を次のように設定してください。

- クライアント環境定義 PDCWAITTIME

  実績やデータ量から CREATE INDEX の実行時間を推測できる場合は，0 以外の余裕を持たせた値を指定してください。

  実行時間を推測できない場合は，0 又は指定を省略してください。

9. HiRDB/パラレルサーバでサーバ間横分割した表にインデクスを定義する場合で，表格納用 RD エリア，又は指定したインデクス格納用 RD エリアが閉塞状態のときは，RD エリアの排他待ちとなります。このとき，排他待ちの時間がシステム定義 pd_lck_wait_timeout オペランドの指定値に達してタイムアウトしても，すぐにエラーリターンされないことがあります。これを避けるため，CREATE INDEX 文を実行する前に，表格納用 RD エリア，又は指定したインデクス格納用 RD エリアの閉塞状態を解除してください。

(7) 使用例

1. 在庫表（ZAIKO）の商品コード（SCODE）列に昇順のインデクス（IDX1）を定義します。ただし，インデクス定義後も行の追加が頻繁に発生すると仮定し，インデクスページ内の未使用領域の比率を 50 パーセントに指定します。

```
CREATE INDEX IDX1
    ON ZAIKO(SCODE ASC)
    PCTFREE = 50
```

2. 在庫表（ZAIKO）の商品名（SNAME）列と色（COL）列を組にして，複数列インデクス（IDX2）を定義します。このとき，インデクスをユーザ用 RD エリア（RDA1）に格納します。

```
CREATE INDEX IDX2
    ON ZAIKO(SNAME,COL)
    IN RDA1
```

3. 在庫表（ZAIKO）の商品コード（SCODE）列に昇順のインデクス（IDX3）を定義します。このとき，インデクスを RDA1，RDA2，RDA3 の RD エリアに分割して格納します。

   ただし，在庫表は商品コードを分割キーにし，三つの RD エリアに分割して格納されているものとします。

```
CREATE INDEX IDX3
    ON ZAIKO(SCODE)
    IN ((RDA1),(RDA2),(RDA3))
```

3.12　CREATE INDEX　形式 2（インデクス定義）

3.12.1　CREATE INDEX 形式 2 の形式と規則

(1)　機能

インデクス型を指定したインデクスを定義します。

(2)　使用権限

表の所有者

公用 RD エリアに自分が所有する表のインデクスを定義できます。

私用 RD エリアの利用権限を持つ表の所有者

利用権限を持つ私用 RD エリアに自分が所有する表のインデクスを定義できます。

(3)　形式 2 ＜インデクス型を指定したインデクスの定義＞

```
CREATE INDEX 〔認可識別子.〕インデクス識別子
      USING TYPE 〔認可識別子.〕インデクス型識別子
      ON 〔認可識別子.〕表識別子（列名）
      IN {RDエリア名
          |（RDエリア名）
          |（（RDエリア名）〔,（RDエリア名）〕…）
          |マトリクス分割インデクス格納用RDエリア指定} ]
       〔インデクスオプション〕
       〔PLUGIN プラグインオプション〕
マトリクス分割インデクス格納用RDエリア指定：：＝2次元格納用RDエリア指定
2次元格納用RDエリア指定：：＝
            （マトリクス分割用RDエリアリスト〔, マトリクス分割用RDエリアリスト〕…）
マトリクス分割用RDエリアリスト：：＝（RDエリア名〔, RDエリア名〕…）
インデクスオプション：：＝EMPTY
```

(4)　オペランド

(a)　〔認可識別子.〕インデクス識別子

認可識別子には，作成するインデクスを所有するユーザの認可識別子を指定します。

インデクス識別子には，定義するインデクスの名称を指定します。

表所有者が定義した自分の所有するインデクス内に同じ名称は指定できません。

(b) USING TYPE 〔認可識別子.〕インデクス型識別子

認可識別子

インデクス型の所有者の認可識別子を指定します。

認可識別子を省略した場合，省略時に仮定される認可識別子に同一名称のインデクス型がないときは，認可識別子'MASTER'に同一名称のインデクス型があれば，そのインデクス型を指定したものとします。

インデクス型識別子

インデクス型を指定します。インデクス型識別子については，各種プラグインマニュアルを参照してください。

(c) 〔認可識別子.〕表識別子（列名）

認可識別子

認可識別子には，表を所有するユーザの認可識別子を指定します。

表識別子

表識別子には，インデクスを作成する実表の名称を指定します。

FIX 表は指定できません。

列名

インデクスを定義する列の名称を指定します。列のデータ型は，抽象データ型だけです。

(d) IN

```
{RDエリア名
 | （RDエリア名）
 | （（RDエリア名）〔，（RDエリア名）〕…）
 |マトリクス分割インデクス格納用RDエリア指定}
```

マトリクス分割インデクス格納用 RD エリア指定：：＝ 2 次元格納用 RD エリア指定

2 次元格納用 RD エリア指定：：＝（マトリクス分割用 RD エリアリスト〔，マトリクス分割用 RD エリアリスト〕…）

マトリクス分割用 RD エリアリスト：：＝（RD エリア名〔，RD エリア名〕…）

インデクスを格納する RD エリアの名称を指定します。

マトリクス分割表にインデクスを定義する場合，マトリクス分割インデクス格納用 RD エリア指定を指定してください。

インデクスを格納する RD エリアについての規則を次に示します。

1. RD エリア名はユーザ LOB 用 RD エリアでなければなりません。
2. RD エリアは，あらかじめデータベース初期設定ユティリティで作成，又はデータベース構成変更ユティリティで追加されていなければなりません。

3. 指定する RD エリア名は，表格納用 RD エリアと同じサーバに定義しているユーザ LOB 用 RD エリアを指定してください。
4. RD エリア名を複数個指定する場合，同じ RD エリア名は指定できません。ただし，境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表で，表格納用 RD エリア名が重複している場合は指定できます。
5. 表が複数の RD エリアに分割して格納されている場合，インデクス格納用 RD エリアの指定は，次に示すとおりです。

   ・RD エリア名は，表を格納している RD エリアと同じ数だけ指定します。このとき，インデクスの格納先は CREATE TABLE で指定した表格納用 RD エリアの指定順に対応します。

   ・境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表で，表格納用 RD エリア名が重複している場合は，それに対応するようにインデクス格納用 RD エリア名を指定します。

(e) EMPTY

未完状態のプラグインインデクスを作成する場合に指定します。

EMPTY を指定すると，プラグインインデクスを定義するときの同時実行性が向上します。また，表の格納データが多く，複数のインデクスの定義を同時に実行する場合に効果があります。表にデータが格納されていない場合は効果がありません。

EMPTY オプションの使用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(f) PLUGIN プラグインオプション

プラグインインデクスに対するパラメタ情報を文字列定数（最大 255 バイト）で指定します。パラメタ情報には，16 進文字列定数は指定できません。

パラメタ情報については，各種プラグインマニュアルを参照してください。

(5) 共通規則

1. インデクスは，一つの表に最大 255 個定義できます。
2. インデクスは，ナル値を含む列，又は行のない列に対しても定義できます。
3. インデクスを格納する RD エリアに，インナレプリカ機能を適用している RD エリアと適用していない RD エリアは混在して指定できません。インナレプリカ機能を適用している RD エリアを指定する場合，RD エリア名にはオリジナル RD エリア名を指定します。
4. インナレプリカ機能を使用している場合の CREATE INDEX の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
5. 一つの RD エリアに格納できるインデクスは，最大 500 個です。

(6) 留意事項

1. インデクスが付いている列の値をユーザが更新すると，インデクスも更新されます。
2. CREATE INDEX は，OLTP 下の X/Open に従った UAP から指定できません。
3. EMPTY を指定してプラグインインデクスを定義した場合，データベース再編成ユティリティでプラグインインデクスを再作成する必要があります。インデクスの再作成については，マニュアル「HiRDB コマンドリファレンス」を参照してください。
4. 大量のデータが登録されている表に対して CREATE INDEX 文でインデクスを作成する場合，処理時間が長くなるおそれがあります。CREATE INDEX 文を実行する前に，タイマ監視に関する指定を次のように設定してください。
   - クライアント環境定義 PDCWAITTIME

     実績やデータ量から CREATE INDEX の実行時間を推測できる場合は，0 以外の余裕を持たせた値を指定してください。

     実行時間を推測できない場合は，0 又は指定を省略してください。
5. HiRDB/パラレルサーバでサーバ間横分割した表にインデクスを定義する場合で，表格納用 RD エリア，又は指定したインデクス格納用 RD エリアが閉塞状態のときは，RD エリアの排他待ちとなります。このとき，排他待ちの時間がシステム定義 pd_lck_wait_timeout オペランドの指定値に達してタイムアウトしても，すぐにエラーリターンされないことがあります。これを避けるため，CREATE INDEX 文を実行する前に，表格納用 RD エリア，又は指定したインデクス格納用 RD エリアの閉塞状態を解除してください。

3.13　CREATE INDEX　形式 3（部分構造インデクス定義）

3.13.1　CREATE INDEX 形式 3 の形式と規則

(1)　機能

XML 型列の値の特定の部分構造をキーとするインデクスを定義します。

使用権限，形式，規則などの詳細については，「CREATE INDEX　形式 3（部分構造インデクス定義）」を参照してください。

3.14 CREATE 〔PUBLIC〕PROCEDURE（手続き定義，パブリック手続き定義）

3.14.1 CREATE PROCEDURE（手続き定義）

(1) 機能

手続きを定義します。

(2) 使用権限

スキーマを所有するユーザ

自分が所有する手続きを定義できます。

(3) 形式

```
  CREATE 手続き本体
  手続き本体：：＝PROCEDURE 〔認可識別子.〕ルーチン識別子
                  （〔 {IN｜OUT｜INOUT} SQLパラメタ名 データ型
                       〔, {IN｜OUT｜INOUT} SQLパラメタ名 データ型〕…〕）
        〔DYNAMIC RESULT SETS 結果集合数〕
        〔LANGUAGE句〕
        〔SQLコンパイルオプション〔SQLコンパイルオプション〕…〕
        {SQL手続き文｜外部ルーチン指定}

  LANGUAGE句：：＝LANGUAGE {SQL｜JAVA｜C}
  SQLコンパイルオプション：：＝ {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE｜COMPAT
IBLE}〕
                               ｜OPTIMIZE LEVEL SQL最適化オプション
                                        〔, SQL最適化オプション〕…
                               ｜ADD OPTIMIZE LEVEL SQL拡張最適化オプション
                                        〔, SQL拡張最適化オプション〕…
                               ｜SUBSTR LENGTH 文字の最大長 }
  外部ルーチン指定：：＝EXTERNAL NAME {外部Javaルーチン名｜外部Cストアドルーチン名}
                       PARAMETER STYLE パラメタスタイル
  パラメタスタイル：：＝ {JAVA｜RDSQL}
```

(4) オペランド

(a) 〔認可識別子.〕ルーチン識別子

認可識別子

定義する手続きの所有者の認可識別子を指定します。

ルーチン識別子

定義する手続きのルーチンの名前を指定します。

(b) （〔{IN | OUT | INOUT} SQL パラメタ名 データ型〔, {IN | OUT | INOUT} SQL パラメタ名 データ型〕…〕）

{IN | OUT | INOUT}

手続きのパラメタの入出力モード（パラメタモード）を指定します。

IN

入力パラメタの場合に指定します。

OUT

出力パラメタの場合に指定します。

INOUT

入力及び出力パラメタの場合に指定します。

SQL パラメタ名

手続きのパラメタの名称を指定します。一つの手続き中で，SQL パラメタ名は重複して指定できません。

データ型

手続きのパラメタのデータ型を指定します。

指定するデータ型が抽象データ型で，認可識別子を省略した場合，省略時に仮定される認可識別子に同一名称の抽象データ型がないときは，認可識別子'MASTER'に同一名称の抽象データ型があれば，その抽象データ型を指定したものとします。

次のデータ型は指定できません。

- LANGUAGE 句で JAVA 又は C を指定した場合，抽象データ型
- LANGUAGE 句で C を指定した場合，BINARY 型及び BLOB 型

指定できるデータ型については，「型マッピング」又は「CREATE 〔PUBLIC〕FUNCTION（関数定義，パブリック関数定義）」を参照してください。

(c) DYNAMIC RESULT SETS 結果集合数

定義する手続きが返す，結果集合の最大数を整数値で指定します。

指定できる値は 0～1023 です。

結果集合数に 0 を指定するか，又はこのオペランドを省略した場合，結果集合を返さない手続きとみなされます。

ただし，LANGUAGE 句で C を指定した場合，このオペランドの指定を無視します。

(d) LANGUAGE {SQL | JAVA | C}

定義する手続きの記述言語を指定します。

外部ルーチン指定を記述する場合，記述言語に SQL は指定しないでください。

SQL

手続きの処理部分を SQL 手続き文にする場合に指定します。

JAVA

手続きの処理部分を外部ルーチン指定にする場合で，手続きを Java クラスのメソッドで実装するときに指定します。

C

手続きの処理部分を外部ルーチン指定にする場合で，手続きを C 言語で記述されたモジュールによって実装するときに指定します。C を指定した場合，SQL パラメタの最大個数が 128 に制限されます。

このオペランドの指定によって，他オペランドの指定要否が変わります。LANGUAGE 句指定による他オペランドの指定要否を次の表に示します。

表 3-20　CREATE PROCEDURE の LANGUAGE 句指定による他オペランドの指定要否

他オペランド	LANGUAGE 句		
	SQL	JAVA	C
EXTERNAL NAME	×	○	○
PARAMETER STYLE	×	JAVA	RDSQL
SQL 手続き文	○	×	×

（凡例）

○：指定してください。

×：指定しないでください。

JAVA：JAVA を指定してください。

RDSQL：RDSQL を指定してください。

(e)　SQL コンパイルオプション

```
: : = {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE} 〕
       | OPTIMIZE LEVEL SQL最適化オプション〔, SQL最適化オプション〕…
       | ADD OPTIMIZE LEVEL SQL拡張最適化オプション〔, SQL拡張最適化オプション〕…
       | SUBSTR LENGTH 文字の最大長 }
```

SQL コンパイルオプションには ISOLATION，OPTIMIZE LEVEL，ADD OPTIMIZE LEVEL，SUBSTR LENGTH をそれぞれ 1 回しか指定できません。

〔ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE}〕〕

SQL のデータ保証レベルを指定します。

データ保証レベル

データ保証レベルとは，トランザクションのどの時点までデータの内容を保証するかのレベルをいいます。次に示すデータ保証レベルを指定できます。

- 0

  データの内容を保証しない場合に指定します。0 レベルを指定すると，ほかのユーザが更新中のデータでも，更新完了を待たないで参照できます。ただし，参照する表が共用表の場合，ほかのユーザが排他モードで LOCK 文を実行しているときには，排他解除待ちとなります。

- 1

  検索処理の終了までデータの内容を保証したい場合に指定します。1 レベルを指定すると，検索処理が終了するまで（HiRDB がページ，又は行を見終わるまで）1 度検索したデータをほかのユーザは更新できません。

- 2

  トランザクションの終了まで 1 度検索したデータの内容を保証したい場合に指定します。2 レベルを指定すると，トランザクションが終了するまで 1 度検索したデータをほかのユーザは更新できません。

〔FOR UPDATE 〔EXCLUSIVE | COMPATIBLE〕〕

手続き中で，FOR UPDATE 句を指定した（FOR UPDATE が仮定される場合を含む）カーソル又は問合せに対して，SQL コンパイルオプションで指定したデータ保証レベルに関係なく，常に WITH EXCLUSIVE LOCK を仮定する場合に指定します。

このオペランドを省略した場合，EXCLUSIVE を仮定します。ただし，0904 互換モードを適用している場合は COMPATIBLE を仮定します。

バージョン 09-50 より前の HiRDB で FOR UPDATE EXCLUSIVE を省略した手続きと同じ動作をさせたい場合は COMPATIBLE を指定します。

バージョン 09-50 より前の HiRDB から，09-50 以降の HiRDB にバージョンアップする場合の注意事項を次に示します。

- 手続きを再定義する場合で，バージョン 09-50 より前の HiRDB で FOR UPDATE EXCLUSIVE を省略した手続きと同じ動作をさせたい場合は次のどちらかの対処をしてください。

  ・ALTER PROCEDURE 又は ALTER ROUTINE で手続きを再定義してください。

  ・手続きを削除し，CREATE PROCEDURE で手続きを再定義する場合は，COMPATIBLE を指定して再定義してください。

このオペランドと，ISOLATION データ保証レベルの関係から決まる FOR UPDATE の排他オプションを次に示します。

ISOLATION データ保証レベル	FOR UPDATE EXCLUSVIE	FOR UPDATE COMPATIBLE
0	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
1	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
2	WITH EXCLUSIVE LOCK	WITH EXCLUSIVE LOCK

《クライアント環境定義との関係》

CREATE PROCEDURE に対して，PDISLLVL，PDFORUPDATEEXLOCK の指定は無効となります。

《SQL との関係》

手続き中の SQL 文に排他オプションを指定している場合は SQL コンパイルオプションで指定するデータ保証レベル，FOR UPDATE EXCLUSIVE，及び FOR UPDATE COMPATIBLE から仮定する排他オプションより SQL 文に指定した排他オプションが優先されます。

《システム定義との関係》

データ保証レベルを省略すると，pd_isolation_level オペランドの指定値が仮定されます。
pd_isolation_level オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

データ保証レベルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

〔OPTIMIZE LEVEL SQL 最適化オプション〔，SQL 最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDSQLOPTLVL」を参照してください。

SQL 最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法がありますが，通常時は識別子で指定する方法をお勧めします。

識別子で指定する場合：

```
OPTIMIZE LEVEL ”識別子”〔，”識別子”〕…
```

＜指定例＞

- ネストループジョイン優先とグループ分け高速化処理を適用する場合
  OPTIMIZE LEVEL "PRIOR_NEST_JOIN","RAPID_GROUPING"
- すべての最適化を適用しない場合
  OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。ただし，同時に"NONE"以外の識別子を指定すると，"NONE"は無効になります。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
OPTIMIZE LEVEL 符号なし整数〔，符号なし整数〕…
```

＜指定例＞

- 複数の SQL オブジェクト作成，AND の複数インデクス利用の抑止，及び複数インデクス利用の強制を適用する場合

  符号なし整数をコンマで区切って指定する場合：

  OPTIMIZE LEVEL 4,10,16

  符号なし整数の和を指定する場合：

  OPTIMIZE LEVEL 30

- 既に 14（4+10）を指定していて，新たに 16 を追加する場合

  OPTIMIZE LEVEL 14,16

- すべての最適化を適用しない場合

  OPTIMIZE LEVEL 0

＜規則＞

1. バージョン 06-00 より前の HiRDB から，バージョン 06-00 以降の HiRDB にバージョンアップする場合，バージョン 06-00 より前の合計値指定も有効となります。最適化オプションを変更する必要がない場合は，バージョン 06-00 以降の HiRDB にバージョンアップしたときにこのオペランドの指定値を変更する必要はありません。
2. 符号なし整数は一つ以上指定してください。
3. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
4. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。ただし，同時に 0 以外の識別子を指定すると，0 は無効になります。
5. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。
6. 複数の最適化方法を指定する場合，その符号なし整数の和を指定することもできます。ただし，同じ最適化方法の値は二つ以上足さないでください（足した結果が別の最適化方法とみなされることもあるため）。
7. 複数の最適化方法の値を足して指定する場合，どの最適方法を指定しているのか分かりにくくなるため，コンマで区切って指定する方法をお勧めします。また，既に複数の最適化方法の値を足して指定している場合で，新たに別の最適化方法が必要になったときは，追加する値をコンマで区切って後ろに指定できます。

《システム定義との関係》

1. SQL 最適化オプションを省略すると，システム定義の pd_optimize_level オペランドの指定値が仮定されます。pd_optimize_level オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

2. システム定義の pd_floatable_bes オペランド，又は pd_non_floatable_bes オペランドを指定している場合，「フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）」及び「フロータブルサーバ対象限定（データ取り出しバックエンドサーバ）」の指定は無効となります。
3. システム定義の pd_indexlock_mode オペランドに KEY を指定している場合（インデクスキー値排他の場合），「更新 SQL の作業表作成抑止」の指定は無効になります。

《クライアント環境定義との関係》

CREATE PROCEDURE に対して，PDSQLOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔ADD OPTIMIZE LEVEL SQL 拡張最適化オプション〔，SQL 拡張最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 拡張最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDADDITIONALOPTLVL」を参照してください。

SQL 拡張最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法があります。

識別子で指定する場合：

```
ADD OPTIMIZE LEVEL ”識別子”〔,”識別子”〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL "COST_BASE_2","APPLY_HASH_JOIN"

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
ADD OPTIMIZE LEVEL 符号なし整数〔,符号なし整数〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL 1,2

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL 0

＜規則＞

1. 符号なし整数は一つ以上指定してください。
2. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。
4. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。

《システム定義との関係》

SQL 拡張最適化オプションを省略すると，システム定義の pd_additional_optimize_level オペランドの指定値が仮定されます。pd_additional_optimize_level オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

《クライアント環境定義との関係》

CREATE PROCEDURE に対して，PDADDITIONALOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔SUBSTR LENGTH 文字の最大長〕

1 文字を表現する最大バイト数を指定します。

文字の最大長に指定できる値は，3～6（pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8_ivs を指定した場合は 3～10）です。

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，又は utf-8_ivs を指定した場合にだけ有効となり，スカラ関数 SUBSTR の結果の長さに影響します。SUBSTR については，「SUBSTR」を参照してください。

《システム定義との関係》

SUBSTR LENGTH を省略すると，システム定義の pd_substr_length オペランドの指定値が仮定されます。pd_substr_length オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

《クライアント環境定義との関係》

CREATE PROCEDURE に対して，PDSUBSTRLEN の指定は無効となります。PDSUBSTRLEN については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

《pdntenv コマンド又は pdsetup コマンドで指定した文字コード種別との関係》

文字コード種別に utf-8，又は utf-8_ivs を指定した場合だけ有効となります。

そのほかの文字コード種別の場合は，構文チェックだけ行い，指定を無視します。

(f)　SQL 手続き文

手続きで実行する SQL 手続き文を指定します。SQL 手続き文については，「7.　ルーチン制御 SQL」の「全般規定」を参照してください。

(g)　EXTERNAL NAME　{外部 Java ルーチン名｜外部 C ストアドルーチン名}

外部 Java ルーチン名

Java 言語で記述した Java メソッドを外部ルーチンとして指定します。外部 Java ルーチン名の指定方法については，「外部 Java ルーチン名」を参照してください。

外部 C ストアドルーチン名

C 言語で記述した C 関数を外部ルーチンとして指定します。外部 C ストアドルーチン名の指定方法については，「外部 C ストアドルーチン名」を参照してください。

(h)　PARAMETER STYLE パラメタスタイル

外部ルーチンを呼び出す場合，パラメタとして引き渡される内容を指定します。

パラメタスタイルが JAVA の場合

SQL データ型で定義している外部 Java 手続きのパラメタは，SQL データ型に対応した Java データ型の Java メソッドのパラメタとして渡されます。

SQL データ型で定義されている外部 Java 手続きの OUT 及び INOUT パラメタが，SQL データ型に対応する Java データ型であり，要素数 1 の配列である Java メソッドのパラメタとして渡されます。Java メソッド終了後，Java メソッドが書き込んだ配列は，外部 Java 手続きの出力パラメタとして扱われます。

パラメタスタイルが RDSQL の場合

SQL パラメタ数を n とすると，外部 C 関数の SQL パラメタは，次の表に示すように，C 関数のパラメタとして渡されます。

表 3-21　C 関数のパラメタとして渡される内容

パラメタ番号 (n は SQL パラメタ数)	内容	内容の詳細	データ型
1 番目～n 番目	SQL パラメタのデータ部	SQL パラメタのデータ部に対応するパラメタです。各パラメタの入出力モードは，対応する SQL パラメタの入出力モードと同じです。	SQL パラメタのデータ型に対応するデータ型※1
n+1 番目～2n 番目	SQL パラメタの標識部	SQL パラメタの標識部に対応するパラメタです。各パラメタの入出力モードは，対応する SQL パラメタの入出力モードと同じです。	short*

パラメタ番号 (n は SQL パラメタ数)	内容	内容の詳細	データ型
		入力パラメタである場合，データがナル値ならば，負の値を設定して C 関数に渡します。 出力パラメタである場合，外部 C ストアドルーチンを実装した C 関数内では，次の説明に従って標識部を設定してください。 • 出力値がナル値の場合 設定値：－1 • 出力値が非ナル値の場合 設定値：0	
2n+1 番目	SQLSTATE	外部 C ストアドルーチンを実装した C 関数が，SQLSTATE の値を設定するための出力パラメタです。領域長は 6 バイトです。領域の先頭から 5 バイト目までに SQLSTATE の値を設定してください。外部 C ストアドルーチンを実装した C 関数内では，次の説明に従って，SQLSTATE の値を設定してください。 • C 関数が正常終了したとき 設定値：‘00000’ SQL 実行結果：正常終了します。 • C 関数が異常終了したとき 設定値：形式’38XYY’の値 X，Y は，それぞれ次の範囲の値です。 X：’I’～’Z’ Y：’0’～’9’又は’A’～’Z’ （例）’38I01’，’38ZCD’ SQL 実行結果：SQL エラーになります。 • C 関数が任意の状態で終了したとき 設定値：‘00000’，形式’38XYY’の値以外 SQL 実行結果：SQL エラーになります。	char*
2n+2 番目	ルーチン名	ルーチン名を表す入力パラメタです。	struct{ short 変数名 1; char 変数名 2[30]; }*※2
2n+3 番目	特定名	手続きを特定するための特定名を表す入力パラメタです。	struct{ short 変数名 1; char 変数名 2[30]; }*※2
2n+4 番目	メッセージテキスト	外部 C ストアドルーチンを実装した C 関数内でエラーが発生した場合，エラーが発生した詳細な理由を設定するための出力パラメタです。	struct{ short 変数名 1; char 変数名 2[80]; }*※2

パラメタ番号 (n は SQL パラメタ数)	内容	内容の詳細	データ型
		外部 C ストアドルーチンの実行終了時，SQLSTATE にクラス 38 の値が設定されていると，メッセージテキストを埋め込んだエラーメッセージが出力されます。 なお，設定できるメッセージテキストの長さは，最大 80 バイトです。	

注※1

外部 C 手続き定義時に指定できる SQL パラメタのデータ型と，外部 C ストアドルーチンを実装した C 関数に渡すパラメタのデータ型の対応関係については，外部 C 関数定義時の対応関係と同じです。詳細は，「CREATE 〔PUBLIC〕FUNCTION（関数定義，パブリック関数定義)」を参照してください。

注※2

変数名 1 に文字列長（バイト数)，変数名 2 に内容を表す文字列を設定します。

(5) 共通規則

1. ルーチン識別子は，以下に示す手続きの識別子と同じ識別子を指定できません。
   - 所有者の手続き
   - パブリック手続きの手続き
2. 入力パラメタは，CALL 文の対象となるルーチンの引数の入出力パラメタが OUT 又は INOUT で定義されたもの，FETCH 文，及び 1 行 SELECT 文の INTO 句，並びに代入文の代入先には指定できません。
3. 出力パラメタは，CALL 文の対象となるルーチンの引数の入出力パラメタが OUT 又は INOUT で定義されたもの，FETCH 文の INTO 句，1 行 SELECT 文の INTO 句，代入文の代入先，及び WRITE LINE 文の値式以外では指定できません。
4. 手続きのパラメタは，30,000 個（LANGUAGE 句が C の場合は 128 個）以下でなければなりません。ただし，LANGUAGE 句に Java を指定した場合，30,000 個以下でも，使用する JavaVM によっては，その仕様上の制限によって実行時にエラーになる場合があります。
5. 手続きを特定するための特定名は，〔認可識別子.〕ルーチン識別子と同じです。
6. 入力パラメタ，及び出力パラメタには，BOOLEAN 型を指定できません。
7. 手続き本体中に指定した CALL 文の手続き名が認可識別子で修飾されている場合，現在定義しようとしている手続きと，認可識別子，ルーチン識別子が一致する手続きは指定できません。

   手続き本体中に指定した CALL 文の手続き名が認可識別子で修飾されていない場合，現在定義しようとしている手続きと，ルーチン識別子が一致する手続きは指定できません。
8. ALTER PROCEDURE，又は ALTER ROUTINE で SQL コンパイルオプションを指定する場合，再作成する手続きの元の CREATE PROCEDURE に SQL コンパイルオプションを反映してできる SQL 文は，SQL 文の最大長を超えないようにしてください。

9. SQL 手続き文中の問合せ指定で WRITE 指定は指定できません。
10. LANGUAGE 句に C を指定している場合，結果集合を返却することはできません。

(6) 留意事項

1. CREATE PROCEDURE は，OLTP 下の X/Open に従った UAP から指定できません。
2. SQL パラメタは，ナル値が持てます。
3. SQL ルーチン中で複数の SQL 文を実行するためには，複合文などのルーチン制御 SQL を使用します。
4. SQL 手続きを定義すると，それを実行するためのアクセス手順を記述した SQL オブジェクトが作成されます。外部 JAVA 手続きを定義した場合，手続き中の SQL 文を実行するためのアクセス手順を記述したオブジェクトは作成されません。
5. 手続き中の SQL 文のデータ型のデータ保証レベル，SQL 最適化オプション， SQL 拡張最適化オプション，及び文字の最大長は，手続きの定義時，又は変更時の指定で決まり，手続き呼び出し時のシステム定義やクライアント環境定義の影響を受けません。
6. ルーチン内でのルーチンの呼び出しを多数又は無限に繰り返す場合，OS のスタックがオーバフローすることがあります。
7. 外部 Java 手続きから次の SQL は実行できません。
   - COMMIT 文，LOCK 文，ROLLBACK 文以外の制御系 SQL
   - ルーチン制御 SQL
8. SQL 手続き中では，結果集合を受け取ることはできません。
9. 外部 Java 手続き中で取得した DatabaseMetaData クラスの，メソッドが返却する結果集合（ResultSet）は，動的結果集合として返却できません。外部 Java 手続きのコール元のコネクションのメタデータを使用して，情報を取得してください。

3.14.2 CREATE PUBLIC PROCEDURE（パブリック手続き定義）

(1) 機能

すべてのユーザから，認可識別子でルーチン識別子を修飾しなくても使用できる手続き（パブリック手続き）を定義できます。

(2) 使用権限

スキーマを所有するユーザ

自分が所有するパブリック手続きを定義できます。

(3) 形式

```
CREATE PUBLIC 手続き本体
```

(4) オペランド

(a) PUBLIC

手続きをパブリック手続きとして定義する場合に指定します。

パブリック手続きにすると，同じ内容の手続きをユーザごとに定義しなくても，一つの手続きをルーチン識別子だけを指定して複数のユーザで使用できます。

(b) 手続き本体

「〔認可識別子.〕ルーチン識別子」以外の説明については，「CREATE PROCEDURE（手続き定義）」と同じです。

〔認可識別子.〕ルーチン識別子

認可識別子

パブリック手続きのため，認可識別子は指定できません。

ルーチン識別子

定義するパブリック手続きのルーチンの名前を指定します。

(5) 共通規則

1. 手続き本体の「〔認可識別子.〕ルーチン識別子」の認可識別子は指定できません。
2. ルーチン識別子は，システム内のすべての手続き中で同じ識別子を使用できません。
3. 手続きを特定するための特定名は，PUBLIC.ルーチン識別子と同じです。
4. 手続き本体中に指定した CALL 文の手続き名が PUBLIC で修飾されている場合，現在定義しようとしているパブリック手続きと，認可識別子，ルーチン識別子が一致するパブリック手続きは指定できません。
   手続き本体中に指定した CALL 文の手続き名が認可識別子で修飾されていない場合，現在定義しようとしているパブリック手続きと，ルーチン識別子が一致する手続きは指定できません。
5. ALTER PROCEDURE，又は ALTER ROUTINE で SQL コンパイルオプションを指定する場合，再作成する手続きの元の CREATE PUBLIC PROCEDURE に SQL コンパイルオプションを反映してできる SQL 文は，SQL 文の最大長を超えないようにしてください。
6. その他の規則は，「CREATE PROCEDURE（手続き定義）」と同じです。

(6) 留意事項

1. CREATE PUBLIC PROCEDURE は，OLTP 下の X/Open に従った UAP から指定できません。
2. ディクショナリ表の所有者を格納する列（SQL_ROUTINES 表の ROUTINE_SCHEMA 列など）には PUBLIC が設定されます。また，パブリック手続きを定義した認可識別子は SQL_ROUTINES 表の ROUTINE_CREATOR 列に格納されます。
3. パブリック手続きの削除は，DROP PUBLIC PROCEDURE で行います。
4. その他の留意事項は，「CREATE PROCEDURE（手続き定義）」と同じです。

3.15 CREATE SCHEMA（スキーマ定義）

3.15.1 CREATE SCHEMA の形式と規則

(1) 機能

スキーマを定義します。

(2) 使用権限

スキーマ定義権限を持つユーザ

ユーザ自身のスキーマを定義できます。

DBA 権限を持つユーザ

CONNECT 権限又は DBA 権限を持つ，ほかのユーザのスキーマを定義できます。ただし，副監査人のスキーマは定義できません。

(3) 形式

```
CREATE SCHEMA スキーマ名句
スキーマ名句：：＝〔AUTHORIZATION 認可識別子〕
```

(4) オペランド

(a) AUTHORIZATION 認可識別子

ユーザの認可識別子を指定します。認可識別子を省略すると，実行するユーザの認可識別子が仮定されます。

副監査人の認可識別子は指定できません。

(5) 留意事項

1. 表，インデクス，抽象データ型，インデクス型，関数，手続き，トリガ，及びアクセス権限は，スキーマを定義した後で定義します。
2. 1 ユーザは，1 個のスキーマだけ所有できます。
3. スキーマ定義中で定義した表，及びインデクスの所有者の認可識別子は，スキーマの認可識別子と同じにしてください。
4. DBA 権限を持つユーザによってスキーマを定義されたユーザは，スキーマ定義権限が与えられます。
5. CREATE SCHEMA は，OLTP 下の X/Open に従った UAP から指定できません。

(6) 使用例

ユーザ（USER1）のスキーマを定義します。

```
    CREATE SCHEMA
        AUTHORIZATION USER1
```

3.16　CREATE SEQUENCE（順序数生成子定義）

3.16.1　CREATE SEQUENCE の形式と規則

(1)　機能

順序数生成子を定義します。

(2)　使用権限

スキーマを所有するユーザ

RD エリア利用権限のある私用 RD エリア，又は公用 RD エリアに順序数生成子を定義できます。

(3)　形式

```
CREATE SEQUENCE 〔認可識別子.〕順序数生成子識別子
                    〔 FOR PUBLIC USAGE〕
                    〔 順序数生成子オプション列〕
                    〔 IN 順序数生成子格納RDエリア名〕

順序数生成子オプション列：：＝順序数生成子オプション〔 順序数生成子オプション〕…
順序数生成子オプション：：＝｛順序数生成子データ型オプション
                            ｜共通順序数生成子オプション列｝
共通順序数生成子オプション列：：＝共通順序数生成子オプション
                                    〔 共通順序数生成子オプション〕…
共通順序数生成子オプション：：＝｛順序数生成子開始オプション
                                  ｜基本順序数生成子オプション｝
基本順序数生成子オプション：：＝｛順序数生成子増分オプション
                                  ｜順序数生成子最大値オプション
                                  ｜順序数生成子最小値オプション
                                  ｜順序数生成子循環オプション
                                  ｜順序数生成子ログ出力間隔オプション｝

順序数生成子データ型オプション　：：＝AS データ型
順序数生成子開始オプション　　：：＝START WITH 開始値
順序数生成子増分オプション　　：：＝INCREMENT BY 増分
順序数生成子最大値オプション　：：＝ {MAXVALUE 最大値｜NO MAXVALUE}
順序数生成子最小値オプション　：：＝ {MINVALUE 最小値｜NO MINVALUE}
順序数生成子循環オプション　　：：＝ {CYCLE｜NO CYCLE}
順序数生成子ログ出力間隔オプション：：＝LOG INTERVAL出力間隔
```

(4)　オペランド

(a)　〔認可識別子.〕順序数生成子識別子

認可識別子

定義する順序数生成子の所有者になるユーザの認可識別子を指定します。

省略した場合，実行するユーザの認可識別子を仮定します。

順序数生成子識別子

順序数生成子の識別子を指定します。

(b) FOR PUBLIC USAGE

すべてのユーザが使用可能な順序数生成子を定義する場合に指定します。

省略した場合，所有者だけが使用できる順序数生成子となります。

(c) IN 順序数生成子格納 RD エリア名

順序数生成子を格納するユーザ用 RD エリア名を指定します。

インナレプリカ機能を適用した RD エリア，及び一時表用 RD エリアは指定できません。

RD エリア名を省略した場合，次に示す優先順位で順序数生成子を格納する RD エリアを決定します。

1. 一時表用 RD エリア以外のインナレプリカ機能を適用しない RD エリアのうち，定義されている表と順序数生成子の合計が一番少ない公用 RD エリア
2. 1 の条件を満たす RD エリアが複数ある場合は，HiRDB が最初に見付けた RD エリア
3. 2 で決定できなかった場合はエラーになります。

(d) 順序数生成子オプション列

順序数生成子オプション列には，同一のオプションを繰り返して指定できません。

(e) 順序数生成子データ型オプション：：＝ AS データ型

順序数生成子が生成する値のデータ型を指定します。

データ型には INTEGER，SMALLINT，位取り 0 の DECIMAL が指定できます。

このオプションを省略した場合，INTEGER が仮定されます。

(f) 順序数生成子開始オプション：：＝ START WITH 開始値

順序数生成子の開始値となる整数値を指定します。

順序数生成子を定義した後，最初に NEXT VALUE 式を実行すると順序数生成子は開始値を返します。

順序数生成子開始オプションには，順序数生成子最小値以上で，かつ，順序数生成子最大値以下の整数値を指定します。

このオプションを省略した場合，昇順順序数生成子ならば順序数生成子最小値，降順順序数生成子ならば順序数生成子最大値が仮定されます。

昇順順序数生成子及び降順順序数生成子については順序数生成子増分オプションを参照してください。

(g) 順序数生成子増分オプション：：＝ INCREMENT BY 増分

順序数生成子が生成した値（現在値）を更新するときに加算する値を指定します。

増分が正なら昇順順序数生成子，負なら降順順序数生成子となります。

順序数生成子増分オプションには 0 以外で次の条件を満たす整数を指定します。

- 増分≦順序数生成子データ型の最大値
- 増分≧順序数生成子データ型の最小値

このオプションを省略した場合，1 が仮定されます。

(h) 順序数生成子最大値オプション：：＝ {MAXVALUE 最大値｜NO MAXVALUE}

順序数生成子が生成する値（順序番号）の最大値を指定します。

順序数生成子最大値オプションに最大値を設定する場合，次の条件を満たす必要があります。

- 順序数生成子最大値≦順序数生成子データ型の最大値
- 順序数生成子最大値＞順序数生成子最小値
- 順序数生成子最大値≧順序数生成子開始値

このオプションを省略した場合，又は NO MAXVALUE を指定した場合は，その順序数生成子データ型の最大値が仮定されます。

(i) 順序数生成子最小値オプション：：＝ {MINVALUE 最小値｜NO MINVALUE}

順序番号の最小値を指定します。

順序数生成子最小値オプションに最小値を設定する場合，次の条件を満たす必要があります。

- 順序数生成子最小値≧順序数生成子データ型の最小値
- 順序数生成子最小値＜順序数生成子最大値
- 順序数生成子最小値≦順序数生成子開始値

このオプションを省略した場合，又は NO MINVALUE を指定した場合は，その順序数生成子データ型の最小値が仮定されます。

(j) 順序数生成子循環オプション：：＝ {CYCLE ｜ NO CYCLE}

順序数生成子が生成できる値が最大値又は最小値を超えたときに，順序番号の循環をするかどうかを指定します。順序数生成子循環オプションに CYCLE を指定した場合の動作を次に示します。

- 昇順順序数生成子の場合
  順序数生成子最大値を超えたときに，順序数生成子は順序数生成子最小値を順序番号として生成します。

- 降順順序数生成子の場合
  順序数生成子最小値を超えたときに，順序数生成子は順序数生成子最大値を順序番号として生成します。

なお，順序数生成子が循環したときは重複した番号が発生します。

このオプションを省略した場合，又は NO CYCLE を指定した場合の動作を次に示します。

- 昇順順序数生成子の場合
  順序数生成子最大値を超えるときにエラーとなります。
- 降順順序数生成子の場合
  順序数生成子最小値を超えるときにエラーとなります。

(k) 順序数生成子ログ出力間隔オプション：：＝ {LOG INTERVAL 出力間隔}

順序数生成子のログを出力する間隔を指定します。

出力間隔を大きく指定することで，ログの出力回数を削減し処理性能が向上します。ただし，システム障害が発生し，出力間隔を 2 以上に指定した場合，最大で出力間隔の数だけ順序番号に欠番が発生する可能性があります。出力間隔に 1 を指定した場合，又はこのオプションを省略した場合は欠番が発生しません。

出力間隔は，1～2^{31}-1 の範囲で，かつ次に示す条件を満たす必要があります。

出力間隔≦（｜（順序数生成子最大値－順序数生成子最小値）÷増分｜）の絶対値

このオプションを省略した場合は 1 が仮定されます。

順序数生成子ログ出力間隔の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. 順序数生成子は，一つの RD エリアに表と合わせて最大 500 個定義できます。
2. 順序番号の生成規則を次に示します。
   - 指定した順序数生成子が 1 回も使用されていない状態にある場合，順序数生成子の開始値を返却し，現在値に開始値を設定します。
   - 指定した順序数生成子が 1 回以上使用されている状態の場合，現在値の次の値を返却し，現在値に増分を加算します。

(6) 留意事項

CREATE SEQUENCE は，OLTP 下の X/Open に従った UAP からは指定できません。

(7) 使用例

順序数生成子（SEQ1）を次に示す条件で定義します。

- RD エリア"RDA1"に格納
- INTEGER 型
- 開始値 1
- 増分 1
- 最大値 999
- 最小値 1
- 循環なし
- ログ出力間隔 20

```
CREATE SEQUENCE USER1.SEQ1
AS INTEGER
START WITH 1
INCREMENT BY 1
MAXVALUE 999
MINVALUE 1
NO CYCLE
LOG INTERVAL 20
IN RDA1
```

3.17　CREATE TABLE（表定義）

3.17.1　CREATE TABLE の形式と規則

(1)　機能

表を定義します。

(2)　使用権限

スキーマを所有するユーザ

RD エリア利用権限のある私用 RD エリア，又は公用 RD エリアに表を定義できます。

(3)　形式

```
CREATE 〔〔SHARE〕 FIX〕 〔GLOBAL TEMPORARY〕 TABLE
     〔認可識別子.〕表識別子
     (表要素〔, 表要素〕…)
     〔 {IN {表格納用RDエリア名
            | (表格納用RDエリア名)
            | ( 〔 (表格納用RDエリア名) 格納条件, 〕…
                (表格納用RDエリア名) 〔格納条件〕 ) }
        |PARTITIONED BY 列名
           IN  ( 〔 (表格納用RDエリア名) 境界値, 〕…
                 (表格納用RDエリア名) 境界値,
                   (表格納用RDエリア名) )
        |PARTITIONED BY MULTIDIM
            (第1次元列名 第1次元境界値リスト
             ,{第2次元列名 第2次元境界値リスト
                 | 〔FIX〕 HASH 〔ハッシュ関数名〕
                    BY 第2次元列名 〔, 第2次元列名〕…})
           IN マトリクス分割表格納用RDエリア指定
        | 〔FIX〕 HASH 〔ハッシュ関数名〕 BY 列名 〔, 列名〕…
           IN (表格納用RDエリア名, 表格納用RDエリア名, …) } 〕
     〔表オプション〕…
     { 〔表制約定義〕… | 〔ON COMMIT {DELETE | PRESERVE} ROWS〕 }
     〔WITH PROGRAM〕
```

≪各項目の詳細≫

```
表要素：：＝ {列定義 | 表制約定義}
  列定義：：＝列名 データ型〔ARRAY ［最大要素数］〕
              〔 {列データ抑制指定 | 〔列回復制約〕
                  {IN {LOB列格納用RDエリア名
                         | (LOB列格納用RDエリア名)
                         | ( (LOB列格納用RDエリア名)
                            〔, (LOB列格納用RDエリア名) 〕…)
                        | マトリクス分割LOB列格納用RDエリア指定}
                   | 抽象データ型定義内LOB格納用RDエリア指定} } 〕
```

```
            〔プラグイン指定〕
            〔圧縮指定〕
            〔DEFAULT句〕
            〔列制約〕…
            〔予備列定義〕
            〔更新可能列属性〕
  列データ抑制指定：：＝SUPPRESS
  列回復制約：：＝RECOVERY 〔 {ALL｜PARTIAL｜NO} 〕
  抽象データ型定義内LOB格納用RDエリア指定：：＝
      ALLOCATE（属性名〔..属性名〕…
          IN  {LOB属性格納用RDエリア名
                ｜（LOB属性格納用RDエリア名）
                ｜（（LOB属性格納用RDエリア名）
                    〔，（LOB属性格納用RDエリア名）〕）
                ｜マトリクス分割LOB属性格納用RDエリア指定}
           〔，属性名〔..属性名〕…
                IN  {LOB属性格納用RDエリア名
                      ｜（LOB属性格納用RDエリア名）
                      ｜（（LOB属性格納用RDエリア名）
                          〔，（LOB属性格納用RDエリア名）〕）
                      ｜マトリクス分割LOB属性格納用RDエリア指定}〕…）
  プラグイン指定：：＝PLUGIN プラグインオプション
  圧縮指定：：＝ COMPRESSED 〔BY 圧縮分割サイズ〕
  圧縮分割サイズ：：＝ n〔 {K｜M｜G} 〕
  DEFAULT句：：＝DEFAULT 〔既定値〕
  既定値：：＝ {定数｜USER｜CURRENT_USER｜CURRENT_DATE｜CURRENT DATE
                ｜CURRENT_TIME｜CURRENT TIME
                ｜CURRENT_TIMESTAMP〔（小数秒精度）〕 〔USING BES〕
                ｜CURRENT TIMESTAMP〔（小数秒精度）〕 〔USING BES〕
                ｜NULL}
  列制約：：＝ {非ナル値制約指定
                ｜単一列一意性制約定義
                  〔インデクスオプション〔インデクスオプション〕〕
                ｜ { 単一列検査制約定義〔制約名定義〕
                      ｜〔制約名定義〕単一列検査制約定義 } ※3
                ｜ { 単一列参照制約定義〔制約名定義〕
                      ｜〔制約名定義〕単一列参照制約定義} } ※3
    予備列定義：：＝FOR RESERVED
  更新可能列属性：：＝UPDATE 〔ONLY FROM NULL〕
    非ナル値制約指定：：＝
          { 〔NULL
            ｜NOT NULL〔WITH DEFAULT〔SYSTEM GENERATED〕〕〕 ※2
            ｜ 〔 〔NOT NULL〕 WITH DEFAULT〔SYSTEM GENERATED〕〕 } ※1
    単一列一意性制約定義：：＝
            { 〔 {UNIQUE｜PRIMARY} 〕 CLUSTER KEY 〔 {ASC｜DESC} 〕
                〔IN {インデクス格納用RDエリア名
                      ｜（インデクス格納用RDエリア名）
                      ｜（（インデクス格納用RDエリア名）
                          〔，（インデクス格納用RDエリア名）〕…）} 〕
              ｜PRIMARY KEY 〔 {ASC｜DESC} 〕
                〔IN {インデクス格納用RDエリア名
                      ｜（インデクス格納用RDエリア名）
                      ｜（（インデクス格納用RDエリア名）
                          〔，（インデクス格納用RDエリア名）〕…）} 〕} 〕
    インデクスオプション：：＝ {PCTFREE＝未使用領域の比率
                                ｜UNBALANCED SPLIT}
    単一列検査制約定義：：＝CHECK（探索条件）
```

```
      単一列参照制約定義：：＝参照指定
    複数列一意性制約定義：：＝
            { 〔 {UNIQUE｜PRIMARY} 〕 CLUSTER KEY
                (列名 〔 {ASC｜DESC} 〕 〔, 列名 〔 {ASC｜DESC} 〕〕…)
                〔IN {インデクス格納用RDエリア名
                      | (インデクス格納用RDエリア名)
                      | ((インデクス格納用RDエリア名)
                        〔, (インデクス格納用RDエリア名)〕…)
                      |マトリクス分割インデクス格納用RDエリア指定} 〕
              |PRIMARY KEY (列名 〔 {ASC｜DESC} 〕
                〔, 列名 〔 {ASC｜DESC} 〕〕…)
                〔IN {インデクス格納用RDエリア名
                      | (インデクス格納用RDエリア名)
                      | ((インデクス格納用RDエリア名)
                        〔, (インデクス格納用RDエリア名)〕…)
                      |マトリクス分割インデクス格納用RDエリア指定} 〕}
    複数列検査制約定義：：＝CHECK (探索条件)
    複数列参照制約定義：：＝FOREIGN KEY (列名 〔, 列名〕…) 参照指定
格納条件：：＝列名 {＝|<>|^＝|!＝|<|<＝|>|>＝}
              {定数| (定数 〔, 定数〕…)}
第1次元列名：：＝列名
第2次元列名：：＝列名
第1次元境界値リスト：：＝境界値リスト
第2次元境界値リスト：：＝境界値リスト
  境界値リスト：：＝ ((境界値) 〔, (境界値)〕…)
マトリクス分割表格納用RDエリア指定：：＝2次元格納用RDエリア指定
マトリクス分割インデクス格納用RDエリア指定：：＝2次元格納用RDエリア指定
マトリクス分割LOB列格納用RDエリア指定：：＝2次元格納用RDエリア指定
マトリクス分割LOB属性格納用RDエリア指定：：＝2次元格納用RDエリア指定
  2次元格納用RDエリア指定：：＝ (マトリクス分割用RDエリアリスト
                                〔, マトリクス分割用RDエリアリスト〕…)
  マトリクス分割用RDエリアリスト：：＝ (RDエリア名 〔, RDエリア名〕…)
ハッシュ関数名：：＝ {HASH1|HASH2|HASH3|HASH4|HASH5|HASH6|HASHZ
                    |HASH0|HASHA|HASHB|HASHC|HASHD|HASHE|HASHF}
参照指定：：＝REFERENCES 被参照表 〔参照制約動作指定〕
被参照表：：＝表名
参照制約動作指定：：＝ {削除動作 〔更新動作〕 | 更新動作 〔削除動作〕}
削除動作：：＝ON DELETE 参照動作
更新動作：：＝ON UPDATE 参照動作
参照動作：：＝ { CASCADE|RESTRICT}
制約名定義：：＝CONSTRAINT 制約名
表オプション：：＝
       {PCTFREE＝ {未使用領域の比率
                  | (〔未使用領域の比率〕, セグメント内の空きページ比率)}
         | {LOCK ROW|LOCK PAGE}
         |SUPPRESS 〔DEC 〔IMAL〕〕
         |WITHOUT ROLLBACK
         |INDEXLOCK {NONE|PAGE}
         |SEGMENT REUSE { 〔セグメント数 〔 {K|M} 〕〕 |NO}
         |INSERT ONLY 〔WHILE {日間隔データ|ラベル付き間隔} BY 列名〕}
```

注※1

FIX 表の列，クラスタキー又は主キーを構成する列の場合

注※2

上記以外の場合

注※3

制約名定義の指定位置は，システム共通定義 pd_constraint_name オペランドの指定値，又はクライアント環境変数 PDCNSTRNTNAME の指定値によって決まります。詳細は，「表制約定義」を参照してください。

(4) オペランド

(a) 〔SHARE〕FIX〔GLOBAL TEMPORARY〕

〔SHARE〕FIX

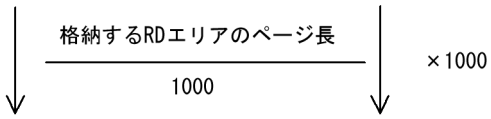
行の長さが固定している表の場合に指定します。表データを共用 RD エリアに格納し，共用表とする場合は SHARE を指定します。

ただし，共用 RD エリアのない HiRDB/シングルサーバでは HiRDB/パラレルサーバと SQL の互換性を保つために SHARE を指定することで共用表を定義できますが，表データは通常のユーザ用 RD エリアに格納します。

行長の変更を伴うデータベースの操作はできませんが，行の格納効率を上げられます。また，行単位インタフェースを用いることで，列数が多い表の場合，アクセス性能を上げられます。

FIX についての規則を次に示します。

1. FIX を指定した場合，次のデータ型は指定できません。
   - VARCHAR
   - NVARCHAR
   - MVARCHAR
   - BLOB
   - BINARY
   - 抽象データ型
2. FIX を指定した場合，繰返し列は指定できません。
3. FIX を指定した場合，行の長さが次の値を超えてはいけません。



   定義する表が一時表の場合は，実体化する際に行の長さが上記の値を超えているかをチェックします。一時表の実体化については，マニュアル「HiRDB システム導入・設計ガイド」の「一時表」を参照してください。
4. FIX を指定して定義した表の各列には，NOT NULL が仮定されます。
5. SHARE と FIX を指定した場合，表データを複数の RD エリアに分割して格納できません。
6. 定義する表が一時表の場合は SHARE を指定できません。

GLOBAL TEMPORARY

定義する実表が一時表の場合に指定します。ただし，表の構成列に次のデータ型は指定できません。

- BLOB 型
- ユーザ定義及びプラグインが提供する抽象データ型

(b) 〔認可識別子.〕表識別子

認可識別子

定義する実表の所有者になるユーザの認可識別子を指定します。

表識別子

定義する実表の名前を指定します。表識別子は，表所有者の表中で同じ識別子は使用できません。

(c) 表要素：：＝ {列定義｜表制約定義}

列定義

表を構成する列自体の定義（列名，データ型など）に指定します。単一列ごとに非ナル値制約，一意性制約，検査制約，参照制約の項目を指定できます。

表制約定義：：＝ {複数列一意性制約定義
〔インデクスオプション〔インデクスオプション〕〕
｜ {複数列検査制約定義〔制約名定義〕
｜ 〔制約名定義〕複数列検査制約定義}
｜ {複数列参照制約定義〔制約名定義〕
｜ 〔制約名定義〕複数列参照制約定義}}

複数の列に一意性制約，検査制約，参照制約を指定します。

制約名定義の指定位置は，システム共通定義 pd_constraint_name オペランドの指定値，又はクライアント環境変数 PDCNSTRNTNAME の指定値によって決まります。制約名定義の指定位置を次の表に示します。

表 3-22 制約名定義の指定位置

クライアント環境変数		システム共通定義		
		pd_constraint_name		
		指定なし	LEADING	TRAILING
PDCNSTRNTNAME	指定なし	前	前	後
	LEADING	前	前	前
	TRAILING	後	後	後

（凡例）

前：制約名定義を制約定義の前に指定します（標準 SQL 仕様）。

後：制約名定義を制約定義の後に指定します（XDM/RD 互換仕様）。

SHARE と FIX を指定している場合，複数列参照制約は指定できません。

(d) {IN～

```
{IN {表格納用RDエリア名
     | (表格納用RDエリア名)
     | ( 〔(表格納用RDエリア名) 格納条件, 〕…
         (表格納用RDエリア名) 〔格納条件〕 ) }
 |PARTITIONED BY 列名
    IN ( 〔(表格納用RDエリア名) 境界値, 〕…
       (表格納用RDエリア名) 境界値, (表格納用RDエリア名) )
 |PARTITIONED BY MULTIDIM
    (第1次元列名 第1次元境界値リスト
     , {第2次元列名 第2次元境界値リスト
          | 〔FIX〕 HASH 〔ハッシュ関数名〕
            BY 第2次元列名 〔, 第2次元列名〕…} )
    IN マトリクス分割表格納用RDエリア指定
 | 〔FIX〕 HASH 〔ハッシュ関数名〕 BY 列名 〔, 列名〕…
    IN (表格納用RDエリア名, 表格納用RDエリア名, …) }
```

IN

表の行を格納する RD エリアを指定します。定義する表が一時表の場合は指定できません。

表格納用 RD エリア名

表の行を格納する一時表用 RD エリア以外のユーザ用 RD エリアの名前を指定します。

ただし，HiRDB/パラレルサーバでは次の制限があります。SHARE を指定している場合，共用 RD エリアの名前を指定します。また，SHARE を指定していない場合，共用 RD エリアの名前を指定できません。

RD エリア名を省略した場合，表を格納する RD エリアは次の表に示すとおりに決定されます。また，HiRDB/パラレルサーバでは SHARE を指定している場合，共用 RD エリアが格納先の候補となります。SHARE を指定していない場合，共用 RD エリアは格納先の候補となりません。

なお，リバランス表を格納しているユーザ用 RD エリアは指定できません。

表 3-23　表格納 RD エリアの指定を省略したときの表格納 RD エリアの決定方法

プライマリキー又はクラスタキーの指定	インデクス格納 RD エリアの指定※	表格納 RD エリアの決定方法
なし	－	次に示す優先順位で表格納 RD エリアを決定します。 1. インナレプリカ機能を適用しない RD エリアのうち，定義されている表と順序数生成子の合計が一番少ない公用 RD エリア 2. 1 の条件を満たす RD エリアが複数ある場合は，HiRDB が最初に見付けた RD エリア 3. 1，2 で決定できなかった場合は，インナレプリカ機能を適用する RD エリア（オリジナル RD エリア）のうち，定義されている表と順序数生成子の合計が一番少ない公用 RD エリア

プライマリキー又はクラスタキーの指定	インデクス格納RDエリアの指定※	表格納RDエリアの決定方法
		4. 3の条件を満たすRDエリアが複数ある場合は，HiRDBが最初に見付けたRDエリア
あり	なし	次に示す優先順位で表格納RDエリアを決定します。 1. インナレプリカ機能を適用しないRDエリアのうち，定義されている表と順序数生成子の合計が一番少ない公用RDエリア 2. 1の条件を満たすRDエリアが複数ある場合は，インデクスの定義数が一番少ない公用RDエリア 3. 1，2の条件を満たすRDエリアが複数ある場合は，HiRDBが最初に見付けたRDエリア 4. 1，2，3で決定できなかった場合は，インナレプリカ機能を適用するRDエリア（オリジナルRDエリア）のうち，定義されている表と順序数生成子の合計が一番少ない公用RDエリア 5. 4の条件を満たすRDエリアが複数ある場合は，インデクスの定義数が一番少ない公用RDエリア 6. 4，5の条件を満たすRDエリアが複数ある場合は，HiRDBが最初に見付けたRDエリア
	インナレプリカ機能を適用しないRDエリア	1. インナレプリカ機能を適用しないRDエリアのうち，定義されている表と順序数生成子の合計が一番少ない公用RDエリア 2. 1の条件を満たすRDエリアが複数ある場合は，HiRDBが最初に見付けたRDエリア
	インナレプリカ機能を適用するRDエリア（オリジナルRDエリア）	1. インナレプリカ機能を適用するRDエリアのうち，オリジナルRDエリアで，かつレプリカRDエリア数の等しい公用RDエリア 2. 1の条件を満たすRDエリアが複数ある場合は，HiRDBが最初に見付けたRDエリア

（凡例）

－：該当しません。

注※

HiRDB/パラレルサーバで，かつインデクス格納用RDエリアの指定がある場合，インデクス格納用RDエリアと同じサーバ内のRDエリアが表格納RDエリアの選定対象になります。

格納条件

表を複数のRDエリアに分割（表の横分割）して格納するための条件を指定します。格納条件を指定した場合，単一列分割になります。

定数の複数指定は格納条件になる列が次に示す形式の場合に有効です。

- 列の値が範囲指定によってグループ分けできないとき（例えば，店番号や部課コードなど）
- 列の値が文字列などの不連続な値で構成されているとき

格納条件についての規則を次に示します。

1. 格納条件を複数指定する場合，格納条件中に同じ列名を指定してください。
2. 格納条件を複数指定した場合に行が格納される RD エリアは，格納条件の指定順に HiRDB が条件を評価して，最初に真になった格納条件に対応した RD エリアです。どちらかの条件でも真にならなかった行は，格納条件を指定しない RD エリアに格納されます。格納条件を指定しない RD エリアが定義されていない場合は格納できません。
3. HiRDB が格納条件を評価した結果，行が格納されない RD エリアがある場合，表の定義はできません。
4. データ挿入時，挿入するデータに該当する RD エリアがない場合はデータを挿入できません。
5. データ更新時，格納条件に指定した列の値は更新できません。
6. 一つの格納条件は一個の RD エリア名と対応します。RD エリアは最大 4,096 個指定できます。ただし，同じ RD エリア名は指定できません。
7. 列名に予備列は指定できません。

PARTITIONED BY 列名

永続実表を境界値によって分割し，複数の RD エリアに格納する場合に指定します。分割できる RD エリアは，重複を含んで最大 4096 個です。

なお，定義する表が一時表の場合は指定できません。

列名

境界値を指定する列の名称を指定します。指定できる列のデータ型は，格納条件で比較演算できる列のデータ型と同じです。PARTITIONED BY 列名を指定した場合，単一列分割になります。

BY 列名についての規則を次に示します。

1. データ更新時，列名で指定した列の値は更新できません。
2. 列名で指定した列は NOT NULL（非ナル値制約，FIX 指定，クラスタキー，又は主キー）にしてください。
3. クラスタキーを指定した列がある場合，それ以外の列に対して境界値は指定できません。
4. クラスタキーが複数列の場合，先頭の列以外の列に対して境界値は指定できません。
5. 繰返し列は指定できません。
6. 予備列は指定できません。

境界値

表の行を分割するときの境界値を指定します。境界値には定数を指定してください。

境界値についての規則を次に示します。

1. 定数には，次のものは指定できません。

   ・長さ 0 の文字列定数，各国文字列定数，混在文字列定数，及び 16 進文字列定数

   ・長さ 256 バイト以上の文字列定数，長さ 128 文字以上の各国文字列定数，長さ 256 バイト以上の混在文字列定数，及び長さ 256 バイト以上の 16 進文字列定数（16 進文字列の文字数 512 文字以上）

2. 境界値は昇順に指定してください。同じ値は指定できません。
3. 最後に指定する境界値には，最大値は指定しないでください。
4. 表格納用 RD エリア名と境界値は交互に指定し，表格納用 RD エリア名で始まり表格納用 RD エリア名で終わるようにしてください。
5. 指定できる表格納用 RD エリア名の延べ数は，最大 4096 個です。
6. 表格納用 RD エリア名は，同一のものを複数指定できます。ただし，同じ表格納用 RD エリア名は続けて指定しないでください。
7. 最初に境界値を指定する RD エリアには，指定する境界値以下の値の行が格納されます。2 番目以降（最後は除く）に指定する RD エリアには，その前に指定した境界値より大きく，かつその後に指定する境界値以下の値の行が格納されます。最後に指定する RD エリアには，その前に指定した境界値より大きい値の行が格納されます。

PARTITIONED BY MULTIDIM

表データをある一つの列（第 1 次元分割列）で分割し，さらに各境界値データを別の列（第 2 次元分割列）で分割する場合に指定します。このような指定による分割方法をマトリクス分割といい，分割した表を，**マトリクス分割表**といいます。

マトリクス分割表を定義する場合，HiRDB Advanced High Availability が必要です。

なお，定義する表が一時表の場合は指定できません。

第 1 次元列名：：＝列名

第 1 次元分割列名を指定します。

第 1 次元分割列名についての規則を次に示します。

1. 指定した列は，NOT NULL にしてください。NOT NULL にする方法を次に示します。
   ・FIX 表として表を定義する。
   ・列定義で NOT NULL を指定する。
   ・クラスタキー，又は主キーを定義する。
2. 指定した列の値は更新できません。
3. 列名に繰返し列は指定できません。
4. 列名に指定する列のデータ型は，格納条件の比較演算できる列のデータ型を参照してください。
5. 列名に予備列は指定できません。

第 1 次元境界値リスト：：＝境界値リスト

境界値リスト：：＝（(境界値)〔,（境界値)〕…)

第 1 次元列名で指定した列の境界値リストを指定します。

境界値には，表の行を分割するときの境界値を指定します。境界値についての規則を次に示します。

1. 境界値には定数を指定してください。
2. 境界値は昇順に指定してください。
3. 境界値には，次の定数はできません。

・長さ 0 の文字列定数，各国文字列定数，混在文字列定数，及び 16 進文字列定数
・長さ 256 バイト以上の文字列定数，及び混在文字列定数
・長さ 128 文字以上の各国文字列定数
・長さ 256 バイト以上の 16 進文字列定数（16 進文字列の文字数 512 文字以上）

また，最後に指定する境界値に最大値は指定できません。

第 2 次元列名：：＝列名

第 2 次元分割列名を指定します。

規則については，第 1 次元列名の説明を参照してください。

第 2 次元境界値リスト：：＝境界値リスト

境界値リスト：：＝（(境界値)〔，(境界値)〕…）

第 2 次元列名で指定した列の境界値リストを指定します。

境界値の規則については，第 1 次元境界値リストの境界値の説明を参照してください。

〔FIX〕 HASH 〔ハッシュ関数名〕

BY 第 2 次元列名 〔，第 2 次元列名〕…

第 2 次元列名：：＝列名

使用するハッシュ関数名，及び第 2 次元分割列名を指定します。

指定方法の詳細，及び規則については，〔FIX〕HASH の項目を参照してください。

マトリクス分割表格納用 RD エリア指定：：＝ 2 次元格納用 RD エリア指定

2 次元格納用 RD エリア指定：：＝

（マトリクス分割用 RD エリアリスト

〔，マトリクス分割用 RD エリアリスト〕…）

マトリクス分割用 RD エリアリスト：：＝

（RD エリア名〔，RD エリア名〕…）

マトリクス分割表，及びマトリクス分割表へのクラスタキー，主キー，BLOB 列，BLOB 属性の抽象データ型を定義する場合，それを格納する RD エリア名を指定します。

マトリクス分割表格納用 RD エリア指定についての規則を次に示します。

1. RD エリア名の指定の規則については，各オペランドの RD エリアの説明を参照してください。
2. 一つのマトリクス分割用 RD エリアリストに指定する RD エリアの数は，第 2 次元境界値リストに指定した境界値の数＋ 1 です。また，第 2 次元分割列にハッシュ関数を使用する場合は，ユーザ任意の数です。
3. 指定できるマトリクス分割用 RD エリアリストの数は，第 1 次元境界値リストに指定した境界値の数＋ 1 です。
4. マトリクス分割表へのクラスタキー，主キー，BLOB 列，及び BLOB 属性を持つ抽象データ型を定義する場合，マトリクス分割表格納用 RD エリア指定に対応するように RD エリアを指定してください。
5. クラスタキーは単一列に指定できません。

6. 指定できる RD エリアの数は重複を含み最大 4096 個です。

7. クラスタキーが 2 列以上の場合，分割に指定した列を先頭から同順で含む必要があります。

〔FIX〕HASH

永続実表をハッシュ関数によって分割し，複数の RD エリアに格納する場合に指定します。同じ表格納用 RD エリア名を連続して指定できます。また，マトリクス分割表の第 2 次元分割列にハッシュ関数を指定する場合にも指定します。

表の分割方式をフレキシブルハッシュ分割にする場合は，HASH を指定します。FIX ハッシュ分割にする場合は，FIX HASH を指定します。

なお，定義する表が一時表の場合は指定できません。

〔ハッシュ関数名〕

表を分割するためのハッシュ関数を指定します。

ハッシュ関数名を省略した場合，分割方法によって次のハッシュ関数が仮定されます。

- 表をハッシュ関数によって分割する場合は，HASH1 を仮定する
- マトリクス分割表の第 2 次元分割列にハッシュ関数を指定する場合は，HASH6 を仮定する

BY 列名 〔，列名〕…

ハッシュ関数の対象となる列の名称を指定します。指定できる列のデータ型は，格納条件で比較演算できる列のデータ型と同じです。

列名を一つ指定すると単一列分割，列名を複数指定すると複数列分割になります。

単一列分割する表の場合：

- クラスタキーを指定した列がある場合，それ以外の列を列名に指定できません。
- クラスタキーが複数列の場合，先頭の列以外の列を列名に指定できません。

複数列分割する表の場合：

- クラスタキーを単一列に指定できません。
- クラスタキーが複数列の場合，先頭から分割に指定した列を同順にすべて含む必要があります。

マトリクス分割表の第 2 次元分割列を単一列分割する場合：

- クラスタキーを単一列に指定できません。
- クラスタキーが複数列の場合，先頭から 2 番目の列以外の列を列名に指定できません。

マトリクス分割表の第 2 次元分割列を複数列分割する場合：

- クラスタキーを単一列に指定できません。
- クラスタキーが複数列の場合，先頭から 2 番目以降の列を同順に指定する必要があります（クラスタキー構成列の先頭から 2 番目の列以降の列をすべて含む必要はありません）。

主キーを指定した列がある場合，表の分割方法によって，主キーの定義可否が決まります。表を横分割する場合の主キーの定義可否（UNIQUE 指定可否）については，「表を横分割する場合の UNIQUE 指定可否」を参照してください。

BY 列名についての規則を次に示します。

1. 複数列分割に指定できる列の最大数は 16 個です。ただし，マトリクス分割表の第 2 次元分割列にハッシュ関数を指定する場合に指定できる列の最大数は 15 個です。
2. 複数列分割する場合は，同じ列名を重複して指定できません。
3. 複数列分割する場合は，列名には相互に依存しない値を持つ列を組み合わせて指定します。
4. 列名で指定した列は NOT NULL（非ナル値制約，FIX 指定，クラスタキー，又は主キー）にしてください。
5. フレキシブルハッシュ分割の表，又はマトリクス分割表の第 2 次元分割列にフレキシブルハッシュ分割を指定する場合には，UNIQUE を指定したクラスタキー，主キー，及び UNIQUE を指定したインデクスは定義できません。
6. FIX ハッシュ分割では，データ更新時，列名で指定した列の値は更新できません。
7. 繰返し列は指定できません。
8. 列名をマトリクス分割表の第 2 次元列名に指定する場合は，第 1 次元列名で指定している列名は指定できません。
9. 予備列は指定できません。

(e) 表オプション

```
::= {PCTFREE= {未使用領域の比率
                  | (〔未使用領域の比率〕, セグメント内の空きページ比率)}
      | {LOCK ROW | LOCK PAGE} | SUPPRESS〔DEC〔IMAL〕〕
      | WITHOUT ROLLBACK
      | INDEXLOCK {NONE | PAGE}
      | SEGMENT REUSE
              {〔セグメント数〔 {K | M} 〕〕 | NO}
      | INSERT ONLY 〔WHILE {日間隔データ | ラベル付き間隔} BY 列名〕}
```

PCTFREE = {未使用領域の比率 | (〔未使用領域の比率〕, セグメント内の空きページ比率)}

同じ表のオプションを繰り返し指定できません。

PCTFREE の指定は，括弧を付けた場合と括弧を付けない場合で指定内容が異なるので注意してください。

PCTFREE = 30 は，未使用領域の比率が 30%を示します。

PCTFREE =（, 30）は，セグメント内の空きページ比率が 30%を示します。

未使用領域の比率

表の初期作成時にデータベース内に設定する未使用領域の比率を指定します。未使用領域の比率は，0～99（%）を指定できます。未使用領域の比率を省略すると，30（%）が仮定されます。仮定値は，システム共通定義の pd_ddl_tbl_pctfree オペランドで変更できます。pd_ddl_tbl_pctfree オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

未使用領域の比率は，データベース作成ユティリティ，及びデータベース再編成ユティリティ実行時に適用されます。INSERT 文や UPDATE 文など，ほかの追加，及び更新では PCTFREE =（0, 0）が仮定されます。

未使用領域の比率についての規則を次に示します。

1. クラスタキーを定義する場合，次に示す理由のため PCTFREE を指定してください。

   ・データの初期作成，又は再編成後の挿入データをクラスタリング可能にするため，表中に空きを作成しておく。

   ・可変長データを含む表に対してデータの初期作成，又は再編成後に行長が長くなるような更新をしたとき，できるだけ近くに格納できるように表中に空きを作成しておく。

2. クラスタキーのない FIX 表に対しては，PCTFREE ＝（0，0）を指定してください。PCTFREE を省略した場合，PCTFREE ＝（0，0）を仮定します。仮定値は，システム共通定義の pd_ddl_tbl_fix_pctfree オペランドで変更できます。pd_ddl_tbl_fix_pctfree オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

3. クラスタキーのない非 FIX 表に対して，PCTFREE を指定すると，ページ内のレコード長が更新のために長くなった場合，該当ページの未使用領域が使用されます。

4. 表の作成後，クラスタキーを定義する表に対して行の追加が頻繁に発生する場合，又は可変長データを含む表に対して，行長の長くなる更新が頻繁に発生する場合，未使用領域の比率を高く指定してください。

セグメント内の空きページ比率

表の作成時にセグメント内に設定する空きページ（未使用ページ）の比率を指定します。

セグメント内の空きページ比率は，0～50（%）を指定できます。

セグメント内の空きページ比率を省略すると，10（%）が仮定されます。

仮定値は，システム共通定義の pd_ddl_tbl_pctfree オペランドで変更できます。pd_ddl_tbl_pctfree オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

セグメント内の空きページ数を求める計算式を次に示します。

$$\left\downarrow \frac{\text{セグメント内ページ数} \times \text{セグメント内空きページ比率}}{100} \right\downarrow$$

ここで指定したセグメント内の空きページ比率は，データベース作成ユティリティ，又はデータベース再編成ユティリティ実行時に適用されます。

クラスタキーを定義する表に対して行の追加が頻繁に発生する場合，及び行長の長くなる更新が頻繁に発生する場合，その増えた分のデータがページ内の未使用領域に納まらないときに，値を指定してください。

{LOCK ROW | LOCK PAGE}

検索時及び更新時の，最小の排他資源単位を指定します。行単位の場合は LOCK ROW，ページ単位の場合は LOCK PAGE を指定します。省略した場合，LOCK ROW を仮定します。

なお，定義する表が一時表の場合は，指定しても無視されます。

LOCK ROW

最小の排他資源単位を行にします。

LOCK PAGE

最小の排他資源単位をページにします。

SUPPRESS〔DEC〔IMAL〕〕

非 FIX 表で，表内のデータの一部を省略して格納するときに指定します。

SUPPRESS オプションの指定は，列に格納するデータの有効けた数が定義長より短い表の場合に，データベース容量の削減に有効です。ただし，DECIMAL データの場合，有効けた数は先頭の 0 部分を除いたけた数になります。

DEC〔IMAL〕

表内の DECIMAL データの先頭 0 を省略して格納するときに指定します。

DECIMAL についての規則を次に示します。

1. SUPPRESS を指定して DECIMAL を省略しても，DECIMAL が仮定されます。
2. 格納する DECIMAL データの有効けた数が定義長と等しい場合，データは「定義長+1」の長さで格納されます。このため，SUPPRESS オプションを省略したときよりデータの長さが長くなるので注意してください。
3. DECIMAL データの定義精度（全けた数）が 1 の場合も，格納されるデータの長さが，SUPPRESS オプションを省略したときより長くなるので注意してください。
4. 抽象データ型の属性中の DECIMAL は，この機能の対象外です。

WITHOUT ROLLBACK

トランザクションのコミットを待たないで，表への更新（追加，削除も含む）処理が完了した時点で，その行の排他を解除する表を定義したい場合に指定します。

WITHOUT ROLLBACK についての規則を次に示します。

1. 該当する表の行の更新処理（追加，削除も含む）の行排他，及びロールバック可否を次の表に示します。

表 3-24　WITHOUT ROLLBACK 指定時の行の更新処理（追加，削除を含む）の行排他，及びロールバック可否

操作対象		システム定義 pd_idx_without_rollback	表に対する操作内容			
			行の挿入	行中の列の更新		行の削除
				更新値が更新前の値と同じ	更新値が更新前の値と異なる	
インデクスが定義された WITHOUT ROLLBACK 指定の表	更新対象列がインデクス構成列である	Y	○	○	○	○
		N	×	○	実行できません。	×
	更新対象列がインデクス構成列ではない	−	該当しません。	○	○	該当しません。

操作対象	システム定義 pd_idx_without_rollback	表に対する操作内容			
		行の挿入	行中の列の更新		行の削除
			更新値が更新前の値と同じ	更新値が更新前の値と異なる	
インデクスが定義されていない WITHOUT ROLLBACK 指定の表	－	○	○	○	○

（凡例）

○：行排他は行の更新処理完了を契機に解除されます。処理完了後は更新内容をロールバックできません。

×：行排他はトランザクション終了時に解除されます。処理完了後もトランザクション未終了時はロールバックできます。

－：定義によりません。

2. データベース作成ユティリティ及びデータベース再編成ユティリティを実行する場合は，このオプションの指定は無視されます。
3. このオプションを指定する場合，FIX 表にする必要があります。ただし，SHARE を指定している場合，このオプションは指定できません。
4. このオプションを指定する場合，BLOB 列の定義はできません。また，LOCK PAGE も指定できません。
5. 定義する表が一時表の場合，このオプションは指定できません。

INDEXLOCK {NONE | PAGE}

XDM/RD との互換のためのオプションであり，指定しても無視されます。

インデクスキー値無排他は，システム定義の pd_indexlock_mode オペランドで指定します。

pd_indexlock_mode オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

SEGMENT REUSE {[セグメント数[{K | M}]] | NO}

定義する表に，空き領域の再利用機能を使用する場合に指定します。

この指定を省略した場合，HiRDB は「セグメント数を指定していない SEGMENT REUSE」が指定されたものとして表を定義します。ただし，リバランス表，一時表の場合は，空き領域の再利用機能を適用しません。

空き領域の再利用機能については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

定義する表が一時表の場合，このオプションは指定できません。

セグメント数[{K | M}]

空き領域の再利用機能を使用し，かつ該当する表のセグメント数に上限を設定する場合，その上限のセグメント数を指定します。セグメント数は，1～268,435,440 の符号なし整数で指定できます。また，単位として，K（キロ），又は M（メガ）を使用できますが，268,435,440 を超える値は指定できません。

行の挿入，削除を頻繁に行う表に対して，行の挿入性能の向上，及び指定したセグメント内での格納効率の向上が期待できます。

セグメント数の指定なし

空き領域の再利用機能を使用し，かつ該当する表のセグメント数に上限を設定しない場合，セグメント数を省略します。

行の挿入，削除を頻繁に行う表で，かつ RD エリアに格納される表が該当する表だけの場合に適用してください。行の挿入性能の向上，及び該当する表を格納する RD エリア内での，空き領域の格納効率の向上が期待できます。

NO

空き領域の再利用機能を使用しない場合に指定します。

行の挿入，削除を頻繁に行わない表に対しては NO を指定してください。

SEGMENT REUSE についての規則を次に示します。

1. 空き領域の再利用機能は，LOB 列，LOB 属性の抽象データ型列，及びインデクスに対しては無効となります。
2. 空き領域の再利用機能は，リバランス表に対しては指定できません。

INSERT ONLY 〔WHILE {日間隔データ｜ラベル付き間隔} BY 列名〕

表を改竄防止表とする場合に指定します。改竄防止表については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

改竄防止表とした場合，その表の値は更新できません。ただし，更新可能列は更新できます。

改竄防止表には，すべての列を更新可能列とすることはできません。

予備列を定義した表を，改竄防止表とすることはできません。

改竄防止表には，行の削除を禁止する期間（行削除禁止期間）を指定できます。行削除禁止期間を指定する場合，WHILE で期間を指定し，列名には挿入履歴保持列（DATE 型の列で，かつ SYSTEM GENERATED の列）を指定してください。削除禁止期間を指定しない場合，永久的に行の削除はできません。

定義する表が一時表の場合，このオプションは指定できません。

日間隔データ

行削除禁止期間を，日間隔データの 10 進数表現で指定します。日間隔データの 10 進数表現については，「日間隔データの 10 進数表現」を参照してください。

なお，日間隔データは正の値だけ指定できます。

ラベル付き間隔

行削除禁止期間を，ラベル付き間隔で指定します。ラベル付き間隔については，「日付演算」を参照してください。

ラベル付き間隔の値式には，括弧で囲まれていない正の整数定数だけ指定できます。

列名

DATE 型で，かつ SYSTEM GENERATED の列を指定してください。

行削除禁止期間には行を挿入した日を含み，行削除禁止期間の計算は，「日付演算」の「日付データと日間隔データとを加減算する場合の規則」に従います。削除禁止最終日付と削除可能日付は，それぞれ次の演算結果となります。

- 削除禁止最終日付＝行挿入日付＋削除禁止期間－1 日
- 削除可能日付＝行挿入日付＋削除禁止期間

行挿入日付と削除禁止期間の指定値での，削除禁止最終日付と削除可能日付の関係を次の表に示します。

表 3-25　削除禁止最終日付と削除可能日付の関係

行挿入日付	削除禁止期間の指定値	削除禁止最終日付	削除可能日付
2002-03-01	1 か年※1	2003-02-28	2003-03-01
1995-03-01	1 か年※1	1996-02-29	1996-03-01
2002-02-28	1 か月※2	2002-03-27	2002-03-28
2002-05-01	1 か日※3	2002-05-01	2002-05-02

注※1

実際の指定形式は，日間隔データの場合は「00010000.」，ラベル付き間隔の場合は「1 YEAR」となります。

注※2

実際の指定形式は，日間隔データの場合は「00000100.」，ラベル付き間隔の場合は「1 MONTH」となります。

注※3

実際の指定形式は，日間隔データの場合は「00000001.」，ラベル付き間隔の場合は「1 DAY」となります。

(f)　{〔表制約定義〕…｜〔ON COMMIT　{DELETE ｜ PRESERVE}　ROWS〕}

〔表制約定義〕…

標準 SQL では，表制約定義は表要素として指定します。したがって，表制約定義は表要素の並び中に表要素として指定することをお勧めします。表制約定義の詳細は，「表要素：：＝ {列定義｜表制約定義}」の「表制約定義」の説明を参照してください。

ON COMMIT　{DELETE ｜ PRESERVE}　ROWS

定義する実表が一時表の場合に指定でき，一時表のデータが存在する期間を指定します。

DELETE

トランザクション固有一時表を作成する場合に指定します。

一時表のデータは，トランザクション終了時に削除されます。

PRESERVE

SQL セッション固有一時表を作成する場合に指定します。

一時表のデータは，SQL セッション終了時に削除されます。

(g) 列定義

```
：：＝列名 データ型〔ARRAY ［最大要素数］〕
  〔｛列データ抑制指定｜〔列回復制約〕
    {IN ｛LOB列格納用RDエリア名
          ｜（LOB列格納用RDエリア名）
          ｜（（LOB列格納用RDエリア名）〔，（LOB列格納用RDエリア名）〕…）
          ｜マトリクス分割LOB列格納用RDエリア指定｝
      ｜抽象データ型定義内LOB格納用RDエリア指定｝｝〕
  〔プラグイン指定〕〔圧縮指定〕〔DEFAULT句〕
  〔列制約…〕〔予備列定義〕〔更新可能列属性〕
```

列名

表を構成する列の名前を指定します。列名に同じ名前は指定できません。

データ型

列のデータ型を指定します。データ型については，「データ型」を参照してください。

スーパタイプで BLOB が定義されている抽象データ型，及び BOOLEAN は指定できません。

指定するデータ型が抽象データ型で，認可識別子を省略した場合，省略時に仮定される認可識別子に同一名称の抽象データ型がないときは，認可識別子'MASTER'に同一名称の抽象データ型があれば，その抽象データ型を指定したものとします。

ARRAY ［最大要素数］

繰返し列を定義する場合に指定します。

最大要素数には，2～30,000 の符号なし整数を指定します。

繰返し列は，次のデータ型には指定できません。

- 文字集合を指定した CHAR，VARCHAR
- BLOB
- BINARY
- 抽象データ型

繰返し列は，圧縮列に指定できません。

LOB 列格納用 RD エリア名

BLOB 列のデータを格納するユーザ LOB 用 RD エリアの名前を指定します。

LOB 列格納用 RD エリアについての規則を次に示します。

1. データ型に BLOB を指定した列は，LOB 列格納用 RD エリア名を必ず指定してください。BLOB 以外のデータ型を指定した列には指定できません。
2. 指定する RD エリア名は，表格納用 RD エリアと同じサーバに定義しているユーザ LOB 用 RD エリアを指定してください。

3. 表を分割する場合は，分割する数と同じ数のユーザ LOB 用 RD エリアを指定してください。その場合，同じサーバのユーザ用 RD エリアとユーザ LOB 用 RD エリアが，同じ順番になるように指定してください。例を次に示します。

   （例）

   ```
   CREATE TABLE MOVIE (ID INT NOT NULL,
   IMAGE BLOB IN ((LU01),(LU02)))
   IN ((RU01) ID < 120,(RU02))
   ```

   （凡例）

   LU01，LU02，RU01，及び RU02 は，RD エリア名を示します。

   注

   RU01 と LU01，RU02 と LU02 は，それぞれ同一サーバの RD エリアです。

4. LOB 列格納用 RD エリアは，あらかじめデータベース初期設定ユティリティで作成，又はデータベース構成変更ユティリティで追加されていなければなりません。

5. LOB 列格納用 RD エリア名を複数個指定する場合，同じ RD エリア名は指定できません。境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表では，表格納用 RD エリアと LOB 列格納用 RD エリアが 1 対 1 に対応するように LOB 列格納用 RD エリア名を指定してください。表格納用 RD エリア名を重複指定した場合は，LOB 列格納用 RD エリアも同一位置が重複するように指定してください。

   上記以外の場合は，LOB 列格納用 RD エリアは重複して指定できません。

マトリクス分割 LOB 列格納用 RD エリア指定：：＝ 2 次元格納用 RD エリア指定

マトリクス分割表を定義する場合に指定します。

指定方法については，PARTITIONED BY MULTIDIM の 2 次元格納用 RD エリア指定の説明を参照してください。また，指定する RD エリアについては，LOB 列格納用 RD エリア名の説明を参照してください。

(h) 列データ抑制指定：：＝ SUPPRESS

列ごとのデータ抑制をします。固定長文字形式で後方空白のデータが多い場合にディスク容量を削除できます。

列データ抑制指定についての規則を次に示します。

1. 列データの一番最後の文字が空白の場合，最後の空白から前方にサーチして連続している空白を，抑制してデータベースに格納します。この場合，1 回空白が途切れたら，そこから前方の空白は抑制しません。
2. CHAR，及び MCHAR は，4 文字以上の半角の空白をデータ抑制の対象とします。
3. NCHAR は，3 文字以上の全角の空白をデータ抑制の対象とします。
4. FIX 表には，指定できません。
5. データ型が抽象データ型の場合は指定できません。

6. データ型が CHAR，MCHAR，又は NCHAR 以外のときは，指定できません。

7. 繰返し列には指定できません。

8. この指定をした列は，データ抑制がされなかった場合でも，1 バイトの付加情報が追加されます。

9. 次に示すデータ型に対しては，この指定をしても無効となります。

   - CHAR の場合，CHAR（5）より列データ長が短いとき
   - MCHAR の場合，MCHAR（5）より列データ長が短いとき
   - NCHAR の場合，NCHAR（3）より列データ長が短いとき

   なお，一度スペースが途切れた場合，そこから前方のスペースの抑制はされません。

(i) 列回復制約：：＝ RECOVERY 〔{ALL | PARTIAL | NO}〕

LOB 列格納用 RD エリア，又は抽象データ型定義内 LOB 格納用 RD エリアに対して，データベースの更新ログ取得方式を指定します。省略すると，更新前ログ取得モード(PARTIAL)が仮定されます。仮定値は，システム共通定義の pd_ddl_tbl_recovery オペランドで変更できます。pd_ddl_tbl_recovery オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

LOB 列格納用 RD エリアの場合：

データ型に BLOB を指定した場合，ユーザ LOB 用 RD エリアのデータベースの更新ログ取得方式を指定します。データ型が BLOB 以外の列には指定できません。

抽象データ型定義内 LOB 格納用 RD エリアの場合：

抽象データ型定義内に BLOB 属性がある場合，ユーザ LOB 用 RD エリアのデータベースの更新ログ取得方式を指定します。

ALL

ログ取得モードでユーザ LOB 用 RD エリアを運用するときに指定します。ログ取得モードで運用すると，ロールバック，及びロールフォワードに必要なデータベースの更新ログを取得します。

PARTIAL

更新前ログ取得モードでユーザ LOB 用 RD エリアを運用するときに指定します。更新前ログ取得モードで運用すると，ロールバックに必要なデータベースの更新ログを取得します。

NO

ログレスモードでユーザ LOB 用 RD エリアを運用するときに指定します。ログレスモードで運用すると，データベースの更新ログを取得しません。

指定するデータベースの更新ログ取得方式によって，UAP を実行するときの運用方法や，障害が発生したときのユーザ LOB 用 RD エリアの回復方法が異なります。ログレスモードの運用については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(j) 抽象データ型定義内 LOB 格納用 RD エリア指定

```
: :＝ALLOCATE（属性名〔..属性名〕…
 IN  {LOB属性格納用RDエリア名
     | （LOB属性格納用RDエリア名）
```

```
    | ( (LOB属性格納用RDエリア名) 〔, (LOB属性格納用RDエリア名) 〕 )
    |マトリクス分割LOB属性格納用RDエリア指定}
  〔, 属性名〔..属性名〕…
   IN  {LOB属性格納用RDエリア名
        | (LOB属性格納用RDエリア名)
        | ( (LOB属性格納用RDエリア名) 〔, (LOB属性格納用RDエリア名) 〕 )
        |マトリクス分割LOB属性格納用RDエリア指定} 〕…)
```

属性名〔..属性名〕

抽象データ型を構成する属性名を指定します。抽象データ型の属性が抽象データ型で，その入れ子になっている抽象データ型の属性に LOB 型の属性がある場合は，その LOB 型の属性名を指定します。

次のような場合に属性名を指定してください。

- 抽象データ型の属性のデータ型が LOB 型の場合
- 抽象データ型の属性が抽象データ型で，その入れ子になっている抽象データ型の属性が LOB 型の属性の場合（その LOB 型の属性名を指定します）

LOB 属性格納用 RD エリア名

抽象データ型の任意の階層にある，BLOB 属性のデータを格納するユーザ LOB 用 RD エリアの名前を指定します。

LOB 属性格納用 RD エリアについての規則を次に示します。

1. データ型に BLOB を含む抽象データ型を指定した場合，各 BLOB 属性に対してユーザ LOB 用 RD エリアの名前を必ず指定してください。BLOB 以外のデータ型を指定した属性には指定できません。
2. 指定する RD エリア名は，表格納用 RD エリアと同じサーバに定義しているユーザ LOB 用 RD エリアを指定してください。
3. 表を分割する場合は，分割する数と同じ数のユーザ LOB 用 RD エリアを指定してください。その場合，同じサーバのユーザ用 RD エリアとユーザ LOB 用 RD エリアが，同じ順番になるようにしてください。
4. LOB 属性格納用 RD エリアは，あらかじめデータベース初期設定ユティリティで作成，又はデータベース構成変更ユティリティで追加されていなければなりません。
5. LOB 属性格納用 RD エリア名を複数個指定する場合，同じ RD エリア名は指定できません。境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表では，表格納用 RD エリアと LOB 属性格納用 RD エリアが 1 対 1 に対応するように LOB 属性格納用 RD エリア名を指定してください。表格納用 RD エリア名を重複指定した場合は，LOB 属性格納用 RD エリアも同一位置が重複するように指定してください。

   上記以外の場合は，LOB 属性格納用 RD エリアは重複して指定できません。

マトリクス分割 LOB 属性格納用 RD エリア：：＝ 2 次元格納用 RD エリア指定

マトリクス分割表を定義する場合に指定します。

指定方法については，PARTITIONED BY MULTIDIM の 2 次元格納用 RD エリア指定の説明を参照してください。また，指定する RD エリアについては，LOB 属性格納用 RD エリア名の説明を参照してください。

(k) プラグイン指定：：＝ PLUGIN プラグインオプション

プラグインオプション

データ型プラグインが実装されている抽象データ型として定義された列に対して，データ型プラグインに渡すためのパラメタ情報を文字列定数（最大 255 バイト）で記述します。パラメタ情報には，16 進文字列定数は指定できません。

パラメタ情報の詳細については，各種プラグインマニュアルを参照してください。

(l) 圧縮指定：：＝ COMPRESSED 〔BY 圧縮分割サイズ〕

圧縮指定は，列データを圧縮する場合に指定します。BINARY 型の列に圧縮を指定する場合は，定義長が 256 バイト以上である必要があります。255 バイト以下の場合，エラーになります。

圧縮分割サイズ：：＝ n〔{K ｜ M ｜ G}〕

圧縮処理や伸張処理のオーバヘッドを削減したい場合，又は圧縮効率を向上させたい場合に指定します。ただし，圧縮分割サイズを大きくすると，メモリ所要量が増加します。詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

データを n バイト単位で分割して圧縮します。n は符号なし整数で指定し，単位として K（キロバイト），M（メガバイト）及び G（ギガバイト）が指定できます（指定省略時はバイト単位）。圧縮分割サイズに指定できる値の範囲を次の表に示します。

表 3-26　圧縮分割サイズに指定できる値の範囲

単位	n に指定できる値の範囲		圧縮分割サイズの値
	BINARY 型	抽象データ型（XML 型）	
指定しない（バイト）	32,000≦n≦2,147,483,647	1,048,576≦n≦2,147,483,647	n
K	32≦n≦2,097,152	1024≦n≦2,097,152	n×1,024
M	1≦n≦2,048	1≦n≦2,048	n×1,048,576
G	1≦n≦2	1≦n≦2	n×1,073,741,824

圧縮分割サイズの注意事項を次に示します。

- 圧縮分割サイズに指定できる値の範囲外の値を指定した場合，エラーになります。ただし，実際の最大長の計算結果が 2,147,483,648 の場合は，2,147,483,647 になります。
- 圧縮分割サイズに定義長より大きい値を指定した場合，分割しないで圧縮します。
- 圧縮分割サイズを省略すると，BINARY 型の場合は 32,000 バイトを仮定し，抽象データ型（XML 型）の場合は 1 メガバイトを仮定します。ALTER TABLE 文で列の定義長を大きくしても，圧縮列定義時に仮定した圧縮分割サイズは変わりません。

圧縮指定についての規則を次に示します。

1. 圧縮指定ができる列は，次のデータ型の列だけです。

- BINARY 型
- 抽象データ型（XML 型）※

注※

抽象データ型（XML 型）の列のデータを圧縮するためには，バージョン 09-03 以降の HiRDB XML Extension が必要です。バージョン 09-02 以前の HiRDB XML Extension を使用している場合，抽象データ型（XML 型）の列に圧縮指定はできません。

(m) DEFAULT 句：：＝ DEFAULT 〔既定値〕

データ挿入時に値を省略した場合に，DEFAULT 句で指定した値が挿入されます。

DEFAULT 句についての規則を次に示します。

1. 一つの項目に対して，DEFAULT 句と WITH DEFAULT は同時には指定できません。
2. BLOB 型，抽象データ型，及び定義長が 32,001 バイト以上の BINARY 型の項目には指定できません。
3. 繰返し列には指定できません。
4. 予備列には指定できません。

既定値：：＝ {定数 | USER | CURRENT_USER
| CURRENT_DATE | CURRENT DATE
| CURRENT_TIME | CURRENT TIME
| CURRENT_TIMESTAMP〔(小数秒精度)〕〔USING BES〕
| CURRENT TIMESTAMP〔(小数秒精度)〕〔USING BES〕 | NULL}

既定値は，指定した項目に挿入できる値でなければなりません。

既定値についての規則を次に示します。

1. 既定値を省略した場合，WITH DEFAULT の既定値が仮定されます。
2. 既定値には，挿入時に上位有効けたが無効となるようなデータ型は指定できません。
3. 非ナル値制約ありの列には，既定値として NULL は指定できません。
4. 既定文字集合以外の列に対して DEFAULT 句を指定する場合，次の既定値は指定できません。
   ・CURRENT_DATE(CURRENT DATE)
   ・CURRENT_TIME(CURRENT TIME)
   ・CURRENT_TIMESTAMP(CURRENT TIMESTAMP)
5. 既定値に USER（CURRENT_USER），CURRENT_DATE（CURRENT DATE），CURRENT_TIME（CURRENT TIME），又は CURRENT_TIMESTAMP〔(小数秒精度)〕(CURRENT TIMESTAMP〔(小数秒精度)〕）を指定した場合，次の値が設定されます。

   USER（CURRENT_USER）の場合：

   行を挿入したときの実行ユーザの認可識別子の値が設定されます。

   CURRENT_DATE（CURRENT DATE）の場合：

行を挿入したときの日付が設定されます。ただし，データベース作成ユティリティ（pdload）の場合は，ユティリティ起動時の日付が設定されます。

CURRENT_TIME（CURRENT TIME）の場合：

行を挿入したときの時刻が設定されます。ただし，データベース作成ユティリティ（pdload）の場合は，ユティリティ起動時の時刻が設定されます。

CURRENT_TIMESTAMP〔(小数秒精度)〕〔USING BES〕（CURRENT TIMESTAMP〔(小数秒精度)〕〔USING BES〕）の場合：

行を挿入したときの時刻印が設定されます。ただし，データベース作成ユティリティ（pdload）の場合は，ユティリティ起動時の時刻印が設定されます。また，HiRDB/パラレルサーバでマルチフロントエンドサーバ構成で運用する場合，時刻印は UAP が接続したフロントエンドサーバで取得します。

USING BES を指定した場合，現在の時刻印を HiRDB/パラレルサーバの場合は更新行又は挿入行を格納する RD エリアを管理しているバックエンドサーバ，HiRDB/シングルサーバの場合はシングルサーバで取得します。

USING BES を省略した場合，HiRDB/パラレルサーバの場合はフロントエンドサーバ，HiRDB/シングルサーバの場合はシングルサーバで現在の時刻印を取得します。

既定値取得サーバ種別指定に USING BES を指定した列を分割キーに指定できません。

(n) 列制約

```
：：＝｛非ナル値制約
        ｜単一列一意性制約定義
          〔インデクスオプション〔インデクスオプション〕〕
        ｜｛ 単一列検査制約定義〔制約名定義〕
            ｜〔制約名定義〕　単一列検査制約定義 ｝
        ｜｛ 単一列参照制約定義〔制約名定義〕
            ｜〔制約名定義〕単一列参照制約定義 ｝｝
```

列単位に次の制約を指定できます。

- 非ナル値制約
- 単一列一意性制約定義
- 単一列検査制約定義
- 単一列参照制約定義

制約名定義の指定位置は，システム共通定義 pd_constraint_name オペランドの指定値，又はクライアント環境変数 PDCNSTRNTNAME の指定値によって決まります。詳細は，表「制約名定義の指定位置」を参照してください。

SHARE と FIX を指定している場合，単一列参照制約は指定できません。

(o) 予備列定義：：= FOR RESERVED

列を予備列として定義する場合に指定します。将来，FIX 表に列を追加する見込みがある場合に予備列を定義します。予備列を定義すると，FIX 表にデータが格納されている状態でも，予備列から切り出して列を追加できます。

予備列には，定義長分の 0x00 が格納されます。0x00 以外の値は格納できません。

予備列定義の規則を次に示します。

1. 非 FIX 表には定義できません。
2. CHAR 型以外の列に指定できません。
3. 文字集合と同時に指定できません。
4. 一つの表に複数指定できません。
5. 表構成列の最後の列以外に指定できません。
6. 予備列だけの表は定義できません。
7. 一時表には定義できません。

(p) 更新可能列属性：：= UPDATE 〔ONLY FROM NULL〕

改竄防止表を定義する場合又は改竄防止表に変更する予定のある表に更新可能列を定義する場合に指定します。

更新可能列属性は改竄防止表の場合だけ有効になります。

改竄防止表の定義については，「INSERT ONLY」オプションを参照してください。改竄防止表への変更については，「ALTER TABLE」の「CHANGE」の「INSERT ONLY」オプションを参照してください。

更新可能列属性についての規則を次に示します。

1. SYSTEM GENERATED を指定した列には指定できません。
2. 次に示す更新できない列には指定できません。
   - クラスタキー構成列
   - 分割キー構成列（フレキシブルハッシュ分割表の分割キー構成列を除く）
   - 予備列
3. 定義する表が一時表の場合，このオプションは指定できません。

UPDATE

改竄防止表で更新可能列を定義する場合に指定します。

UPDATE ONLY FROM NULL

改竄防止表での行の値がナル値から非ナル値へ一度だけ更新可能な列を定義する場合に指定します。

改竄防止表で UPDATE ONLY FROM NULL 指定した列の値の更新可否を次に示します。

更新前の列の値	更新後の列の値	更新可否
ナル値	ナル値	○
ナル値	非ナル値	○
非ナル値	ナル値	×
非ナル値	非ナル値※	×

（凡例）

○：更新できます。

×：更新できません。

注

繰返し列の場合は，ナル値（要素数が 0 の値）から添え字指定無しでの列単位更新だけ実行できます。

注※

更新前の値と同値を含みます。

UPDATE ONLY FROM NULL 指定についての規則を次に示します。

- NOT NULL を指定した列には指定できません。
- FIX 表の場合指定できません。
- 主キー又はクラスタキー構成列には指定できません。
- 分割キー構成列には指定できません。
- BLOB 型及び定義長が 32,001 バイト以上の BINARY 型の列には指定できません。

更新可能列属性の指定可否と指定時の列の値の更新可否は次のようになります。

表種別	UPDATE 指定		UPDATE ONLY FROM NULL 指定		左記の指定なし	
	指定可否	列の値の更新可否	指定可否	列の値の更新可否	指定可否	列の値の更新可否
非改竄防止表	○	○	○	○	－	○
改竄防止表	○	○	○	○※	－	×

（凡例）

○：更新できます。

×：更新できません。

－：該当しません。

注※

ナル値から非ナル値へ一度だけ更新できます。

(q) 非ナル値制約指定

```
::= { 〔NULL
      |NOT NULL 〔WITH DEFAULT 〔SYSTEM GENERATED〕 〕 〕 ※2
      | 〔 〔NOT NULL〕 WITH DEFAULT 〔SYSTEM GENERATED〕 〕 ※1}
```

注※1　FIX 表の列，クラスタキー又は主キーを構成する列の場合

注※2　上記以外の場合

NULL

列名で指定した列にナル値を許す場合に指定できます。

FIX 表の列，クラスタキー又は主キーを構成する列には指定できません。

NOT NULL

列名で指定した列にナル値を許さないように制約（非ナル値制約）する必要がある場合に指定します。

データ型が抽象データ型の場合，NOT NULL は指定できません。

NOT NULL は，繰返し列には指定できません。

WITH DEFAULT

INSERT 文，又はデータベース作成ユティリティを使用したデータロード時に，挿入する列名と挿入値の指定を省略したとき，非ナル値制約を持つ列に対して既定値を挿入する場合に指定します。

WITH DEFAULT についての規則を次に示します。

1. FIX 表に WITH DEFAULT を指定した場合，NOT NULL を省略できます。
2. データ型が抽象データ型の場合，WITH DEFAULT は指定できません。
3. 予備列には指定できません。
4. WITH DEFAULT を指定した場合，設定される列の既定値を次の表に示します。

表 3-27　WITH DEFAULT を指定したときの既定値

データ型		列の既定値
INTEGER		0
SMALLINT		
FLOAT		
SMALLFLT		
DECIMAL		
CHAR		空白
NCHAR		
MCHAR		
VARCHAR	既定文字集合	1 バイトの空白

データ型		列の既定値
	文字集合(UTF16 以外)	
	文字集合(UTF16)	1 文字の空白
NVARCHAR		1 文字の空白
MVARCHAR		1 バイトの空白
DATE		行追加時の現在の日付
TIME		行追加時の現在の時刻
TIMESTAMP		行追加時の現在の時刻印
INTERVAL YEAR TO DAY		0 か年 0 か月 0 か日間
INTERVAL HOUR TO SECOND		0 時間 0 分 0 秒間
BLOB		長さ 0 バイトのデータ
BINARY		

注　WITH DEFAULT を指定していない場合，ナル値が列の既定値になります。

SYSTEM GENERATED

列のデータ型が DATE 型，又は TIME 型の場合だけ指定できます。SYSTEM GENERATED を指定した列を，挿入履歴保持列といいます。また，SYSTEM GENERATED を指定した列は，改竄防止表の行削除禁止期間を指定するときに使用します。

SYSTEM GENERATED を指定した列は，INSERT 文でのデータ挿入時に，値の指定有無に関係なく，DATE 型の場合は現在の日付（CURRENT_DATE），TIME 型の場合は現在の時刻（CURRENT_TIME）を挿入します。

ただし，SYSTEM GENERATED を指定した列に対する挿入値としてキーワード NULL を指定するとエラーになります。

(r)　単一列一意性制約定義

```
: : = { 〔 {UNIQUE | PRIMARY} 〕 CLUSTER KEY 〔 {ASC | DESC} 〕
      〔IN {インデクス格納用RDエリア名
           | (インデクス格納用RDエリア名)
           | ( (インデクス格納用RDエリア名)
               〔, (インデクス格納用RDエリア名) 〕…) } 〕
       | PRIMARY KEY 〔 {ASC | DESC} 〕
         〔IN {インデクス格納用RDエリア名
              | (インデクス格納用RDエリア名)
              | ( (インデクス格納用RDエリア名)
                  〔, (インデクス格納用RDエリア名) 〕…) } 〕 }
```

〔{UNIQUE ｜ PRIMARY}〕 CLUSTER KEY〔{ASC ｜ DESC}〕

〔IN {インデクス格納用 RD エリア名

｜ (インデクス格納用 RD エリア名)

｜ ((インデクス格納用 RD エリア名)

〔,（インデクス格納用 RD エリア名)〕…)}〕

列名で指定した列をクラスタキーとして定義する場合に指定します。

クラスタキーについての規則を次に示します。

1. クラスタキーを指定した場合，それらを構成する列には，次に示すデータ型は指定できません。
   ・BLOB
   ・BINARY
   ・抽象データ型
2. クラスタキーを構成する列には，繰返し列は指定できません。
3. クラスタキーを指定すると，指定した列に対してインデクスが定義されます。定義されたインデクスは削除できません。
4. クラスタキーを構成する列には NOT NULL が仮定されます。
5. UNIQUE 又は PRIMARY を指定したクラスタキーに重複するデータは挿入できません。
6. 一つの表に二つ以上のクラスタキーは定義できません。
7. クラスタキーを指定した場合，ハッシュ関数に HASHA～HASHF は指定できません。
8. クラスタキーを構成する列の長さの合計は，次の計算式を満たすようにしてください。
   列の長さの合計
   ≦MIN((インデクス格納用 RD エリアのページサイズ÷2)－1242，4036)
9. 分割表にクラスタキーを指定する場合の規則を次に示します。
   キーレンジ分割（格納条件，境界値）
   クラスタキーを構成する列には，分割キーを指定してください。
   ハッシュ分割
   単一列分割
   クラスタキーを構成する列には，分割キーを指定してください。
   複数列分割
   クラスタキーを単一列に指定できません。
   マトリクス分割
   クラスタキーを単一列に指定できません。
10. 定義する表が一時表の場合は指定できません。
11. クラスタキーを構成する列には，予備列は指定できません。

UNIQUE

クラスタキーを構成する列の値がすべての行で異なるように制約する必要がある場合に指定します。

PRIMARY

クラスタキーを構成する列を主キーとして定義する場合に指定します。

ASC

クラスタキーのインデクスを昇順に作成する場合に指定します。

DESC

クラスタキーのインデクスを降順に作成する場合に指定します。

インデクス格納用 RD エリア名

クラスタキーのインデクスを格納する RD エリア名を指定します。

インデクスを横分割して格納する場合は，表を横分割する個数分のインデクス格納用 RD エリア名を指定します。

指定するインデクス格納用 RD エリア名は，一時表用 RD エリア以外のユーザ用 RD エリアにしてください。

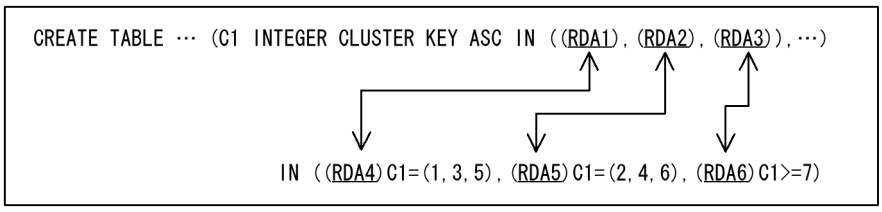
ただし，HiRDB/パラレルサーバでは次の制限があります。

- SHARE を指定している場合，インデクス格納用 RD エリア名は，共用 RD エリアにしてください。
- SHARE を指定していない場合，インデクス格納用 RD エリア名は，共用 RD エリアを指定できません。

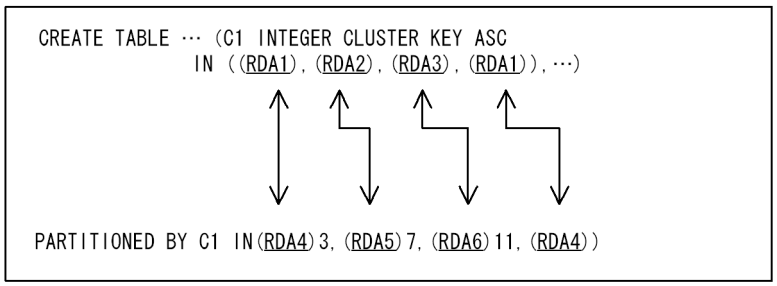
インデクス格納用 RD エリア名についての規則を次に示します。

1. RD エリアは，あらかじめデータベース初期設定ユティリティで作成，又はデータベース構成変更ユティリティで追加されていなければなりません。
2. インデクス格納用 RD エリア名を省略すると，定義した表を格納する RD エリアにインデクスも格納されます。
3. インデクス格納用 RD エリア名を複数個指定する場合，同じ RD エリア名は指定できません。境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表では，表格納用 RD エリアとインデクス格納用 RD エリアが 1 対 1 に対応するようにインデクス格納用 RD エリア名を指定してください。表格納用 RD エリア名を重複指定した場合は，インデクス格納用 RD エリアも同一位置が重複するように指定してください。
4. リバランス表を格納しているユーザ用 RD エリアは指定できません。また，リバランス表に対してクラスタキーを定義する場合，RD エリア名の指定は省略できません。
5. インデクス格納用 RD エリア名を指定する場合，その数は横分割した表の格納数と同じにして，同じサーバの RD エリアを指定する必要があります。このとき，インデクス格納用 RD エリア名は表格納用 RD エリア名の指定順に対応します。例を次に示します。

   1.格納条件によって横分割する場合



   2.境界値によって横分割する場合



（凡例）

RDA1～6：RDエリア名を示します。

：表のRDエリアRDA4，RDA5，及びRDA6が，インデクスのRDエリア RDA1，RDA2，及びRDA3とそれぞれ対応していることを示します。

PRIMARY KEY〔{ASC ｜ DESC}〕

〔IN {インデクス格納用 RD エリア名

｜ (インデクス格納用 RD エリア名)

｜ ((インデクス格納用 RD エリア名)

〔, (インデクス格納用 RD エリア名)〕…)}〕

列名で指定した列を主キーとして定義する場合に指定します。

主キーについての規則を次に示します。

1. 主キーを指定した場合，それらを構成する列には，次に示すデータ型は指定できません。
   - BLOB
   - BINARY
   - 抽象データ型
2. 主キーを構成する列には，繰返し列は指定できません。
3. 主キーを構成する列には，予備列は指定できません。
4. 主キーを指定すると，指定した列に対してインデクスが定義されます。定義されたインデクスを削除する場合は，主キーを削除してください。
5. 一時表の場合，ALTER TABLE で主キーを削除できません。一時表に定義された主キーを削除する場合は，一時表を削除して，再度定義してください。
6. 主キーを構成する列には NOT NULL が仮定されます。
7. 主キーに重複するデータは挿入できません。
8. 一つの表に二つ以上の主キーは定義できません。
9. 横分割表に対する主キーの定義可否（UNIQUE 指定可否）については，「表を横分割する場合の UNIQUE 指定可否」を参照してください。
10. 主キーを構成する列の長さの合計は，次の計算式を満たすようにしてください。

    列の長さの合計

    ≦MIN（(インデクス格納用 RD エリアのページサイズ÷2）－1242，4036）

定義するインデクスが一時インデクスの場合は，一時表を実体化する際に上記を満たすかどうかをチェックします。

一時表の実体化については，マニュアル「HiRDB システム導入・設計ガイド」の「一時表」を参照してください。

11. マトリクス分割表には，主キーを単一列に指定できません。

ASC

主キーのインデクスを昇順に作成する場合に指定します。

DESC

主キーのインデクスを降順に作成する場合に指定します。

インデクス格納用 RD エリア名

主キーのインデクスを格納する RD エリア名を指定します。

指定するインデクス格納用 RD エリアは，一時表用 RD エリア以外のユーザ用 RD エリアにしてください。

ただし，HiRDB/パラレルサーバでは次の制限があります。

- SHARE を指定している場合，インデクス格納用 RD エリア名は，共用 RD エリアにしてください。
- SHARE を指定してない場合，インデクス格納用 RD エリア名は，共用 RD エリアを指定できません。

インデクス格納用 RD エリア名についての規則を次に示します。

1. RD エリアは，あらかじめデータベース初期設定ユティリティで作成，又はデータベース構成変更ユティリティで追加されていなければなりません。
2. RD エリア名を省略すると，定義した表を格納する RD エリアにインデクスも格納されます。ただし，HiRDB/シングルサーバでの横分割表の場合，又は HiRDB/パラレルサーバでの同じバックエンドサーバ内の横分割表の場合，インデクスは次のように格納されます。

   単一列分割する表の場合：

   分割キー以外の列を主キーに指定した場合，最初に分割条件を指定した表格納用 RD エリアに格納されます。

   複数列分割する表の場合：

   最初に分割条件を指定した表格納用 RD エリアに格納されます。

   マトリクス分割表の場合：

   主キーを単一列に指定できません。
3. 指定する RD エリア名は，表格納用 RD エリアと同じサーバに定義しているユーザ用 RD エリアを指定してください。
4. インデクス格納用 RD エリア名を複数個指定する場合，同じ RD エリア名は指定できません。境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表では，表格納用 RD エリアとインデクス格納用 RD エリアが 1 対 1 に対応するようにインデクス格納用 RD エリア名を指定して

ください。表格納用 RD エリア名を重複指定した場合は，インデクス格納用 RD エリアも同一位置が重複するように指定してください。

5. リバランス表を格納しているユーザ用 RD エリアは指定できません。また，リバランス表に対して主キーを定義する場合，RD エリア名の指定は省略できません。

6. 表を複数の RD エリアに分割格納している場合，インデクス格納用 RD エリアは次のどれかの指定方法となります。

   (a) 単一列分割する表で，かつ分割キーに指定した列を主キーの先頭に指定する場合

   (b) 複数列分割する表で，かつ分割キーに指定したすべての列を主キーの先頭から同順に指定する場合

   (c) マトリクス分割表の場合

   (d) (a)～(c)以外で，分割キーに指定した列を主キーの構成列にすべて指定する場合（順不同）

   (a)～(c)の場合，RD エリア名は表格納用 RD エリアと同じ数だけ指定します。この場合，インデクスの格納先は指定した表格納用 RD エリアの指定順に対応します。

   (d)の場合，RD エリア名は表を分割して格納しているサーバの数だけ指定します。HiRDB/シングルサーバの場合は一つだけ指定し，HiRDB/パラレルサーバの場合は，表があるバックエンドサーバごとに一つの RD エリアを指定します。

   上記以外の場合，分割格納する表に対して主キーは定義できません。

7. 定義する表が一時表の場合は指定できません。

(s) インデクスオプション：：＝ {PCTFREE ＝未使用領域の比率 | UNBALANCED SPLIT}

PCTFREE ＝未使用領域の比率

インデクス作成時のインデクスページ内の未使用領域の比率を指定します。未使用領域の比率は，0～99（%）を指定できます。省略すると 30%が仮定されます。仮定値は，システム共通定義の pd_ddl_idx_pctfree オペランドで変更できます。pd_ddl_idx_pctfree オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

データベース作成ユティリティ，及びデータベース再編成ユティリティでのインデクス一括作成時に，未使用領域の比率でインデクスが作成されます。INSERT 文や UPDATE 文など，ほかの追加及び更新では PCTFREE ＝ 0 が仮定されます。

インデクスの定義後，行の追加が頻繁に発生する場合，未使用領域の比率を高く指定してください。

UNBALANCED SPLIT

ページスプリットする場合，両ページに対するキーの振り分けを不均衡にします。

スプリット対象ページへのキーの追加位置が，ページの前半部の場合はスプリット後の左側のページの空きを多く確保し，後半部の場合は右側のページの空きを多く確保するように分割します。このようなインデクスをアンバランスインデクススプリットといいます。

アンバランスインデクススプリットについては，マニュアル「HiRDB システム運用ガイド」を参照してください。

(t)　単一列検査制約定義：：＝ CHECK（探索条件）

列名で指定した列に対して検査制約を定義する場合に指定します。

探索条件

列の値を制約する条件を指定します。この条件が偽になる場合，この表に対して挿入，更新はできません。

探索条件中に指定する列は，列名で指定した列を指定してください。

探索条件には，次のものを含むことはできません。

- 副問合せ
- 集合関数又は SQL/XML 集合関数
- ウィンドウ関数
- 繰返し列
- 予備列
- 関数呼出し
- USER，CURRENT_USER
- CURRENT DATE，CURRENT_DATE
- CURRENT TIME，CURRENT_TIME
- CURRENT TIMESTAMP，CURRENT_TIMESTAMP
- ？パラメタ，埋込み変数
- TIME から TIMESTAMP への変換を指定した CAST 指定
- 値式に TIME 型を指定したスカラ関数 VARCHAR_FORMAT
- 抽象データ型の値式
- システム定義スカラ関数
- スカラ関数 IS_USER_CONTAINED_IN_HDS_GROUP
- 構造化繰返し述語
- XML コンストラクタ関数
- SQL/XML スカラ関数

(u)　単一列参照制約定義：：＝参照指定

列名で指定した列を外部キーとして定義する場合に指定します。

外部キーについての規則を次に示します。

1. 外部キーを指定した場合，列には，次に示すデータ型は指定できません。
   - BLOB

- BINARY
- 抽象データ型

2. 外部キー列には，繰返し列は指定できません。
3. 外部キー列には，予備列は指定できません。
4. 外部キーと参照する主キーは次に示す事項をすべて同じにしてください。
   - 対応するデータ型
   - 対応するデータ長

(v) 複数列一意性制約定義

```
: : = { 〔 {UNIQUE | PRIMARY} 〕 CLUSTER KEY
        (列名 〔 {ASC | DESC} 〕 〔, 列名 〔 {ASC | DESC} 〕〕…)
        〔IN {インデクス格納用RDエリア名
             | (インデクス格納用RDエリア名)
             | ((インデクス格納用RDエリア名)
                 〔, (インデクス格納用RDエリア名)〕…)
             | マトリクス分割インデクス格納用RDエリア指定} 〕
      | PRIMARY KEY (列名 〔 {ASC | DESC} 〕
        〔, 列名 〔 {ASC | DESC} 〕〕…)
        〔IN {インデクス格納用RDエリア名
             | (インデクス格納用RDエリア名)
             | ((インデクス格納用RDエリア名)
                 〔, (インデクス格納用RDエリア名)〕…)
             | マトリクス分割インデクス格納用RDエリア指定} 〕}
```

〔{UNIQUE | PRIMARY}〕 CLUSTER KEY (列名〔{ASC | DESC}〕
〔, 列名〔{ASC | DESC}〕〕…)
〔IN {インデクス格納用 RD エリア名
| (インデクス格納用 RD エリア名)
| ((インデクス格納用 RD エリア名)
〔, (インデクス格納用 RD エリア名)〕…)
| マトリクス分割インデクス格納用 RD エリア指定}〕

複数の列をクラスタキーとして定義する場合に指定します。

クラスタキーについての規則を次に示します。

1. クラスタキーを指定した場合，それらを構成する列には，次に示すデータ型は指定できません。
   ・BLOB
   ・BINARY
   ・抽象データ型
2. クラスタキーを複数の列で構成する場合，項番 1 に加えて次に示すデータ型も指定できません。
   ・FLOAT
   ・SMALLFLT

3. クラスタキーを構成する列には，繰返し列は指定できません。

4. クラスタキーを構成する列には，予備列は指定できません。

5. クラスタキーを指定すると，指定した列に対してインデクスが定義されます。定義されたインデクスを削除する場合，表を削除してください。

6. クラスタキーを構成する列には NOT NULL が仮定されます。

7. UNIQUE 又は PRIMARY を指定したクラスタキーに同じデータは挿入できません。

8. 一つの表に二つ以上のクラスタキーは定義できません。

9. クラスタキーを構成する列の数は，最大 64 個です。

10. クラスタキーを指定した場合，ハッシュ関数に HASHA～HASHF は指定できません。

11. インデクスを構成する列の長さの合計は，次の計算式を満たすようにしてください。

    列の長さの合計

    ≦MIN((インデクス格納用 RD エリアのページサイズ÷2)－1242，4036)

12. 分割表にクラスタキーを指定する場合の規則を次に示します。

    キーレンジ分割（格納条件，境界値）

    クラスタキーを構成する列の先頭には，分割キーを指定してください。

    ハッシュ分割

    単一列分割

    クラスタキーを構成する列の先頭には，分割キーを指定してください。

    複数列分割

    クラスタキーを構成する列の先頭から，分割キーを同順にすべて含むように指定してください。

    マトリクス分割

    クラスタキーを構成する列の先頭から，第 1 次元分割列，第 2 次元分割列の順にすべて含むように指定してください。

    第 2 次元分割列を複数列分割する場合も，第 1 次元分割列から同順にすべての分割キーを含むように指定してください。

13. 定義する表が一時表の場合は指定できません。

UNIQUE

クラスタキーを構成する複数の列の値がすべての行で異なるように制約する必要がある場合に指定します。

PRIMARY

クラスタキーを構成する複数の列を主キーとして定義する場合に指定します。

列名

クラスタキーを定義する列の名前を指定します。

クラスタキーを構成する列名は同じ列名を指定できません。複数の列にクラスタキーを定義する場合，各列の指定順序にキー値が作成されます。

ASC

クラスタキーのインデクスを昇順に作成する場合に指定します。

DESC

クラスタキーのインデクスを降順に作成する場合に指定します。

インデクス格納用 RD エリア名

クラスタキーのインデクスを格納する RD エリア名を指定します。

インデクスを横分割して格納するときは，表を横分割する個数分のインデクス格納用 RD エリア名を指定します。

指定するインデクス格納用 RD エリア名は，一時表用 RD エリア以外のユーザ用 RD エリアにしてください。

インデクス格納用 RD エリアについての規則を次に示します。

1. RD エリアは，あらかじめデータベース初期設定ユティリティで作成，又はデータベース構成変更ユティリティで追加されていなければなりません。
2. インデクス格納用 RD エリア名を省略すると，定義した表を格納する RD エリアにインデクスも格納されます。
3. 指定する RD エリア名は，表格納用 RD エリアと同じサーバに定義しているユーザ用 RD エリアを指定してください。
4. インデクス格納用 RD エリア名を複数個指定する場合，同じ RD エリア名は指定できません。境界値指定の横分割表，ハッシュ分割表，又はマトリクス分割表では，表格納用 RD エリアとインデクス格納用 RD エリアが 1 対 1 に対応するようにインデクス格納用 RD エリア名を指定してください。表格納用 RD エリア名を重複指定した場合は，インデクス格納用 RD エリアも同一位置が重複するように指定してください。
5. インデクス格納用 RD エリア名を複数個指定する場合，その数は横分割した表の格納数と同じにする必要があります。このとき，インデクス格納用 RD エリア名は表格納用 RD エリア名の指定順に対応します。例については単一列一意性制約定義のインデクス格納用 RD エリア名を参照してください。
6. リバランス表を格納しているユーザ用 RD エリアは指定できません。また，リバランス表に対してクラスタキーを定義する場合，RD エリア名の指定は省略できません。

マトリクス分割インデクス格納用 RD エリア指定：：＝ 2 次元格納用 RD エリア指定

マトリクス分割表を定義する場合に指定します。

指定方法については，PARTITIONED BY MULTIDIM の 2 次元格納用 RD エリア指定の説明を参照してください。また，指定する RD エリアについては，インデクス格納用 RD エリア名の説明を参照してください。

PRIMARY KEY（列名〔{ASC ｜ DESC}〕
〔，列名〔{ASC ｜ DESC}〕〕…）
〔IN｛インデクス格納用 RD エリア名
｜（インデクス格納用 RD エリア名）

| ((インデクス格納用 RD エリア名)
〔,(インデクス格納用 RD エリア名)〕…)
| マトリクス分割インデクス格納用 RD エリア指定}〕

複数の列を主キーとして定義する場合に指定します。

主キーについての規則を次に示します。

1. 主キーを指定した場合，それらを構成する列には，次に示すデータ型は指定できません。
   ・BLOB
   ・BINARY
   ・抽象データ型
2. 主キーを複数の列で構成する場合，項番 1 に加えて次に示すデータ型も指定できません。
   ・FLOAT
   ・SMALLFLT
3. 主キーを構成する列には，繰返し列は指定できません。
4. 主キーを構成する列には，予備列は指定できません。
5. 主キーを指定すると，指定した列に対してインデクスが定義されます。定義されたインデクスを削除する場合は，主キーを削除してください。
6. 一時表の場合，ALTER TABLE で主キーを削除できません。一時表に定義された主キーを削除する場合は，一時表を削除して，再度定義してください。
7. 主キーを構成する列には NOT NULL が仮定されます。
8. 主キーに同じデータは挿入できません。
9. 一つの表に二つ以上の主キーは定義できません。
10. 主キーを構成する列の数は，最大 64 個です。
11. 横分割表に対する主キーの定義可否（UNIQUE 指定可否）については，「表を横分割する場合の UNIQUE 指定可否」を参照してください。
12. インデクスを構成する列の長さの合計は，次の計算式を満たすようにしてください。

    列の長さの合計
    ≦MIN（(インデクス格納用 RD エリアのページサイズ÷2）－1242，4036）

    定義するインデクスが一時インデクスの場合は，一時表を実体化する際に上記を満たすかどうかをチェックします。

    一時表の実体化については，マニュアル「HiRDB システム導入・設計ガイド」の「一時表」を参照してください。

列名

主キーを定義する列の名前を指定します。

主キーを構成する列名は同じ列名を指定できません。複数の列に主キーを定義する場合，各列の指定順序にキー値が作成されます。

ASC

主キーのインデクスを昇順に作成する場合に指定します。

DESC

主キーのインデクスを降順に作成する場合に指定します。

インデクス格納用 RD エリア名

主キーのインデクスを格納する RD エリア名を指定します。

1. インデクス格納用 RD エリア名を省略すると，定義した表を格納する RD エリアにインデクスも格納されます。ただし，HiRDB/シングルサーバでの横分割表の場合，又は HiRDB/パラレルサーバでの同じバックエンドサーバ内の横分割表の場合，インデクスは次のように格納されます。

   単一列分割する表の場合：

   定義した表を格納する RD エリアにインデクスも格納されます。

   複数列分割する表の場合：

   分割キーに指定したすべての列を，主キーの先頭から同じ順番で指定しないと，最初に分割条件を指定した表格納用 RD エリアに格納されます。

   マトリクス分割表の場合：

   分割キーに指定したすべての列を，主キーの先頭から同じ順番で指定しないと，表定義はできません。分割キーに指定したすべての列を，主キーの先頭から同じ順番で指定しない場合，インデクス格納用 RD エリア名を指定してください。

インデクス格納用 RD エリア名の説明，規則については，単一列一意性制約定義の PRIMARY KEY のインデクス格納用 RD エリア名の説明を参照してください。

マトリクス分割インデクス格納用 RD エリア指定：：＝ 2 次元格納用 RD エリア指定

マトリクス分割表を定義する場合に指定します。

指定方法については，PARTITIONED BY MULTIDIM の 2 次元格納用 RD エリア指定の説明を参照してください。また，指定する RD エリアについては，複数列一意性制約定義の〔{UNIQUE ｜ PRIMARY}〕 CLUSTER KEY のインデクス格納用 RD エリア名の説明を参照してください。

(w) 複数列検査制約定義：：＝ CHECK（探索条件）

複数の列に対して検査制約を定義する場合に指定します。

探索条件

複数の列を制約する条件を指定します。この条件が偽になる場合，この表に対して挿入，更新はできません。

探索条件中に指定する列は，表定義で指定した列を指定してください。

探索条件中の制限は，「単一列検査制約定義」の「探索条件」を参照してください。

(x) 複数列参照制約定義：：＝ FOREIGN KEY（列名〔，列名〕…） 参照指定

複数列を外部キーとして定義する場合に指定します。

外部キーについての規則を次に示します。

1. 外部キーを指定した場合，それらを構成する列には，次に示すデータ型は指定できません。
   - BLOB
   - BINARY
   - 抽象データ型
2. 外部キーを構成する列には，繰返し列は指定できません。
3. 外部キーを構成する列には，予備列は指定できません。
4. 外部キーと参照する主キーは次に示す事項をすべて同じにしてください。
   - 対応するデータ型
   - 対応するデータ長
   - 列数

   なお，外部キーとして複数の列を指定した場合，HiRDB は指定順に列の対応をチェックします。

FOREIGN KEY（列名〔，列名〕…） 参照指定

外部キーを構成する列名を指定します。

列名は最大 64 個指定できます。

列名は同じ列名を指定できません。

(y) 格納条件：：＝列名 {= | <> | ^= | != | < | <= | > | >=} {定数 | (定数〔, 定数〕)}

表を複数の RD エリアに分割（表の横分割）して格納するための条件を指定します。

格納条件についての規則を次に示します。

1. 定数を複数個指定できるのは，比較演算子で＝を用いた場合だけです。
2. 定数を複数個指定する場合，同じ値は指定しないでください。
3. 比較演算できる列のデータ型を次に示します。
   - INTEGER
   - SMALLINT
   - DECIMAL
   - FLOAT
   - SMALLFLT
   - CHARACTER※1
   - VARCHAR※1
   - NCHAR※2

- NVARCHAR※2
- MCHAR※1
- MVARCHAR※1
- DATE
- TIME
- TIMESTAMP※4
- INTERVAL YEAR TO DAY※3
- INTERVAL HOUR TO SECOND※3

注※1

定義長が 255 バイト以下の列だけ指定できます。

注※2

定義長が 127 文字以下の列だけ指定できます。

注※3

データディクショナリ表 SQL_DIV_TABLE 中の DCVALUES 列（格納振り分け条件値）には，補正した値が格納されます。

（例）

19921225. → 19930025.

99981315. → 99990115.

注※4

DEFAULT 句中に USING BES を含む既定値を指定した場合は指定できません。

4. 定数には，次のものは指定できません。
   - 長さ 0 の文字列定数，各国文字列定数，混在文字列定数，及び 16 進文字列定数
   - 長さ 256 バイト以上の文字列定数，長さ 128 文字以上の各国文字列定数，長さ 256 バイト以上の混在文字列定数，及び長さ 256 バイト以上の 16 進文字列定数（16 進文字列の文字数 512 文字以上）
5. 列名で指定した列の値は更新できません。
6. 一つの格納条件に指定する定数の数は，すべての格納条件に指定した定数の総数が 15,000 以下の範囲で指定してください。ただし，格納条件を省略した場合も 1 と数えます。
7. クラスタキーを指定した列がある場合，それ以外の列に対して格納条件を指定できません。
8. 列名で指定した列は NOT NULL（非ナル値制約，FIX 指定，クラスタキー，又は主キー）にしてください。
9. クラスタキーが複数列の場合，先頭の列名以外の列に対して格納条件は指定できません。
10. 繰返し列には，格納条件を指定できません。
11. 予備列には，格納条件を指定できません。

(z) ハッシュ関数名：：＝ {HASH1 | HASH2 | HASH3 | HASH4 | HASH5 | HASH6 | HASHZ | HASH0 | HASHA | HASHB | HASHC | HASHD | HASHE | HASHF}

リバランス表でない場合，又はマトリクス分割表の第 2 次元にハッシュ関数を指定する場合：

HASH1～HASH6，HASHZ，又は HASH0 のどれかを指定してください。

HASH6 が最も均等にハッシングされるので，通常は HASH6 を指定してください。ただし，分割キーのデータによっては均等にならない場合もあるので，そのときにはほかのハッシュ関数を指定してください。

HASH0 は年月の値で，データ格納先 RD エリアを月単位に循環させる場合に指定してください。

HASHZ は年月日の値で，データ格納先 RD エリアを日単位に循環させる場合に指定してください。

HASH0 と HASHZ の使い分け

運用方法	推奨
月単位で分割する	HASH0
日単位で分割する	HASHZ

リバランス表の場合：

HASHA～HASHF のどれかを指定してください。

HASHF が最も均等にハッシングされるので，通常は HASHF を指定してください。ただし，分割キーのデータによっては均等にならない場合もあるので，そのときにはほかのハッシュ関数を指定してください。

HASH1，HASHA

このハッシュ関数は，分割に指定した列のすべてのデータ型に使用できます。分割に指定したすべての列のデータの全バイト※を使用してハッシュします。データ長が 0 バイト以上の列に指定できます。

HASH2，HASHB

このハッシュ関数は，分割に指定した列のすべてのデータ型に使用できます。分割に指定したすべての列のデータの全バイト※を使用してハッシュします。データ長が 0 バイト以上の列に指定できます。

HASH1 又は HASHA を指定してデータを均等に RD エリアに格納できなかった場合に指定します。

HASH3，HASHC

このハッシュ関数は，分割に指定した列のデータ型が INTEGER，又は SMALLINT 専用です。分割したすべての列の末尾 2 バイト※を使用してハッシュします。データ長が 2 バイト以上の列に指定できます。

HASH4，HASHD

このハッシュ関数は，分割に指定した列のデータ型が DATE 専用です。分割に指定したすべての列の先頭 4 バイト※を使用してハッシュします。データ長が 4 バイト以上の列に指定できます。

HASH5，HASHE

このハッシュ関数は，分割に指定した列のデータ型が TIME 専用です。分割に指定したすべての列の先頭 3 バイト※を使用してハッシュします。データ長が 3 バイト以上の列に指定できます。

HASH6，HASHF

このハッシュ関数は，分割に指定した列のすべてのデータ型に使用できます。DECIMAL の場合に最も適しています。分割に指定したすべての列の，データの全バイト※を使用してハッシュします。データ長が 0 バイト以上の列に指定できます。

HASHZ

このハッシュ関数は，分割列中の年月日の値を使用してハッシングし，データを格納する RD エリアを日単位で循環させて割り当てる場合に使用します。

このハッシュ関数を指定する場合には，分割キーは 1 列で，そのデータ型は DATE，TIMESTAMP，CHAR(8)（文字集合が UTF16 の場合は CHAR(16)）でなければなりません。文字の長さが 8 文字の場合の文字形式は，’YYYYMMDD’ にしてください。

また，YYYY，MM，及び DD は，次の値にしてください。

YYYY：0001～9999（年）

MM：01～12（月）

DD：01～該当年月の最終日（日）

HASH0

このハッシュ関数は，分割列中の年月の値を使用してハッシングし，データを格納する RD エリアを月単位で循環させて割り当てる場合に使用します。

このハッシュ関数を指定する場合には，分割キーは 1 列で，そのデータ型は DATE，TIMESTAMP，CHAR(8)（文字集合が UTF16 の場合は CHAR(16)），又は CHAR(6)（文字集合が UTF16 の場合は CHAR(12)）でなければなりません。文字の長さが 8 文字の場合の文字形式は，’YYYYMMDD’，文字の長さが 6 文字の場合の文字形式は，’YYYYMM’ にしてください。

また，YYYY，MM，及び DD は，次の値にしてください。

YYYY：0001～9999（年）

MM：01～12（月）

DD：01～該当年月の最終日（日）

注※

VARCHAR 型，MVARCHAR 型，又は NVARCHAR 型は，後続の空白を無視してハッシュします。また，DECIMAL 型，INTERVAL YEAR TO DAY 型，又は INTERVAL HOUR TO SECOND 型で符号部が'F'の場合は，'C'に変換してハッシュします。

(aa)　参照指定：：＝ REFERENCES　被参照表　〔参照制約動作指定〕

参照する被参照表を指定します。制約動作を指定したい場合は参照制約動作指定を指定します。

参照指定を指定した場合，参照指定を指定した表（参照表）は，次の表に示す操作規則が適用されます。参照指定を指定した表が参照する表（被参照表）には，表「CASCADE 指定時の被参照表に対する操作，

及びそれに伴う参照表への影響」及び表「RESTRICT 指定時の被参照表に対する操作」に示す操作規則が適用されます。

表 3-28　参照指定指定時の参照表に対する操作

外部キー構成列に対する操作	外部キー構成列の行と，外部キーが参照する被参照表の行の関係		操作結果
追加（INSERT）	挿入する行の外部キー構成列の値と等しい主キー構成列の値を持つ行が被参照表中に存在する。		○
	挿入する行の外部キー構成列の値と等しい主キー構成列の値を持つ行が被参照表中に存在しない。	挿入する行の外部キー構成列中にナル値がある。	○
		挿入する行の外部キー構成列中にナル値がない。	×
更新（UPDATE）	更新後の外部キー構成列の値と等しい主キー構成列の値を持つ行が被参照表中に存在する。		○
	更新後の外部キー構成列の値と等しい主キー構成列の値を持つ行が被参照表中に存在しない。	更新後の外部キー構成列中にナル値がある。	○
		更新後の外部キー構成列中にナル値がない。	×

（凡例）

○：参照表に対する操作ができます。

×：制約違反エラーとなります。

被参照表：：＝表名

参照する表名を指定します。

表名の規則を次に示します。

- 主キーを持つ表の表名を指定してください。
- 表名は実表にしてください。
- 自分の所有する表を指定してください。
- 定義する表識別子は指定できません。

参照制約動作指定：：＝ 〔削除動作〔更新動作〕｜更新動作〔削除動作〕〕

主キーの更新，削除に同期して行われる外部キーに対する動作を指定します。

削除動作：：＝ ON DELETE 参照動作

被参照表の行を削除したときの動作を指定します。

更新動作：：＝ ON UPDATE 参照動作

被参照表の行を更新したときの動作を指定します。

参照動作

CASCADE

データの整合性を保つため，主キーに対する操作を外部キーに伝える場合に指定します。

被参照表に対する操作，及びそれに伴う参照表への影響を次の表に示します。

表 3-29　CASCADE 指定時の被参照表に対する操作，及びそれに伴う参照表への影響

外部キー構成列が参照する被参照表の主キー構成列に対する操作	外部キー構成列の行と外部キーが参照する被参照表の行の関係	参照表に対する影響
削除（DELETE）	削除する行の主キー構成列の値と等しい外部キー構成列の値を持つ行が参照表に存在する。	行削除
	削除する行の主キー構成列の値と等しい外部キー構成列の値を持つ行が参照表に存在しない。	影響しない
更新（UPDATE）	更新前の主キー構成列の値と等しい外部キー構成列の値を持つ行が参照表に存在する。	主キーと等しい値で更新
	更新前の主キー構成列の値と等しい外部キー構成列の値を持つ行が参照表に存在しない。	影響しない

RESTRICT

主キーに対する操作が外部キーに影響を与えるかどうかチェックして，データの整合性を維持するように操作を制約する場合に指定します。

被参照表に対する操作を次の表に示します。

表 3-30　RESTRICT 指定時の被参照表に対する操作

外部キー構成列が参照する被参照表の主キー構成列に対する操作	外部キー構成列の行と外部キーが参照する被参照表の行の関係	操作結果
削除（DELETE）	削除する行の主キー構成列の値と等しい外部キー構成列の値を持つ行が参照表に存在する。	×
	削除する行の主キー構成列の値と等しい外部キー構成列の値を持つ行が参照表に存在しない。	○
更新（UPDATE）	更新前の主キー構成列の値と等しい外部キー構成列の値を持つ行が参照表に存在する。	×
	更新前の主キー構成列の値と等しい外部キー構成列の値を持つ行が参照表に存在しない。	○

（凡例）

○：被参照表に対する操作ができます。

×：制約違反エラーとなります。

参照制約動作自体を省略した場合は，ON DELETE RESTRICT ON UPDATE RESTRICT を仮定します。

削除指定を省略した場合は ON DELETE RESTRICT，更新指定を省略した場合は ON UPDATE RESTRICT を仮定します。

参照制約動作に CASCADE を指定した場合，HiRDB が制約操作を行うためのトリガを生成します。

作成するトリガの名称を次の表に示します。なお，トリガの名称の長さは共に 21 バイトです。

表 3-31　HiRDB が作成するトリガの名称

参照制約動作	トリガ名称
削除動作	(DRAyyyymmddhhmmssth)
更新動作	(URAyyyymmddhhmmssth)

注　yyyymmddhhmmssth は，トリガ生成時のタイムスタンプです（100 分の 1 秒単位までの情報を持ちます）。

また，トリガの SQL コンパイルオプションは，トリガ定義で SQL コンパイルオプションを省略した場合の値と同じです。省略時の値については「CREATE TRIGGER（トリガ定義）」を参照してください。

外部キーが複数指定された場合は，次に示す順序で制約が行われます。

1. CASCADE
2. RESTRICT

CASCADE を複数指定した場合，表に指定した順に CASCADE の制約を行います。

RESTRICT を複数指定した場合，最適な制約チェックができるように，HiRDB が制約を行う順序を決定し，その順序で制約を行います。

(ab)　制約名定義：：＝ CONSTRAINT 制約名

指定した制約に対して，制約名を定義する場合に指定します。

制約名

スキーマ内で同一制約名は指定できません。

制約名を省略した場合，HiRDB が制約名を仮定します。

HiRDB が仮定する制約名を次の表に示します。

表 3-32　制約名省略時に HiRDB が仮定する制約名

種類		制約名	備考
参照制約	単一列 参照制約定義	制約を指定する列名	なし
	複数列 参照制約定義	外部キーに指定した最初の列名	なし
検査制約		CK_表番号_yyyymmddhhmmssth	30 文字固定（表番号 10 文字，時刻 16 文字）

注　yyyymmddhhmmssth は，制約定義時のタイムスタンプです（100 分の 1 秒単位までの情報を持ちます）。

表番号は，10 文字で右詰め，先頭から 0 が埋められた値です。

ユーザが制約名を指定する場合は，重複する可能性があるため，上記のような形式の制約名は指定しないでください。

(ac) WITH PROGRAM

外部キーを定義する場合，該当する手続き，及びトリガの有効な SQL オブジェクトを無効にする場合に指定します。外部キーを定義しない場合，WITH PROGRAM の指定は無視します。無効になるオブジェクトを次の表に示します。

表 3-33 無効になるオブジェクト

作成バージョン	オブジェクトの内容	
	オブジェクトの種類	無効になる条件
07-00 以降	手続き，及びトリガのオブジェクト	REFERENCES で指定した表を使用した UPDATE 文，又は DELETE 文を含む場合
07-00 より前		REFERENCES で指定した表を使用した SQL 文を含む場合

(5) 共通規則

1. 表は，順序数生成子と合わせて一つの RD エリアに最大 500 個定義できます。
2. クラスタキーを定義した列は更新できません。
   ただし，クラスタキーを定義した列に，定義長 256 バイト以上の可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）の列を含む場合は更新できます。クラスタキーを更新すると，更新した行はクラスタリング効果がなくなる可能性があります。
3. クラスタキーを定義した列にナル値は挿入できません。
4. クラスタキーを定義した列は更新できません。
5. 指定する LOB 列格納用 RD エリア，又は LOB 属性格納用 RD エリアには，既にほかの BLOB 列，又は BLOB 属性に割り当てられている RD エリアは指定できません。
6. 同じ列構成に対して CLUSTER KEY 句と PRIMARY KEY 句は指定できません。同じ列構成にクラスタキーと主キーを定義したい場合は，PRIMARY CLUSTER KEY 句を指定してください。この場合の同じ列構成とは，次の条件をすべて満たすものをいいます。
   - CLUSTER KEY 句と PRIMARY KEY 句に指定した列名の並び，及び指定した列数が一致している。
   - 昇順，降順の指定がすべて一致しているか，又はすべて逆になっている。
7. 表格納用 RD エリア，LOB 列格納用 RD エリア，LOB 属性格納用 RD エリア，及びインデクス格納用 RD エリアに，インナレプリカ機能を適用している RD エリアと適用していない RD エリアは混在して指定できません。インナレプリカ機能を適用している RD エリアを指定する場合，RD エリア名にはオリジナル RD エリア名を指定します。
8. インナレプリカ機能を使用している場合の CREATE TABLE の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
9. 次の表に対して，HiRDB Dataextractor，及び HiRDB Datareplicator を使用したデータ反映はしないでください。
   - 改竄防止表

- 非ナル値制約を指定し，かつ SYSTEM GENERATED を指定した列がある表

10. クラスタキー又は主キーを定義すると，定義されるインデクスのインデクス識別子は次の表に示す規則で決定されます。

表 3-34　定義されるインデクス識別子

指定項目	インデクス識別子
CLUSTER	（CLUSTER 表番号）
PRIMARY	（PRIMARY 表番号）
PRIMARY CLUSTER	（PRI－CLS 表番号）

注

表番号の部分は 10 文字の右詰めで，先頭から 0 が埋められた値になります。

合計で 19 文字（このうち 9 文字は上記に示した括弧と文字）の固定値が指定されます。

11. 一時表を定義する場合，格納用 RD エリアは指定できません。

12. 列は一つの表に最大 30,000 個定義できます。

また，列の長さ（データ長）の合計が，次の式を満たす必要があります。各列の長さ（データ長）を次の表に示します。

- FIX 指定のない表（表操作時）

$$\sum_{i=1}^{n}(2+ai) \leqq \text{格納するRDエリアのページ長} - 56$$

（ai：各列の長さ，n：上記列の個数）

- FIX 指定のある表（表定義時）

$$\sum_{i=1}^{n} ai < \left\downarrow \frac{\text{格納するRDエリアのページ長}}{1000} \right\downarrow \times 1000$$

（ai：各列の長さ，n：上記列の個数）

表 3-35　既定義型データ長一覧

分類	データ型及び条件	データ長（単位：バイト）
数値データ	INTEGER	4
	SMALLINT	2
	LARGE DECIMAL(m,n)※1	↓m÷2↓ ＋1※2
	FLOAT 又は DOUBLE PRECISION	8

分類	データ型及び条件			データ長（単位：バイト）
	SMALLFLT 又は REAL			4
文字データ	CHARACTER(n)			n※3
	VARCHAR(n)	d≦255	繰返し列の要素	d＋2
			上記以外	d＋1
		d≧256		6
	VARCHAR(n) ノースプリットオプション指定あり	n≦255	抽象データ型の属性	d＋3
			繰返し列の要素	d＋2
			上記以外	d＋1
		n≧256		6
各国文字データ	NCHAR(n)又は NATIONAL CHARACTER(n)			2×n ※4
	NVARCHAR(n)	d≦127	繰返し列の要素	2×d＋2
			上記以外	2×d＋1
		d≧128		6
	NVARCHAR(n) ノースプリットオプション指定あり	n≦127	抽象データ型の属性	2×d＋3
			繰返し列の要素	2×d＋2
			上記以外	2×d＋1
		n≧128		6
混在文字データ	MCHAR(n)			n※3
	MVARCHAR(n)	d≦255	繰返し列の要素	d＋2
			上記以外	d＋1
		d≧256		6
	MVARCHAR(n) ノースプリットオプション指定あり	n≦255	抽象データ型の属性	d＋3
			繰返し列の要素	d＋2
			それ以外	d＋1
		n≧256		6
日付データ	DATE			4
時刻データ	TIME			3
時刻印データ	TIMESTAMP(n)			7＋（n÷2）
日間隔データ	INTERVAL YEAR TO DAY			5
時間隔データ	INTERVAL HOUR TO SECOND			4

分類	データ型及び条件		データ長（単位：バイト）
長大データ	BLOB		9
バイナリデータ	BINARY(n)	n≦255	d＋3
		n≧256	15

（凡例）

d：実際のデータ長（文字数）

m，n：正の整数

注※1

全体のけた数が m けたで，小数点以下のけた数が n けたの固定小数点数です。m を省略した場合は 15 を仮定します。

注※2

表定義時に表オプションに SUPPRESS DECIMAL を指定した場合，データ長は「↓k÷2↓＋2」になります。k は，格納時の有効けた数（先頭の 0 の部分を除いたけた数）を示します。なお，次に示す場合は SUPPRESS DECIMAL を使用しないでください。計算式中の a は，SUPPRESS DECIMAL 又は列データ抑制指定を使用しない場合の，表中の列のデータ長の合計値です。

32717 ＜（a ＋表中の列数×2 ＋ 8）

注※3

列データ抑制指定をして，データ抑制された場合，n は「n－b ＋ 4」になります。なお，データ抑制は，列データ抑制指定時，列データの最後の文字が空白の場合，その最後の文字と連続している半角の空白が 4 文字以上あるときだけ実行されます。b は，列データの最後の文字と連続している空白の数を示します。

ただし，列データ抑制指定をして，データ抑制されなかった場合は，列ごとに 1 バイトの付加情報が追加されます。

なお，次に示す場合は列データ抑制指定を使用しないでください。計算式中の a は，SUPPRESS DECIMAL 又は列データ抑制指定を使用しない場合の，表中の列のデータ長の合計値です。

32717 ＜（a ＋表中の列数×2 ＋ 8）

注※4

列データ抑制指定をして，データ抑制された場合，2×n は「2×n－2×b ＋ 5」になります。なお，データ抑制は，列データ抑制指定時，列データの最後の文字が空白の場合，その最後の文字と連続している全角の空白が 3 文字以上あるときだけ実行されます。b は，列データの最後の文字と連続している空白の数を示します。

ただし，列データ抑制指定をして，データ抑制されなかった場合は，列ごとに 1 バイトの付加情報が追加されます。

なお，次に示す場合は列データ抑制指定を使用しないでください。計算式中の a は，SUPPRESS DECIMAL 又は列データ抑制指定を使用しない場合の，表中の列のデータ長の合計値です。

32717 ＜（a ＋表中の列数×2 ＋ 8）

(6) 参照制約に関する規則

1. 同じ外部キー構成列（並びが同じでなくてもよい）の外部キーから，同じ被参照表を参照することはできません。
2. WITHOUT ROLLBACK を指定して定義した表には外部キーを指定できません。
3. WITHOUT ROLLBACK を指定して定義した表に主キーがある場合，その主キーを参照する外部キーを定義できません。
4. 共用表には外部キーを指定できません。
5. 改竄防止表には外部キーを指定できません。
6. 外部キーは一つの表に最大 255 個定義できます。
7. 一つの主キーには最大 255 個外部キーが定義できます。
8. 外部キーの文字集合と，その外部キーから参照される表の主キーの文字集合は同じにしてください。
9. 一時表には外部キーを指定できません。
10. 被参照表が一時表である参照表には，外部キーを指定できません。

(7) 検査制約に関する規則

1. 表中に定義できる検査制約は最大 254 個です。一つの表で検査制約中で指定した論理演算子の数（CASE 式の WHEN 探索条件中の AND，OR を除く，AND 及び OR の数）と検査制約の数の合計（検査制約制限値と呼ぶ）も最大 254 個です。
2. 複数の条件を指定する場合は，複数の条件をまとめて一つの検査制約を定義しないで，複数に分けて検査制約を定義することを推奨します。これによって，制約違反となった場合に，違反となった条件を制約名から容易に判別できます。
3. 改竄防止表には検査制約定義を指定できません。
4. 検査制約の探索条件に BLOB 型を指定する場合，又は定義長が 32,001 バイト以上の BINARY 型の列を指定する場合，次の SQL を実行できません。
   - 検査制約の探索条件に指定した BLOB 型，又は定義長が 32,001 バイト以上の BINARY 型の列に対する UPDATE 文での連結演算による更新
5. 一時表には検査制約を定義できません。

(8) 留意事項

1. CREATE TABLE は，OLTP 下の X/Open に従った UAP から指定できません。
2. 表の未使用領域の比率，及びセグメント内空きページ比率の両方に 0%を指定する場合，PCTFREE = （0，0）を指定してください。
3. 行単位インタフェースを使用して，日付データを CHAR（10）で受け渡しする場合は，日付けデータ型を使用しないで，CHAR（10）で列を定義してください。

4. 行単位インタフェースを使用して，時刻データを CHAR（8）で受け渡しする場合は，時刻データ型を使用しないで，CHAR（8）で列を定義してください。

5. 行単位インタフェースを使用して，時刻印データを 19，22，24，又は 26 バイトの CHAR で受け渡しをする場合，時刻印データ型を使用しないで 19，22，24，又は 26 バイトの CHAR で列を指定してください。

6. 改竄防止表を定義する場合，表格納用 RD エリアには改竄防止表だけを格納することをお勧めします。改竄防止表に対する pdrorg が異常終了した場合，再編成が完了するまでその RD エリアは閉塞が解除できません。そのため，改竄防止表が格納されている RD エリアにほかの表，及びインデクスがあると，それらも使用できなくなります。

7. 参照動作に CASCADE を指定した参照表に対して表定義変更などを行った場合，HiRDB が生成した制約操作を行うためのトリガが無効になる場合があります。次にトリガが無効になる条件を示します。

   - 参照制約動作指定が ON UPDATE CASCADE の場合に生成するトリガが，無効となる条件
     ・参照表に対して表定義変更（列に対する SPLIT の変更，列に対する既定値の変更）を行った場合
     ・参照表に対してインデクスを定義した場合
     ・参照表のインデクスを削除した場合
     ・参照表に対して，トリガ契機が UPDATE のトリガを定義した場合
     ・参照表に定義したトリガ契機が UPDATE のトリガを削除した場合
     ・参照表が参照する被参照表の主キー構成列に対して，表定義変更を行った場合
   - 参照制約動作指定が ON DELETE CASCADE の場合に生成するトリガが，無効となる条件
     ・参照表に対して表定義変更（列に対する SPLIT の変更，列に対する既定値の変更）を行った場合
     ・参照表に対してインデクスを定義した場合
     ・参照表のインデクスを削除した場合
     ・参照表に対して，トリガ契機が DELETE のトリガを定義した場合
     ・参照表に定義したトリガ契機が DELETE のトリガを削除した場合

   無効になったトリガは，ALTER ROUTINE を使用して再作成してください。

8. 参照動作として ON UPDATE CASCADE を指定した参照制約を複数指定する場合は，被参照表に同一表名を指定しないでください。

   ただし，以下に示す条件をすべて満たす場合を除きます。

   - 該当する複数の参照指定の外部キー構成列が重複していない。
   - 該当する複数の参照指定の外部キー構成列に関連する検査制約，参照制約を定義していない。

(9) 使用例

1. 在庫表（ZAIKO）を定義します。

```
CREATE TABLE ZAIKO
    (SCODE CHAR(4),SNAME NCHAR(8),
    COL NCHAR(1),TANKA INTEGER,ZSURYO INTEGER)
```

2. 在庫表（ZAIKO）を次に示す条件で定義します。

- FIX 表にします。
- 表データはユーザ用 RD エリア（RDA1）に格納します。
- 在庫表はクラスタキーのない FIX 表のため，未使用領域の比率，及びセグメント内の空きページ比率を 0%に指定します。

```
CREATE FIX TABLE ZAIKO
   (SCODE CHAR(4),SNAME NCHAR(8),
   COL NCHAR(1),TANKA INTEGER,ZSURYO INTEGER)
 IN RDA1
 PCTFREE=(0,0)
```

3. 在庫表（ZAIKO）を次に示す条件で定義します。

- 商品コード（SCODE）列を一意性制約のあるクラスタキーとして定義します。
- 表データ，及びインデクスをそれぞれ三つの RD エリアに分割して格納します。格納条件は次のとおりとします。

格納条件	格納する RD エリア	
	表データ	インデクス
101M ≦ SCODE ≦ 202M	RDA1	RDA4
302S ≦ SCODE ≦ 412M	RDA2	RDA5
591L ≦ SCODE ≦ 591S	RDA3	RDA6

```
CREATE TABLE ZAIKO
    (SCODE CHAR(4)
        UNIQUE CLUSTER KEY ASC
        IN ((RDA4),(RDA5),(RDA6)),
    SNAME NCHAR(8),
    COL   NCHAR(1),
    TANKA INTEGER,
    ZSURYO INTEGER)
  IN ((RDA1)SCODE<='202M',
      (RDA2)SCODE<='412M',
      (RDA3))
```

4. 在庫表（ZAIKO）を次に示す条件で定義します。

- 商品名（SNAME）列，及び色（COL）列をクラスタキーとして定義します。
- クラスタキーのインデクスは，商品名の昇順，色の降順にします。
- 表データは，ユーザ用 RD エリア（RDA1）に格納します。
- インデクスは，ユーザ用 RD エリア（RDA2）に格納します。

```
CREATE TABLE ZAIKO
    (SCODE CHAR(4),SNAME NCHAR(8),
    COL NCHAR(1),TANKA INTEGER,ZSURYO INTEGER)
```

```
 IN RDA1
 CLUSTER KEY (SNAME ASC,COL DESC) IN RDA2
```

5. 抽象データ型 t_従業員を含む社員表を定義します。

```
CREATE TABLE 社員表
    (社員No INTEGER NOT NULL,
     文書データ BLOB (6000) IN ((LRDA1),(LRDA2)),
     従業員 t_従業員 ALLOCATE(顔写真
       IN ((LRDA03),(LRDA04))))
       IN ((RDA1)社員No<=700000,(RDA2))
```

6. 受注表（JUTYU）を，次の条件で定義します。

- 改竄防止表とする。
- 行削除禁止期間を 10 年とする。
- 挿入履歴保持列として，JINSERTDATE 列を定義する。

```
CREATE TABLE JUTYU
    (DNO CHAR(6),TCODE CHAR(5),SCODE CHAR(4),
     JSURYO INTEGER,JDATE DATE,JTIME TIME,
     JINSERTDATE DATE NOT NULL WITH DEFAULT SYSTEM GENERATED)
 INSERT ONLY WHILE 10 YEARS BY JINSERTDATE
```

7. 在庫表（ZAIKO）を次に示す条件で定義します。

- マトリクス分割表として，6 個のユーザ用 RD エリアに分割して格納します。
- 格納条件を次に示します。

格納条件	格納する RD エリア
SCODE≦202M AND TANKA≦5000	RDA1
SCODE≦202M AND TANKA > 5000	RDA2
202M < SCODE≦412M AND TANKA≦5000	RDA3
202M < SCODE≦412M AND TANKA > 5000	RDA4
SCODE > 412M AND TANKA≦5000	RDA5
SCODE > 412M AND TANKA > 5000	RDA6

```
CREATE TABLE ZAIKO
    (SCODE CHAR(4) NOT NULL,
     TANKA INTEGER NOT NULL)
  PARTITIONED BY MULTIDIM (SCODE(('202M'),('412M')),
                           TANKA((5000)))
  IN ((RDA1,RDA2),(RDA3,RDA4),(RDA5,RDA6))
```

8. 在庫表（ZAIKO）を次に示す条件で定義します。

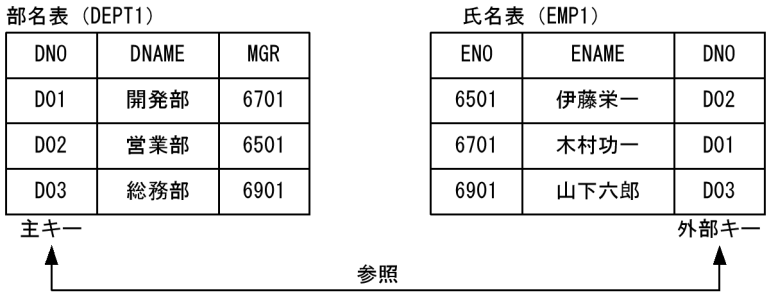
- 商品コード（SCODE）列にサイズが S，M，L 以外の商品で挿入・更新できないように検査制約を定義します。

```
  CREATE TABLE ZAIKO
      (SCODE CHAR(4) CONSTRAINT CHECK_SIZE
CHECK(SCODE LIKE '%S' OR SCODE LIKE '%M' OR SCODE LIKE '%L'),
SNAME NCHAR(8),
      COL NCHAR(1),TANKA INTEGER,ZSURYO INTEGER)
```

9. 単一の列に参照制約を定義します。

- 部名表の行を削除した場合，名前表に対応する行も削除します。

```
被参照表（部名表（DEPT1））を定義し，主キーを部コード（DNO）列にします。
  CREATE TABLE DEPT1
      (DNO CHAR(3) PRIMARY KEY,DNAME NVARCHAR(20),MGR CHAR(8))

参照表（氏名表（EMP1））を定義し，外部キーを部コード（DNO）列にします。
  CREATE TABLE EMP1
      (ENO CHAR(8),ENAME NVARCHAR(25),
       DNO CHAR(3)
           CONSTRAINT EMP1_K REFERENCES DEPT1 ON DELETE CASCADE)
```



10. 複数の列に参照制約を定義します。

- 課名表の行を削除した場合，名前表の行に，削除する課名表の行と同じ値がある場合，課名表の削除を抑止します。また，課名表の行を更新する場合，名前表の行に，更新する課名表の行と同じ値がある場合，課名表の更新を抑止します。

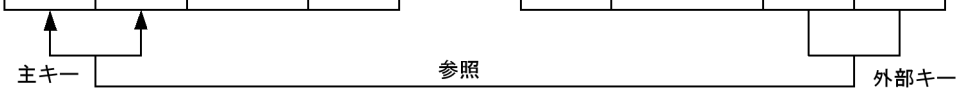
```
被参照表（課名表（DEPT2））を定義し，主キーを部コード（DNO）列，課コード（SNO）列にします。
  CREATE TABLE DEPT2
      (DNO CHAR(3),SNO CHAR(3),SNAME CHAR(20),CHIEF CHAR(8),
      PRIMARY KEY(DNO,SNO))

参照表（氏名表（EMP2））を定義し，外部キーを部コード（DNO）列，課コード（SNO）列にします。
  CREATE TABLE EMP2
      (ENO CHAR(8),ENAME CHAR(25),DNO CHAR(3),SNO CHAR(3),
      CONSTRAINT EMP2_K FOREIGN KEY(DNO,SNO) REFERENCES DEPT2)
```

部名表（DEPT2）

DNO	SNO	SNAME	CHIEF
D01	S11	開発課	6701
D02	S21	営業一課	6501
D03	S31	総務課	6901

氏名表（EMP2）

ENO	ENAME	DNO	SNO
6501	佐藤勝	D01	S11
6702	高橋功	D03	S31
6903	南悟	D02	S21



3.18　CREATE TRIGGER（トリガ定義）

3.18.1　CREATE TRIGGER の形式と規則

(1)　機能

指定した表への操作（INSERT 文，UPDATE 文，及び DELETE 文）を契機に，自動的に SQL を実行する動作（トリガ）を定義します。

(2)　使用権限

自分が所有する表に対してトリガを定義できます。

(3)　形式

```
  CREATE TRIGGER 〔認可識別子.〕トリガ識別子
        トリガ動作時期
        トリガ契機
        ON 〔認可識別子.〕表識別子
         〔REFERENCING 新旧値別名リスト〕
        トリガ動作
         〔SQLコンパイルオプション〔SQLコンパイルオプション〕…〕
         〔WITH PROGRAM〕

  トリガ動作時期：：＝ {BEFORE｜AFTER}
  トリガ契機：：＝ {INSERT｜DELETE｜UPDATE〔OF 列名〔, 列名〕…〕}
  新旧値別名リスト：：＝新旧値別名〔新旧値別名〕
  新旧値別名：：＝ {OLD 〔ROW〕 〔AS〕 旧値相関名
                   ｜NEW 〔ROW〕 〔AS〕 新値相関名}
  旧値相関名，新値相関名：：＝相関名
  トリガ動作：：＝〔 {FOR EACH ROW｜FOR EACH STATEMENT}〕
                 〔WHEN （探索条件）〕
                トリガSQL文
  トリガSQL文：：＝SQL手続き文
  SQLコンパイルオプション：：＝ {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE｜COMPAT
IBLE}〕
                               ｜OPTIMIZE LEVEL SQL最適化オプション
                                          〔, SQL最適化オプション〕…
                               ｜ADD OPTIMIZE LEVEL SQL拡張最適化オプション
                                          〔, SQL拡張最適化オプション〕…
                               ｜SUBSTR LENGTH 文字の最大長 }
```

(4) オペランド

(a) 〔認可識別子.〕トリガ識別子

認可識別子

定義するトリガの所有者の認可識別子を指定します。省略した場合，CREATE TRIGGER を実行するユーザの認可識別子が仮定されます。

トリガ識別子

定義するトリガの名前を指定します。

(b) トリガ動作時期：：＝ {BEFORE | AFTER}

BEFORE

表に対する操作の前に，トリガ動作を実行します。

BEFORE を指定した場合，データを更新する SQL（INSERT 文，UPDATE 文，及び DELETE 文），CALL 文，及びデフォルトコンストラクタ関数以外の関数呼出しは，トリガ SQL 文中には指定できません。

トリガ動作時期に BEFORE を指定したトリガを，**BEFORE トリガ**といいます。

AFTER

表に対する操作の後に，トリガ動作を実行します。

トリガ動作時期に AFTER を指定したトリガを，**AFTER トリガ**といいます。

(c) トリガ契機：：＝ {INSERT | DELETE | UPDATE〔OF 列名〔，列名〕…〕}

どのような操作をしたときにトリガを実行するかを指定します。

INSERT

表に行を挿入したときにトリガを実行します。トリガ契機に INSERT を指定したトリガを INSERT トリガといいます。

トリガ契機に INSERT を指定した場合，新旧値別名に OLD〔ROW〕〔AS〕旧値相関名は指定できません。

DELETE

表の行を削除したときにトリガを実行します。トリガ契機に DELETE を指定したトリガを DELETE トリガといいます。

トリガ契機に DELETE を指定した場合，新旧値別名に NEW〔ROW〕〔AS〕新値相関名は指定できません。

UPDATE

表の行を更新したときにトリガを実行します。トリガ契機に UPDATE を指定したトリガを UPDATE トリガといいます。

トリガ契機に UPDATE を指定した場合，トリガ動作条件を満たしていれば，更新前後で値が同じでもトリガを実行します。

OF 列名〔，列名〕…

特定の列を更新したときにトリガを実行したい場合，OF 列名〔，列名〕…を指定します。ここで指定した列を，トリガ契機列といいます。

トリガ契機列についての規則を次に示します。

1. 列名には，トリガを定義する表の列名を指定します。
2. 列名は重複して指定できません。
3. 繰返し列を指定した場合，その列に対する ADD 句，又は DELETE 句だけの UPDATE 文でも，トリガを実行します。
4. トリガ契機列を省略した場合，対象表のすべての列名（予備列を除く）が仮定されます（トリガ定義後に追加した列も含まれます)。また，ADD 句，又は DELETE 句だけの UPDATE 文でも，トリガを実行します。
5. 予備列は指定できません。

(d) 〔認可識別子.〕表識別子

トリガを定義する実表の表名を指定します。

トリガは，自分の永続実表にだけ定義できます。ビュー表及び一時表に対してトリガは定義できません。

(e) REFERENCING 新旧値別名リスト：：＝新旧値別名〔新旧値別名〕

トリガ定義中で更新前後の行を参照する場合に，別名を指定します。

新旧値別名には，OLD〔ROW〕〔AS〕旧値相関名，又は NEW〔ROW〕〔AS〕新値相関名を，それぞれ 1 回だけ指定できます。

(f) 新旧値別名：：＝ {OLD 〔ROW〕 〔AS〕 旧値相関名｜NEW 〔ROW〕 〔AS〕 新値相関名}

旧値相関名：：＝相関名

新値相関名：：＝相関名

更新前，又は更新後の行に名前を付けて参照する場合に指定します。

新旧値別名についての規則を次に示します。

1. ROW 及び AS は，指定してもしなくても同じです。
2. 旧値相関名と新値相関名には，同じ相関名は指定できません。
3. 旧値相関名，又は新値相関名に指定した相関名の有効範囲は，トリガ定義全体になります。

4. BEFORE トリガで新値相関名で修飾した列を更新した場合は，その更新内容が表に反映されます。ただし，BEFORE トリガでは，次の列を新値相関名で修飾した更新はできません。更新した場合は実行時エラーとなります。

   ・SYSTEM GENERATED を指定した列

   また，トリガ契機が INSERT の BEFORE トリガでは，次の列を新値相関名で修飾した更新はできません。更新した場合は実行時エラーとなります。

   ・分割表（フレキシブルハッシュ分割表を除く）の分割キーに指定した列

   BEFORE トリガの場合，新値相関名で修飾して更新するときでも，更新する前の挿入値（トリガ契機が INSERT の場合），又は更新値（トリガ契機が UPDATE の場合）は，指定した列に挿入又は更新できる値でなければなりません。例えば，非ナル値制約ありの列には，トリガ動作で更新する前の挿入値又は更新値として NULL は指定できません。ただし，一意性制約については，トリガ動作で更新した後の値が一意性制約を満たしていれば挿入又は更新ができます。

5. 繰返し列及び抽象データ型列は，新値相関名，及び旧値相関名で修飾して参照できません（トリガ定義中で，トリガを定義する表中の繰返し列及び抽象データ型列は参照できません）。

6. 旧値相関名，及び新値相関名には，ROW は指定できません。

7. 予備列は，新値相関名，及び旧値相関名で修飾して参照できません。

OLD 〔ROW〕 〔AS〕 旧値相関名

更新前の行に名前を付けて参照する場合に指定します。

旧値相関名で修飾した列が保持する値は，トリガの契機となる SQL の実行前の値です。UPDATE 文の場合は更新前の値，DELETE 文の場合は削除する行の列値となります。

NEW 〔ROW〕 〔AS〕 新値相関名

更新後の行に名前を付けて参照する場合に指定します。

新値相関名で修飾した列が保持する値は，トリガの契機となる SQL の実行結果の値です。UPDATE 文の場合は更新後の値，INSERT 文の場合は挿入値となります。ただし，トリガ中で新値相関名で修飾した列を更新した場合は，更新した値を引き継ぎます。

(g) トリガ動作：：＝〔{FOR EACH ROW ｜ FOR EACH STATEMENT}〕〔WHEN (探索条件)〕トリガ SQL 文

FOR EACH ROW

トリガを，更新した行単位に実行する場合に指定します。トリガ動作に FOR EACH ROW を指定したトリガを，行単位トリガといいます。

FOR EACH ROW を指定した場合，表を 1 行更新するごとにトリガを実行します。

FOR EACH STATEMENT

トリガを，SQL 文単位に実行する場合に指定します。トリガ動作に FOR EACH STATEMENT を指定したトリガを，文単位トリガといいます。

FOR EACH STATEMENT を指定した場合，一つの SQL 文ごとにトリガを実行します。更新対象の行がない場合でもトリガを実行します。

FOR EACH STATEMENT を指定した場合，新旧値別名リストは指定できません。

探索条件

トリガ契機が発生したとき，ここに指定した条件が真になる場合にトリガ SQL 文を実行します。この探索条件を，トリガ動作条件といいます。

トリガ動作条件を省略した場合，指定したトリガ契機が発生すると必ずトリガ SQL 文を実行します。

トリガ動作条件には，次のものは指定できません。

- 副問合せ
- 集合関数又は SQL/XML 集合関数
- ウィンドウ関数
- 繰返し列
- 埋込み変数，?パラメタ，SQL 変数，及び SQL パラメタ
- 次のデータ型の列

  ・BLOB（ただし，スカラ関数 LENGTH，スカラ関数 POSITION，及びユーザ定義関数の関数呼出しの中では指定できます）

  ・最大長 32,001 バイト以上の BINARY（ただし，スカラ関数 LENGTH，スカラ関数 POSITION，及びユーザ定義関数の関数呼出しの中では指定できます）

  ・抽象データ型
- 構造化繰返し述語
- プラグイン関数
- 結果が抽象データ型となる値式（値式中にも指定できません）
- XML コンストラクタ関数
- SQL/XML スカラ関数
- 予備列

トリガ動作条件中でトリガを定義する表の列を参照する場合は，新旧値相関名で修飾してください（修飾なし，又は表名での修飾は指定できません）。

トリガ SQL 文

SQL 手続き文を指定します。SQL 手続き文については，「ルーチン制御 SQL」を参照してください。

トリガ SQL 文として指定する SQL 手続き文には，次の制限があります。

1. トリガを定義している表の表名は指定できません。
2. トリガを定義している表の列を，修飾なし，又は表名で修飾して指定することはできません（新旧値別名で修飾した列は指定できます）。列を新旧値相関名で修飾した場合，トリガ SQL 文中でこの列指定を指定できる箇所は，SQL 文中で SQL パラメタを指定できる箇所と同じになります。SQL パラメタを指定できる箇所については，「埋込み変数，標識変数，?パラメタ，SQL パラメタ，及び SQL 変数」を参照してください。ただし，LIMIT 句にはこの列指定は指定できません。

3. ROLLBACK 文，COMMIT 文，及び PURGE TABLE 文は指定できません。また，CALL 文でこれらを指定した手続きが呼び出された場合は，実行時エラーとなります。
4. JAVA ストアドプロシジャ，及び GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。また，CALL 文で JAVA ストアドプロシジャを指定した手続きが呼び出された場合は，実行時エラーとなります。
5. トリガ SQL 文中で，新値相関名，又は旧値相関名で修飾して指定する列の個数は 30,000 以下にしてください。

(h) SQL コンパイルオプション

```
::= {ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE} 〕
      | OPTIMIZE LEVEL SQL最適化オプション〔, SQL最適化オプション〕…
      | ADD OPTIMIZE LEVEL SQL拡張最適化オプション〔, SQL拡張最適化オプション〕…
      | SUBSTR LENGTH 文字の最大長 }
```

SQL コンパイルオプションには ISOLATION，OPTIMIZE LEVEL，ADD OPTIMIZE LEVEL，SUBSTR LENGTH をそれぞれ 1 回しか指定できません。

〔ISOLATION データ保証レベル 〔FOR UPDATE {EXCLUSIVE | COMPATIBLE}〕〕

SQL のデータ保証レベルを指定します。

データ保証レベル

データ保証レベルとは，トランザクションのどの時点までデータの内容を保証するかのレベルをいいます。次に示すデータ保証レベルを指定できます。

- 0

  データの内容を保証しない場合に指定します。0 レベルを指定すると，ほかのユーザが更新中のデータでも，更新完了を待たないで参照できます。ただし，参照する表が共用表の場合，ほかのユーザが排他モードで LOCK 文を実行しているときには，排他解除待ちとなります。

- 1

  検索処理の終了までデータの内容を保証したい場合に指定します。1 レベルを指定すると，検索処理が終了するまで（HiRDB がページ，又は行を見終わるまで）1 度検索したデータをほかのユーザは更新できません。

- 2

  トランザクションの終了まで 1 度検索したデータの内容を保証したい場合に指定します。2 レベルを指定すると，トランザクションが終了するまで 1 度検索したデータをほかのユーザは更新できません。

〔FOR UPDATE {EXCLUSIVE | COMPATIBLE}〕

トリガ中で FOR UPDATE 句を指定した（FOR UPDATE が仮定される場合を含む）カーソル又は問合せに対して，SQL コンパイルオプションで指定したデータ保証レベルに関係なく，常に WITH EXCLUSIVE LOCK を仮定する場合に指定します。

このオペランドを省略した場合，EXCLUSIVE を仮定します。ただし，0904 互換モードを適用している場合は COMPATIBLE を仮定します。

バージョン 09-50 より前の HiRDB で FOR UPDATE EXCLUSIVE を省略したトリガと同じ動作をさせたい場合は COMPATIBLE を指定します。

バージョン 09-50 より前の HiRDB から，09-50 以降の HiRDB にバージョンアップする場合の注意事項を次に示します。

- トリガを再定義する場合で，09-50 より前の HiRDB で FOR UPDATE EXCLUSIVE を省略したトリガと同じ動作をさせたい場合は，次のどちらかの対処をしてください。

  ・ALTER TRIGGER 又は ALTER ROUTINE でトリガの SQL オブジェクトを再作成してください。

  ・トリガを削除し，CREATE TRIGGER でトリガを再定義する場合は，COMPATIBLE を指定して再定義してください。

このオペランドと，ISOLATION データ保証レベルの関係から決まる FOR UPDATE の排他オプションを次に示します。

ISOLATION データ保証レベル	FOR UPDATE EXCLUSVIE	FOR UPDATE COMPATIBLE
0	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
1	WITH EXCLUSIVE LOCK	WITHOUT LOCK WAIT
2	WITH EXCLUSIVE LOCK	WITH EXCLUSIVE LOCK

《クライアント環境定義との関係》

CREATE TRIGGER に対して，PDISLLVL，PDFORUPDATEEXLOCK の指定は無効となります。

《SQL との関係》

手続き中の SQL 文に排他オプションを指定している場合は SQL コンパイルオプションで指定するデータ保証レベル，FOR UPDATE EXCLUSIVE，及び FOR UPDATE COMPATIBLE から仮定する排他オプションより SQL 文に指定した排他オプションが優先されます。

《システム定義との関係》

データ保証レベルを省略すると，pd_isolation_level オペランドの指定値が仮定されます。
pd_isolation_level オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

データ保証レベルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

〔OPTIMIZE LEVEL SQL 最適化オプション〔，SQL 最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDSQLOPTLVL」を参照してください。

SQL 最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法がありますが，通常時は識別子で指定する方法をお勧めします。

識別子で指定する場合：

```
OPTIMIZE LEVEL ”識別子”〔, ”識別子”〕…
```

＜指定例＞

- ネストループジョイン優先とグループ分け高速化処理を適用する場合
  OPTIMIZE LEVEL "PRIOR_NEST_JOIN","RAPID_GROUPING"
- すべての最適化を適用しない場合
  OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。ただし，同時に"NONE"以外の識別子を指定すると，"NONE"は無効になります。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
OPTIMIZE LEVEL 符号なし整数〔, 符号なし整数〕…
```

＜指定例＞

- 複数の SQL オブジェクト作成，AND の複数インデクス利用の抑止，及び複数インデクス利用の強制を適用する場合
  符号なし整数をコンマで区切って指定する場合：
  OPTIMIZE LEVEL 4,10,16
  符号なし整数の和を指定する場合：
  OPTIMIZE LEVEL 30
- 既に 14（4+10）を指定していて，新たに 16 を追加する場合
  OPTIMIZE LEVEL 14,16
- すべての最適化を適用しない場合
  OPTIMIZE LEVEL 0

＜規則＞

1. バージョン 06-00 より前の HiRDB から，バージョン 06-00 以降の HiRDB にバージョンアップする場合，バージョン 06-00 より前の合計値指定も有効となります。最適化オプションを変更する必要がない場合は，バージョン 06-00 以降の HiRDB にバージョンアップしたときにこのオペランドの指定値を変更する必要はありません。
2. 符号なし整数は一つ以上指定してください。

3. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。

4. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。ただし，同時に 0 以外の識別子を指定すると，0 は無効になります。

5. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。

6. 複数の最適化方法を指定する場合，その符号なし整数の和を指定することもできます。ただし，同じ最適化方法の値は二つ以上足さないでください（足した結果が別の最適化方法とみなされることもあるため）。

7. 複数の最適化方法の値を足して指定する場合，どの最適方法を指定しているのか分かりにくくなるため，コンマで区切って指定する方法をお勧めします。また，既に複数の最適化方法の値を足して指定している場合で，新たに別の最適化方法が必要になったときは，追加する値をコンマで区切って後ろに指定できます。

《システム定義との関係》

1. SQL 最適化オプションを省略すると，システム定義の pd_optimize_level オペランドの指定値が仮定されます。pd_optimize_level オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

2. システム定義の pd_floatable_bes オペランド，又は pd_non_floatable_bes オペランドを指定している場合，「フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）」及び「フロータブルサーバ対象限定（データ取り出しバックエンドサーバ）」の指定は無効となります。

3. システム定義の pd_indexlock_mode オペランドに KEY を指定している場合（インデクスキー値排他の場合），「更新 SQL の作業表作成抑止」の指定は無効になります。

《クライアント環境定義との関係》

CREATE TRIGGER に対して，PDSQLOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔ADD OPTIMIZE LEVEL SQL 拡張最適化オプション〔，SQL 拡張最適化オプション〕…〕

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための最適化の方法を指定します。

SQL 拡張最適化オプションの指定値及びその内容については，マニュアル「HiRDB UAP 開発ガイド」の「PDADDITIONALOPTLVL」を参照してください。

SQL 拡張最適化オプションは，識別子（文字列）で指定する方法と，数値で指定する方法があります。

識別子で指定する場合：

```
ADD OPTIMIZE LEVEL ”識別子”〔,”識別子”〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL "COST_BASE_2","APPLY_HASH_JOIN"

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL "NONE"

＜規則＞

1. 識別子は一つ以上指定してください。
2. 識別子を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，識別子に"NONE"を指定してください。
4. 識別子は大文字及び小文字で指定できます。
5. 同じ識別子を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ識別子は指定しないようにしてください。

数値で指定する場合：

```
ADD OPTIMIZE LEVEL 符号なし整数〔,符号なし整数〕…
```

＜指定例＞

- 「コストベース最適化モード 2 の適用」及び「ハッシュジョイン，副問合せのハッシュ実行」を適用する場合

  ADD OPTIMIZE LEVEL 1,2

- すべての最適化を適用しない場合

  ADD OPTIMIZE LEVEL 0

＜規則＞

1. 符号なし整数は一つ以上指定してください。
2. 符号なし整数を二つ以上指定する場合は，コンマ（,）で区切ってください。
3. すべての最適化を適用しない場合は，符号なし整数に 0 を指定してください。
4. 同じ符号なし整数を二つ以上指定しても，一つ指定したものとみなされますが，なるべく同じ符号なし整数は指定しないようにしてください。

《システム定義との関係》

SQL 拡張最適化オプションを省略すると，システム定義の pd_additional_optimize_level オペランドの指定値が仮定されます。pd_additional_optimize_level オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

《クライアント環境定義との関係》

CREATE TRIGGER に対して，PDADDITIONALOPTLVL の指定は無効となります。

《SQL との関係》

SQL 文中に SQL 最適化指定を指定している場合は，SQL 最適化オプションよりも SQL 最適化指定が優先されます。SQL 最適化指定については，「SQL 最適化指定」を参照してください。

〔SUBSTR LENGTH 文字の最大長〕

1 文字を表現する最大バイト数を指定します。

文字の最大長に指定できる値は，3～6（pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8_ivs を指定した場合は 3～10）です。

pdntenv コマンド（UNIX 版の場合は pdsetup コマンド）で文字コード種別に utf-8，又は utf-8_ivs を指定した場合にだけ有効となり，スカラ関数 SUBSTR の結果の長さに影響します。SUBSTR については，「SUBSTR」を参照してください。

《システム定義との関係》

SUBSTR LENGTH を省略すると，システム定義の pd_substr_length オペランドの指定値が仮定されます。pd_substr_length オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

《クライアント環境定義との関係》

CREATE TRIGGER に対して，PDSUBSTRLEN の指定は無効となります。PDSUBSTRLEN については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

《pdntenv コマンド又は pdsetup コマンドで指定した文字コード種別との関係》

文字コード種別に utf-8，又は utf-8_ivs を指定した場合だけ有効となります。

そのほかの文字コード種別の場合は，構文チェックだけ行い，指定を無視します。

(i) WITH PROGRAM

トリガ定義時に，そのトリガを定義する表を使用し，次のどれかの条件を満たす手続き，及びトリガの有効な SQL オブジェクトがあれば，その SQL オブジェクトを無効にする場合に指定します。

- トリガ契機が INSERT の場合，トリガを定義する表に対して行を挿入する手続き，及びトリガ
- トリガ契機が UPDATE の場合，トリガを定義する表に対して行を更新する手続き，及びトリガ
- トリガ契機が DELETE の場合，トリガを定義する表に対して行を削除する手続き，及びトリガ

バージョン 07-00 より前の HiRDB で SQL オブジェクトを作成した関数及び手続きは，上記の条件を満たしていなくてもすべて無効にします。

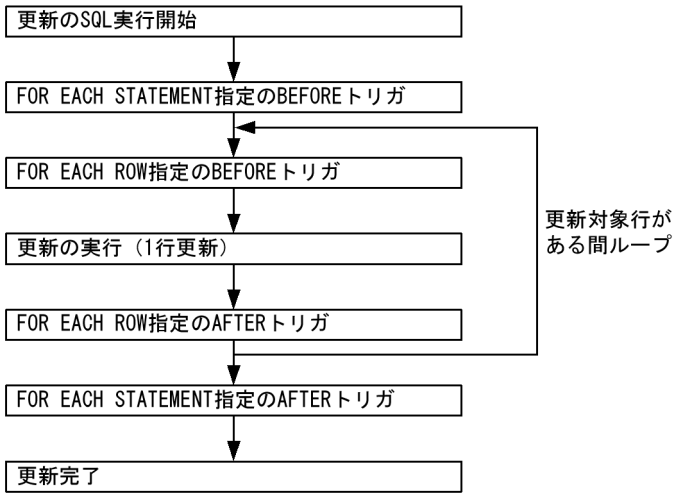
(5) 共通規則

1. トリガを定義すると，トリガ動作手続きと呼ばれる手続きの SQL オブジェクトが作成され，データディクショナリ LOB 用 RD エリアに格納されます。このため，トリガを定義するためには，あらかじめデータディクショナリ LOB 用 RD エリアに十分な容量を確保しておく必要があります。データディクショナリ LOB 用 RD エリアの容量の見積もりについては，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

2. トリガ動作手続きのルーチン識別子は，次のような名称となります。なお，長さは 22 バイトです。
   '(TRIGyyyymmddhhmmssth)'
   yyyymmddhhmmssth：トリガ定義時のタイムスタンプ（100 分の 1 秒単位）
3. トリガ動作手続きの特定名は，〔認可識別子.〕トリガ動作手続きのルーチン識別子と同じです。トリガ動作手続きのルーチン識別子は，ディクショナリ表 SQL_TRIGGERS の SPECIFIC_NAME 列に格納されます。
4. トリガを定義するユーザは，トリガ SQL 文を実行するために必要な権限を保持している必要があります。トリガを定義したユーザがその権限を保持しているかどうかは，トリガの定義時及び実行時にチェックされます。
5. WITH PROGRAM を省略した場合，トリガを定義する表を使用し，SQL オブジェクトを変更する必要のある手続き，及びトリガの有効な SQL オブジェクトがあれば，そのトリガは定義できません。
6. ALTER TRIGGER 又は ALTER ROUTINE で SQL コンパイルオプションを指定する場合，SQL オブジェクトを再作成するトリガの元の CREATE TRIGGER に，SQL コンパイルオプションを反映してできる SQL は，SQL の最大長を超えないようにしてください。
7. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から CREATE TRIGGER は実行できません。

(6) トリガを引き起こす SQL を実行するときの規則

1. トリガは，指定したトリガ契機（INSERT 文，UPDATE 文，又は DELETE 文）でだけ発生します。PURGE TABLE 文，データベース作成ユティリティ，データベース再編成ユティリティ，及び RD エリアの再初期化などでは動作しません。
2. トリガの実行順序を次に示します。



3. トリガ動作時期，トリガ契機，及びトリガの動作単位（文単位，又は行単位）が同じトリガを複数定義している場合は，定義した順番（SQL_TRIGGERS 表の CREATE_TIME 列の値の順番）にトリガ動作を実行します。
4. トリガ実行時にエラーが発生した場合，そのトランザクションは無効となります（暗黙的ロールバックを引き起こします）。また，トリガを引き起こす SQL の実行時，1 回でもトリガが実行された後でエ

ラーが発生すると，そのエラーがトリガ実行時かどうか，又はトランザクションを無効にするかどうかに関係なく，そのトランザクションを無効にします。ただし，WITHOUT ROLLBACK を指定して定義した表については，行更新（追加，削除を含む）の処理完了後はロールバックしない場合があります。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。

5. トリガを引き起こす SQL を実行するユーザは，トリガ SQL 文を実行するために必要な権限を保持している必要はありません。
6. トリガ SQL 文中の SQL を契機として，ほかのトリガを実行することもできます（トリガのネスト）。ただし，ネストして動作するのは 16 段目のトリガまでで，16 段目のトリガでほかのトリガを引き起こす操作が発生すると，エラーになります。
7. トリガ契機に UPDATE を指定した行単位トリガは，その更新対象の行に対して 1 回だけ実行されます。トリガがネストして起動し，その延長で再度同じトリガが起動する場合でも，実行されるのは最初の 1 回だけです。
8. UPDATE トリガが定義されている表に対して次の指定はできません。
   - 新値相関名を指定した UPDATE トリガを定義した表に対する，コンポネント指定を使用した更新
   - UPDATE トリガを定義した表に対する，連結演算を使用した BLOB 型，又は定義長が 32,001 バイト以上の BINARY 型の更新
9. 行単位 AFTER トリガを引き起こす SQL の副問合せ中で，トリガ動作又はその延長で更新される表を参照する場合，その副問合せには外への参照を含めないようにしてください。

(7) 留意事項

1. CREATE TRIGGER は，OLTP 下の X/Open に従った UAP から指定できません。
2. トリガ SQL 文に更新系 SQL（UPDATE 文，DELETE 文，又は INSERT 文）を指定する場合，指定した更新系 SQL の実行時に次のような SQL エラーになることがあります。
   - 更新系 SQL を指定した FOR EACH ROW のトリガを実行する SQL 中に，更新系 SQL で指定した表が含まれている
   - 更新系 SQL を指定した FOR EACH ROW のトリガを実行する FOR EACH ROW のトリガ定義（複数のトリガ定義がネストする場合も含む）中に，更新系 SQL 中に指定した表が含まれている

   このような場合は，システム定義でインデクスキー値無排他を指定するか，又はトリガ定義を変更してください。
3. トリガ SQL 文に更新系 SQL を指定する場合，指定した更新系 SQL の実行中に SQL の検索に影響を与えることがあります。実行中の SQL の検索に影響させないようにするためには，更新系 SQL が，実行中の SQL の検索条件に合致しないように，探索条件やデータを工夫する必要があります。特に，トリガ定義が入れ子になっている場合に注意してください。
4. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中から無効となった手続き，及びトリガに関する行は削除されます。

5. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して，手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
6. トリガ中のトリガ SQL 文のデータ保証レベル，SQL 最適化オプション，SQL 拡張最適化オプション，及び文字の最大長は，トリガの定義時，又は変更時の指定で決まり，トリガ動作実行時のシステム定義やクライアント環境定義の影響を受けません。
7. ON 〔認可識別子.〕表識別子で指定した表を基表とするビュー表に対しても，トリガ契機に指定した操作を実行すれば，トリガ動作は実行されます。
8. トリガの SQL オブジェクトが無効な場合，そのトリガを定義した表を使用し，次のどれかに該当する SQL を実行すると，トリガ動作条件の真偽に関係なくエラーとなります。また，トリガの SQL オブジェクトのインデクスが無効な場合は，そのトリガを定義した表を使用し，次のどれかに該当する SQL を実行すると，トリガ動作条件が真のときにエラーとなります。
   - トリガ契機が INSERT の場合，トリガを定義した表に対して行を挿入する INSERT 文
   - トリガ契機が UPDATE の場合，トリガを定義した表に対して行を更新する UPDATE 文
   - トリガ契機が DELETE の場合，トリガを定義する表に対して行を削除する DELETE 文
9. トリガがネストしている場合，性能が劣化する場合があります。できるだけネストはしないようにしてください。
10. ネストするトリガを作成する場合，先頭のトリガから作成すると，ネストするトリガ作成時にエラーとなります。また，WITH PROGRAM 指定をすると，ネストするトリガは作成できますが，先頭のトリガが無効状態となります。このため，ネストするトリガを作成する場合は，ネストの末端のトリガから順番に作成してください。

(8) 使用例

CREATE TRIGGER の使用例では，在庫表（ZAIKO）のほかに，在庫履歴表（RZAIKO），広島在庫表，仙台在庫表を使用します。広島在庫表と仙台在庫表は，在庫表と同じ構成です。在庫履歴表の構成は次のとおりです。

在庫履歴表（RZAIKO）

SCODE（品番）	BSURYO（更新前数量）	ASURYO（更新後数量）	RDATE（更新日時）	RTIME（更新時間）
101M	26	10	2003-06-20	10:15:23

1. 在庫表（ZAIKO）に行が挿入された後，挿入された行の情報を在庫履歴表（RZAIKO）に挿入するトリガ（INSERTTRIG1）を定義します。

```
CREATE TRIGGER INSERTTRIG1
  AFTER INSERT ON ZAIKO
  REFERENCING NEW ROW X1
  FOR EACH ROW
  INSERT INTO RZAIKO
    VALUES(X1.SCODE,NULL,X1.ZSURYO,CURRENT_DATE,CURRENT_TIME)
```

2. 在庫表（ZAIKO）の数量（ZSURYO）が更新された後，更新前後の値を在庫履歴表（RZAIKO）に挿入するトリガ（INSERTTRIG2）を定義します。

```
CREATE TRIGGER INSERTTRIG2
  AFTER UPDATE OF ZSURYO ON ZAIKO
  REFERENCING NEW ROW X1 OLD ROW Y1
  FOR EACH ROW
  INSERT INTO RZAIKO
    VALUES(Y1.SCODE,Y1.ZSURYO,X1.ZSURYO,CURRENT_DATE,CURRENT_TIME)
```

3. 在庫表（ZAIKO）から行が削除された後，削除された行の情報を在庫履歴表（RZAIKO）に挿入するトリガ（INSERTTRIG3）を定義します。

```
CREATE TRIGGER INSERTTRIG3
  AFTER DELETE ON ZAIKO
  REFERENCING OLD ROW Y1
  FOR EACH ROW
  INSERT INTO RZAIKO
    VALUES(Y1.SCODE,Y1.ZSURYO,NULL,CURRENT_DATE,CURRENT_TIME)
```

4. トリガ動作にルーチン制御 SQL（複合文）を使用します。在庫表（ZAIKO）が更新された後，その更新内容を広島在庫表，及び仙台在庫表に反映するトリガ（UPDATELOCAL）を定義します。なお，複合文を使用しない場合は，在庫表の更新を契機にしたトリガを二つ定義しなければなりません。

```
CREATE TRIGGER UPDATELOCAL
  AFTER UPDATE OF ZSURYO ON ZAIKO
  REFERENCING NEW ROW X1 OLD ROW Y1
  BEGIN
    UPDATE 広島在庫表 SET ZSURYO=X1.ZSURYO WHERE SCODE=Y1.SCODE;
    UPDATE 仙台在庫表 SET ZSURYO=X1.ZSURYO WHERE SCODE=Y1.SCODE;
  END
```

5. トリガ動作にルーチン制御 SQL（代入文）を使用します。在庫表（ZAIKO）に挿入される行に対し，商品コード（SCODE）が 101M，201M，又は 301M の場合は挿入される単価（TANKA）に 5,000 円を加算した金額を，商品コード（SCODE）が 101M，201M，又は 301M 以外の場合は挿入される単価を 1.2 倍にした金額を新しく単価に設定するトリガ（SETTANKA）を定義します。

```
CREATE TRIGGER SETTANKA
  BEFORE INSERT ON ZAIKO
  REFERENCING NEW ROW AS X1
  FOR EACH ROW
  SET X1.TANKA=CASE X1.SCODE WHEN '101M' THEN X1.TANKA+5000
      WHEN '201M' THEN X1.TANKA+5000
      WHEN '301M' THEN X1.TANKA+5000
      ELSE X1.TANKA * 1.2
    END
```

6. トリガ動作に SQL 診断文（SIGNAL 文）を使用します。在庫表（ZAIKO）の行を削除する前に SIGNAL 文でエラーを発生させ，在庫表からの行の削除を抑止するトリガ（SIGNALTRIG）を定義します。

```
CREATE TRIGGER SIGNALTRIG
  BEFORE DELETE ON ZAIKO
  SIGNAL SQLSTATE '99001'
```

3.19 CREATE TYPE（型定義）

3.19.1 CREATE TYPE の形式と規則

(1) 機能

抽象データ型を定義します。

(2) 使用権限

スキーマを所有するユーザ

自分が所有する抽象データ型を定義できます。

(3) 形式

```
CREATE TYPE 〔認可識別子.〕データ型識別子
                〔サブタイプ句〕
                〔デフォルトコンストラクタオプション〕
                〔メンバリスト〕

サブタイプ句：：＝UNDER 〔認可識別子.〕データ型識別子

デフォルトコンストラクタオプション：：＝
                 CONSTRUCTOR  {PRIVATE｜PROTECTED｜PUBLIC}

メンバリスト：：＝（メンバ〔, メンバ〕…）
    メンバ：：＝｛属性定義｜ルーチン宣言｝
        属性定義：：＝〔隠蔽レベル〕 属性名 データ型 〔NO SPLIT〕
        隠蔽レベル：：＝｛PRIVATE｜PROTECTED｜PUBLIC｝
        ルーチン宣言：：＝〔隠蔽レベル〕 ルーチン本体
        ルーチン本体：：＝｛関数本体｜手続き本体｝
```

(4) オペランド

(a) 〔認可識別子.〕データ型識別子

認可識別子

定義する抽象データ型の所有者の認可識別子を指定します。

データ型識別子

定義する抽象データ型の名称を指定します。

(b) サブタイプ句：：＝ UNDER 〔認可識別子.〕データ型識別子

サブタイプ句は，指定した抽象データ型を継承するサブタイプを定義する場合に指定します。スーパタイプとなる抽象データ型の認可識別子とデータ型識別子を指定してください。

サブタイプ句を指定すると，そのスーパタイプで定義されているすべての属性及びルーチンが，定義した抽象データ型に継承されます。

認可識別子.

スーパタイプの抽象データ型の所有者の認可識別子を指定します。

認可識別子を省略した場合，省略時に仮定される認可識別子に同一名称の抽象データ型がないときは，認可識別子'MASTER'に同一名称の抽象データ型があれば，その抽象データ型を指定したものとします。

データ型識別子

スーパタイプの抽象データ型を指定します。

(c) デフォルトコンストラクタオプション：：＝ CONSTRUCTOR {PRIVATE | PROTECTED | PUBLIC}

デフォルトコンストラクタオプションは，デフォルトコンストラクタ関数の隠蔽レベルを指定します。省略した場合，PRIVATE が仮定されます。

デフォルトコンストラクタ関数は，定義した抽象データ型の名称と同一の関数名を持つ関数が定義されることになります。デフォルトコンストラクタ関数には，引数がありません。この関数では，抽象データ型のそれぞれ各属性のナル値が設定された抽象データ型の値が戻されます。

PRIVATE

PRIVATE を指定した場合，抽象データ型のデフォルトコンストラクタ関数がこの抽象データ型定義文内でだけ利用できるようになります。

PROTECTED

PROTECTED を指定した場合，部分的に隠蔽されることになります。抽象データ型のデフォルトコンストラクタ関数が，定義される抽象データ型の中と，その抽象データ型のすべてのサブタイプの定義内でだけ利用できるようになります。

PUBLIC

PUBLIC を指定した場合，抽象データ型のデフォルトコンストラクタ関数が継承に関係なく利用できるようになります。

(d) 属性定義：：＝〔隠蔽レベル〕 属性名 データ型 〔NO SPLIT〕

属性定義では，抽象データ型を構成する属性を指定します。

隠蔽レベル

隠蔽レベルは三つあります。この隠蔽レベルは，属性又は抽象データ型に対する操作を記述するルーチンに対して指定できます。省略した場合は，その定義部分より上の隠蔽レベルが仮定されます。また，最初の属性に隠蔽レベルを指定していない場合，PUBLIC が仮定されます。

PRIVATE

PRIVATE を指定した場合，隠蔽されることになります。指定された属性は，その抽象データ型の定義中でだけ利用できます。

PROTECTED

PROTECTED を指定した場合，部分的に隠蔽されることになります。指定された属性は，その抽象データ型の定義中と，その抽象データ型のすべてのサブタイプの定義内でだけ利用できます。

PUBLIC

PUBLIC を指定した場合，指定された属性は継承関係に関係なく利用できるようになります。

属性名

抽象データ型の属性の名称を指定します。

データ型

抽象データ型の属性のデータ型を指定します。

指定するデータ型が抽象データ型で，認可識別子を省略した場合，省略時に仮定される認可識別子に同一名称の抽象データ型がないときは，認可識別子'MASTER'に同一名称の抽象データ型があれば，その抽象データ型を指定したものとします。

なお，次のデータ型は指定できません。

- 文字集合を指定した CHAR，VARCHAR

NO SPLIT

可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）の実際のデータ長が 256 バイト以上の場合，1 行を 1 ページに格納したいときに指定します。NO SPLIT を指定すると，データベース容量を削減できる場合があります。これをノースプリットオプションといいます。

ノースプリットオプションについては，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(e) ルーチン宣言：：＝〔隠蔽レベル〕 ルーチン本体

ルーチン本体：：＝ {関数本体｜手続き本体}

ルーチン宣言では，データへの操作を指定するルーチンを書きます。隠蔽レベルについては，属性定義と同じです。

ルーチン本体には，関数又は手続きを書きます。関数及び手続きについては，それぞれ「CREATE 〔PUBLIC〕 FUNCTION（関数定義，パブリック関数定義）」及び「CREATE 〔PUBLIC〕 PROCEDURE（手続き定義，パブリック手続き定義）」を参照してください。

(5) 共通規則

1. ルーチンで認可識別子を指定する場合，実行ユーザの認可識別子と同じものを指定します。
2. 関数本体及び手続き本体中で指定する CALL 文の手続き，及び関数呼出しの関数は，次のものでなければなりません。
   - この CREATE TYPE 外で，既に定義されている関数及び手続き
   - このルーチン宣言より前のルーチン宣言で定義されている関数及び手続き
3. 抽象データ型の属性名は，継承関係にあるすべての抽象データ型内でユニークでなければなりません。
4. 継承するサブタイプの世代数は，最大 30,000 です。
5. CREATE TYPE で定義する抽象データ型は，次の式を満たす必要があります。各属性の長さ（データ長）を次の表に示します。

$$2 \times n + \sum_{i=1}^{n} Ai <= 32757$$

n ：CREATE TYPEで定義した抽象データ型の属性数

$\sum_{i=1}^{n} Ai$ ：メンバリストの各属性のデータ長の総和

表 3-36　データ長一覧

分類	データ型	データ長 (単位：バイト)
数データ	INTEGER（4 バイト 2 進形式の整数）	4
	SMALLINT（2 バイト 2 進形式の整数）	2
	[LARGE] DECIMAL［m，n］※ （パック 10 進形式の固定小数点数）	↓m/2↓+1
	FLOAT（8 バイトの浮動小数点数）	8
	SMALLFLT（4 バイトの浮動小数点数）	4
文字データ	CHARACTER［n］（n バイトの固定長文字列）	n
	VARCHAR［n］（n バイトの可変長文字列）	35
各国文字データ	NCHAR［n］（n 文字の固定長各国文字列）	2n
	NVARCHAR［n］（n 文字の可変長各国文字列）	35
混在文字データ	MCHAR［n］（n バイトの固定長混在文字列）	n
	MVARCHAR［n］（n バイトの可変長混在文字列）	35
日付データ	DATE（日付）	4
日間隔データ	INTERVAL YEAR TO DAY（日間隔）	5
時刻データ	TIME（時刻）	3

分類	データ型	データ長 (単位：バイト)
時間隔データ	INTERVAL HOUR TO SECOND（時間隔）	4
時刻印データ	TIMESTAMP〔p〕（p けたの小数秒）（時刻印）	7+p/2
長大データ	BLOB（n バイトのバイナリ列）	35
バイナリデータ	BINARY〔n〕（n バイトのバイナリデータ列）	35
抽象データ	CREATE TYPE で定義したユーザ定義型	35

（凡例）

m，n：正の整数

p：0，2，4，又は 6 の整数

注※

全体のけた数が m けたで，小数点以下のけた数が n けたの固定小数点数です。m を省略すると 15 が仮定されます。

6. ALTER PROCEDURE，又は ALTER ROUTINE で SQL コンパイルオプションを指定する場合，再作成するルーチンの元の CREATE TYPE に SQL コンパイルオプションを反映してできる SQL 文は，SQL 文の最大長を超えないようにしてください。
7. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から CREATE TYPE は実行できません。

(6) 留意事項

1. CREATE TYPE は，OLTP 下の X/Open に従った UAP から指定できません。
2. 型定義に伴う SQL オブジェクトの無効化について示します。
   - 抽象データ型を定義する場合，その抽象データ型と同じ継承関係にある抽象データ型（最上位のデータ型の，すべてのサブタイプ）を使用する関数，手続き，及びトリガの有効な SQL オブジェクトがあれば，その SQL オブジェクトは無効となります。
   - 抽象データ型を定義する場合，定義する抽象データ型と，同じデータ型が認可識別子'MASTER'である場合，認可識別子'MASTER'のデータ型を使用する関数，手続き，及びトリガの有効な SQL オブジェクトのうち，定義する抽象データ型の認可識別子の SQL オブジェクトは無効になります。また，パブリック関数及びパブリック手続きの場合，定義する抽象データ型の認可識別子と，パブリック関数及びパブリック手続きを定義した認可識別子が同じならば，そのパブリック関数及びパブリック手続きの SQL オブジェクトは無効となります。
   - 関数本体を指定して関数を定義する場合の SQL オブジェクトの無効化については，「CREATE〔PUBLIC〕FUNCTION（関数定義，パブリック関数定義）」の留意事項を参照してください。
   - 手続き本体を指定して手続きを定義する場合の SQL オブジェクトの無効化については，「CREATE〔PUBLIC〕PROCEDURE（手続き定義，パブリック手続き定義）」の留意事項を参照してください。
3. 関数，手続き，及びトリガの有効な SQL オブジェクトが無効となった場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった関数，手続き，及びトリガの行は削除されます。
4. 2 に示した無効になる関数のうち，次のどちらかの条件を満たす関数をビュー定義で使用している場合，型定義がエラーになります。

- 引数のデータ型に抽象データ型を使用している
- 戻り値のデータ型に抽象データ型を使用している

5. 無効となった関数，手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して，関数，手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
6. 型定義に伴って無効となった関数を使用したビュー表がある場合，そのビュー表を操作するには ALTER ROUTINE を実行して関数の SQL オブジェクトを再作成しておく必要があります。
7. プラグイン開発者によって提供され，pdplgrgst コマンドによって HiRDB に登録される抽象データ型を継承した場合，その抽象データ型のサブタイプは定義できません。

3.20　CREATE 〔PUBLIC〕VIEW（ビュー定義，パブリックビュー定義）

3.20.1　CREATE VIEW（ビュー定義）

(1)　機能

ビュー表を定義します。

(2)　使用権限

導出問合せ式を使用できるユーザ

使用できる導出問合せ式を指定し，自分が所有するビュー表を定義できます。

導出問合せ式を使用できるユーザを次に示します。導出問合せ式については，「問合せ式」を参照してください。また，導出問合せ式中の問合せ指定の権限については，「問合せ指定」の使用権限を参照してください。

- 実表又はビュー表の所有者
- 実表又はビュー表に対する SELECT 権限を持つユーザ

ただし，他人が所有する表への SELECT 権限を受け取ってビュー表を定義した場合，以降このビュー表からのビュー定義は自分だけできます。SELECT 権限を与えたユーザはできません。

(3)　形式

```
CREATE 〔READ ONLY〕 VIEW 〔認可識別子.〕表識別子
        〔（列名〔, 列名〕…）〕
  AS  導出問合せ式

導出問合せ式：：＝  問合せ式本体
問合せ式本体：：＝｛問合せ指定
                  ｜（問合せ式本体）
                  ｜問合せ式本体 {UNION｜EXCEPT} 〔ALL〕
                    {問合せ指定｜（問合せ式本体）｝｝
```

(4)　オペランド

(a)　〔READ ONLY〕

定義するビュー表を読み込み専用にする場合に指定します。

(b)　〔認可識別子.〕表識別子〔(列名〔, 列名〕…)〕

認可識別子

定義するビュー表の所有者となるユーザの認可識別子を指定します。

表識別子

定義するビュー表の名前を指定します。

定義するビュー表の所有者になるユーザが既に所有している表（実表及びビュー表）の中に同じ名前を指定しないでください。

列名

ビュー表を構成する列を指定します。

列名を省略した場合，次のようになります。

- 導出問合せ式に集合演算を指定しない場合，問合せ指定によって指定された導出表の各列の列名（AS 列名を指定している場合は，AS 句で指定した列名）が，ビュー表を構成する各列の列名になります。
- 導出問合せ式に集合演算を指定した場合，導出問合せ式の 1 番目の問合せ指定によって指定された導出表の各列の列名（AS 列名を指定している場合は，AS 句で指定した列名）が，ビュー表を構成する各列の列名になります。

ただし，導出問合せ式によって指定された導出表が，列名が同じである二つ以上の列を含む，又は名前のない列を含む場合は省略できません。

列名についての規則を次に示します。

1. 導出表の列が次に示す項目から導き出された列で，AS 列名を省略した場合，その列は名前のない列になります。
   ・スカラ演算
   ・関数呼出し
   ・集合関数
   ・定数
   ・USER 値関数
   ・CURRENT_DATE 値関数
   ・CURRENT_TIME 値関数
   ・CURRENT_TIMESTAMP 値関数
   ・コンポネント指定
2. 同じ列名は指定できません。列名で指定する列の数は，導出問合せ式の結果で得られる導出表の列と同じ数にしてください。
3. 列名で指定できる列の数の最大値は 30,000 個です。

(c) 導出問合せ式

ビュー表の定義内容を表す導出問合せ式を指定します。導出問合せ式については「問合せ式」を参照してください。

導出問合せ式についての規則を次に示します。

1. ビュー定義中の導出問合せ式では，選択式に［表指定.］ROW は指定できません。

2. ビュー定義中の導出問合せ式では，直接含まれる SELECT 句に添字付きの繰返し列は指定できません。
3. 導出問合せ式に直接含まれる SELECT 句に*，又は表.*を指定しても，ビュー定義後にビュー表の基表に追加した列はビュー表には追加されません。
4. ビュー定義の導出問合せ式中には，実表及びビュー表を指定できます。また，ビュー表を基にして新しいビュー表も定義できます。
5. ビュー定義の導出問合せ式中では，埋込み変数，及び?パラメタは指定できません。
6. ビュー定義中の導出問合せ式では，次に示す値式は指定できません。
   - XML コンストラクタ関数
   - SQL/XML スカラ関数
   - SQL/XML 述語
   - SQL/XML 集合関数
7. ビュー定義中の導出問合せ式に，予備列を指定できません。

(5) 共通規則

1. ビュー表は，読み込み専用のビュー表と更新できるビュー表に分けられます。

   読み込み専用のビュー表は行の挿入，更新，削除，及びカーソル宣言での FOR UPDATE 句の指定はできません。

   読み込み専用のビュー表を次に示します。
   - ビュー定義文で，READ ONLY を指定して定義したビュー表
   - ビュー定義文で，最も外側の問合せ指定に対する表の結合，SELECT DISTINCT，GROUP BY 句，HAVING 句，又は集合関数を含むビュー表
   - ビュー定義文で，最も外側の問合せ指定の SELECT 句に基の表の同一列を複数指定して定義したビュー表
   - ビュー定義文で，最も外側の問合せ指定の SELECT 句に列指定以外の値式を含むビュー表
   - バージョン 07-02 より前に定義したビュー表のうち，ビュー定義文で，最も外側の問合せ指定の FROM 句に指定した表と同じ表を FROM 句に指定した副問合せを含むビュー表（FROM 句に指定する表がビュー表の場合，ビュー表の基となる表も含む）

     注　このビュー表を更新できるビュー表にするには，そのビュー表を削除してから，定義し直してください。
   - ビュー定義文で，最も外側の問合せ指定の FROM 句に導出表を指定して定義したビュー表
   - ビュー定義文で集合演算を指定して定義したビュー表

   また，上記以外のビュー表（読み込み専用でないビュー表）が更新できるビュー表です。
2. ビュー定義の導出問合せ式中に指定した表は，そのビュー表を構成する基の表です。ビュー定義の導出問合せ式に含まれる FROM 句に指定した表は，そのビュー表を導き出すための操作対象の基になる表です。

3. ビュー定義の導出問合せ式中の，最も外側の問合せに含まれる FROM 句に指定した表が，そのビュー表に対する操作の対象の基になる表です。

   更新できるビュー表の所有者は，そのビュー表の操作の対象の基になる表に対して持つ，次のアクセス権限をそのまま引き継ぎます。

   - SELECT 権限
   - INSERT 権限
   - DELETE 権限
   - UPDATE 権限

   読み込み専用のビュー表の所有者は，そのビュー表の操作の対象になる表の基になる表に対して持つ SELECT 権限だけを引き継ぎます。

   また，すべて自分が所有する表から定義したビュー表の所有者はほかのユーザに対して自分の持つアクセス権限を与えたり，取り消したりできます。

4. ビュー表を使用して行の追加や更新をする場合，追加する行，又は更新後の行は，ビュー表定義時に導出問合せ式中で指定した探索条件を満足してもしなくてもかまいません。ただし，ビュー表定義時に指定した探索条件を満足しない行は，そのビュー表を使用して検索，更新，及び削除はできません。

5. ビュー表を構成する列の属性（データ型，データ長，非ナル値制約の有無，及び最大要素数）は，ビュー定義の導出問合せ式で指定した導出表の対応する列の属性と同じになります。

6. ビュー定義を実行する場合，そのビュー表の基になる表をあらかじめ定義してください。

7. ビュー定義文は，OLTP 下の X/Open に従った UAP から指定できません。

8. 文字列表現された日付，時刻，又は時刻印の定数，及び 10 進数で表現された日間隔や時間隔の定数から定義したビュー表の列は，日付データ，時刻データ，時刻印データ，日間隔データ，及び時間隔データが要求される場所で指定しても，その列のデータ型で扱い，それぞれの要求されるデータ型には変換されません。ただし，スカラ関数 DATE，TIME，及び TIMESTAMP の引数に指定した場合を除きます。

   (例)

   ```
   UPDATE T1 SET C1=(SELECT VC2 FROM V1 WHERE VC1='E')
   …指定できません。
   ```

   (C1 は日付データ型の列，VC2 は日付の文字列で表現された定数から定義した VARCHAR（10）の列)

9. ビュー定義中の導出問合せ式では，直接含まれる SELECT 句に添字付きの繰返し列は指定できません。

10. ビュー定義中の導出問合せ式では，最も外側の問合せ指定の選択式に CASE 式を指定した場合，その CASE 式の探索条件には繰返し列を指定できません。

11. ビュー定義の導出問合せ式の選択式には，次の項目は指定できません。

    - WRITE 指定
    - GET_JAVA_STORED_ROUTINE_SOURCE 指定
    - ウィンドウ関数

12. 関数呼出しを指定して定義したビュー表を操作する場合，呼び出す候補となるユーザ定義関数は，そのビュー表の定義よりも前に定義したユーザ定義関数だけとなります。

13. SQL オブジェクト移行ユティリティで，32 ビットモードから 64 ビットモードの HiRDB へ移行した場合，次に示す条件をすべて満たすビュー表は移行前後で検索結果が異なることがあります。

    (1) ビュー定義の導出問合せ式にユーザ定義関数を指定して，ビュー表を定義する

    (2) (1)のビュー表を定義した後，(1)の導出問合せ式に指定したユーザ定義関数に対して，呼出し候補となるユーザ定義関数を定義する

    (3) (2)で定義したユーザ定義関数は，(1)の導出問合せ式に指定したユーザ定義関数よりも呼出し優先度が高い

    上記の(1)～(3)の後，SQL オブジェクト移行ユティリティで，32 ビットモードから 64 ビットモードの HiRDB へ移行する

    ユーザ定義関数の呼出し決定規則については，「呼び出す関数の決定規則と結果のデータ型」を参照してください。

(6) 使用例

1. 在庫表（ZAIKO）から商品名（SNAME）がソックスの行で商品コード（SCODE），在庫量（ZSURYO），単価（TANKA）の列で構成されるビュー表（VZAIKO1）を定義します。なお，列の並びは，商品コード，在庫量，単価の順とします。

```
CREATE VIEW VZAIKO1
    AS SELECT SCODE,ZSURYO,TANKA
    FROM ZAIKO
    WHERE SNAME = N'ソックス'
```

2. 在庫表（ZAIKO）と同じ構成のビュー表（VZAIKO2）を読み込み専用として定義します。

```
CREATE READ ONLY VIEW VZAIKO2
    AS SELECT * FROM ZAIKO
```

3.20.2 CREATE PUBLIC VIEW（パブリックビュー定義）

(1) 機能

すべてのユーザから，認可識別子で表識別子を修飾しなくても使用できるビュー表（パブリックビュー）を定義できます。

(2) 使用権限

実表の所有者

自分が所有する実表から，自分が所有するパブリックビューを定義できます。

ビュー表の所有者

自分が所有する実表が基表となるビュー表から，自分が所有するパブリックビューを定義できます。

(3) 形式

```
CREATE PUBLIC〔READ ONLY〕 VIEW 表識別子
        〔（列名〔，列名〕…）〕
  AS  導出問合せ式

導出問合せ式：：＝  問合せ式本体
問合せ式本体：：＝｛問合せ指定
                    ｜（問合せ式本体）
                    ｜問合せ式本体 {UNION｜EXCEPT} 〔ALL〕
                      {問合せ指定｜（問合せ式本体）｝｝
```

(4) オペランド

PUBLIC，表識別子以外の説明については，「CREATE VIEW（ビュー定義）」と同じです。

(a) PUBLIC

ビュー表をパブリックビューとして定義する場合に指定します。

パブリックビューにすると，同じ内容のビュー表をユーザごとに定義しなくても，一つのビュー表を表識別子だけを指定して複数のユーザで使用できます。

(b) 表識別子

定義するパブリックビューの名前を指定します。

既に定義してあるパブリックビューと同じ名前は指定できません。

(5) 共通規則

1. 定義するパブリックビューの名前には，既に定義されている表（実表及びビュー表）と同じ表識別子を指定できます。ただし，認可識別子を省略して表識別子を使用する場合，UAP 実行ユーザが所有している表（実表及びビュー表）がパブリックビューより優先されます。パブリックビューを明示的に修飾する場合は，認可識別子に PUBLIC を指定してください。
2. そのほかの規則は，「CREATE VIEW（ビュー定義）」の共通規則と同じです。

(6) 留意事項

1. ディクショナリ表の所有者を格納する列（SQL_TABLES 表の TABLE_SCHEMA 列など）には PUBLIC が設定されます。また，パブリックビューを定義した認可識別子は，SQL_TABLES 表の TABLE_CREATOR 列に格納します。

2. パブリックビューを使用した SQL 文に対する前処理が有効な間に，使用しているパブリックビューと同じ名前の表（実表及びビュー表）を指定した定義系 SQL 文を発行すると，定義系 SQL は排他待ちの状態になります。

3. パブリックビューを使用した手続き及びトリガを定義した後で，パブリックビューと同じ名前の表（実表及びビュー表）を定義しても，該当する手続き及びトリガは無効にならないで，パブリックビューを使用した手続き及びトリガとして動作します。ただし，その手続き及びトリガを再作成した（インデクス無効状態のプロシジャを HiRDB が内部的に再作成する場合も含む）場合，パブリックビューと同じ名前の表（実表及びビュー表）を使用した手続き及びトリガとして動作します。

4. パブリックビューの削除は，DROP PUBLIC VIEW で行います。

(7) 使用例

1. 在庫表（ZAIKO）から商品名（SNAME）がソックスの行で商品コード（SCODE），在庫量（ZSURYO），単価（TANKA）の列で構成されるパブリックビュー（PVZAIKO1）を定義します。なお，列の並びは，商品コード，在庫量，単価の順とします。

```
CREATE PUBLIC VIEW PVZAIKO1
    AS SELECT SCODE,ZSURYO,TANKA
    FROM ZAIKO
    WHERE SNAME = N'ソックス'
```

3.21　DROP AUDIT（監査対象イベントの削除）

3.21.1　DROP AUDIT の形式と規則

(1)　機能

CREATE AUDIT で定義した監査対象イベントと内容が一致する定義を，監査対象から削除します。

(2)　使用権限

監査権限を持つユーザ

監査権限を持つユーザが，DROP AUDIT の各定義文を実行できます。

(3)　形式

```
DROP AUDIT
       〔AUDITTYPE  {PRIVILEGE | EVENT | ANY}〕
      FOR  操作種別
       〔選択オプション〕
       〔WHENEVER   {SUCCESSFUL | UNSUCCESSFUL | ANY} 〕
```

- 各項目の詳細

```
操作種別：：＝  {ANY
               | SESSION     〔 {セッション種別 | ANY} 〕
               | PRIVILEGE   〔 {権限操作種別 | ANY} 〕
               | DEFINITION  〔 {オブジェクト定義イベント種別 | ANY} 〕
               | ACCESS      〔 {オブジェクト操作イベント種別 | ANY} 〕
               | UTILITY     〔 {ユティリティイベント種別 | ANY} 〕 }
選択オプション：：＝ {ON オブジェクト名
               | BY AUTHORIZATION 認可識別子}
    オブジェクト名：：＝
                  {FUNCTION  認可識別子.ルーチン識別子
                | INDEX  認可識別子.インデクス識別子
                | LIST  認可識別子.表識別子
                | PROCEDURE  認可識別子.ルーチン識別子
                | RDAREA  RDエリア名
                | SCHEMA  認可識別子
                | TABLE  〔認可識別子.〕表識別子
                | TRIGGER  認可識別子.トリガ識別子
                | TYPE  認可識別子.データ型識別子
                | VIEW  認可識別子.表識別子
                | SEQUENCE  認可識別子.順序数生成子識別子}
セッション種別：：＝
     {CONNECT | DISCONNECT | AUTHORIZATION}
権限操作種別：：＝
     {GRANT | REVOKE}
オブジェクト定義イベント種別：：＝
     {CREATE | DROP | ALTER}
```

```
オブジェクト操作イベント種別：：＝
      {SELECT｜INSERT｜UPDATE｜DELETE｜PURGE｜ASSIGN｜CALL｜LOCK
        ｜NEXT VALUE}
ユティリティイベント種別：：＝
      {PDLOAD｜PDRORG｜PDEXP｜PDCONSTCK}
```

(4)　オペランド

それぞれの項目の詳細については，「CREATE AUDIT（監査対象イベントの定義）」を参照してください。

(a)　AUDITTYPE　{PRIVILEGE ｜ EVENT ｜ ANY}

削除する監査証跡の種別を指定します。

(b)　FOR 操作種別

監査対象から削除する操作種別を指定します。

(c)　WHENEVER　{SUCCESSFUL ｜ UNSUCCESSFUL ｜ ANY}

CREATE AUDIT で指定した WHENEVER 句の指定を，監査対象から削除します。

(d)　セッション種別

監査対象から削除する，HiRDB に対する接続，接続中のユーザ変更，又は切断操作を指定します。

(e)　権限操作種別

監査対象から削除する，HiRDB に対する権限操作を指定します。

(f)　オブジェクト定義イベント種別

監査対象から削除する，HiRDB に対するオブジェクト定義操作を指定します。

(g)　オブジェクト操作イベント種別

監査対象から削除する，HiRDB に対するオブジェクト操作を指定します。

(h)　ユティリティイベント種別

監査対象から削除する，HiRDB に対するユティリティイベントを指定します。

(i)　選択オプション

監査対象から削除する選択オプションを指定します。

(5)　規則

1. セキュリティ監査機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

2. 実際に監査証跡を記録するには，システム定義 pd_audit オペランドの指定，又は pdaudbegin コマンドを実行する必要があります。
3. CREATE AUDIT，及び DROP AUDIT を実行した際の監査証跡は，セキュリティ監査機能が有効な場合は必ず記録します。
4. CREATE AUDIT 時に指定した AUDITTYPE, FOR ＜操作種別＞, WHENEVER の組み合わせと同じ指定で DROP AUDIT を実行できます。

   (例)

   CREATE AUDIT AUDITTYPE EVENT FOR SESSION を削除する場合は，DROP AUDIT AUDITTYPE EVENT FOR SESSION と指定すると削除できます。
5. DROP AUDIT では，定義された監査対象範囲の一部分だけを，監査対象から外すような使い方はできません。その場合，KFPA11909-E メッセージを出力します。

   （例 1 ）

   CREATE AUDIT FOR ANY で，すべての監査イベントを監査対象として定義した場合，表への SELECT を監査対象から削除するため，DROP AUDIT FOR ACCESS に SELECT を指定して実行できません。

   表への SELECT を監査対象から削除したい場合は，表への SELECT 以外の必要な監査対象の定義を行ってから，DROP AUDIT FOR ANY を実行してください。

   （例 2 ）

   CREATE AUDIT AUDITTYPE PRIVILEGE FOR ANY で，権限チェック時のすべての監査イベントを監査対象として定義した場合，AUDITTYPE の PRIVILEGE を監査対象から削除するため，DROP AUDIT AUDITTYPE PRIVILEGE を指定して実行できません。

(6) 留意事項

1. DROP AUDIT は，OLTP 下の X/Open に従った UAP からは指定できません。

(7) 使用例

1. CREATE AUDIT FOR ANY で定義した監査対象を削除します。

```
DROP AUDIT FOR ANY
```

2. CREATE AUDIT FOR SESSION で定義した監査対象を削除します。

```
DROP AUDIT FOR SESSION CONNECT
```

3. CREATE AUDIT FOR PRIVILEGE で定義した監査対象を削除します。

```
DROP AUDIT FOR PRIVILEGE GRANT
```

4. CREATE AUDIT FOR DEFINITION で定義した監査対象を削除します。

```
DROP AUDIT FOR DEFINITION CREATE WHENEVER ANY
```

5. CREATE AUDIT FOR ACCESS で定義した監査対象を削除します。

```
DROP AUDIT FOR ACCESS INSERT
```

6. CREATE AUDIT AUDITTYPE ANY FOR ANY で定義した監査対象を削除します。

```
DROP AUDIT AUDITTYPE ANY FOR ANY
```

7. CREATE AUDIT AUDITTYPE PRIVILEGE FOR ANY で定義した監査対象を削除します。

```
DROP AUDIT AUDITTYPE PRIVILEGE FOR ANY
```

8. CREATE AUDIT AUDITTYPE EVENT FOR ANY ON TABLE “USER1”.”T1”で定義した監査対象を削除します。

```
DROP AUDIT AUDITTYPE EVENT FOR ANY ON TABLE “USER1”.”T1”
```

9. CREATE AUDIT AUDITTYPE ANY FOR ANY BY AUTHORIZATION “USER1”で定義した監査対象を削除します。

```
DROP AUDIT AUDITTYPE ANY FOR ANY BY AUTHORIZATION “USER1”
```

3.22 DROP CONNECTION SECURITY（CONNECT 関連セキュリティ機能の削除）

3.22.1 DROP CONNECTION SECURITY の形式と規則

(1) 機能

CONNECT 関連セキュリティ機能に関するセキュリティ項目を削除します。

(2) 使用権限

DBA 権限を持つユーザ

DBA 権限を持つユーザが，DROP CONNECTION SECURITY の各定義文を実行できます。

(3) 形式

```
DROP CONNECTION SECURITY FOR {セキュリティ対象〔, セキュリティ対象 〕
                            |パスワード有効期間
                            |CONSTRAINT 接続制約名}

セキュリティ対象：：＝ { CONNECT|PASSWORD }
パスワード有効期間：：＝ INTERVAL PASSWORD FROM 認可識別子
```

(4) オペランド

(a) セキュリティ対象：：＝ {CONNECT | PASSWORD}

セキュリティ対象には CONNECT，PASSWORD をそれぞれ 1 回しか指定できません。

CONNECT

連続認証失敗回数制限に関する設定を削除する場合に指定します。

PASSWORD

パスワード文字制限強化に関する設定を削除する場合に指定します。

パスワード有効期間：：＝ INTERVAL PASSWORD FROM 認可識別子 パスワード有効期間定義

パスワード有効期間設定済みのユーザから，パスワード有効期間に関する設定を削除する場合に指定します。

パスワード有効期間に関する設定を削除すると，指定したユーザの有効期限切れ状態は解除されます。

(b) CONSTRAINT 接続制約名

IP アドレスによる接続許可・拒否に関する制約を削除する場合に指定します。

(5) 留意事項

1. 連続認証失敗回数制限に関する設定を削除した場合，全ユーザの連続認証失敗アカウントロック状態は解除されます（SQL_USERS の連続認証失敗アカウントロック日時，及び連続認証失敗回数にナル値を設定します）。
2. パスワード文字制限に関する設定を削除した場合，全ユーザのパスワード無効アカウントロック状態は解除されます（SQL_USERS のパスワード無効アカウントロック日時にナル値を設定します）。
3. 定義されていないセキュリティ対象又は接続制約名を指定することはできません。
4. CONNECT，又は PASSWORD のどちらか一方だけを定義した状態で CONNECT，PASSWORD の両方を削除しようとした場合はエラーとなります。削除の対象に指定した定義済みの項目は削除されません。

(6) 使用例

1. パスワード文字制限強化に関する設定を削除します。

```
DROP CONNECTION SECURITY FOR PASSWORD
```

2. IP アドレスによる接続制限に関する設定を削除する場合は，マニュアル「HiRDB システム運用ガイド」の「IP アドレスによる接続制限」を参照してください。

3.23　DROP DATA TYPE（ユーザ定義型削除）

3.23.1　DROP DATA TYPE の形式と規則

(1)　機能

抽象データ型を削除します。

(2)　使用権限

抽象データ型の所有者

自分が所有する抽象データ型を削除できます。

DBA 権限を持つユーザ

他ユーザが所有する抽象データ型を削除できます。

(3)　形式

```
DROP DATA TYPE 〔認可識別子.〕データ型識別子
                  〔WITH PROGRAM〕
```

(4)　オペランド

(a)　〔認可識別子.〕データ型識別子

認可識別子

削除するデータ型識別子の所有者の認可識別子を指定します。

データ型識別子

削除するデータ型識別子を指定します。

(b)　WITH PROGRAM

抽象データ型を削除する場合，次に示す関数，手続き，及びトリガの有効な SQL オブジェクトを無効にするときに指定します。

- 削除する抽象データ型と同じ継承関係にある抽象データ型（最も上位のデータ型のすべてのサブタイプ）を使用する関数，手続き，及びトリガの有効な SQL オブジェクト
- 削除する抽象データ型で定義した関数及び手続きを使用する，関数，手続き，及びトリガの有効な SQL オブジェクト
- 削除する抽象データ型で定義した関数と所有者，ルーチン識別子，及びパラメタ数が同じ関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト

WITH PROGRAM を省略した場合，次に示す関数，手続き，及びトリガの SQL オブジェクトがあるときは，その抽象データ型は削除できません。

- 削除する抽象データ型と同じ継承関係にある抽象データ型を使用する関数，手続き，及びトリガの有効な SQL オブジェクト
- 削除する抽象データ型で定義した関数及び手続きを使用する関数，手続き，及びトリガの有効な SQL オブジェクト
- 削除する抽象データ型で定義した関数と所有者，ルーチン識別子，及びパラメタ数が同じ関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト

(5) 共通規則

1. 指定した抽象データ型を利用している実表，インデクス，抽象データ型，抽象データ型内のルーチン，ルーチン，及びトリガがある場合は削除しません。
2. 抽象データ型定義内で定義した関数をビュー定義で使用している場合は，その抽象データ型は削除できません。
3. WITH PROGRAM の指定によって無効となる関数のうち，次のどれかの条件を満たす関数をビュー定義に使用している場合，抽象データ型の削除はできません。
   - 引数のデータ型に抽象データ型を指定している
   - 戻り値のデータ型に抽象データ型を指定している
4. 抽象データ型削除に伴って実行中の SQL オブジェクトが無効になる場合，Java 手続き中から DROP DATA TYPE は実行できません。

(6) 留意事項

1. DROP DATA TYPE は，OLTP 下の X/Open に従った UAP から指定できません。
2. 指定した抽象データ型内で抽象データ型関数が指定されている場合，プラグイン情報も削除されます。
3. WITH PROGRAM を指定して関数，手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった関数，手続き，及びトリガの行は削除されます。
4. WITH PROGRAM を指定して無効にした関数，手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE 又は ALTER PROCEDURE を実行して関数，手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
5. WITH PROGRAM を指定して無効にしたトリガの SQL オブジェクトを実行するためには，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成しておく必要があります。ただし，削除した抽象データ型定義内で定義した関数及び手続きを使用していたトリガについては，SQL オブジェクトを実行するためには，次のどちらかの操作をする必要があります。
   - 抽象データ型を定義し直して，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成する。

- 無効にしたトリガを DROP TRIGGER で削除してから，削除した抽象データ型定義内で定義した関数及び手続きを使用しないように CREATE TRIGGER でトリガを再定義します。ただし，次のすべての条件を満たすトリガがある場合は，それらもすべて DROP TRIGGER で削除し，定義していた順に CREATE TRIGGER で再定義しないと，トリガ動作の実行順序が変わります。

  (条件)

  ・無効にしたトリガよりも定義したのが後である。

  ・無効にしたトリガと定義した表が同じである。

  ・無効にしたトリガとトリガ契機（INSERT，UPDATE，又は DELETE）が同じである（ただし，UPDATE の場合は，トリガ契機列の指定の有無，及び内容に関係なく同じとみなされます）。

  ・無効にしたトリガとトリガ動作時期（BEFORE 又は AFTER）が同じである。

  ・無効にしたトリガとトリガの動作する単位（行単位又は文単位）が同じである。

6. WITH PROGRAM を指定して無効にした関数を使用したビュー表がある場合，そのビュー表を操作するためには，ALTER ROUTINE を実行して関数の SQL オブジェクトを再作成しておく必要があります。

(7) 使用例

ユーザ（認可識別子：USER1）が定義した SGML 型を削除します。

```
DROP DATA TYPE USER1.SGML
```

3.24 DROP 〔PUBLIC〕FUNCTION（関数削除，パブリック関数削除）

3.24.1 DROP FUNCTION（関数削除）

(1) 機能

関数を削除します。

(2) 使用権限

関数の所有者

自分が所有する関数を削除できます。

DBA 権限を持つユーザ

他ユーザが所有する関数を削除できます。

(3) 形式

```
DROP FUNCTION 〔認可識別子.〕ルーチン識別子（〔データ型〔，データ型〕…〕）
                  〔WITH PROGRAM〕
```

(4) オペランド

(a) 〔認可識別子.〕ルーチン識別子

認可識別子

削除する関数の所有者の認可識別子を指定します。

ルーチン識別子

削除する関数の名称を指定します。

(b) データ型

削除する関数のパラメタで指定したデータ型を指定します。削除する関数のパラメタに指定したデータ型が，固定長文字データ型又は可変長文字データ型でかつ，文字集合指定を指定している場合，その文字集合指定を指定してください。

(c) WITH PROGRAM

関数を削除する場合，次に示す関数，手続き，及びトリガの有効な SQL オブジェクトを無効にするときに指定します。

- 削除する関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト

- 削除する関数と所有者，ルーチン識別子，及びパラメタ数が同じ関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト

WITH PROGRAM を省略した場合，次に示す関数，手続き，及びトリガの SQL オブジェクトがあるときは，その関数は削除できません。

- 削除する関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト
- 削除する関数と所有者，ルーチン識別子，及びパラメタ数が同じ関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト

(5) 共通規則

1. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から DROP FUNCTION は実行できません。
2. 抽象データ型内で定義した関数は削除できません。
3. 同一名の関数が複数ある場合，削除する関数を一意に特定できるようにパラメタのデータ型を指定します。
4. 指定した関数を使用する，ビュー表（パブリックビューも含む）がある場合，関数は削除できません。
5. WITH PROGRAM の指定によって無効となる関数のうち，次のどれかの条件を満たす関数をビュー定義に使用している場合，その関数は削除できません。
   - 引数のデータ型に抽象データ型を使用している
   - 戻り値のデータ型に抽象データ型を使用している

(6) 留意事項

1. DROP FUNCTION は，OLTP 下の X/Open に従った UAP から指定できません。
2. WITH PROGRAM を指定して関数，手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった関数，手続き，及びトリガの行は削除されます。
3. WITH PROGRAM を指定して無効にした関数，手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE 又は ALTER PROCEDURE を実行して関数，手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
4. WITH PROGRAM を指定して無効にしたトリガの SQL オブジェクトを実行するためには，次のどちらかの操作をする必要があります。ただし，削除した関数と所有者，ルーチン識別子，及びパラメタ数が同じ関数を使用していたトリガについては，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成してもかまいません。
   - 関数を定義し直して，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成する。
   - 無効にしたトリガを DROP TRIGGER で削除してから，削除した関数を使用しないように CREATE TRIGGER でトリガを再定義します。ただし，次のすべての条件を満たすトリガがある場合は，そ

れらもすべて DROP TRIGGER で削除し，定義していた順に CREATE TRIGGER で再定義しないと，トリガ動作の実行順序が変わります。

（条件）

・無効にしたトリガよりも定義したのが後である。

・無効にしたトリガと定義した表が同じである。

・無効にしたトリガとトリガ契機（INSERT，UPDATE，又は DELETE）が同じである（ただし，UPDATE の場合は，トリガ契機列の指定の有無，及び内容に関係なく同じとみなされます）。

・無効にしたトリガとトリガ動作時期（BEFORE 又は AFTER）が同じである。

・無効にしたトリガとトリガの動作する単位（行単位又は文単位）が同じである。

5. WITH PROGRAM を指定してトリガ動作条件中で使用している関数を削除した場合，無効にしたトリガの SQL オブジェクトを実行するときだけでなく，そのトリガを引き起こす SQL を前処理するときにもエラーになります。

6. WITH PROGRAM を指定して無効にした関数を使用したビュー表がある場合，そのビュー表を操作するためには，ALTER ROUTINE を実行して関数の SQL オブジェクトを再作成しておく必要があります。

3.24.2 DROP PUBLIC FUNCTION（パブリック関数削除）

(1) 機能

すべてのユーザから，認可識別子でルーチン識別子を修飾しなくても使用できる関数（パブリック関数）を削除します。

(2) 使用権限

パブリック関数の所有者

自分が所有する（定義した）パブリック関数を削除できます。

DBA 権限を持つユーザ

他ユーザが所有するパブリック関数を削除できます。

(3) 形式

```
DROP PUBLIC FUNCTION ルーチン識別子（〔データ型〔，データ型〕…〕）
                    〔WITH PROGRAM〕
```

(4) オペランド

PUBLIC，ルーチン識別子，WITH PROGRAM 以外の説明については，「DROP FUNCTION（関数削除）」を参照してください。

(a) PUBLIC

パブリック関数を削除する場合に指定します。

(b) ルーチン識別子

削除するパブリック関数の名称を指定します。

(c) WITH PROGRAM

パブリック関数を削除する場合，次に示す関数，手続き，及びトリガの有効な SQL オブジェクトを無効にするときに指定します。

- 削除するパブリック関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト
- 削除するパブリック関数と，ルーチン識別子，及びパラメタ数が同じパブリック関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト

WITH PROGRAM を省略した場合，次に示す関数，手続き，及びトリガの SQL オブジェクトがあるときは，その関数は削除できません。

- 削除するパブリック関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト
- 削除するパブリック関数とルーチン識別子，及びパラメタ数が同じ関数を使用する関数，手続き，及びトリガの有効な SQL オブジェクト

(5) 共通規則

1. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から DROP PUBLIC FUNCTION は実行できません。
2. その他の共通規則は，「DROP FUNCTION（関数削除）」と同じです。

(6) 留意事項

1. DROP PUBLIC FUNCTION は，OLTP 下の X/Open に従った UAP から指定できません。
2. その他の留意事項は，「DROP FUNCTION（関数削除）」と同じです。

3.25　DROP INDEX（インデクス削除）

3.25.1　DROP INDEX の形式と規則

(1)　機能

インデクスを削除します。

(2)　使用権限

インデクスの所有者

ユーザ自身のインデクスを削除できます。

DBA 権限を持つユーザ

他ユーザのインデクスを削除できます。

(3)　形式

```
DROP INDEX 〔認可識別子.〕インデクス識別子〔WITH PROGRAM〕
```

(4)　オペランド

(a)　〔認可識別子.〕インデクス識別子〔WITH PROGRAM〕

認可識別子

インデクスを持つユーザの識別子を指定します。

省略した場合，実行するユーザの認可識別子を仮定します。

インデクス識別子

削除するインデクス識別子を指定します。

WITH PROGRAM

インデクスを削除するときに，そのインデクスを使用する手続き及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

WITH PROGRAM を省略した場合，そのインデクスを使用する手続き及びトリガの有効な SQL オブジェクトがあると，そのインデクスは削除できません。

(5)　共通規則

1. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から DROP INDEX は実行できません。

2. インナレプリカ機能を使用している場合の DROP INDEX の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
3. SQL セッション固有一時表を使用している SQL セッション中で，使用中の SQL セッション固有一時表のインデクスは削除できません。

(6) 留意事項

1. DROP INDEX は，OLTP 下の X/Open に従った UAP から指定できません。
2. WITH PROGRAM を指定して手続き及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった手続き及びトリガの情報は削除されます。
3. WITH PROGRAM を指定して無効にした手続き及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して，手続き及びトリガの SQL オブジェクトを再作成しておく必要があります。
4. WITH PROGRAM 指定の有無に関係なく，削除するインデクスを定義している表に対する手続き及びトリガ（その表に定義したトリガを除く）がある場合，その SQL オブジェクト中のインデクス情報は無効となります。この場合，このトリガは実行できなくなります。また，手続き又はトリガからこの手続きを実行できなくなるため，SQL オブジェクトを再作成する必要があります。

(7) 使用例

在庫表（ZAIKO）の商品コード（SCODE）列に定義したインデクス（IDX1）を削除します。

```
DROP INDEX IDX1
```

3.26 DROP 〔PUBLIC〕PROCEDURE（手続き削除，パブリック手続き削除）

3.26.1 DROP PROCEDURE（手続き削除）

(1) 機能

手続きを削除します。

(2) 使用権限

手続きの所有者

自分が所有する手続きを削除できます。

DBA 権限を持つユーザ

他ユーザの所有する手続きを削除できます。

(3) 形式

```
DROP PROCEDURE 〔認可識別子.〕ルーチン識別子
                    〔WITH PROGRAM〕
```

(4) オペランド

(a) 〔認可識別子.〕ルーチン識別子

認可識別子

削除する手続きの所有者の認可識別子を指定します。

省略した場合，実行するユーザの認可識別子を仮定します。

ルーチン識別子

削除する手続きのルーチンの名前を指定します。

(b) WITH PROGRAM

手続きを削除する場合，その手続きを使用する手続き，及びトリガの有効な SQL オブジェクトがあれば，その SQL オブジェクトを無効にするときに指定します。

WITH PROGRAM を省略した場合，その手続きを使用する手続き，及びトリガの有効な SQL オブジェクトがあるときは，その手続きは削除できません。

(5) 共通規則

1. 指定した手続きを呼び出す SQL ルーチン，及び抽象データ型内のルーチン，トリガがある場合は削除しません。
2. 抽象データ型内で定義した手続きは削除できません。
3. 次のような場合，Java 手続き中から DROP PROCEDURE は実行できません。
   - 実行中の SQL オブジェクトが無効になる場合，又は削除される場合
   - 実行中の Java 手続きが削除される場合

(6) 留意事項

1. DROP PROCEDURE は，OLTP 下の X/Open に従った UAP から指定できません。
2. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった手続き，及びトリガの行は削除されます。
3. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE 又は ALTER PROCEDURE を実行して手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
4. WITH PROGRAM を指定して無効にしたトリガの SQL オブジェクトを実行するためには，次のどちらかの操作をする必要があります。
   - 手続きを定義し直して，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成する。
   - 無効にしたトリガを DROP TRIGGER で削除してから，削除した手続きを使用しないように CREATE TRIGGER でトリガを再定義します。ただし，次のすべての条件を満たすトリガがある場合は，それらもすべて DROP TRIGGER で削除し，定義していた順に CREATE TRIGGER で再定義しないと，トリガ動作の実行順序が変わります。

     （条件）

     ・無効にしたトリガよりも定義したのが後である。

     ・無効にしたトリガと定義した表が同じである。

     ・無効にしたトリガとトリガ契機（INSERT，UPDATE，又は DELETE）が同じである（ただし，UPDATE の場合は，トリガ契機列の指定の有無，及び内容に関係なく同じとみなされます）。

     ・無効にしたトリガとトリガ動作時期（BEFORE 又は AFTER）が同じである。

     ・無効にしたトリガとトリガの動作する単位（行単位又は文単位）が同じである。
5. ルーチン識別子には，トリガ動作手続きの識別子は指定できません。トリガを削除する場合には，DROP TRIGGER を実行してください。

3.26.2　DROP PUBLIC PROCEDURE（パブリック手続き削除）

(1)　機能

すべてのユーザから，認可識別子でルーチン識別子を修飾しなくても使用できる手続き（パブリック手続き）を削除します。

(2)　使用権限

手続きの所有者

自分が所有する（定義した）パブリック手続きを削除できます。

DBA 権限を持つユーザ

他ユーザの所有するパブリック手続きを削除できます。

(3)　形式

```
DROP PUBLIC PROCEDURE ルーチン識別子
                      〔WITH PROGRAM〕
```

(4)　オペランド

PUBLIC，ルーチン識別子以外の説明については，「DROP PROCEDURE（手続き削除）」を参照してください。

(a)　PUBLIC

パブリック手続きを削除する場合に指定します。

(b)　ルーチン識別子

削除するパブリック手続きのルーチンの名前を指定します。

(5)　共通規則

1. 次のような場合，Java 手続き中から DROP PUBLIC PROCEDURE は実行できません。
   - 実行中の SQL オブジェクトが無効になる場合，又は削除される場合
   - 実行中の Java 手続きが削除される場合
2. その他の共通規則は，「DROP PROCEDURE（手続き削除）」と同じです。

(6)　留意事項

1. DROP PUBLIC PROCEDURE は，OLTP 下の X/Open に従った UAP から指定できません。
2. その他の留意事項は，「DROP PROCEDURE（手続き削除）」と同じです。

3.27　DROP SCHEMA（スキーマ削除）

3.27.1　DROP SCHEMA の形式と規則

(1)　機能

スキーマ定義で定義したスキーマを削除します。

(2)　使用権限

スキーマの所有者

ユーザ自身のスキーマを削除できます。

DBA 権限を持つユーザ

他ユーザのスキーマを削除できます。

(3)　形式

```
DROP SCHEMA AUTHORIZATION 認可識別子〔WITH PROGRAM〕
```

(4)　オペランド

(a)　認可識別子〔WITH PROGRAM〕

認可識別子

スキーマの所有者の認可識別子を指定します。

WITH PROGRAM

スキーマを削除するときに，指定したスキーマ内の実表，ビュー表，インデクス，抽象データ型，ルーチン，トリガ，及び順序数生成子を使用する，関数，手続き，及びトリガの有効な他ユーザの SQL オブジェクトを無効にしたいときに指定します。

WITH PROGRAM を省略した場合，スキーマ内の実表，ビュー表，インデクス，抽象データ型，手続き，トリガ，及び順序数生成子を使用する，関数，手続き，及びトリガの有効な他ユーザの SQL オブジェクトがあると，そのスキーマは削除できません。

(5)　共通規則

1. 指定した認可識別子のスキーマ内のすべての実表，ビュー表（パブリックビューも含む），インデクス，コメント，アクセス権限，ルーチン（DROP SCHEMA で指定した認可識別子が定義したパブリック手続き，パブリック関数を含む），トリガ，抽象データ型，インデクス型，及び順序数生成子が削除されます。

2. WITH PROGRAM を指定しても，指定したスキーマ内の抽象データ型，関数，手続き，トリガ，及び順序数生成子を使用する，関数，手続き，トリガ，及び順序数生成子の有効な他ユーザの SQL オブジェクトがあると，そのスキーマは削除できません。
3. 指定したスキーマ内の関数を使用する，他ユーザのビュー表，又は他ユーザが定義したパブリックビューがある場合，そのスキーマは削除できません。
4. スキーマ内の抽象データ型を使用する，他ユーザの表又はインデクスがある場合，そのスキーマは削除できません。
5. 次のような場合，Java 手続き中から DROP SCHEMA は実行できません。
   - 実行中の SQL オブジェクトが無効になる場合，又は削除される場合
   - 実行中の Java 手続きが削除される場合
6. インナレプリカ機能を使用している場合の DROP SCHEMA の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
7. 指定したスキーマ内に改竄防止表があり，かつその改竄防止表に行がある場合，そのスキーマは削除できません。
8. DBA 権限を持つユーザは監査人のスキーマを削除できますが，監査人のスキーマ内に監査証跡表がある場合は監査人のスキーマを削除できません。
9. SQL セッション固有一時表を使用している SQL セッション中で，使用中の SQL セッション固有一時表を含むスキーマは削除できません。

(6) 留意事項

1. DROP SCHEMA は，OLTP 下の X/Open に従った UAP から指定できません。
2. WITH PROGRAM を指定して関数，手続き，トリガ，及び順序数生成子の有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった関数，手続き，トリガ，及び順序数生成子の情報は削除されます。
3. WITH PROGRAM を指定して無効にした関数，手続き，トリガ，及び順序数生成子の有効な SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して，関数，手続き，トリガ，及び順序数生成子の有効な SQL オブジェクトを再作成しておく必要があります。
4. 認可識別子に指定したユーザが，他ユーザに付与したスキーマ操作権限はすべて削除されます。

(7) 使用例

ユーザ（認可識別子：USER1）が持っているスキーマを削除します。

```
DROP SCHEMA
    AUTHORIZATION USER1
```

3.28　DROP SEQUENCE（順序数生成子削除）

3.28.1　DROP SEQUENCE の形式と規則

(1)　機能

順序数生成子を削除します。

(2)　使用権限

指定する順序数生成子の所有者

自分が所有する順序数生成子を削除できます。

DBA 権限を持つユーザ

自分が所有する順序数生成子，及び他ユーザが所有する順序数生成子を削除できます。

(3)　形式

```
DROP SEQUENCE 〔認可識別子.〕順序数生成子識別子〔WITH PROGRAM〕
```

(4)　オペランド

(a)　〔認可識別子.〕順序数生成子識別子

認可識別子

順序数生成子を所有するユーザの認可識別子を指定します。

省略した場合，実行するユーザの認可識別子を仮定します。

順序数生成子識別子

削除する順序数生成子の名称を指定します。

WITH PROGRAM

順序数生成子を削除するときに，順序数生成子を使用する手続き，及びトリガの有効な他ユーザの SQL オブジェクトを無効にしたいときに指定します。

WITH PROGRAM を省略した場合，順序数生成子を使用する手続き，及びトリガの有効な他ユーザの SQL オブジェクトがあると，その順序数生成子は削除できません。

(5)　共通規則

1. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から DROP SEQUENCE は実行できません。

(6) 留意事項

1. DROP SEQUENCE は，OLTP 下の X/Open に従った UAP からは指定できません。
2. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった手続き，及びトリガの情報は削除されます。
3. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE 又は ALTER PROCEDURE を実行して手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。

(7) 使用例

順序数生成子（SEQ1）を削除します。

```
DROP SEQUENCE SEQ1
```

3.29 DROP TABLE（表削除）

3.29.1 DROP TABLE の形式と規則

(1) 機能

表を削除します。

(2) 使用権限

指定する表の所有者

自分が所有する表を削除できます。

DBA 権限を持つユーザ

他ユーザの所有する表を削除できます。

(3) 形式

```
DROP TABLE 〔認可識別子.〕表識別子〔WITH PROGRAM〕
```

(4) オペランド

(a) 〔認可識別子.〕表識別子〔WITH PROGRAM〕

認可識別子

表を所有するユーザの認可識別子を指定します。

省略した場合，実行するユーザの認可識別子を仮定します。

表識別子

削除する表の名称を指定します。

WITH PROGRAM

表を削除する場合，その表を SQL 手続き文で使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

WITH PROGRAM を省略した場合，表を使用する手続き，及びトリガの有効な SQL オブジェクト（削除する表が参照する被参照表に，内部的に定義されている参照制約動作を行うためのトリガを除きます）があると，その表は削除できません。

参照表を削除した場合，次の表に示す手続き，及びトリガの有効な SQL オブジェクトを無効にします。

表 3-37　無効になるオブジェクト

作成バージョン	オブジェクトの内容	
	オブジェクトの種類	無効になる条件
07-00 以降	手続き，及びトリガのオブジェクト	参照表が参照する被参照表を使用した UPDATE 文又は，DELETE 文を含む場合
07-00 より前		参照表が参照する被参照表を使用した SQL 文を含む場合

(5)　共通規則

1. 表を削除すると，その表に対して定義したインデクス，ビュー表（パブリックビューも含む），コメント，アクセス権限，及びトリガは，削除されます。

   参照表を削除した場合は，被参照表に内部的に定義されている参照制約動作を行うためのトリガも合わせて削除されます。

2. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から DROP TABLE は実行できません。
3. インナレプリカ機能を使用している場合の DROP TABLE の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
4. 表に定義している LOB 列，又は LOB 属性の列を格納する RD エリアに更新凍結指定を指定している場合，その表は削除できません。
5. 指定した表が改竄防止表で，かつその改竄防止表に行がある場合，表は削除できません。
6. DBA 権限を持つユーザは，他ユーザが所有する表を削除できますが，監査人の所有する監査証跡表は削除できません。
7. 外部キーで参照される被参照表は削除できません。
8. SQL セッション固有一時表を使用している SQL セッション中で，使用中の SQL セッション固有一時表は削除できません。

(6)　留意事項

1. DROP TABLE は，OLTP 下の X/Open に従った UAP から指定できません。
2. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった手続き，及びトリガの情報は削除されます。
3. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE 又は ALTER PROCEDURE を実行して手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
4. WITH PROGRAM を指定して無効にしたトリガの SQL オブジェクトを実行するためには，次のどちらかの操作をする必要があります。

- 表を定義し直して，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成する。
- 無効にしたトリガを DROP TRIGGER で削除してから，削除した表を使用しないように CREATE TRIGGER でトリガを再定義します。ただし，次のすべての条件を満たすトリガがある場合は，それらもすべて DROP TRIGGER で削除し，定義していた順に CREATE TRIGGER で再定義しないと，トリガ動作の実行順序が変わります。

  (条件)

  ・無効にしたトリガよりも定義したのが後である。

  ・無効にしたトリガと定義した表が同じである。

  ・無効にしたトリガとトリガ契機（INSERT，UPDATE，又は DELETE）が同じである（ただし，UPDATE の場合は，トリガ契機列の指定の有無，及び内容に関係なく同じとみなされます）。

  ・無効にしたトリガとトリガ動作時期（BEFORE 又は AFTER）が同じである。

  ・無効にしたトリガとトリガの動作する単位（行単位又は文単位）が同じである。

(7) 使用例

在庫表（ZAIKO）を削除します。

```
DROP TABLE ZAIKO
```

3.30 DROP TRIGGER（トリガ削除）

3.30.1 DROP TRIGGER の形式と規則

(1) 機能

トリガを削除します。

(2) 使用権限

トリガの所有者

自分が所有するトリガを削除できます。

DBA 権限を持つユーザ

自分が所有するトリガ，及び他ユーザの所有するトリガを削除できます。

(3) 形式

```
DROP TRIGGER 〔認可識別子.〕トリガ識別子〔WITH PROGRAM〕
```

(4) オペランド

(a) 〔認可識別子.〕トリガ識別子〔WITH PROGRAM〕

認可識別子

削除するトリガの所有者の認可識別子を指定します。

省略した場合，DROP TRIGGER を実行するユーザの認可識別子を仮定します。

トリガ識別子

削除するトリガの名称を指定します。

WITH PROGRAM

トリガを削除するときに，そのトリガを使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

WITH PROGRAM を省略した場合，トリガを使用する手続き，及びトリガの有効な SQL オブジェクトがあると，そのトリガは削除できません。

(5) 共通規則

1. 次の条件を満たす場合，Java 手続き中から DROP TRIGGER は実行できません。
   - 実行中の SQL オブジェクトが無効になる，又は削除される場合

- 実行中の Java 手続きが削除される場合

(6) 留意事項

1. DROP TABLE は，OLTP 下の X/Open に従った UAP から指定できません。
2. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった手続き，及びトリガの情報は削除されます。
3. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。

(7) 使用例

トリガ（TRIG1）を削除します。

```
DROP TRIGGER TRIG1
```

3.31 DROP 〔PUBLIC〕VIEW（ビュー表の削除，パブリックビュー表の削除）

3.31.1 DROP VIEW（ビュー表の削除）

(1) 機能

ビュー表を削除します。

(2) 使用権限

指定するビュー表の所有者

自分が所有するビュー表を削除できます。

DBA 権限を持つユーザ

他ユーザの所有するビュー表を削除できます。

(3) 形式

```
DROP VIEW 〔認可識別子.〕表識別子〔WITH PROGRAM〕
```

(4) オペランド

(a) 〔認可識別子.〕表識別子〔WITH PROGRAM〕

認可識別子

ビュー表を所有するユーザの認可識別子を指定します。省略した場合，実行するユーザの認可識別子を仮定します。

表識別子

削除するビュー表の名前を指定します。

WITH PROGRAM

ビュー表を削除する場合，そのビュー表を使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

WITH PROGRAM を省略した場合，そのビュー表を使用する手続き，及びトリガの有効な SQL オブジェクトがあると，そのビュー表は削除できません。

(5) 共通規則

1. ビュー表を削除すると，そのビュー表に対するアクセス権限，及び削除するビュー表から定義したビュー表も削除されます。

2. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から DROP VIEW は実行できません。

(6) 留意事項

1. DROP VIEW は，OLTP 下の X/Open に従った UAP から指定できません。
2. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった手続き，及びトリガの情報は削除されます。
3. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE 又は ALTER PROCEDURE を実行して手続き，及びトリガの SQL オブジェクトを再作成しておく必要があります。
4. WITH PROGRAM を指定して無効にしたトリガの SQL オブジェクトを実行するためには，次のどちらかの操作をする必要があります。
   - ビュー表を定義し直して，ALTER TRIGGER 又は ALTER ROUTINE を実行してトリガの SQL オブジェクトを再作成する。
   - 無効にしたトリガを DROP TRIGGER で削除してから，削除したビュー表を使用しないように CREATE TRIGGER でトリガを再定義します。ただし，次のすべての条件を満たすトリガがある場合は，それらもすべて DROP TRIGGER で削除し，定義していた順に CREATE TRIGGER で再定義しないと，トリガ動作の実行順序が変わります。

     (条件)

     ・無効にしたトリガよりも定義したのが後である。

     ・無効にしたトリガと定義した表が同じである。

     ・無効にしたトリガとトリガ契機（INSERT，UPDATE，又は DELETE）が同じである（ただし，UPDATE の場合は，トリガ契機列の指定の有無，及び内容に関係なく同じとみなされます）。

     ・無効にしたトリガとトリガ動作時期（BEFORE 又は AFTER）が同じである。

     ・無効にしたトリガとトリガの動作する単位（行単位又は文単位）が同じである。

(7) 使用例

在庫表（ZAIKO）から定義したビュー表（VZAIKO1）を削除します。

```
DROP VIEW VZAIKO1
```

3.31.2 DROP PUBLIC VIEW（パブリックビューの削除）

(1) 機能

すべてのユーザから，認可識別子で表識別子を修飾しなくても使用できるビュー表（パブリックビュー）を削除します。

(2) 使用権限

指定するパブリックビューの所有者（定義者）

自分が所有する（定義した）パブリックビューを削除できます。

DBA 権限を持つユーザ

他ユーザの所有するパブリックビューを削除できます。

(3) 形式

```
DROP PUBLIC VIEW 表識別子〔WITH PROGRAM〕
```

(4) オペランド

PUBLIC，表識別子以外の説明については，「DROP VIEW（ビュー表の削除）」を参照してください。

(a) PUBLIC

パブリックビューを削除する場合に指定します。

(b) 表識別子

削除するパブリックビューの名前を指定します。

(5) 共通規則

「DROP VIEW（ビュー表の削除）」の共通規則と同じです。

(6) 留意事項

「DROP VIEW（ビュー表の削除）」の留意事項と同じです。

(7) 使用例

在庫表（ZAIKO）から定義したパブリックビュー（PVZAIKO1）を削除します。

```
DROP PUBLIC VIEW PVZAIKO1
```

3.32.1 GRANT DBA（DBA 権限定義），GRANT SCHEMA（スキーマ定義権限定義），GRANT SCHEMA OPERATION（スキーマ操作権限定義），GRANT CONNECT（CONNECT 権限定義），GRANT RDAREA（RD エリア利用権限定義）

(1) 機能

ユーザに DBA 権限，スキーマ定義権限，スキーマ操作権限，CONNECT 権限，私用 RD エリアの利用権限を与えます。

(2) 使用権限

DBA 権限を持つユーザ

DBA 権限，スキーマ定義権限，CONNECT 権限，及び私用 RD エリアの利用権限を与えます。

CONNECT 権限を持つユーザ

パスワードを変更できます。

スキーマの所有者

ユーザ自身のスキーマの操作権限を与えます。

(3) 形式

```
GRANT {DBA   TO 認可識別子〔, 認可識別子〕…
            〔IDENTIFIED  {BY パスワード〔, パスワード〕…|USING OS} 〕
        |SCHEMA  TO 認可識別子〔, 認可識別子〕…
        |SCHEMA OPERATION TO 認可識別子〔, 認可識別子〕…
        |CONNECT TO 認可識別子〔, 認可識別子〕…
            〔IDENTIFIED  {BY パスワード〔, パスワード〕…|USING OS} 〕
        |RDAREA RDエリア名〔, RDエリア名〕…
           TO {認可識別子〔, 認可識別子〕…|  PUBLIC } }
```

(4) オペランド

(a) DBA TO 認可識別子〔, 認可識別子〕…〔IDENTIFIED {BY パスワード〔, パスワード〕…| USING OS}〕

DBA TO

ユーザに DBA 権限を与える場合に指定します。

認可識別子〔，認可識別子〕…

DBA 権限を与えるユーザの認可識別子を指定します。

IDENTIFIED BY パスワード〔，パスワード〕…

DBA 権限を与えるユーザのパスワードを指定します。

既に DBA 権限，又は CONNECT 権限を持つ簡易認証ユーザの場合，指定できません。

IDENTIFIED BY 句とユーザの権限の関係を次の表に示します。

表 3-38　GRANT DBA 文の IDENTIFIED BY 句とユーザの権限の関係

ユーザの権限		GRANT DBA 文の IDENTIFIED BY	
		指定あり	指定なし
既に CONNECT 権限あり	パスワードあり	DBA 権限を与えて，指定されたパスワードに変更	DBA 権限を与える
	パスワードなし	DBA 権限を与えて，指定されたパスワードに変更	KFPA11571 エラー
CONNECT 権限なし		CONNECT 権限，DBA 権限，及び指定されたパスワードを与える	KFPA11571 エラー

IDENTIFIED USING OS

簡易認証ユーザとして DBA 権限を与える場合に指定します。

既に DBA 権限，又は CONNECT 権限を持つ非簡易認証ユーザの場合，指定できません。

(b)　SCHEMA TO 認可識別子〔，認可識別子〕…

SCHEMA TO

ユーザにスキーマ定義権限を与える場合に指定します。

認可識別子〔，認可識別子〕…

スキーマ定義権限を与える場合，ユーザの認可識別子を指定します。

(c)　SCHEMA OPERATION TO 認可識別子〔，認可識別子〕…

SCHEMA OPERATION TO

他ユーザに，自分の所有するスキーマのスキーマ操作権限を与える場合に指定します。

監査人は実行できません。

認可識別子〔，認可識別子〕…

スキーマ操作権限を与える認可識別子を指定します。

DBA 権限，又は CONNECT 権限のないユーザを指定することはできません。

(d) CONNECT TO 認可識別子〔，認可識別子〕…〔IDENTIFIED {BY パスワード〔，パスワード〕… | USING OS}〕

CONNECT TO

ユーザに CONNECT 権限を与える場合に指定します。

CONNECT 権限を与える場合の規則を次に示します。

1. 簡易認証ユーザとして CONNECT 権限を与える場合は，IDENTIFIED USING OS を指定してください。
2. 簡易認証ユーザに対して，パスワードの変更をする指定（IDENTIFIED BY 句を指定，又は IDENTIFIED 句の指定を省略）はできません。
3. 非簡易認証ユーザに対して，簡易認証ユーザに変更する指定（IDENTIFIED USING OS 句の指定）はできません。

認可識別子〔，認可識別子〕…

CONNECT 権限を与えるユーザの認可識別子を指定します。

IDENTIFIED BY パスワード〔，パスワード〕…

CONNECT 権限を与えるユーザのパスワードを指定します。

IDENTIFIED BY 句とユーザの権限の関係を次の表に示します。

表 3-39　GRANT CONNECT 文の IDENTIFIED BY 句とユーザの権限の関係

ユーザの権限			パスワード文字制限定義	パスワード有効期限	GRANT CONNECT 文の IDENTIFIED BY 句	
					指定あり	指定なし
既に CONNECT 権限あり	パスワードあり	DBA 権限あり	－	－	指定されたパスワードに変更	KFPA11571-E エラー
		DBA 権限なし	なし	なし	指定されたパスワードに変更	パスワードなしに変更
				あり	指定されたパスワードに変更	KFPA19634-E エラー
			あり	－	指定されたパスワードに変更	KFPA19634-E エラー
	パスワードなし	DBA 権限なし	なし	－	指定されたパスワードに変更	パスワードなしに変更
			あり	－	指定されたパスワードに変更	KFPA19634-E エラー
CONNECT 権限なし			なし	－	CONNECT 権限とパスワードを与える	パスワードなしで CONNECT 権限だけを与える
			あり	－	CONNECT 権限とパスワードを与える	KFPA19634-E エラー

（凡例）　－：該当しません。

IDENTIFIED USING OS

簡易認証ユーザとして CONNECT 権限を与える場合に指定します。

(e) RDAREA RD エリア名〔，RD エリア名〕…TO {認可識別子〔，認可識別子〕… | PUBLIC}

RDAREA RD エリア名〔，RD エリア名〕

ユーザに利用権限を与える RD エリアの名称を指定します。

認可識別子〔，認可識別子〕…

利用権限を与えるユーザの認可識別子を指定します。

PUBLIC

指定した RD エリアを公用 RD エリアにする場合に指定します。

(5) 共通規則

1. GRANT RDAREA でレプリカ RD エリアに利用権限は与えられません。
2. インナレプリカ機能を使用している場合の GRANT の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
3. パスワード文字制限強化機能を使用している場合，GRANT DBA 文，及び GRANT CONNECT 文実行時にパスワード文字制限のチェックを行います。
4. RD エリア名には，一時表用 RD エリアを指定できません。
5. 副監査人にスキーマ定義権限を与えることはできません。

(6) 留意事項

1. この SQL 文中に，最大 16 個の私用 RD エリアと，最大 1,600 個の認可識別子が指定できます。
2. DBA 権限，又は CONNECT 権限のないユーザには，スキーマ定義権限を与えられません。
3. PUBLIC 指定の RD エリア利用権限のある RD エリアに対して，個別のユーザへ RD エリア利用権限を与えられません。また，個別のユーザに権限を与えた RD エリアは，PUBLIC を指定して公用 RD エリアとして定義できません。
4. DBA 権限を与えると，CONNECT 権限も与えられます。
5. DBA 権限を与えるユーザには，パスワードを指定してください。なお，DBA 権限が与えられていても，パスワードのないユーザは DBA 権限を使用できません。
6. 自分自身に GRANT CONNECT を指定すると，パスワードの変更ができます。この場合，GRANT 実行者は CONNECT 権限だけでパスワードの変更ができます。
7. GRANT は，OLTP 下の X/Open に従った UAP から指定できません。
8. GRANT DBA 文と GRANT CONNECT 文で，複数の認可識別子を指定し，同時に複数のユーザに権限を与える場合，IDENTIFIED BY 句で指定するパスワードは一部だけの省略はできません。パスワードは指定した認可識別子の数だけ指定するか，又は IDENTIFIED BY 句全体を省略してください。ただし，GRANT CONNECT 文では IDENTIFIED BY 句全体を省略した場合，パスワードのない DBA を登録するような指定があるとエラーとなり（表「GRANT DBA 文の IDENTIFIED BY 句と

ユーザの権限の関係」，表「GRANT CONNECT 文の IDENTIFIED BY 句とユーザの権限の関係」を参照），GRANT 文全体が無効になります。この場合，エラーとなるユーザを別の GRANT 文で IDENTIFIED BY 句を指定して実行してください。

9. パスワード文字制限機能使用時に GRANT DBA 文と GRANT CONNECT 文で，複数の認可識別子を指定し，同時に複数のユーザに権限を与える場合，一人でもパスワード文字制限に違反する指定をした場合，エラーとなり（表「GRANT DBA 文の IDENTIFIED BY 句とユーザの権限の関係」，表「GRANT CONNECT 文の IDENTIFIED BY 句とユーザの権限の関係」を参照），GRANT 文全体が無効となります。
10. パスワード無効アカウントロック状態のユーザには GRANT DBA 文で DBA 権限の付与，又はパスワードの変更はできません。GRANT CONNECT 文でパスワード無効アカウントロック状態を解除後，GRANT DBA 文を実行してください。
11. 監査人には，DBA 権限は与えられません。
12. 監査人には，スキーマ操作権限は与えられません。
13. RD エリア利用権限が主監査人にだけ与えられている RD エリアに対して，他ユーザの RD エリア利用権限は与えられません。
14. 既に CONNECT 権限を持つ簡易認証ユーザに対して DBA 権限を与える場合は，USING OS を指定してください。
15. 既にスキーマ操作権限を持つユーザに対して，スキーマ操作権限は与えられません。

(7) 使用例

1. ユーザ（認可識別子：USER1）に DBA 権限を与えます。このとき，パスワードは，PSWD とします。

```
GRANT DBA TO USER1 IDENTIFIED BY PSWD
```

2. ユーザ（認可識別子：USER2）にスキーマ定義権限を与えます。

```
GRANT SCHEMA TO USER2
```

3. ユーザ（認可識別子：USER3）に CONNECT 権限を与えます。このとき，パスワードは，PSWD とします。

```
GRANT CONNECT TO USER3
    IDENTIFIED BY PSWD
```

4. ユーザ（認可識別子：USER3）のパスワードを ABCD に変更します。

```
GRANT CONNECT TO USER3
    IDENTIFIED BY ABCD
```

5. ユーザ（認可識別子：USER4～6）に RD エリア（RDA1，RDA2）の利用権限を与えます。

```
GRANT RDAREA RDA1,RDA2
    TO USER4,USER5,USER6
```

6. RD エリア（RDA3）を公用 RD エリアとして定義します。

```
GRANT RDAREA RDA3 TO PUBLIC
```

7. ユーザ（認可識別子：USER7，USER8）にスキーマ操作権限を与えます。

```
GRANT SCHEMA OPERATION TO USER7, USER8
```

3.32.2 GRANT アクセス権限（アクセス権限定義）

(1) 機能

ユーザに表のアクセス権限を与えます。

(2) 使用権限

表の所有者

自分が所有する実表，及び自分の実表，ビュー表から定義した自分の所有するビュー表に対して，自分が持つアクセス権限をほかのユーザに与えられます。ただし，他人が所有する表に対する SELECT 権限を受け取って，ビュー表を定義すると，以降このビュー表に対して自分が持つアクセス権限を，ほかのユーザに与えることはできません。

(3) 形式

```
GRANT {アクセス権限〔, アクセス権限〕… | ALL〔PRIVILEGES〕}
    ON〔認可識別子.〕表識別子
    TO {認可識別子〔, 認可識別子〕… | PUBLIC}

アクセス権限：：＝ {SELECT | INSERT | DELETE | UPDATE}
```

(4) オペランド

(a) {アクセス権限〔, アクセス権限〕… | ALL〔PRIVILEGES〕}

アクセス権限

認可するアクセス権限を指定します。なお，同じアクセス権限は指定できません。

ALL〔PRIVILEGES〕

指定した表に対して，アクセス権限で指定できるすべてのアクセス権限を与える場合，指定します。

ビュー表に対して指定した場合，ビュー表所有者の持つアクセス権限をすべて与えます。

(b) アクセス権限：：＝ {SELECT | INSERT | DELETE | UPDATE}

SELECT

SELECT 権限を与える場合に指定します。

INSERT

INSERT 権限を与える場合に指定します。

DELETE

DELETE 権限を与える場合に指定します。

UPDATE

UPDATE 権限を与える場合に指定します。

(c) ON〔認可識別子.〕表識別子

認可識別子

アクセス権限を与える表の所有者の認可識別子を指定します。

パブリックビューのアクセス権限を与える場合は，認可識別子に PUBLIC を指定してください。

表識別子

アクセス権限を与える表の名前を指定します。

(d) TO {認可識別子〔, 認可識別子〕… | PUBLIC}

認可識別子

アクセス権限を与えるユーザの認可識別子を指定します。ユーザの認可識別子は，最大 1,600 個指定できます。ただし，同じ認可識別子は指定できません。

PUBLIC

指定したアクセス権限を，すべてのユーザに与える場合に指定します。

(5) 共通規則

1. インナレプリカ機能を使用している場合の GRANT の実行条件については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。
2. 一時表に対するアクセス権限をほかのユーザに与える場合，指定できるのは ALL 〔PRIVILEDGE〕だけです。

(6) 留意事項

1. GRANT は，OLTP 下の X/Open に従った UAP から指定できません。
2. 監査人以外のユーザに監査証跡表又は監査証跡表を基表としたビュー表の INSERT，UPDATE，DELETE 権限を与えることはできません。

(7) 使用例

1. ユーザ（認可識別子：USER1）に在庫表（ZAIKO）を検索する権限（SELECT 権限）を与えます。

```
GRANT SELECT
    ON ZAIKO TO USER1
```

2. すべてのユーザに対して，在庫表（ZAIKO）をアクセスするすべての権限を与えます。

```
GRANT ALL
    ON ZAIKO TO PUBLIC
```

3.33　GRANT　形式 2（監査人のパスワード変更）

3.33.1　GRANT 形式 2 の形式と規則

(1)　機能

監査人のパスワードを変更します。

(2)　使用権限

監査権限を持つユーザ

監査権限を持つユーザが，自分のパスワードを変更できます。

監査権限を持つほかのユーザのパスワードは変更できません。

(3)　形式

```
GRANT AUDIT IDENTIFIED BY パスワード
```

(4)　オペランド

(a)　パスワード

監査人の新しいパスワードを指定します。

英字の大文字，小文字を区別したい場合は，引用符（"）で囲んで指定してください。

(5)　使用例

監査人のパスワードを，「a0h7Fc3」に変更します。

```
GRANT AUDIT IDENTIFIED BY "a0h7Fc3"
```

3.34 REVOKE（権限削除）

3.34.1 REVOKE DBA（DBA 権限削除），REVOKE SCHEMA（スキーマ定義権限削除），REVOKE SCHEMA OPERATION（スキーマ操作権限削除），REVOKE CONNECT（CONNECT 権限削除），REVOKE RDAREA（RD エリア利用権限削除）

(1) 機能

ユーザに与えている DBA 権限，スキーマ定義権限，スキーマ操作権限，CONNECT 権限，私用 RD エリアの利用権限を取り消します。

(2) 使用権限

DBA 権限を持つユーザ

DBA 権限，スキーマ定義権限，CONNECT 権限，及び RD エリアの利用権限を取り消します。

スキーマの所有者

スキーマ操作権限を取り消します。

(3) 形式

```
REVOKE { DBA    FROM 認可識別子〔, 認可識別子〕…
           |SCHEMA  FROM 認可識別子〔, 認可識別子〕…
           |SCHEMA OPERATION FROM 認可識別子〔, 認可識別子〕…
           |CONNECT FROM 認可識別子〔, 認可識別子〕…
           |RDAREA  RDエリア名〔, RDエリア名〕…
              FROM {認可識別子〔, 認可識別子〕…|PUBLIC} }
```

(4) オペランド

(a) DBA FROM 認可識別子〔, 認可識別子〕…

DBA FROM

DBA 権限を取り消す場合に指定します。

認可識別子〔, 認可識別子〕…

DBA 権限を取り消すユーザの認可識別子を指定します。

(b) SCHEMA FROM 認可識別子〔,認可識別子〕…

SCHEMA FROM

スキーマ定義権限を取り消す場合に指定します。

認可識別子〔,認可識別子〕…

スキーマ定義権限を取り消すユーザの認可識別子を指定します。

(c) SCHEMA OPERATION FROM 認可識別子〔,認可識別子〕…

SCHEMA OPERATION FROM

他ユーザに付与したスキーマ操作権限を取り消す場合に指定します。

認可識別子〔,認可識別子〕…

スキーマ操作権限を取り消すユーザの認可識別子を指定します。

(d) CONNECT FROM 認可識別子〔,認可識別子〕…

CONNECT FROM

CONNECT 権限を取り消す場合に指定します。

認可識別子〔,認可識別子〕…

CONNECT 権限を取り消すユーザの認可識別子を指定します。

(e) RDAREA RD エリア名〔,RD エリア名〕…FROM {認可識別子〔,認可識別子〕… | PUBLIC }

RD エリア名〔,RD エリア名〕

利用権限を取り消す RD エリアの名称を指定します。

認可識別子〔,認可識別子〕…

利用権限を取り消すユーザの認可識別子を指定します。

PUBLIC

指定した公用 RD エリアの利用権限を取り消す場合に指定します。

(5) 留意事項

1. 認可していない権限，又は既に取り消している権限を再度，取り消しできます。
2. RD エリアの利用権限を取り消されるユーザがその RD エリア中に表，インデクス，又は順序数生成子を持っている場合，RD エリアの権限を取り消すことはできません。
3. スキーマに表がある場合，スキーマ権限を取り消すことはできません。
4. 最大 16 個の私用 RD エリア名を指定できます。
5. 最大 1,600 個のユーザの認可識別子を指定できます。

6. GRANT の PUBLIC で与えた権限は，REVOKE の PUBLIC で取り消してください。
7. 自分自身の DBA 権限を取り消すことはできません。
8. DBA 権限，又はスキーマを持っているユーザの CONNECT 権限を取り消すことはできません。
9. CONNECT 権限を取り消すと，次に示す権限及び接続制約も併せて取り消されます。
   - スキーマ定義権限
   - 認可識別子を指定して定義した接続制約。接続制約の詳細については「CREATE CONNECTION SECURITY（CONNECT 関連セキュリティ機能の定義）」を参照してください。
10. REVOKE は，OLTP 下の X/Open に従った UAP から指定できません。
11. 監査人のスキーマ定義権限，及び CONNECT 権限を取り消すことはできません。
12. RD エリア名には，一時表用 RD エリアを指定できません。
13. pdplgrgst を実行する場合は，DBA 権限を持つ非簡易認証ユーザが必要です。
14. スキーマ定義権限を取り消すと，付与したスキーマ操作権限も取り消されます。

(6) 使用例

1. ユーザ（認可識別子：USER1）に与えている DBA 権限を取り消します。

```
REVOKE DBA FROM USER1
```

2. ユーザ（認可識別子：USER2）に与えているスキーマ定義権限を取り消します。

```
REVOKE SCHEMA FROM USER2
```

3. ユーザ（認可識別子：USER3）に与えている CONNECT 権限を取り消します。

```
REVOKE CONNECT FROM USER3
```

4. ユーザ（認可識別子：USER4～6）に与えている RD エリア（RDA1，RDA2）の利用権限を取り消します。

```
REVOKE RDAREA RDA1,RDA2
    FROM USER4,USER5,USER6
```

5. 公用 RD エリアとして定義した RD エリア（RDA3）を私用 RD エリアとして再定義します（USER1 に RD エリア利用権限を与えます）。

```
REVOKE RDAREA RDA3 FROM PUBLIC
    GRANT RDAREA RDA3 TO USER1
```

6. ユーザ（認可識別子：USER7，USER8）に与えているスキーマ操作権限を取り消します。

```
REVOKE SCHEMA OPERATION FROM USER7,USER8
```

3.34.2 REVOKE アクセス権限（アクセス権限削除）

(1) 機能

ユーザに与えている表のアクセス権限を取り消します。

(2) 使用権限

表の所有者

GRANT（アクセス権限定義）で与えたアクセス権限を取り消すことができます。

(3) 形式

```
REVOKE {アクセス権限〔, アクセス権限〕… | ALL〔PRIVILEGES〕}
    ON 〔認可識別子.〕表識別子
    FROM {認可識別子〔, 認可識別子〕… | PUBLIC}
         〔WITH PROGRAM〕

アクセス権限：：＝ {SELECT | INSERT | DELETE | UPDATE}
```

(4) オペランド

(a) {アクセス権限〔, アクセス権限〕… | ALL〔PRIVILEGES〕}

アクセス権限

アクセス権限を取り消す場合に指定します。なお，同じアクセス権限は指定できません。

ALL〔PRIVILEGES〕

指定した表に対して，アクセス権限で指定できるすべてのアクセス権限を取り消す場合に指定します。

(b) アクセス権限：：＝ {SELECT | INSERT | DELETE | UPDATE}

SELECT

SELECT 権限を取り消す場合に指定します。

INSERT

INSERT 権限を取り消す場合に指定します。

DELETE

DELETE 権限を取り消す場合に指定します。

UPDATE

UPDATE 権限を取り消す場合に指定します。

(c) ON〔認可識別子.〕表識別子

認可識別子

アクセス権限を取り消す表の所有者の認可識別子を指定します。

パブリックビューのアクセス権限を取り消す場合は，認可識別子に PUBLIC を指定してください。

表識別子

アクセス権限を取り消す表の名前を指定します。

(d) FROM｛認可識別子〔，認可識別子〕…｜PUBLIC｝〔WITH PROGRAM〕

認可識別子

アクセス権限を取り消すユーザの認可識別子を指定します。ユーザの認可識別子は，最大 1,600 個指定できます。ただし，同じ認可識別子は指定できません。

PUBLIC

GRANT の PUBLIC 指定で与えたアクセス権限を取り消す場合に指定します。

WITH PROGRAM

ビュー表の基になる表の SELECT 権限を取り消すときに，SELECT 権限の取消によって削除されるビュー表を使用する手続き，及びトリガの有効な SQL オブジェクトを無効にしたいときに指定します。

(5) 共通規則

1. 自分自身の持っているアクセス権限を取り消すことはできません。
2. 与えていない権限，又は既に取り消した権限を再度取り消すこともできます。
3. GRANT の PUBLIC で与えた権限は，REVOKE の PUBLIC で取り消してください。

   PUBLIC を指定して権限が与えられている場合，特定のユーザのアクセスを許さないようにするためには，PUBLIC を指定して権限を削除し，認可識別子を指定して必要なユーザに対して権限を与え直してください。
4. 表の所有者が他人に与えた SELECT 権限を取り消した場合，その他人がその表を利用して定義したビュー表は削除されます。
5. ビュー定義で使用した表に対するアクセス権限，又はビュー表の基の表に対するアクセス権限を，ビュー表所有者から取り消す場合の規則を次に示します。

   <ビュー表 V1 の定義>

```
CREATE VIEW V1※1(VSCODE,VSNAME,VTANKA)
    AS SELECT SCODE,SNAME,TANKA
        FROM ZAIKO※2 WHERE SCODE =
            (SELECT SCODE FROM JUTYU※3)
```

   <ビュー表 V2 の定義>

```
CREATE VIEW V2※4(VSSCODE,VSNYUKOBI)
    AS SELECT SCODE,NYUKOBI
        FROM NYUKO※5
```

<ビュー表 V1 からのビュー表 VV1 の定義>

```
CREATE VIEW VV1※6(VVSCODE,VVTANKA)
    AS SELECT VSCODE,VTANKA
        FROM V1※1 WHERE SCODE =
            (SELECT VSSCODE FROM V2※4
                WHERE VSNYUKOBI > DATE('1995-09-21'))
```

- ビュー表所有者からビュー定義で使用した表（※2，※3，※5）に対する SELECT 権限を取り消す場合，そのビュー表（※1，※4），及びそのビュー表を使用して定義するビュー表（※6）も削除されます。

  （例 1）

  ビュー表所有者から JUTYU（※3）に対する SELECT 権限を取り消す場合，V1（※1）及び VV1（※6）が削除されます。

  （例 2）

  ビュー表所有者から NYUKO（※5）に対する SELECT 権限を取り消す場合，V2（※4），及び VV1（※6）が削除されます。

- ビュー表所有者からビュー表の基になる表（※2，※5）に対するアクセス権限を取り消す場合，そのビュー表（※1，※4），及びそのビュー表を基に定義するビュー表（※6）に対するアクセス権限も取り消されます。なお，ここでいうアクセス権限とは，SELECT 権限を除いた権限です。

  （例 1）

  ビュー表所有者から ZAIKO（※2）に対するアクセス権限を取り消す場合，V1（※1），及び VV1（※6）に対するアクセス権限も削除されます。

  （例 2）

  ビュー表所有者から NYUKO（※5）に対するアクセス権限を取り消す場合，V2（※4）に対するアクセス権限も削除されます。

  取り消されたアクセス権限を再度与えるには，取り消したビュー表の基になる表に対するアクセス権限をビュー表所有者に再度与えて，そのビュー表を削除してからビュー表を再度定義してください。

  （例 1）の場合，ZAIKO（※2）に対するアクセス権限をビュー表所有者に与えて，V1（※1）を削除してから（V1 を削除すると VV1 も削除されます），V1（※1），VV1（※6）の順で再度定義します。

  （例 2）の場合，NYUKO（※5）に対するアクセス権限をビュー表所有者に与えて，V2（※4）を削除してから V2（※4）を再度定義します。

6. WITH PROGRAM を省略した場合，SELECT 権限の取り消しによって削除される，ビュー表を使用する手続き及びトリガの有効な SQL オブジェクトがあると，その権限は削除できません。

7. 実行中の SQL オブジェクトが無効になる場合，Java 手続き中から REVOKE は実行できません。

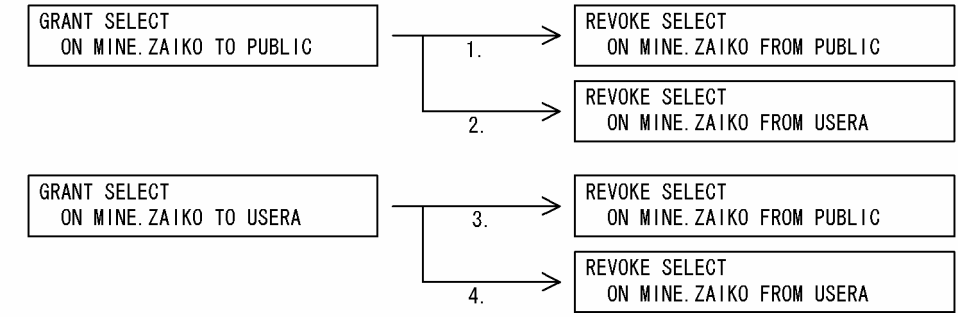
8. 一時表に対するアクセス権限を取り消す場合，指定できるのは ALL ［PRIVILEDGE］だけです。

9. 監査証跡表に対する副監査人のアクセス権を取り消すことはできません。

(6) 留意事項

1. PUBLIC に対して与えた権限，又は特定ユーザに与えた権限の取り消しについては，組み合わせによって結果が異なります。組み合わせを次の図に示します。

図 3-1 PUBLIC に与えた権限，又は特定ユーザに与えた権限の取り消し



［説明］
1. すべてのユーザに与えている権限が取り消されます。
2. USERAに与えている権限はないので，何も取り消されません。
3. すべてのユーザに与えている権限はないので，何も取り消されません。
4. USERAに与えている権限が取り消されます。

2. REVOKE は，OLTP 下の X/Open に従った UAP から指定できません。
3. WITH PROGRAM を指定して手続き，及びトリガの有効な SQL オブジェクトを無効にした場合，ディクショナリ表 SQL_ROUTINE_RESOURCES 中の無効となった手続き，及びトリガの情報は削除されます。
4. WITH PROGRAM を指定して無効にした手続き，及びトリガの SQL オブジェクトを実行するためには，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER を実行して手続き，及びトリガの有効な SQL オブジェクトを再作成しておく必要があります。

(7) 使用例

1. ユーザ（認可識別子：USER1）に与えた在庫表（ZAIKO）を検索する権限 （SELECT 権限）を取り消します。

```
REVOKE SELECT ON ZAIKO FROM USER1
```

2. すべてのユーザに対して与えた在庫表（ZAIKO）をアクセスするすべての権限の中で，DELETE 権限だけを取り消します。

```
REVOKE DELETE ON ZAIKO FROM PUBLIC
```

4 操作系 SQL

この章では，操作系 SQL の構文形式，及び文法について説明します。

4.1　全般規則

4.1.1　操作系 SQL の全般規定

(1)　操作系 SQL の種類と機能

操作系 SQL は，表のデータを操作（検索，追加，削除，更新）するときに使用します。

操作系 SQL の種類と機能を次の表に示します。

表 4-1　操作系 SQL の種類と機能

種　類	機　能
ALLOCATE CURSOR 文 形式 1（文カーソル割当て）	PREPARE 文で前処理した SELECT 文，又は手続きから返却された結果集合の組に対してカーソルを割り当てます。
ALLOCATE CURSOR 文 形式 2（結果集合カーソル割当て）	
ASSIGN LIST 文　形式 1（リスト作成）	実表からリストを作成します。
ASSIGN LIST 文　形式 2（リスト作成）	
CALL 文（手続きの呼び出し）	手続きを呼び出します。
CLOSE 文（カーソルクローズ）	カーソルを閉じます。
DEALLOCATE PREPARE 文（SQL の前処理無効化）	PREPARE 文で前処理した SQL を無効にし，SQL 文識別子，又は拡張文名の割当てを解除します。
DECLARE CURSOR　形式 1（カーソル宣言）	SELECT 文の検索結果を FETCH 文で 1 行ずつ取り出すために，カーソルを宣言します。
DECLARE CURSOR　形式 2（カーソル宣言）	
DELETE 文　形式 1（行削除）	指定した探索条件を満足する行，又はカーソルが指している行を削除します。
DELETE 文　形式 2（配列を使用した行削除）	
準備可能動的 DELETE 文：位置付け（前処理可能なカーソルを使用した行削除）	指定したカーソルが指している行を削除します。動的に実行する場合に使用します。
DESCRIBE 文　形式 1（検索情報，入出力情報の受け取り）	PREPARE 文で前処理した SQL の検索情報，若しくは出力情報，又は入力情報を，SQL 記述領域に返します。
DESCRIBE 文　形式 2（検索情報，入出力情報の受け取り）	
DESCRIBE CURSOR 文（カーソルの検索情報の受け取り）	手続きから返却された結果集合を参照するカーソルの検索情報を，SQL 記述領域に返します。

種　類	機　能
DESCRIBE TYPE 文（ユーザ定義型の定義情報の受け取り）	PREPARE 文で前処理した SQL の検索項目情報に直接又は間接的に含まれるユーザ定義型の定義情報（各属性のデータコード，データ長など）を SQL 記述領域に受け取ります。
DROP LIST 文（リスト削除）	リストを削除します。
EXECUTE 文　形式 1（SQL の実行）	PREPARE 文で前処理した SQL を実行します。
EXECUTE 文　形式 2（配列を使用した SQL の実行）	PREPARE 文で前処理した SQL を，配列を使用して複数行分一括して実行します。
EXECUTE IMMEDIATE 文（SQL の前処理と実行）	文字列で与えられた SQL を，前処理して実行します。
FETCH 文　形式 1（データの取り出し）	取り出す行を示すカーソルの位置を次の行に進め，その行の列の値を INTO 句で指定した埋込み変数に読み込みます。
FETCH 文　形式 2（データの取り出し）	検索結果中の 1 行，又は複数行を，SQL 記述領域に指定した受け取り領域に読み込みます。
FETCH 文　形式 3（データの取り出し）	検索結果中の複数行を，INTO 句で指定した埋込み変数に一度に読み込みます。
FREE LOCATOR 文（位置付け子の無効化）	位置付け子を無効にします。
INSERT 文　形式 1（行挿入）	表に，行を列単位で挿入します。直接，値を指定して一つの行の挿入ができます。また，問合せ式本体を使用して，一つ，又は複数の行の挿入もできます。
INSERT 文　形式 2（行挿入）	行全体を一つのデータとみなして，FIX 属性の表に，行を行単位で挿入します。直接，値を指定して一つの行の挿入ができます。また，問合せ式本体を使用して，一つ，又は複数の行の挿入もできます。
INSERT 文　形式 3（配列を使用した行挿入）	配列形式の埋込み変数を指定して複数の行の挿入ができます。表に，行を列単位で複数行挿入します。
INSERT 文　形式 4（配列を使用した行挿入）	配列形式の埋込み変数を指定して複数の行の挿入ができます。行全体を一つのデータとみなして，FIX 属性の表に，行を行単位で複数行挿入します。
OPEN 文　形式 1（カーソルオープン）	カーソルを開きます。DECLARE CURSOR で宣言したカーソル，又は ALLOCATE CURSOR 文で割り当てたカーソルを，検索結果の先頭の行の直前に位置づけて，検索結果を取り出せる状態にします。
OPEN 文　形式 2（カーソルオープン）	
PREPARE 文（SQL の前処理）	文字列で与えられた SQL を実行するための前処理をしてその SQL に名称（SQL 文識別子，又は拡張文名）を付けます。
PURGE TABLE 文（全行削除）	実表中のすべての行を削除します。
1 行 SELECT 文（1 行検索）	表のデータを検索します。表から 1 行だけデータを取り出す場合は，カーソルを使用しないでデータを取り出す 1 行 SELECT 文を指定します。
動的 SELECT 文　形式 1（動的検索）	表のデータを検索します。動的 SELECT 文は，PREPARE 文で前処理します。検索するときは，DECLARE CURSOR によってカーソルを宣言するか，ALLOCATE CURSOR 文によってカーソルを割り当ててから，そのカーソルを使用して検索結果を 1 行ずつ取り出します。
動的 SELECT 文　形式 2（動的検索）	リストを介して表を検索します。

種　類	機　能
UPDATE 文　形式 1（データ更新）	表内の，指定した探索条件を満足する行，又はカーソルが指している行の指定した列の値を更新します。
UPDATE 文　形式 2（データ更新）	表内の，指定した探索条件を満足する行，又はカーソルが指している行の値を行単位で更新します。
UPDATE 文　形式 3（配列を使用した行更新）	配列形式の埋込み変数を指定して，複数の更新処理を実行できます。表内の，指定した探索条件を満足する行の値を列単位で複数回更新します。
UPDATE 文　形式 4（配列を使用した行更新）	配列形式の埋込み変数を指定して，複数の更新処理を実行できます。FIX 指定の表内の，指定した探索条件を満足する行の値を行単位で複数回更新します。
準備可能動的 UPDATE 文：位置付け 形式 1（前処理可能なカーソルを使用したデータ更新）	表内のカーソルが指している行の指定した列を更新します。PREPARE 文で前処理してから EXECUTE 文で実行，又は EXECUTE IMMEDIATE 文で前処理と実行を一度に行う場合に使用します。
準備可能動的 UPDATE 文：位置付け 形式 2（前処理可能なカーソルを使用したデータ更新）	FIX 指定の表内のカーソルが指している行の指定した列を更新します。PREPARE 文で前処理してから EXECUTE 文で実行，又は EXECUTE IMMEDIATE 文で前処理と実行を一度に行う場合に使用します。
代入文　形式 1（SQL 変数，又は SQL パラメタへの値の代入）	SQL 変数，又は SQL パラメタに値を代入します。
代入文　形式 2（埋込み変数，又は？パラメタへの値の代入）	埋込み変数，又は？パラメタに値を代入します。

(2)　留意事項

リードレプリカ機能を使用する場合の参照 DB 及び更新 DB での操作系 SQL の実行可否を次の表に示します。リードレプリカ機能の詳細は，マニュアル「HiRDB システム運用ガイド」の「リードレプリカ機能の運用」を参照してください。

表 4-2　参照 DB 及び更新 DB での操作系 SQL の実行可否

操作系 SQL	参照 DB での実行可否	更新 DB での実行可否
DELETE 文	×	○
INSERT 文	×	○
PURGE TABLE 文	×	○
UPDATE 文	×	○
上記以外	○	○

（凡例）

○：実行できます。

×：実行できません。

4.2 ALLOCATE CURSOR 文 形式 1（文カーソル割当て）

4.2.1 ALLOCATE CURSOR 文 形式 1 の形式と規則

(1) 機能

PREPARE 文で前処理した SELECT 文（動的 SELECT 文）に対してカーソルを定義して割当てます。

(2) 使用権限

なし。

(3) 形式 1 ＜ PREPARE 文で前処理した SELECT 文（動的 SELECT 文）に対するカーソル割当て＞

```
ALLOCATE 拡張カーソル名 CURSOR 〔WITH HOLD〕FOR 拡張文名
```

(4) オペランド

(a) 拡張カーソル名

割当てるカーソルの拡張カーソル名を指定します。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(b) 〔WITH HOLD〕

ホールダブルカーソルとして割当てる場合に指定します。ただし，次の場合，ホールダブルカーソルは使用できません。

- FROM 句に，ホールダブルカーソルに未対応のプラグイン※を使用した抽象データ型を含む表を指定した場合
- ホールダブルカーソルに未対応のプラグイン※を使用した関数呼出しを指定して導出した，名前付きの導出表に対する問合せ
- リストを介した検索
- ON COMMIT DELETE ROWS を指定して定義した一時表に対する問合せ

なお，ホールダブルカーソルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注※

プラグインのホールダブルカーソルへの対応状況については，ディクショナリ表 SQL_PLUGINS の PLUGIN_HOLDABLE 列で確認できます。ディクショナリ表 SQL_PLUGINS については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(c) 拡張文名

有効範囲内で，PREPARE 文で前処理した SELECT 文を識別する拡張文名を指定します。

拡張文名については，「拡張文名」を参照してください。

(5) 共通規則

1. 割当てられたカーソルは，閉じた状態になります。
2. 同じ SELECT 文を識別する拡張文名に対して，複数のカーソルを割当てた場合，WITH HOLD 指定のあるカーソル割当てと，WITH HOLD 指定のないカーソル割当てを混在させることはできません。

(6) 留意事項

1. 指定した拡張カーソル名が既にその有効範囲内で割当てられている場合はエラーとなります。

(7) 使用例

1. 在庫表（ZAIKO）から，行を 1 行ずつ取り出すためにカーソル（拡張カーソル名 cr（有効範囲：GLOBAL））を割当てます。

```
PREPARE GLOBAL :sel FROM  ‘SELECT * FROM ZAIKO’
ALLOCATE GLOBAL :cr CURSOR FOR GLOBAL :sel
```

2. 在庫表（ZAIKO）から，カーソル（拡張カーソル名：cr（有効範囲：GLOBAL，値：’CR1’））を使用してすべての行を検索しながら，動的にカーソル位置の行の単価（TANKA）を 1 割引に更新します。

```
<埋込み変数selに任意の名称を設定>
PREPARE GLOBAL :sel FROM  ‘SELECT * FROM ZAIKO FOR UPDATE’
＜埋込み変数crに’CR1’を設定＞
ALLOCATE GLOBAL :cr CURSOR FOR GLOBAL :sel
PREPARE PRE1 FROM
 ‘UPDATE SET TANKA = ＜単価の 1 割引の値＞ WHERE CURRENT OF GLOBAL CR1’
OPEN GLOBAL :cr
FETCH GLOBAL :cr INTO ＜各列を取り出す変数名＞
EXECUTE PRE1
CLOSE GLOBAL :cr
DEALLOCATE PREPARE GLOBAL :sel
```

4.3 ALLOCATE CURSOR 文 形式 2（結果集合カーソル割当て）

4.3.1 ALLOCATE CURSOR 文 形式 2 の形式と規則

(1) 機能

手続きから返却された順序付けられた結果集合の組に対してカーソルを割り当てます。

(2) 使用権限

なし。

(3) 形式 2 ＜手続きから返却された結果集合の組に対するカーソル割当て＞

```
ALLOCATE 拡張カーソル名 FOR
      PROCEDURE 〔認可識別子.〕ルーチン識別子
```

(4) オペランド

(a) 拡張カーソル名

割り当てるカーソルの拡張カーソル名を指定します。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(b) 〔認可識別子.〕ルーチン識別子

認可識別子

カーソルを割り当てる結果集合を返却した手続きの，所有者の認可識別子を指定します。

パブリック手続きに対してカーソルを割り当てる場合は，認可識別子に引用符（"）で囲んだ大文字の PUBLIC を指定します。

ルーチン識別子

カーソルを割り当てる結果集合を返却した手続きの，ルーチンの名前を指定します。

(5) 共通規則

1. 現行 SQL セッション（HiRDB の接続から切断まで）内で既に呼び出した手続きを指定してください。
2. 現行 SQL セッション内で指定した手続きを 2 回以上呼び出した場合は，最後に呼び出したときに手続きが返却した結果集合の組にカーソルを割り当てます。
3. 指定した手続きのうち，最後に呼び出した手続きが返却した結果集合の組に，既にカーソルを割り当てている場合はエラーとなります。

4. 指定した手続きが一つも結果集合を返却しない場合，次のリターンコードが設定されます。
   - SQL 連絡領域の SQLCODE 領域に 100
   - SQLCODE 変数に 100
   - SQLSTATE 変数に’02001’
5. ALLOCATE CURSOR 文を実行すると，カーソルは，手続きから返却された結果集合のうち，最初の結果集合を参照し，FETCH 文によってその結果集合のデータを取り出せます。2 個目以降の結果集合を参照するには，CLOSE 文を実行します。CLOSE 文を実行し，次のリターンコードが設定された場合，次の結果集合が存在し，カーソルは次の結果集合を参照します。
   - SQL 連絡領域の SQLCODE 領域に 121
   - SQLCODE 変数に 121
   - SQLSTATE 変数に’0100D’

   また，CLOSE 文を実行し，次のリターンコードが設定された場合，次の結果集合は存在しません。この場合，拡張カーソル名はどのカーソルも識別しなくなります。
   - SQL 連絡領域の SQLCODE 領域に 100
   - SQLCODE 変数に 100
   - SQLSTATE 変数に’02001’
6. 割り当てられたカーソルの定義は，参照している結果集合を生成した手続き中のカーソル宣言と同じになります。
7. 割り当てられたカーソルは，開いた状態となります。
8. 割り当てられたカーソルの位置は，手続き終了時のカーソル位置となります。
9. パブリック手続きに対してカーソルを割り当てる場合は，認可識別子に引用符（"）で囲んだ大文字の PUBLIC を必ず指定してください。

(6) 留意事項

1. 指定した拡張カーソル名が既にその有効範囲内で割り当てられている場合はエラーとなります。

(7) 使用例

使用例については，「結果集合返却機能」を参照してください。

4.4 ASSIGN LIST 文　形式 1（リスト作成）

4.4.1 ASSIGN LIST 文 形式 1 の形式と規則

(1) 機能

実表からリストを作成します。

(2) 使用権限

実表に対する SELECT 権限を持つユーザが，その実表からリストを作成できます。

(3) 形式 1 ＜実表からのリストの作成＞

```
ASSIGN LIST リスト名 FROM （〔認可識別子.〕表識別子）
          〔WHERE 探索条件〕
          〔WITHOUT LOCK 〔 {WAIT｜NOWAIT} 〕〕
          〔 {WITH ROLLBACK｜NO WAIT} 〕
```

(4) オペランド

(a) リスト名

作成するリストの名前を指定します。

既にあるリスト名を指定した場合，そのリストを削除し新しく作成したリストを有効とします。

(b) （〔認可識別子.〕表識別子）

認可識別子には，表の所有者の認可識別子を指定します。

表識別子には，作成するリストの基になる実表の名前を指定します。

表識別子には，以下の表は指定できません。

- 共用表
- WITHOUT ROLLBACK を指定した表
- ビュー表
- 既定文字集合以外の文字集合を指定した列を含む表
- 一時表

(c) 〔WHERE 探索条件〕

探索条件には，取り出す行を選択する条件を指定します。

探索条件を省略した場合は，指定した表のすべての行がリストになります。

ASSIGN LIST 文の探索条件には，論理演算子として ANDNOT も指定できます。論理演算の評価の優先順位は，AND と同じです。

ANDNOT 論理演算をした場合の述語の結果を次に示します。

左辺	右辺		
	T	F	?
T	F	T	T※
F	F	F	F
?	F	?	?

（凡例）

T：真であることを示します。

F：偽であることを示します。

？：不定であることを示します。

注※

C1=V1 ANDNOT C2=V2 は，C1=V1 AND NOT C2=V2 と等価ではありません。

述語 1＝（述語 2 ANDNOT 述語 3）とすると，述語 1 を満たす集合は，述語 2 を満たす集合と述語 3 を満たす集合の差となります。

述語 1＝（述語 2 AND NOT 述語 3）とすると，述語は述語 2 と NOT 述語 3 との AND 論理演算をした結果になります。

したがって，AND NOT の場合，上記真理値表の※は？（不定）となります。

探索条件についての規則を次に示します。

1. 探索条件中に，次のものは指定できません。
   - 副問合せ（NOT 指定のない IN 述語中での，外への参照のない表副問合せは指定できます）
   - コストベース最適化モード 1 の場合，四則演算，日付演算，時刻演算，連結演算，スカラ関数，CASE 式，及び CAST 指定
   - コストベース最適化モード 1 の場合，列指定　比較演算子　列指定
   - 両辺が列指定を含まない値式だけから成る述語
   - コストベース最適化モード 1 の場合，値式 1 が列指定で，値式 2，又は値式 3 が列指定の BETWEEN 述語
   - コンポネント指定
   - コストベース最適化モード 1 の場合，関数呼出し（インデクスを使用するロジックを実装したプラグイン提供関数は指定できます）

- IS FALSE，IS UNKNOWN の論理述語（コストベース最適化モード 2 の場合で，値式がインデクス型プラグイン専用関数ではなく，かつ，値式中に列指定を含むときは指定できます）
- XMLPARSE，及び列指定の XML 問合せ文脈項目の指定がない XMLQUERY，XMLEXISTS 述語
- 予備列
- OR 論理演算子の中にインデクスを使用しないで評価する条件，又はインデクスを一部使用しない条件を指定する場合，その OR 論理演算子の中に次のどれかを指定している。
  ・構造化繰返し述語
  ・インデクス型プラグイン専用関数
  ・XMLEXISTS 述語（XQuery に hi-fn:contains を指定）
- BLOB 型，又は定義長が 32,000 バイトより大きい BINARY 型のどちらかの列

2. 探索条件中の列名には，表名，及び相関名を指定できません。
3. コストベース最適化モード 1 の場合，探索条件中に繰返し列を指定した場合は，添字として ANY を指定する必要があります。
4. コストベース最適化モード 1 の場合，探索条件で指定したすべての列にインデクス（IS NULL 述語に指定した列に対して，除外キーを持つインデクスを除く）が定義されていなければなりません。コストベース最適化モード 2 では，すべての列にインデクスを定義する必要はありません。ただし，すべての列にインデクスを定義した方が ASSIGN LIST 文の性能がよいため，インデクスを定義することを推奨します。
5. 構造化繰返し述語に対しては，コストベース最適化モードに関係なく，構造化繰返し述語の探索条件中に指定した，すべての繰返し列を構成列に含む複数列インデクスが定義されていなければなりません。
6. 繰返し列と繰返し列以外の列の両方を構成列に含むインデクスを使用して，繰返し列でない列の述語を評価する場合，繰返し列のどれかの列に条件を指定していればインデクスは使用できます。
7. 表副問合せを指定した IN 述語中の列には，次のどちらかのインデクスが定義されていなければなりません。なお，表副問合せを指定した IN 述語中の列を先頭とする複数列インデクスは，SQL 最適化モードがコストベース最適化モード 2 の場合に限ります。
   - 単一列インデクス
   - 表副問合せを指定した IN 述語中の列を先頭とする複数列インデクス

   表副問合せを指定した IN 述語中の列が複数列インデクスの先頭でない場合でも，この列より前のインデクス構成列に，比較述語（＝），NULL 述語（IS NULL），又は右辺が値指定の IN 述語（IN）のどれかが指定されていればかまいません。ただし，繰返し列をインデクス構成列に含むインデクスは除きます。また，右辺が値指定の IN 述語（IN）の場合は，値指定の個数がシステム共通定義 pd_apply_search_ats_num に対して，次のどちらかの条件を満たしている必要があります。
   - IN が 1 列だけの場合，値指定の個数は⌊pd_apply_search_ats_num の値÷44⌋以下
   - IN が 2 列以上の場合，それぞれの列に指定した値指定の個数の積が⌊pd_apply_search_ats_num の値÷44⌋以下

8. コストベース最適化モード 1 の場合，探索条件を指定しない場合は，リストの基になるすべての表について，繰返し列以外のどれかの列にインデクス（プラグインインデクス，除外キーを持つインデクスを除く）が定義されていなければなりません。
9. コストベース最適化モード 1 の場合，繰返し列を指定した述語は論理演算子 NOT で否定できません。
10. ANDNOT 論理演算を含む述語は論理演算子 NOT で否定できません。
11. 副問合せ中の探索条件，及び探索 CASE 式の WHEN の探索条件には ANDNOT は指定できません。
12. ANDNOT 論理演算の左右のどちらかにインデクスを使用しないで評価する条件，又はインデクスを一部使用しないで評価する条件を指定している場合，ANDNOT は指定できません。
13. 値式がインデクス型プラグイン専用関数の論理述語は NOT で否定できません。
14. 表副問合せを指定した IN 述語は論理演算子 NOT で否定できません。

(d) 〔WITHOUT LOCK 〔{WAIT | NOWAIT}〕〕

このオペランドを省略した場合は，一度検索したデータの内容をトランザクションが終了するまで保証します。

WITHOUT LOCK WAIT

一度検索したデータの内容をトランザクション終了まで保証する必要がない場合に指定します。

WITHOUT LOCK〔WAIT〕を指定した場合，HiRDB はトランザクション終了を待たないで排他制御を解除するので，同時実行性を向上できます。

WITHOUT LOCK NOWAIT

ほかのユーザが更新中のデータでも，更新の完了を待たないで参照し，一度検索したデータの内容をトランザクション終了まで保証する必要がない場合に指定します。

WITHOUT LOCK NOWAIT を指定した場合，HiRDB は排他制御をしません。更新中の表に対しても排他待ちしないで検索できるので，より同時実行性を向上できます。しかし，更新中の表を検索した場合は，タイミングによっては不正なデータが得られる場合があります。

(e) 〔{WITH ROLLBACK | NO WAIT}〕

このオペランドを省略した場合は，検索する表をほかのユーザが使用中ならば，使用中のユーザのトランザクションが終了するまで待ち，その後実行します（WITHOUT LOCK NOWAIT 指定の場合を除く）。

WITH ROLLBACK

検索する表をほかのユーザが使用中の場合，トランザクションを取り消して無効にしたいときに指定します。

NO WAIT

検索する表をほかのユーザが使用中の場合，トランザクションを取り消さないでエラーにしたいときに指定します。ただし，この SQL の実行中に掛けた排他は解除されません。

(5) 共通規則

1. リストは所有者だけ使用でき，権限定義，権限削除の対象とはなりません。実行ユーザとリストの所有者が異なる場合は，SET SESSION AUTHORIZATION 文を使用して，実行ユーザを変更してください。
2. リストは HiRDB から切り離しをしても削除されません。
3. リストは HiRDB 停止時に自動的に削除されます。
4. 障害発生時には，リストは次の条件のとおり検索できなくなるため（検索するとエラー），検索できないリストは削除，又は再作成する必要があります。

   条件 1

   - すべてのユニットが異常終了した場合

     すべてのリストが自動的に削除されます。

   条件 2

   - 一部のユニットが異常終了した場合

     異常終了したユニットで作成したリストが検索できなくなります。
5. リストがある間，リストの基表に対して次の操作はできません。
   - スキーマの削除
   - 表の削除
   - 表の定義変更
6. リストを作成した場合，作成した行数を SQLCA の SQLERRD[2]に返します。
7. リストは，SQL ルーチン中には指定できません。
8. ASSIGN LIST 文は動的に実行できますが，直接ホストプログラムに埋め込んで実行できません。
9. 同一ユーザが，複数同時に HiRDB と接続してリストを操作できません。
10. 一つのリスト用 RD エリアに格納できるリストの最大数は，データベース初期設定ユティリティ（pdinit），又はデータベース構成変更ユティリティ（pdmod）で指定する最大リスト登録数（500～50,000）で決まります。

(6) 留意事項

1. リストは基表と同一サーバ上のリスト用 RD エリアに作成します。あらかじめリスト用 RD エリアを準備しておいてください。
2. リストは HiRDB から切り離しをしても削除されません。不要になったリストは，DROP LIST 文で削除してください。
3. リストがある間，リストの基表に対して次の操作をした場合は，リストを作り直してください。
   - PURGE TABLE 文
   - 表の再編成

- 作成モードのデータロード
- RD エリアの再初期化

(7) 使用例

使用例については，「ASSIGN LIST 文　形式 2（リスト作成）」の使用例を参照してください。

4.5　ASSIGN LIST 文　形式 2（リスト作成）

4.5.1　ASSIGN LIST 文 形式 2 の形式と規則

(1)　機能

リストの複写，又はリスト間の集合演算をして，結果をリストとして作成します。また，リストの名前も変更できます（リスト名変更）。

(2)　使用権限

リストの所有者

自分の所有するリストからリストを作成できます。

(3)　形式 2 ＜リストからのリストの作成，リストの名前の変更＞

```
ASSIGN LIST リスト名 FROM リスト名１
          〔 { { AND｜OR｜AND NOT｜ANDNOT}  リスト名２｜FOR ALTER LIST} 〕
```

(4)　オペランド

(a)　リスト名

作成するリストの名前を指定します。

既にあるリスト名を指定した場合，そのリストを削除して，新しく作成したリストを有効とします。

(b)　リスト名１ 〔{{AND ｜ OR ｜ AND NOT ｜ ANDNOT}　リスト名２｜ FOR ALTER LIST}〕

リスト名１，リスト名２に指定するリストの基表は，同じでなければなりません。

リスト名，リスト名１，及びリスト名２に指定するリストの名前はすべて異なっていなければなりません。

リスト名１

リストの複写をする場合に指定します。

リスト名１ AND リスト名２

リスト間の積集合を求める場合に指定します。

リスト名１ OR リスト名２

リスト間の和集合を求める場合に指定します。

リスト名1 { AND NOT | ANDNOT } リスト名2

リスト間の差集合を求める場合に指定します。

リスト名1 FOR ALTER LIST

リストの名前を変更する場合に指定します。

(5) 共通規則

形式1の共通規則と同じです。

(6) リストの名前を変更する場合の規則

リストの名前を変更した場合，システム共通定義 pd_list_rowcount_in_rename オペランドの指定値に従って次の値を SQLCA の SQLERRD[2]に返します。

表 4-3 リスト名変更時に SQLCA の SQLERRD[2]に返却する値

システム共通定義 pd_list_rowcount_in_rename オペランドの指定値	SQLCA の SQLERRD[2]への返却値
Y	リストの行数
N	0

(7) 留意事項

形式1の留意事項と同じです。

(8) 使用例

表T1

C1	C2	CHECK1	CHECK2
A	1	Y	Y
B	2	Y	Y
C	3	Y	Y
D	4	Y	N
E	5	Y	N
F	6	N	Y
G	7	N	N
H	8	N	N

1. 表 T1 から列 CHECK1 が Y のリスト（LIST1）を作成します。

```
ASSIGN LIST LIST1 FROM (T1) WHERE CHECK1=’Y’
```

↓

LIST1

C1	C2	CHECK1	CHECK2
A	1	Y	Y
B	2	Y	Y
C	3	Y	Y
D	4	Y	N
E	5	Y	N

2. 表 T1 から列 CHECK2 が Y のリスト（LIST2）を作成します。

```
ASSIGN LIST LIST2 FROM (T1) WHERE CHECK2=’Y’
```

↓

LIST2

C1	C2	CHECK1	CHECK2
A	1	Y	Y
B	2	Y	Y
C	3	Y	Y
F	6	N	Y

3. LIST1 と LIST2 の積集合を求めて，リスト（LIST3）を作成します。

```
ASSIGN LIST LIST3 FROM LIST1 AND LIST2
```

↓

LIST3

C1	C2	CHECK1	CHECK2
A	1	Y	Y
B	2	Y	Y
C	3	Y	Y

4.6 CALL 文（手続きの呼び出し）

4.6.1 CALL 文の形式と規則

(1) 機能

手続きを呼び出します。

(2) 使用権限

DBA 権限又は CONNECT 権限を持つユーザ

手続きを呼び出せます。ただし，手続き中の各 SQL 文が実行されるときに，それぞれの SQL の実行権限が必要です。

(3) 形式

```
CALL 〔認可識別子.〕ルーチン識別子
      （〔引数〔, 引数〕…〕）
引数：：＝ { {IN｜OUT｜INOUT}  ：埋込み変数〔：標識変数〕
           ｜〔 {IN｜OUT｜INOUT} 〕  {SQL変数｜SQLパラメタ｜？パラメタ}
           ｜〔IN〕 値式}
```

(4) オペランド

(a) 〔認可識別子.〕ルーチン識別子

認可識別子

呼び出す手続きの所有者の認可識別子を指定します。

パブリック手続きを呼び出す場合は，認可識別子に PUBLIC を指定します。

ルーチン識別子

呼び出す手続きのルーチンの名前を指定します。

(b) 引数

```
：：＝ { {IN｜OUT｜INOUT}  ：埋込み変数〔：標識変数〕
       ｜〔 {IN｜OUT｜INOUT} 〕  {SQL変数｜SQLパラメタ｜？パラメタ}
       ｜〔IN〕 値式}
```

呼び出す手続きのパラメタに対する引数を指定します。IN，OUT，又は INOUT は，CREATE PROCEDURE で指定した手続きのパラメタの入出力モード（パラメタモード）を指定します。

IN，OUT，及び INOUT の指定規則を次の表に示します。

表 4-4　IN，OUT，及び INOUT の指定規則

指定する引数の種別	CALL 文の種別		
	動的実行	UAP 埋込み	ルーチン定義中
：埋込み変数［：標識変数］	×	○※2	×
SQL 変数，SQL パラメタ	×	×	○※1
SQL 変数，SQL パラメタを基にしたコンポネント指定	×	×	○※1
？パラメタ	○※1	×	×
上記以外の値式	○※3	○※3	○※3

（凡例）

○：指定できます。

×：指定できません。

注※1

IN，OUT，又は INOUT を指定できます。また，省略もできます。

注※2

IN，OUT，又は INOUT を必ず指定します。

注※3

IN を指定できます。また，省略もできます。

(5)　共通規則

1. 引数は，指定した順序でパラメタと対応します。
2. 引数のデータ型は，対応するパラメタのデータ型と互換性がなければなりません。
3. 対応するパラメタのパラメタモードが OUT，又は INOUT の場合で，そのパラメタ値として NULL 値が出力される場合，受け取る引数には標識変数が必要です。
4. パラメタモードが IN，OUT，又は INOUT のパラメタに対する引数として指定した?パラメタは，それぞれ入力?パラメタ，出力?パラメタ，又は入力?パラメタかつ出力?パラメタになります。

   値式中の？パラメタは，入力？パラメタになります。
5. 入力パラメタ，及び出力パラメタには，BOOLEAN 型を指定できません。
6. 埋込み変数（標識変数），？パラメタは単純構造にしてください。
7. パラメタモードが IN の引数に，次の指定は記述できません。
   - 集合関数を含む値式
   - ウィンドウ関数を含む値式
   - 列指定を含む値式
   - 抽象データ型列の属性を示したコンポネント指定を含む値式

8. パラメタモードが OUT 又は INOUT の引数に，抽象データ型列の属性を示したコンポネント指定は記述できません。
9. 引数には，結果のデータ型が BLOB，又は最大長が 32,001 バイト以上の BINARY となるスカラ関数 SUBSTR を，単独の値式として指定できません。
10. 引数に指定する値式中に，副問合せは指定できません。
11. 手続き定義で DYNAMIC RESULT SETS 句に 1 以上を指定した手続きを呼び出した場合，手続きから次の表に示す結果集合の組が返却されます。ただし，次の表に示す結果集合が DYNAMIC RESULT SETS 句に指定した数より多い場合は，DYNAMIC RESULT SETS 句に指定した数までの結果集合だけが返却されます。

表 4-5　手続きから返却される結果集合とその順序

手続きの種類	返却される結果集合	返却順序
Java 手続き	手続き定義の外部ルーチン指定で指定した Java メソッドの java.sql.ResultSet[]型のパラメタに設定した結果集合	パラメタ指定順序
SQL 手続き	手続き中で宣言された結果集合カーソルのうち，手続き終了時に開いた状態であるカーソルの結果集合	カーソルをオープンした順序

12. 呼び出した手続きが結果集合を返却した場合は，次のリターンコードが設定されます。
    - SQL 連絡領域の SQLCODE 領域に 120
    - SQLCODE 変数に 120
    - SQLSTATE 変数に’0100C’
      ただし，手続き終了時に開いた状態である結果集合カーソルの数が手続き定義の DYNAMIC RESULT SETS 句に指定した数より多い場合は，SQLSTATE 変数には’0100E’が設定されます。

(6)　留意事項

1. インデクスの追加又は削除によって，手続きの SQL オブジェクト中のインデクス情報が無効になった場合は手続きを実行できなくなるため，手続きの SQL オブジェクトを再作成する必要があります。
2. PURGE TABLE 文，COMMIT 文，及び ROLLBACK 文を使用した手続きは，次の環境では実行できません。
    - OLTP 下の UAP から手続きを呼び出す場合
3. 認可識別子を省略した場合の仮定値については，「スキーマパス」を参照してください。

4.7 CLOSE 文（カーソルクローズ）

4.7.1 CLOSE 文の形式と規則

(1) 機能

カーソルを閉じ，FETCH 文による検索結果の取り出しを終わらせます。

(2) 使用権限

なし。

(3) 形式

```
CLOSE {カーソル名 | 拡張カーソル名}
```

(4) オペランド

(a) {カーソル名 | 拡張カーソル名}

カーソル名

閉じるカーソル名を指定します。閉じるカーソルとは，OPEN 文で開いているカーソルです。

拡張カーソル名

閉じるカーソルを識別する拡張カーソル名を指定します。閉じるカーソルとは，OPEN 文で開いているカーソル，又は手続きから返却された結果集合の組に割り当てられ，結果集合を参照しているカーソルです。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(5) 共通規則

1. 次の SQL を実行した場合，その時点で開いているカーソルはすべて閉じられます。また，暗黙的ロールバックありのエラーが発生した場合にも，カーソルはすべて閉じられます。
   - 定義系 SQL（クライアント環境定義 PDCMMTBFDDL に YES を指定している場合）
   - PURGE TABLE 文
   - COMMIT 文
   - DISCONNECT 文
   - ROLLBACK 文
   - PREPARE 文（クライアント環境定義 PDPRPCRCLS に YES を指定している場合）

- 内部 DISCONNECT（DISCONNECT 文を実行しないで UAP を終了する）

ただし，ホールダブルカーソルは，COMMIT 文を実行した場合は閉じられません。PURGE TABLE 文を実行し，ホールダブルカーソルで開いている表が検査保留状態に設定された場合，ホールダブルカーソルは閉じられます。

2. ALLOCATE CURSOR 文 形式 2 で手続きが返却した結果集合の組に割り当てられたカーソルに対して CLOSE 文を実行した場合，現在参照している結果集合の次の結果集合が存在するときは，現在参照している結果集合は閉じられます。カーソルは次の結果集合を参照し，次のリターンコードが設定されます。

   - SQL 連絡領域の SQLCODE 領域に 121
   - SQLCODE 変数に 121
   - SQLSTATE 変数に’0100D’

   また，このときカーソルは開いた状態となります。

   一方，次の結果集合が存在しない場合は，現在参照している結果集合は閉じられ，次のリターンコードが設定されます。

   - SQL 連絡領域の SQLCODE 領域に 100
   - SQLCODE 変数に 100
   - SQLSTATE 変数に’02001’

   また，このとき拡張カーソル名はどのカーソルも識別しなくなります。

   手続きから返却された結果集合の組に対してカーソルを割り当てた場合の一連の操作については，「結果集合返却機能」を参照してください。

(6) 留意事項

1. カーソル名は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって使用できません。

(7) 使用例

カーソル（CR1）を閉じます。

```
CLOSE CR1
```

4.8 DEALLOCATE PREPARE 文（SQL の前処理無効化）

4.8.1 DEALLOCATE PREPARE 文の形式と規則

(1) 機能

PREPARE 文で前処理した SQL を無効にし，SQL 文識別子，又は拡張文名の割当てを解除します。

(2) 使用権限

なし。

(3) 形式

```
DEALLOCATE PREPARE {SQL文識別子｜拡張文名}
```

(4) オペランド

(a) {SQL 文識別子｜拡張文名}

SQL 文識別子

PREPARE 文で前処理した SQL を識別するために付けた名称を指定します。

SQL 文識別子については，「名前の指定」を参照してください。

HiRDB の予約語も使用できますが，使用する場合，予約語と同じ名称でも引用符（”）で囲まないでください。ただし，’SELECT’，及び’ WITH’ は使用できません。

拡張文名

PREPARE 文で前処理した SQL を識別するために付けた拡張文名を指定します。

拡張文名については，「拡張文名」を参照してください。

(5) 共通規則

1. 指定した SQL 文識別子，又は拡張文名が識別している SQL に対して DECLARE CURSOR によって宣言された，又は ALLOCATE CURSOR 文によって割り当てられたカーソルが存在し，そのカーソルが開いた状態である場合は無効にできません。
2. 指定した SQL 文識別子，又は拡張文名に対して宣言された，又は割り当てられた閉じた状態のカーソルもすべて無効となります。また，そのカーソルを参照する前処理された SQL もすべて無効となります。

(6) 留意事項

1. 無効にした SQL 文識別子の SQL がホールダブルカーソルの場合は，ロールバックしても SQL 文識別子は有効にはなりません。
2. SQL 文識別子は，埋込み変数名と同様にコンパイル単位のモジュール内で有効な名前であり，同じ SQL 文識別子に対する複数の SQL を，複数のモジュールにわたって使用できません。

(7) 使用例

PREPARE 文に指定した SQL 文識別子（PRESQL）が識別する SQL 文の前処理結果を解放します。

```
DEALLOCATE PREPARE PRESQL
```

4.9 DECLARE CURSOR 形式 1（カーソル宣言）

4.9.1 DECLARE CURSOR 形式 1 の形式と規則

(1) 機能

問合せ指定の検索結果を FETCH 文で 1 行ずつ取り出すために，カーソルを宣言します。

形式 1 では，直接指定したカーソル指定に対するカーソルの宣言をします。

(2) 使用権限

なし。

(3) 形式 1<直接指定したカーソル指定に対するカーソルの宣言>

```
 DECLARE カーソル名 CURSOR
〔WITH HOLD〕〔 {WITH RETURN | WITHOUT RETURN} 〕 FOR
 〈カーソル指定　形式1〉
  〈問合せ式〉
   〈問合せ指定〉
   {SELECT 〔 {ALL | DISTINCT} 〕 {選択式〔, 選択式〕… | *}
    〈表式〉
    FROM 表参照〔, 表参照〕…
          〔WHERE  探索条件〕
          〔GROUP  BY  値式〔, 値式〕…〕
          〔HAVING  探索条件〕
     | 問合せ式 }
     〔ORDER BY  {列指定 | ソート項目指定番号} 〔 {ASC | DESC} 〕
          〔,  {列指定 | ソート項目指定番号} 〔 {ASC | DESC} 〕〕… 〕
     〔LIMIT  {〔オフセット行数,〕 {リミット行数 | ALL}
            |  {リミット行数 | ALL}  〔OFFSET オフセット行数〕} 〕
 〈排他オプション〉
  〔〔 {WITH  {SHARE | EXCLUSIVE} LOCK
     | WITHOUT LOCK 〔 {WAIT | NOWAIT} 〕} 〕
  〔 {WITH ROLLBACK | NO WAIT} 〕〕
     〔FOR {UPDATE 〔OF列名〔, 列名〕…〕〔NOWAIT〕 | READ ONLY} 〕
     〔UNTIL DISCONNECT〕
```

(4) オペランド

(a) カーソル名

カーソルの名称を指定します。

UAP 中に指定する場合は，SQL の予約語と同じ名称でも，引用符（”）で囲まないでください。ただし，SQL の予約語と同じ名称を手続き中に指定する場合は，引用符（”）で囲んでください。

カーソル名については，「名前の指定」を参照してください。

(b) 〔WITH HOLD〕

ホールダブルカーソルを使用する場合に指定します。

WITH HOLD は，UNTIL DISCONNECT を指定した場合の機能と同じなので，説明は UNTIL DISCONNECT を参照してください。また，この指定が UNTIL DISCONNECT と重複してもしなくても，機能差はありません。

(c) 〔{WITH RETURN ｜ WITHOUT RETURN}〕

SQL 手続き中のカーソル宣言で，そのカーソルの結果集合の返却可能性を指定します。

WITH RETURN を指定して宣言したカーソルを結果集合カーソルといいます。

SQL 手続き中で宣言された結果集合カーソルを，開いた状態のまま手続きを終了すると，カーソルの結果集合を手続きの呼出し元に返却できます。

結果集合カーソルを宣言する SQL 手続き定義の DYNAMIC RESULT SETS 句には 1 以上の値を指定してください。

返却した結果集合の使用方法については，「結果集合返却機能」を参照してください。

(d) カーソル指定　形式 1

問合せの内容を表すカーソルを指定します。

カーソル指定については「カーソル指定　形式 1」を，問合せ式については「問合せ式」を，問合せ指定については「問合せ指定」を，表式については「表式」を，探索条件については「探索条件」を参照してください。

(e) 排他オプション

問合せをする場合の排他制御モードと，ほかのユーザが資源を占有していた場合の処置を指定します。

排他オプションについては，「排他オプション」を参照してください。

(f) 〔FOR {UPDATE 〔OF 列名〔，列名〕…〕〔NOWAIT〕 ｜ READ ONLY}〕

FOR UPDATE 〔OF 列名〔，列名〕…〕を FOR UPDATE 句といいます。

FOR UPDATE

カーソルを使用して検索中の表に対して，そのカーソルを使用した行の更新，又は削除をして，更にカーソルを使用しない行の更新，削除，又は追加をする場合に指定します。

指定したカーソルを使用して行を更新する UPDATE 文，又は削除する DELETE 文がモジュール中に含まれている場合に，FOR UPDATE OF を省略すると FOR UPDATE が仮定され，すべての列を更新，追加，又は削除できます。

カーソルを使用して検索中の表に対して，そのカーソル，及びほかのカーソルを使用した行の更新，削除がなく，カーソルを使用しない行の更新，削除，及び追加もしない場合は，オペランドを省略してください。SQL 中の排他オプションを省略した場合，排他オプションは pd_isolation_level の指定値，PDISLLVL の指定値，又は SQL コンパイルオプションで指定するデータ保証レベルの指定値によって決まりますが，PDFORUPDATEEXLOCK に YES，又は SQL コンパイルオプションのデータ保証レベルの後に FOR UPDATE EXCLUSIVE を指定することで，この指定のあるカーソルに対する排他オプションは WITH EXCLUSIVE LOCK に仮定されます。詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

OF 列名〔，列名〕…

カーソルを使用して検索中の表に対して，そのカーソルを使用した検索行の更新だけをする場合に，更新する列を指定します。

SELECT 文の選択式で指定していない列でも指定できます。ただし，同じ列を 2 回以上指定できません。また，予備列は指定できません。

カーソルを使用して検索中の表に対して，そのカーソル及びほかのカーソルを使用した行の更新，削除がなく，カーソルを使用しない行の更新，削除，及び追加もしない場合には，オペランドを省略してください。

指定する列名は，AS 列名に指定した列名ではなく，最も外側の問合せ指定の FROM 句に指定した表の列を指定します。

〔NOWAIT〕

FOR UPDATE 句を指定し，かつ排他オプションに WITH EXCLUSIVE LOCK NO WAIT を指定した場合と同じ動作をします。ただし，排他オプションを指定した場合，NOWAIT は指定できません。

排他オプションに WITH EXCLUSIVE LOCK NO WAIT を指定した場合の動作については，「排他オプション」を参照してください。

FOR READ ONLY

カーソルを使用して検索中にほかのカーソル，又は直接探索条件を指定して更新する場合に指定します。この指定は，検索中の更新が検索結果に影響を与えないようにするためです。

(g) 〔UNTIL DISCONNECT〕

ホールダブルカーソルを使用する場合に指定します。

この指定をしたときの機能は，WITH HOLD を指定した場合と全く同じです。また，この指定が WITH HOLD と重複してもしなくても，機能差は生じません。

なお，ホールダブルカーソルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

ホールダブルカーソルについての規則を次に示します。

1. 次の場合，ホールダブルカーソルは使用できません。
   - ホールダブルカーソルに未対応のプラグイン※を使用した抽象データ型の列を指定した場合
   - ホールダブルカーソルに未対応のプラグイン※を使用した関数呼出しを指定した場合

- ホールダブルカーソルに未対応のプラグイン※を使用した関数呼出しを指定して導出した，名前付きの導出表に対する問合せを指定した場合
- ON COMMIT DELETE ROWS を指定して定義した一時表に対する問合せ

2. ホールダブルカーソルが開いている場合，定義系 SQL は実行できません。
3. ホールダブルカーソルを使用した SELECT 文に対して OPEN 文を実行後，その SELECT 文中で使用している表に対して PURGE TABLE 文を実行すると，カーソルは閉じた状態になります。
4. ホールダブルカーソルを使用した SELECT 文に対して OPEN 文を実行してから DISCONNECT するまでの間に，その SELECT 文中で使用している表に対して，ほかのユーザが定義系 SQL 文を発行すると，定義系 SQL は排他待ちの状態になります。また，ホールダブルカーソルを使用した SELECT 文に対する前処理が有効な間に，その SELECT 文中で使用している表に対して，ほかのユーザが定義系 SQL 文を発行すると，定義系 SQL は排他待ちの状態になります。

注※

プラグインのホールダブルカーソルへの対応状況については，ディクショナリ表 SQL_PLUGINS の PLUGIN_HOLDABLE 列で確認できます。ディクショナリ表 SQL_PLUGINS については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. 宣言したカーソルは，閉じた状態になります。
2. DECLARE CURSOR 中の SELECT 文で指定した埋込み変数，SQL 変数，又は SQL パラメタの値は，カーソルを開いてからカーソルを閉じるまでこのカーソルの OPEN 文実行時の値が有効です。値を変更したい場合は，いったんカーソルを閉じてから，再度開いてください。
3. 1 個の UAP では，最大 1,023 個のカーソルを宣言できます。
4. カーソル指定又は排他オプション中に次のどれかの指定を含む場合は，そのカーソルを使用した更新及び削除はできません。また，FOR UPDATE 句の指定もできません。

   (5-a) UNION 〔ALL〕，又は EXCEPT〔ALL〕

   (5-b) 最も外側の問合せ指定の FROM 句に指定した表を，副問合せの FROM 句に指定

   (5-c) 最も外側の問合せ指定での表の結合

   (5-d) 最も外側の問合せ指定での FROM 句の導出表

   (5-e) 最も外側の問合せ指定での SELECT DISTINCT

   (5-f) 最も外側の問合せ指定での GROUP BY 句

   (5-g) 最も外側の問合せ指定での HAVING 句

   (5-h) 最も外側の問合せ指定に対する集合関数

   (5-i) 最も外側の問合せ指定に対するウィンドウ関数

   (5-j) 最も外側の問合せ指定の FROM 句中で，次に示すどれかのビュー表を指定

   - ビュー定義文で上記(5-a)～(5-i)を指定して定義したビュー表

- ビュー定義文で最も外側の問合せ指定の SELECT 句に，列指定以外の値式を指定して定義したビュー表
- ビュー定義文で READ ONLY を指定したビュー表

(5-k) WITHOUT LOCK NOWAIT

(5-l) WITH 句を指定した問合せ式本体の最も外側の問合せ指定の FROM 句に問合せ名を指定

5. そのカーソルを指定した行の更新，又は削除がある場合 FOR READ ONLY 句は指定できません。
6. FOR READ ONLY 句を指定した場合，次に示す制限があります。

(5-a) 選択式中に，結果が次のデータ型になるスカラ演算，関数呼出し，及びコンポネント指定は指定できません。

- BLOB
- 最大長が 32,001 バイト以上の BINARY
- BOOLEAN
- 抽象データ型

(5-b) 選択式中の WRITE 指定の出力 BLOB 値には，列指定だけ指定できます。

(5-c) GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。

(6) 参照制約に関する規則

1. ホールダブルカーソルを使用して，外部キーの定義されている表を検索している場合，検索中の表が検査保留状態となったときは，カーソルは閉じた状態になります。

(7) 留意事項

1. カーソル名は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって使用できません。
2. カーソル宣言は，この宣言で使用したカーソル名を参照するどの SQL 文よりも先行して記述する必要があります。
3. DECLARE CURSOR は実行文ではないため，SQLCODE にリターンコードは返されません。したがって，リターンコードの判定はしないでください。
4. SQL 最適化オプションの更新 SQL の作業表作成抑止を適用して，更にインデクスキー値無排他機能を使用すると，FOR UPDATE 又は FOR UPDATE OF を指定しないカーソルの使用中に，行の更新，追加，又は削除ができます。
5. FOR READ ONLY を指定している場合，HiRDB が作業表を作成することがあります。このとき，作業表の行長によっては，FOR READ ONLY の処理が制限されることがあります。作業表の行長については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(8) 使用例

1. 在庫表（ZAIKO）から，行を 1 行ずつ取り出すためにカーソル（CR1）を宣言します。

```
DECLARE CR1 CURSOR FOR
    SELECT SCODE,SNAME,COL,TANKA,ZSURYO
        FROM ZAIKO
```

2. 在庫表（ZAIKO）から，単価（TANKA）が 5,000 円以上の行を 1 行ずつ取り出すためにカーソル（CR1）を宣言します。

```
DECLARE CR1 CURSOR FOR
    SELECT * FROM ZAIKO
        WHERE TANKA >= 5000
```

3. 在庫表（ZAIKO）から，カーソル（CR1）を使用してすべての行を検索しながら単価（TANKA）を 1 割引にし，更に行を挿入します。

```
DECLARE CR1 CURSOR FOR
    SELECT * FROM ZAIKO
        FOR UPDATE
OPEN CR1
  FETCH CR1 INTO ＜各列を取り出す変数名＞
UPDATE ZAIKO
    SET TANKA = ＜単価の 1 割引の値＞
        WHERE CURRENT OF CR1
INSERT INTO ZAIKO VALUES(＜各列の挿入値＞)
CLOSE CR1
```

4. 在庫表（ZAIKO）から，カーソル（CR1）を使用してすべての行を検索しながら単価（TANKA）を 1 割引にします。

```
DECLARE CR1 CURSOR FOR
    SELECT * FROM ZAIKO
        FOR UPDATE OF TANKA
OPEN CR1
FETCH CR1 INTO ＜各列を取り出す変数名＞
UPDATE ZAIKO
    SET TANKA = ＜単価の 1 割引の値＞
        WHERE CURRENT OF CR1
CLOSE CR1
```

5. 在庫表（ZAIKO）からカーソル（CR1）を使用してすべての行を検索中に，カーソルを使用しないで，直接商品名（SNAME）がセーターの行を削除します。

```
DECLARE CR1 CURSOR FOR
    SELECT * FROM ZAIKO
        FOR READ ONLY
OPEN CR1
FETCH CR1 INTO ＜各列を取り出す変数名＞
DELETE FROM ZAIKO
    WHERE SNAME=N’セーター’
CLOSE CR1
```

4.10 DECLARE CURSOR 形式 2（カーソル宣言）

4.10.1 DECLARE CURSOR 形式 2 の形式と規則

(1) 機能

問合せ指定の検索結果を FETCH 文で 1 行ずつ取り出すために，カーソルを宣言します。

形式 2 では，PREPARE 文で前処理した SELECT 文（動的 SELECT 文）に対するカーソルの宣言をします。

(2) 使用権限

なし。

(3) 形式 2 <PREPARE 文で前処理した SELECT 文（動的 SELECT 文）に対するカーソルの宣言>

```
DECLARE カーソル名 CURSOR
    〔WITH HOLD〕〔 {WITH RETURN | WITHOUT RETURN} 〕 FOR SQL文識別子
```

(4) オペランド

(a) カーソル名

カーソルの名称を指定します。

UAP 中で指定する場合は，SQL の予約語と同じ名称でも，引用符（”）で囲まないでください。ただし，SQL の予約語と同じ名称を手続き中に指定する場合は，引用符（”）で囲んでください。

カーソル名については，「名前の指定」を参照してください。

(b) 〔WITH HOLD〕

ホールダブルカーソルを使用する場合に指定します。ただし，次の場合，ホールダブルカーソルは使用できません。

- ホールダブルカーソルに未対応のプラグイン※を使用した抽象データ型の列を指定した場合
- ホールダブルカーソルに未対応のプラグイン※を使用した関数呼出しを指定した場合
- ホールダブルカーソルに未対応のプラグイン※を使用した関数呼出しを指定して導出した，名前付きの導出表に対する問合せを指定した場合
- リストを介した検索の場合

なお，ホールダブルカーソルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注※

プラグインのホールダブルカーソルへの対応状況については，ディクショナリ表 SQL_PLUGINS の PLUGIN_HOLDABLE 列で確認できます。ディクショナリ表 SQL_PLUGINS については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(c) 〔{WITH RETURN ｜ WITHOUT RETURN}〕

SQL 手続き中のカーソル宣言で，そのカーソルの結果集合の返却可能性を指定します。WITH RETURN を指定して宣言したカーソルを結果集合カーソルといいます。SQL 手続き中で宣言された結果集合カーソルを，開いた状態のまま手続きを終了すると，カーソルの結果集合を手続きの呼出し元に返却できます。返却した結果集合の使用方法については，「結果集合返却機能」を参照してください。

(d) SQL 文識別子

PREPARE 文で前処理した SELECT 文に付けられた SQL 文識別子を指定します。

(5) 共通規則

1. 宣言したカーソルは，閉じた状態になります。
2. 宣言したカーソルを，UPDATE 文，又は DELETE 文で使用できません。
3. 同じ名前の SQL 文識別子に対して，複数のカーソル宣言をした場合，WITH HOLD 指定のあるカーソル宣言と，WITH HOLD 指定のないカーソル宣言を混在させることはできません。

(6) 参照制約に関する規則

1. ホールダブルカーソルを使用して，外部キーの定義されている表を検索している場合，検索中の表が検査保留状態となったときは，カーソルは閉じた状態になります。

(7) 留意事項

1. カーソル名は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって使用できません。
2. 対応する PREPARE 文は，カーソル宣言より先に記述してください。
3. カーソル宣言は，この宣言で使用したカーソル名を参照するどの SQL 文よりも先に記述する必要があります。
4. DECLARE CURSOR は実行文ではないため，SQLCODE にリターンコードは返されません。したがって，リターンコードの判定はしないでください。
5. SQL 最適化オプションの更新 SQL の作業表作成抑止を適用して，更にインデクスキー値無排他機能を使用すると，FOR UPDATE 又は FOR UPDATE OF を指定しないカーソルの使用中に，行の更新，追加，又は削除ができます。

(8) 使用例

前処理した SELECT 文（SQL 文識別子：SEL）によって指定した行を 1 行ずつ取り出すために，カーソル（CR1）を宣言します。

```
DECLARE CR1 CURSOR FOR SEL
```

4.11　DELETE 文　形式 1（行削除）

4.11.1　DELETE 文 形式 1 の形式と規則

(1)　機能

表から指定した探索条件を満足する行，又はカーソルが指している行を削除します。

(2)　使用権限

表に対する DELETE 権限を持つユーザが，その表の行を削除できます。

ただし，探索条件中に副問合せを指定する場合は，その副問合せ指定の表に対して SELECT 権限が必要です。

(3)　形式

```
DELETE FROM 〔認可識別子.〕表識別子
                〔IN （RDエリア名指定）〕〔〔AS〕相関名〕
                〔使用インデクスのSQL最適化指定〕
                〔WHERE {探索条件｜CURRENT OF {カーソル名｜拡張カーソル名}}〕
                〔WITH ROLLBACK〕
                〔WRITE IMMEDIATE〕
```

(4)　オペランド

(a)　〔認可識別子.〕表識別子

認可識別子

表を所有するユーザの認可識別子を指定します。

認可識別子に，"MASTER"は指定できません。認可識別子を省略した場合については，「名前の修飾」を参照してください。

表識別子

削除する行がある表を指定します。

表識別子についての規則を次に示します。

1. 読み込み専用のビュー表は，行の挿入，更新，及び削除ができません。
2. 読み込み専用のビュー表については，「CREATE 〔PUBLIC〕VIEW（ビュー定義，パブリックビュー定義）」の共通規則を参照してください。
3. ビュー表からの行の削除を指定した場合は，そのビュー表の操作の対象の基になる実表から行を削除します。

4. 表名の有効範囲は DELETE 文全体です。

(b) 〔IN （RD エリア名指定)〕

IN

アクセス対象の RD エリアを指定します。なお，表識別子に指定した表が一時表の場合は指定できません。

RD エリア名指定：：＝値指定

表識別子に指定した表を格納している RD エリアのうち，アクセスする RD エリアの名称を VARCHAR 型，CHAR 型，MVARCHAR 型，又は MCHAR 型の値指定で指定します。複数の RD エリア名を指定する場合はコンマ（,）で区切って指定してください。RD エリア名は重複して指定できません。重複して指定した場合はエラーとなります。値指定で指定する RD エリア名に許される文字については，「名前の指定」を参照してください。また，値指定で指定した RD エリア名の前後の空白は無視されます。RD エリア名を引用符（"）で囲んだ場合は，引用符（"）の外側の空白だけを無視します。

カーソル名又は拡張カーソル名の指定がある場合は，カーソル宣言で宣言したカーソルで指定した RD エリアと同じ RD エリアの集合を指定してください（順不同)。指定しない場合はエラーとなります。

インナレプリカ機能を適用している RD エリアを指定する場合，オリジナル RD エリア名を指定してください。レプリカ RD エリアを対象とする場合は，カレント切り替えコマンド（pddbchg コマンド)，又はクライアント環境定義の PDDBACCS オペランドでアクセス対象 RD エリアをレプリカ RD エリアに切り替えてください。

(c) 〔AS〕相関名

削除対象の表に対して相関名を使用する場合に指定します。

相関名の有効範囲は DELETE 文全体です。削除対象の表識別子は有効範囲を持ちません。

(d) 使用インデクスの SQL 最適化指定

使用インデクスの SQL 最適化指定については，「SQL 最適化指定」を参照してください。

(e) WHERE {探索条件｜CURRENT OF {カーソル名｜拡張カーソル名}}

WHERE

WHERE 句を省略すると，指定した表のすべての行が削除されます。SQL 連絡領域の SQLWARN4 領域に警告フラグ”W”が設定されます。

探索条件

削除する行を選択する条件を指定します。

探索条件中に埋込み変数，?パラメタ，SQL 変数，又は SQL パラメタを指定できます。ただし，PREPARE 文で前処理される DELETE 文の探索条件中では，?パラメタだけ指定できます。

SQL 手続き中では，SQL 変数，又は SQL パラメタを使用します。Java 手続き中の指定値については，マニュアル「HiRDB UAP 開発ガイド」の JDBC ドライバ又は SQLJ を参照してください。

探索条件については，「探索条件」を参照してください。

カーソル名

削除する行を指すカーソルの名称を指定します。

カーソル名で指定するカーソルは，カーソル宣言で宣言されたカーソルです。

カーソル名で指定するカーソルは，OPEN 文で開き，FETCH 文で削除する行に位置づけてください。なお，そのカーソルに対する OPEN 文，FETCH 文，CLOSE 文及び DELETE 文は，同一トランザクション内で実行してください（ただし，ホールダブルカーソルの場合を除く）。

カーソル名で指定するカーソルは，DELETE 文実行後は，指している行がありません。削除した行より後の行を更新，又は削除する場合は，そのカーソルに対して FETCH 文を実行して，カーソルを移動させてください。

拡張カーソル名

削除する行を指すカーソルを識別する拡張カーソル名を指定します。

ALLOCATE CURSOR 文で割り当てられたカーソルを識別している拡張カーソル名を指定してください。ただし，結果集合カーソルは指定できません。

拡張カーソル名が識別するカーソルは，開いた状態であり，かつ削除する行に位置づけられている必要があります。

拡張カーソル名が識別するカーソルは，DELETE 文実行後は指している行がありません。削除した行より後の行を更新，又は削除する場合は，そのカーソルに対して FETCH 文を実行して，カーソルを移動させてください。

拡張カーソル名を指定する場合は，FOR UPDATE 句の指定のある問合せに対する拡張カーソルを指定しなければなりません。FOR UPDATE 句については，「動的 SELECT 文　形式 1（動的検索）」のオペランド規則の FOR UPDATE 句を参照してください。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(f)　WITH ROLLBACK

削除する表がほかのユーザで使用されているとき，トランザクションを取り消して無効にする場合に指定します。

WITH ROLLBACK を省略した場合は，削除する表がほかのユーザのトランザクションで使用されているとき，そのトランザクションが終了してから実行します。

(g)　WRITE IMMEDIATE

表オプションに WITHOUT ROLLBACK の指定がある表に対する更新処理で，更新完了時にシステムログを書き出す場合，このオプションを指定してください。WITHOUT ROLLBACK 指定のない表に対して，このオプションを指定しても無効になります。

更新完了時にシステムログを書き出す効果については，マニュアル「HiRDB UAP 開発ガイド」の採番業務で使用する表を参照してください。

(5) 共通規則

1. DELETE 文が正常に実行された場合は，SQL 連絡領域の SQLERRD[2]領域に削除した行数が設定されます。
2. 削除の対象になる行がない場合は，次のリターンコードが設定されます。
   - SQL 連絡領域の SQLCODE 領域に 100
   - SQLCODE 変数に 100
   - SQLSTATE 変数に’02000’
3. LOB 列又は LOB 属性を格納するユーザ LOB 用 RD エリアが更新凍結状態の場合，その LOB 列又は LOB 属性は削除できません（削除しようとすると凍結済みエラーとなります）。
4. 表が改竄防止表であり，かつ指定した探索条件を満足する行の中に削除禁止期間の行がある場合は，探索条件を満足するすべての行を削除しないでエラーとなります。
5. 表が改竄防止表であり，かつ指定したカーソルが指している行が削除禁止期間である場合，行を削除しないでエラーとなります。
6. WITHOUT ROLLBACK を指定した表に対して，DELETE 文を実行する場合，インデクスの定義有無によって行排他の解除タイミングが異なります。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
7. 共用表の行を削除する場合，事前にその表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表の行を削除しようとすると，エラーとなり行を削除できません。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。
8. RD エリア名指定を指定した場合，分割数が表の分割数と異なるインデクスは利用できません。RD エリア名指定を指定する問合せのためにインデクスを定義する場合は，分割数が表の分割数と等しいインデクスを定義してください。

(6) 参照制約に関する規則

1. 被参照表，参照表の行を削除する場合の規則は「CREATE TABLE（表定義）」の参照動作を参照してください。
2. 制約動作が RESTRICT で定義された被参照表の行を削除する場合，削除対象になる行中の主キー構成列の値が参照表の外部キー構成列の値に含まれるかどうかを確認するため，参照表を検索します。このとき，参照表の検索時のデータ保証レベルは共用モードになります。そのため，制約動作が RESTRICT で定義された被参照表の行を削除するときに，ほかのトランザクションによって参照表に対する操作が行われていると，そのトランザクションが決着するまでその行削除は待ち状態になります。
3. 次の条件が重なった場合，参照制約での被参照表と参照表間でデータの不整合が発生することがあります。また，制約動作が RESIRICT 又は CASCADE のどちらかの場合でも発生することがあります。参照制約に関する規則については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

- 参照表の行を削除するトランザクションと，被参照表の行を更新，又は削除するトランザクションが異なる
- 上記二つのトランザクションが同時に実行される
- 参照表で削除する行の主キー構成列の値と，被参照表で更新，又は削除する行の外部キー構成列の値と同じである
- 参照表の行を削除するトランザクションをコミット，被参照表の行を更新，又は削除するトランザクションをロールバックする

(7) 留意事項

1. カーソル名は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって使用できません。
2. 表が改竄防止表であり，かつ削除禁止期間を指定している場合，削除対象行の挿入日付と削除禁止期間の和が 9999 年 12 月 31 日を超えるとオーバフローが発生します。

(8) 使用例

1. 在庫表（ZAIKO）から商品コード（SCODE）列が 302S の行を削除します。

```
DELETE FROM ZAIKO
     WHERE SCODE =  '302S'
```

2. 在庫表（ZAIKO）から埋込み変数（：XSCODE）に読み込まれた商品コード（SCODE）列の行を削除します。

```
DELETE FROM ZAIKO
    WHERE SCODE = :XSCODE
```

3. 在庫表（ZAIKO）からカーソル（CR1）で指定した行を削除します。

```
DELETE FROM ZAIKO
    WHERE CURRENT OF CR1
```

4.12　DELETE 文　形式 2（配列を使用した行削除）

4.12.1　DELETE 文 形式 2 の形式と規則

(1)　機能

表から，指定した探索条件を満足する行を削除します。配列形式の埋込み変数を指定して，複数回の削除処理を一括して実行できます。

(2)　使用権限

表に対する DELETE 権限を持つユーザが，その表の行を削除できます。

ただし，探索条件中に副問合せを指定する場合は，その副問合せ指定の表に対して SELECT 権限が必要です。

(3)　形式<埋込み変数配列を指定して，複数回の削除を実行>

```
FOR ：埋込み変数
DELETE FROM 〔認可識別子.〕表識別子
               〔IN （RDエリア名指定）〕〔〔AS〕相関名〕
                〔使用インデクスのSQL最適化指定〕
               WHERE 探索条件
                〔WITH ROLLBACK〕
                〔WRITE IMMEDIATE〕
```

(4)　オペランド

FOR，IN (RD エリア名指定)，探索条件以外のオペランド，及びオペランド規則については，「DELETE 文　形式 1（行削除)」を参照してください。

(a)　FOR ：埋込み変数

埋込み変数配列を使用して削除を行う回数を設定した埋込み変数を指定します。SMALLINT 型の埋込み変数を指定してください。設定値は 1 から 4,096 までの範囲で，かつ埋込み変数配列，及び標識変数配列の要素数以下にしてください。0 及び負の値は設定できません。範囲外の値を設定した場合は実行時にエラーとなります。

埋込み変数配列

配列形式で宣言した埋込み変数です。NULL 値以外の値で探索条件として使用するための配列変数を指定します。探索条件として使用する値を埋込み変数配列の各要素に設定してください。探索条件として使用する値に NULL 値を含む場合は，埋込み変数配列と標識変数配列を両方指定します。

標識変数配列

配列形式で宣言した標識変数です。埋込み変数配列の各要素の値が NULL 値かどうかを示す値を，標識変数配列の対応する要素に設定してください。設定する値については，「標識変数の値の設定」を参照してください。

(b) 〔IN （RD エリア名指定)〕

RD エリア名指定を埋込み変数で指定する場合は，埋込み変数配列で指定してください。配列でない埋込み変数を指定した場合はエラーとなります。そのほかのオペランド規則については，「DELETE 文　形式 1（行削除)」を参照してください。

(c) WHERE 探索条件

探索条件

削除する行を選択する条件を指定します。探索条件に配列形式でない埋込み変数は指定できません。

(5) 共通規則

1. FOR 句以外に埋込み変数配列を一つ以上指定してください。指定しない場合はエラーとなります。
2. FOR 句以外に配列形式でない埋込み変数を指定するとエラーとなります。
3. 埋込み変数配列のデータ型は対応する列のデータ型，又は変換できるデータ型にしてください。
4. 埋込み変数配列，及び標識変数配列の要素数は 1 から 4,096 までの範囲にしてください。範囲外の値を指定した場合はエラーとなります。また，”FOR ：埋込み変数”で指定する要素数の最大値以上になるようにしてください。
5. 各埋込み変数配列，及び標識変数配列で 1 回の削除処理によって評価される要素は同じ要素番号の要素となります。
6. DELETE 文形式 2 は埋込み変数配列，及び標識変数配列を含むため，PREPARE 文で前処理することはできません。動的に実行する場合については，「EXECUTE 文　形式 2（配列を使用した SQL の実行)」を参照してください。
7. 配列を使用した DELETE は手続きの中では使えません。
8. 配列を使用した DELETE では，BLOB 型，最大長が 32,001 バイト以上の BINARY 型及び抽象データ型は扱えません。
9. DELETE 文が正常に実行された場合は，SQL 連絡領域の SQLERRD[2]領域に削除した行数が設定されます。
10. 削除の対象になる行がない場合は，次のリターンコードが設定されます。
    - SQL 連絡領域の SQLCODE 領域に 100
    - SQLCODE 変数に 100
    - SQLSTATE 変数に’02000’

11. 表が改竄防止表であり，かつ指定した探索条件を満足する行の中に削除禁止期間の行がある場合は，探索条件を満足するすべての行を削除しないでエラーとなります。
12. 削除するどれかの行でエラーが発生した場合，ロールバックされます。
13. WITHOUT ROLLBACK を指定した表に対して，DELETE 文を実行する場合，インデクスの定義有無によって行排他の解除タイミングが異なります。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
14. 共用表の行を削除する場合，事前にその表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表の行を削除削除しようとすると，エラーとなり行を削除できません。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。
15. RD エリア名指定を指定した場合，分割数が表の分割数と異なるインデクスは利用できません。RD エリア名指定を指定する問合せのためにインデクスを定義する場合は，分割数が表の分割数と等しいインデクスを定義してください。

(6) 参照制約に関する規則

1. 被参照表，参照表の行を削除する場合の規則は「CREATE TABLE（表定義）」の参照動作を参照してください。
2. 制約動作が RESTRICT で定義された被参照表の行を削除する場合，削除対象になる行中の主キー構成列の値が参照表の外部キー構成列の値に含まれるかどうかを確認するため，参照表を検索します。このとき，参照表の検索時のデータ保証レベルは共用モードになります。そのため，制約動作が RESTRICT で定義された被参照表の行を削除するときに，ほかのトランザクションによって参照表に対する操作が行われていると，そのトランザクションが決着するまでその行削除は待ち状態になります。
3. 次の条件が重なった場合，参照制約での被参照表と参照表間でデータの不整合が発生することがあります。また，制約動作が RESIRICT 又は CASCADE のどちらかの場合でも発生することがあります。参照制約に関する規則については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。
   - 参照表の行を削除するトランザクションと，被参照表の行を更新，又は削除するトランザクションが異なる
   - 上記二つのトランザクションが同時に実行される
   - 参照表で削除する行の主キー構成列の値と，被参照表で更新，又は削除する行の外部キー構成列の値と同じである
   - 参照表の行を削除するトランザクションをコミット，被参照表の行を更新，又は削除するトランザクションをロールバックする

(7) 使用例

1. C言語の配列変数に設定した商品コード（SCODE）の値別に，在庫表（ZAIKO）の行の削除を複数回分一括して実行します。

```
XDELETE_NUM = 5;
EXEC SQL FOR :XDELETE_NUM
      DELETE FROM ZAIKO
WHERE SCODE = :XSCODE:ISCODE;
```

4.13　準備可能動的 DELETE 文：位置付け（前処理可能なカーソルを使用した行削除）

4.13.1　準備可能動的 DELETE 文：位置付けの形式と規則

(1)　機能

カーソルが指している行を削除します。PREPARE 文で前処理してから EXECUTE 文で実行，又は EXECUTE IMMEDIATE 文で前処理と実行を一度に行う場合に使用します。

(2)　使用権限

表に対する DELETE 権限を持つユーザが，その表の行を削除できます。

(3)　形式

```
DELETE 〔FROM 〔認可識別子.〕表識別子
              〔IN （RDエリア名指定）〕〔〔AS〕 相関名〕
              〔使用インデクスのSQL最適化指定〕〕
                WHERE CURRENT OF GLOBAL カーソル名
              〔WITH ROLLBACK〕
              〔WRITE IMMEDIATE〕
```

(4)　オペランド

GLOBAL，カーソル名以外のオペランド，及びオペランド規則については，「DELETE 文　形式 1（行削除）」を参照してください。

(a)　WHERE CURRENT OF GLOBAL カーソル名

GLOBAL

カーソル名の有効範囲として GLOBAL を指定します。

カーソル名

削除する行を指すカーソルの名称を指定します。

カーソル名で指定するカーソルは，ALLOCATE CURSOR 文で指定した拡張カーソル名が識別するカーソルです。ALLOCATE CURSOR 文で指定した拡張カーソル名の値を指定してください。ただし，結果集合カーソルは指定できません。

実行時には，カーソル名で指定するカーソルは，開いた状態であり，かつ削除する行に位置づけられている必要があります。

カーソル名で指定するカーソルは，DELETE 文実行後は，指している行がありません。削除した行より後の行を更新，又は削除する場合は，そのカーソルに対して FETCH 文を実行して，カーソルを移動させてください。

拡張カーソル名を指定する場合は，FOR UPDATE 句の指定のある問合せに対する拡張カーソルを指定しなければなりません。FOR UPDATE 句については，「動的 SELECT 文　形式 1（動的検索）」のオペランド規則の FOR UPDATE 句を参照してください。

(5)　共通規則

1. PREPARE 文で前処理してから，EXECUTE 文で実行，又は EXECUTE IMMEDIATE 文で前処理と実行を一度に行ってください。
2. 表識別子を省略する場合は，前処理する前に，ALLOCATE CURSOR 文によって動的 SELECT 文にカーソルが割り当てられている必要があります。このとき，カーソルを割り当てた動的 SELECT 文に指定している検索対象の表を仮定します。表識別子を指定する場合には，前処理する前に，動的 SELECT 文にカーソルが割り当てられている必要はありません。
3. そのほかの共通規則については，DELETE 文 形式 1 の共通規則が適用されます。

(6)　参照制約に関する規則

1. DELETE 文 形式 1 の参照制約に関する規則が適用されます。

(7)　留意事項

1. DELETE 文 形式 1 の留意事項が適用されます。

(8)　使用例

1. 在庫表（ZAIKO）からカーソル（cr（値：’CR1’））で指定した行を削除します。

```
PREPARE :sel FOR  ‘SELECT * FROM ZAIKO FOR UPDATE’
＜埋込み変数crに’CR1’を設定＞
ALLOCATE CURSOR GLOBAL :cr FOR GLOBAL :sel
OPEN GLOBAL :cr
FETCH GLOBAL :cr INTO ＜各列を取り出す変数名＞
PREPARE PRE1 FOR
      ‘DELETE FROM ZAIKO WHERE CURRENT OF GLOBAL CR1’
EXECUTE PRE1
DEALLOCATE PREPARE GLOBAL :sel
```

4.14　DESCRIBE 文　形式 1（検索情報，入出力情報の受け取り）

4.14.1　DESCRIBE 文 形式 1 の形式と規則

(1)　機能

PREPARE 文で前処理した SQL の検索項目情報，又は出力?パラメタ情報を SQL 記述領域（データコード，データ長など）に受け取ります。

受け取り情報の詳細は，マニュアル「HiRDB UAP 開発ガイド」を参照してください。PREPARE 文で前処理した SQL が SQL の検索項目情報，又は出力?パラメタ情報を持たない場合には，SQL 記述領域の SQLD 領域に 0 が設定されます。

(2)　使用権限

なし。

(3)　形式 1 ＜検索項目情報，又は出力?パラメタ情報の受け取り＞

```
DESCRIBE 〔OUTPUT〕 {SQL文識別子｜拡張文名} INTO
            〔：〕SQL記述領域名〔〔：〕列名記述領域名〕
            〔TYPE〔：〕型名記述領域名〕
            〔CHARACTER_SET〔：〕文字集合名記述領域名〕
```

(4)　オペランド

(a)　{SQL 文識別子｜拡張文名}

SQL 文識別子

PREPARE 文で指定した SQL 文識別子を指定します。

拡張文名

PREPARE 文で前処理した SQL 文を識別する拡張文名を指定します。

拡張文名については，「拡張文名」を参照してください。

(b)　〔：〕SQL 記述領域名〔〔：〕列名記述領域名〕

SQL 記述領域名

SQL の検索項目情報（前処理した SQL が SELECT 文の場合），又は出力?パラメタ情報（前処理した SQL が CALL 文の場合）を受け取る SQL 記述領域の名称を指定します。

SQL 記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

列名記述領域名

検索項目の名称，又はルーチンのパラメタ名を受け取る列名記述領域を指定します。

列名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(c) 〔TYPE〔：〕型名記述領域名〕

型名記述領域名

検索項目のユーザ定義型名を受け取る型名記述領域を指定します。

型名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(d) 〔CHARACTER_SET〔：〕文字集合名記述領域名〕

文字集合名記述領域名

検索項目情報（前処理した SQL が SELECT 文の場合），又は出力？パラメタ情報（前処理した SQL が CALL 文の場合）の文字集合名を受け取る文字集合名記述領域を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. UAP は，DESCRIBE 文を実行する前に，SQL 記述領域に SQLVAR の数（SQLN 領域）を設定してください。
2. SQLDATA と SQLIND は DESCRIBE 文実行時にクリアされるので，DESCRIBE 文を使用する場合は，その実行後に値を設定してください。
3. 列名記述領域名を指定する場合は，対応する PREPARE 文で WITH SQLNAME OPTION を指定してください。
4. 列名記述領域名は，検索項目の名称，又はルーチンのパラメタ名を受け取る場合だけ指定してください。ただし，ルーチンのパラメタ名は，CALL 文の引数に単独で？パラメタを指定した場合にだけ受け取ることができます。？パラメタを含む値式を指定した場合は，列名記述領域の名称の長さが 0 になります。
5. 型名記述領域名を指定する場合は，対応する PREPARE 文で WITH〔ALL〕TYPE OPTION を指定してください。
6. 型名記述領域名は，検索項目のユーザ定義型名を受け取る場合だけ指定してください。
7. 文字集合名記述領域は，検索項目情報（前処理した SQL が SELECT 文の場合），又は出力？パラメタ情報（前処理した SQL が CALL 文の場合）のデータ型が文字データ型で，文字集合を指定している場合だけ指定してください。

(6) 留意事項

1. SQL 文識別子は，埋込み変数名と同様に，コンパイル単位のファイル内で有効な名前であり，同じ SQL 文識別子に対する複数の SQL を，複数のモジュールにわたって使用できません。

2. 出力？パラメタがユーザ定義型の手続きの場合，ユーザ定義型の出力？パラメタに対して型名情報は設定されません。
3. DESCRIBE〔OUTPUT〕文を使用しなくても，PREPARE 文で OUTPUT を指定した場合，DESCRIBE〔OUTPUT〕で得られる情報と同様の情報を得ることもできます。
4. ユーザ定義型の所有者名の長さが９文字以上の場合，型名記述領域を使用できません。

(7) 使用例

1. PREPARE 文で前処理した SELECT 文（SQL 文識別子：PRESQL）の検索項目情報，及び検索項目の名称，又はルーチンのパラメタ名を，SQL 記述領域，及び列名記述領域に指定します。

```
DESCRIBE PRESQL INTO :SQLDA :SQLCNDA
```

2. PREPARE 文で前処理した SELECT 文（拡張文名：pre）の検索項目情報，及び検索項目の名称を，SQL 記述領域，及び列名記述領域に指定します。

```
PREPARE GLOBAL :pre FOR :sel WITH SQLNAME OPTION
DESCRIBE GLOBAL :pre INTO :SQLDA :SQLCNDA
```

4.15　DESCRIBE 文　形式 2（検索情報，入出力情報の受け取り）

4.15.1　DESCRIBE 文 形式 2 の形式と規則

(1)　機能

PREPARE 文で前処理した SQL の入力?パラメタ情報を SQL 記述領域（データコード，データ長など）に受け取ります。

受け取り情報の詳細は，マニュアル「HiRDB UAP 開発ガイド」を参照してください。PREPARE 文で前処理した SQL が入力?パラメタ情報を持たない場合には，SQL 記述領域の SQLD 領域に 0 が設定されます。

(2)　使用権限

なし。

(3)　形式 2 <入力?パラメタ情報の受け取り>

```
DESCRIBE INPUT  {SQL文識別子｜拡張文名}  INTO
            〔:〕SQL記述領域名〔〔:〕列名記述領域名〕
            〔CHARACTER_SET〔:〕文字集合名記述領域名〕
```

(4)　オペランド

(a)　{SQL 文識別子｜拡張文名}

SQL 文識別子

PREPARE 文で指定した SQL 文識別子を指定します。

拡張文名

PREPARE 文で前処理した SQL 文を識別する拡張文名を指定します。

拡張文名については，「拡張文名」を参照してください。

(b)　〔:〕SQL 記述領域名〔〔:〕列名記述領域名〕

SQL 記述領域名

入力?パラメタ情報を受け取る SQL 記述領域の名称を指定します。

SQL 記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

列名記述領域名

検索項目の名称，又はルーチンのパラメタ名を受け取る列名記述領域を指定します。

列名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(c) 〔CHARACTER_SET〔：〕文字集合名記述領域名〕

文字集合名記述領域名

入力？パラメタ情報の文字集合名を受け取る文字集合名記述領域を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. UAP は，DESCRIBE 文を実行する前に，SQL 記述領域に SQLVAR の数（SQLN 領域）を設定してください。
2. SQLDATA と SQLIND は DESCRIBE 文実行時にクリアされるので，DESCRIBE 文を使用する場合は，その実行後に値を設定してください。
3. 列名記述領域名を指定する場合は，対応する PREPARE 文で WITH SQLNAME OPTION を指定してください。
4. 列名記述領域名は，検索項目の名称，又はルーチンのパラメタ名を受け取る場合だけ指定してください。ただし，ルーチンのパラメタ名は，CALL 文の引数に単独で？パラメタを指定した場合にだけ受け取ることができます。？パラメタを含む値式を指定した場合は，列名記述領域の名称の長さが 0 になります。
5. 文字集合名記述領域は，入力？パラメタ情報のデータ型が文字データ型で，文字集合を指定している場合だけ指定してください。

(6) 留意事項

1. SQL 文識別子は，埋込み変数名と同様に，コンパイル単位のファイル内で有効な名前であり，同じ SQL 文識別子に対する複数の SQL を，複数のモジュールにわたって使用できません。
2. DESCRIBE INPUT 文を使用しなくても，PREPARE 文で INPUT を指定した場合，DESCRIBE INPUT で得られる情報と同様の情報を得ることができます。

4.16　DESCRIBE CURSOR 文（カーソルの検索情報の受け取り）

4.16.1　DESCRIBE CURSOR 文の形式と規則

(1)　機能

手続きから返却された結果集合を参照するカーソルの問合せの検索項目情報を SQL 記述領域（データコード，データ長など）に受け取ります。

受け取り情報の詳細は，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(2)　使用権限

なし。

(3)　形式 1 <カーソルの検索項目情報の受け取り>

```
DESCRIBE 〔OUTPUT〕 CURSOR 拡張カーソル名 STRUCTURE  INTO
          〔：〕SQL記述領域名〔CHARACTER_SET〔：〕文字集合名記述領域名〕
```

(4)　オペランド

CURSOR 拡張カーソル名 STRUCTURE 以外のオペランドについては，「DESCRIBE 文　形式 1（検索情報，入出力情報の受け取り）」を参照してください。

(a)　CURSOR 拡張カーソル名 STRUCTURE

拡張カーソル名

ALLOCATE CURSOR 文 形式 2 で手続から返却された結果集合の組に割り当てられたカーソルを識別する拡張カーソル名を指定します。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(b)　〔：〕SQL 記述領域名〔CHARACTER_SET〔：〕文字集合名記述領域名〕

SQL 記述領域名

カーソルの検索項目情報を受け取る SQL 記述領域の名称を指定します。

SQL 記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

文字集合名記述領域名

カーソルの検索項目情報の文字集合名を受け取る文字集合名記述領域を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. UAP は，DESCRIBE 文を実行する前に，SQL 記述領域に SQLVAR の数（SQLN 領域）を設定してください。
2. SQLDATA と SQLIND は DESCRIBE 文実行時にクリアされるので，DESCRIBE 文を使用する場合は，その実行後に値を設定してください。

(6) 使用例

1. 手続き PROC1 から返却された結果集合の組みに対して割り当てたカーソル（拡張カーソル名：cr，有効範囲：GLOBAL）の検索項目情報を，SQL 記述領域に設定します。

```
CALL PROC1()
ALLOCATE GLOBAL :cr FOR PROCEDURE PROC1
DESCRIBE CURSOR GLOBAL :cr STRUCTURE INTO :SQLDA
```

4.17　DESCRIBE TYPE 文（ユーザ定義型の定義情報の受け取り）

4.17.1　DESCRIBE TYPE 文の形式と規則

(1)　機能

PREPARE 文で前処理した SQL の検索項目情報にユーザ定義型が直接又は間接的に含まれる場合，そのユーザ定義型の定義情報（各属性のデータコード，データ長など）を SQL 記述領域に受け取ります。※

PREPARE 文で WITH ALL TYPE OPTION を指定しないで前処理した場合，ユーザ定義型の定義情報は受け取れません。また，受け取ろうとするユーザ定義型の属性の隠蔽レベルがすべて PUBLIC 以外の場合，SQL 記述領域の SQLD 領域に 0 が設定されます。

注※

直接ユーザ定義型が含まれる場合とは，検索項目情報の列がユーザ定義型のことをいい，間接的にユーザ定義型が含まれる場合とは，検索項目情報の列がユーザ定義型で，更にその列にユーザ定義型の属性がある状態（つまりユーザ定義型がネストしている状態）をいいます。

(2)　使用権限

なし。

(3)　形式

```
DESCRIBE TYPE ：埋込み変数1 ：埋込み変数2 FOR {SQL文識別子｜拡張文名} INTO
                〔：〕SQL記述領域名 〔〔：〕列名記述領域名〕
                〔TYPE 〔：〕型名記述領域名〕
```

(4)　オペランド

(a)　TYPE ：埋込み変数 1 ：埋込み変数 2 FOR {SQL 文識別子｜拡張文名}

埋込み変数 1

ユーザ定義型の所有者を格納した VARCHAR(30)の埋込み変数を指定します。

埋込み変数 2

ユーザ定義型のデータ型識別子を格納した VARCHAR(30)の埋込み変数を指定します。

SQL 文識別子

PREPARE 文で指定した SQL 文識別子を指定します。

拡張文名

PREPARE 文で前処理した SQL 文を識別する拡張文名を指定します。

拡張文名については，「拡張文名」を参照してください。

(b) 〔：〕SQL 記述領域名 〔〔：〕列名記述領域名〕 〔TYPE 〔：〕型名記述領域名〕

SQL 記述領域名

ユーザ定義型の属性の情報を受け取る SQL 記述領域の名称を指定します。SQL 記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

列名記述領域名

ユーザ定義型の属性名を受け取る列名記述領域を指定します。列名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

型名記述領域名

ユーザ定義型の属性がユーザ定義型の場合，そのユーザ定義型のユーザ定義型名を受け取る型名記述領域を指定します。型名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. UAP 中では，DESCRIBE TYPE 文を実行する前に，SQL 記述領域に SQLVAR の数（SQLN 領域）を設定しておく必要があります。
2. ユーザ定義型の属性のデータ型が SQL 記述領域に，属性の名称が列名記述領域に，属性のデータ型がユーザ定義型の場合のユーザ定義型名が型名記述領域にそれぞれ設定されます。
3. 指定した抽象データ型が上位のデータ型の属性を継承している場合，継承している属性の情報も設定されます。この場合，設定される順序は，継承している属性，固有の属性の順です。
4. PREPARE 文で指定した SQL の検索項目に直接又は間接的に含まれるユーザ定義型以外のユーザ定義型は指定できません。
5. 隠蔽レベルが PUBLIC 以外の属性は定義情報を取得できません。
6. 列名記述領域名を指定する場合は，対応する PREPARE 文で WITH SQLNAME OPTION を指定してください。

(6) 留意事項

1. SQL 文識別子は，埋込み変数と同様にコンパイル単位のファイル内で有効な名前であり，同じ SQL 文識別子に対する複数の SQL を，複数のモジュールにわたって使用できません。
2. ユーザ定義型の所有者名の長さが 9 文字以上の場合，型名記述領域を使用できません。

(7) 使用例

PREPARE 文で前処理した SELECT 文（SQL 文識別子：PRESQL）で取得したユーザ定義型名（埋込み変数 1：WUSERID，埋込み変数 2：WTYPENAME）の属性情報，属性名，及び属性のユーザ定義型名を，SQL 記述領域，列名記述領域，及び型名記述領域に指定します。

```
DESCRIBE TYPE :WUSERID :WTYPENAME
    FOR PRESQL INTO :SQLDA :SQLCNDA :SQLTNDA
```

4.18　DROP LIST 文（リスト削除）

4.18.1　DROP LIST 文の形式と規則

(1)　機能

リストを削除します。

(2)　使用権限

リストの作成者

自分の作成したリストを削除できます。

(3)　形式

```
DROP { LISTリスト名 | ALL LIST }
```

(4)　オペランド

(a)　LIST リスト名

削除するリストの名前を指定します。

(b)　ALL LIST

自分の所有するリストをすべて削除します。

(5)　共通規則

1. DROP LIST 文は ROLLBACK 文の対象とはなりません。
2. DROP LIST 文は動的に実行できますが，直接ホストプログラムに埋め込んで実行できません。
3. 同一ユーザが，複数同時に HiRDB と接続してリストを操作できません。

(6)　留意事項

1. DROP ALL LIST を実行した場合，削除するリストがなくてもエラーにはなりません。

(7)　使用例

1. 扶養家族（FAMILYT）リストを削除します。

```
DROP LIST FAMILYT
```

2. 自分の所有するリストをすべて削除します。

```
DROP ALL LIST
```

4.19　EXECUTE 文　形式 1（SQL の実行）

4.19.1　EXECUTE 文 形式 1 の形式と規則

(1)　機能

PREPARE 文で前処理した SQL を実行します。

(2)　使用権限

なし。

(3)　形式 1 <前処理した SQL 文の実行>

```
EXECUTE　{SQL文識別子｜拡張文名}
 〔 {INTO　：埋込み変数〔：標識変数〕
             〔, ：埋込み変数〔：標識変数〕〕…
     ｜INTO DESCRIPTOR　〔：〕SQL記述領域名
                       〔CHARACTER_SET〔：〕文字集合名記述領域名〕} 〕
 〔 {USING　：埋込み変数〔：標識変数〕
              〔, ：埋込み変数〔：標識変数〕〕…
     ｜USING DESCRIPTOR　〔：〕SQL記述領域名
      〔CHARACTER_SET〔：〕文字集合名記述領域名〕} 〕
```

(4)　オペランド

(a)　{SQL 文識別子｜拡張文名}

SQL 文識別子

PREPARE 文で前処理した SQL に付けられた SQL 文識別子を指定します。

拡張文名

PREPARE 文で前処理した SQL 文を識別する拡張文名を指定します。

拡張文名については，「拡張文名」を参照してください。

(b)　INTO：埋込み変数〔：標識変数〕〔,：埋込み変数〔：標識変数〕〕…

埋込み変数

PREPARE 文で前処理した 1 行 SELECT 文の検索結果を受け取る場合，又は PREPARE 文で前処理した CALL 文に出力?パラメタが含まれる場合，その値を埋込み変数で受け取るときに，その検索結果の列の値，又は出力?パラメタを受け取る埋込み変数を指定します。

標識変数

埋込み変数に返される検索結果の列，又は出力パラメタの値がナル値かどうかを示す値が返される標識変数を指定します。

標識変数は，SMALLINT のデータ型を持つ埋込み変数として，埋込み SQL 宣言節で宣言してください。

標識変数を省略した場合は，ナル値を受け取れません。

(c) INTO DESCRIPTOR〔：〕SQL 記述領域名〔CHARACTER_SET〔：〕文字集合名記述領域名〕

SQL 記述領域名

PREPARE 文で前処理した 1 行 SELECT 文の検索結果，又は CALL 文の出力?パラメタの値を SQL 記述領域を用いて受け取るときに，その検索結果の列の値，又は出力?パラメタの値を受け取るための変数を記述した SQL 記述領域の名称を指定します。

文字集合名記述領域名

PREPARE 文で前処理した 1 行 SELECT 文の検索結果，又は CALL 文の出力?パラメタの値を SQL 記述領域を用いて受け取るときに，その検索結果の列の値，又は出力?パラメタの値を受け取るための変数の文字集合名を記述した文字集合名記述領域の名称を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(d) USING：埋込み変数〔：標識変数〕〔,：埋込み変数〔：標識変数〕〕…

埋込み変数

PREPARE 文で前処理した SQL に入力?パラメタが含まれる場合，その値を埋込み変数で与えるときに，その入力?パラメタに与える値を持つ埋込み変数を指定します。

USING 句で指定する埋込み変数の数と，EXECUTE 文で実行する SQL に含まれる入力?パラメタの数を同じにしてください。なお，埋込み変数と入力?パラメタは，それぞれ並びの順に先頭から対応付けられます。

USING 句で指定する埋込み変数のデータ型は，対応する入力?パラメタに対して許されるデータ型にしてください。

標識変数

埋込み変数の値がナル値かどうかを示す標識変数を指定します。

標識変数は，SMALLINT のデータ型を持つ埋込み変数として，埋込み SQL 宣言節で宣言してください。

標識変数を省略した場合は，埋込み変数の値はナル値以外と仮定されます。

(e) USING DESCRIPTOR〔：〕SQL 記述領域名〔CHARACTER_SET〔：〕文字集合名記述領域名〕

SQL 記述領域名

PREPARE 文で前処理した SQL に入力?パラメタが含まれる場合，その値を SQL 記述領域で与えるときに，その入力?パラメタの情報を格納した SQL 記述領域名を指定します。

文字集合名記述領域名

PREPARE 文で前処理した SQL の入力?パラメタの値を，SQL 記述領域を用いて与えるときに，その入力?パラメタの値を与えるための変数の文字集合名を記述した文字集合名記述領域の名称を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. EXECUTE 文で実行する SQL は，PREPARE 文で前処理しておく必要があります。
2. PREPARE 文とその PREPARE 文で前処理した SQL を実行する EXECUTE 文は，同じトランザクション内で実行してください。
3. SQL 文識別子は，埋込み変数と同様に，コンパイル単位のファイル内で有効な名前であり，同じ SQL 文識別子に対する複数の SQL を，複数のモジュールにわたって使用できません。
4. INTO 句で指定する埋込み変数の数と，EXECUTE 文で実行する 1 行 SELECT 文の検索結果の列の数，又は CALL 文の出力?パラメタの数を同じにしてください。実行する SQL が 1 行 SELECT 文の場合，埋込み変数の数と，列の数が同じでないときは，SQL 連絡領域の SQLWARN3 領域に警告フラグ”W”が設定されます。なお，埋込み変数と検索結果の列，又は埋込み変数と出力?パラメタは，それぞれ並びの順に先頭から対応付けられます。
5. INTO 句で指定する埋込み変数のデータ型は，対応する検索結果の列，又は出力?パラメタに対して許されるデータ型にしてください。
6. INTO 句で指定する固定長文字列（各国文字列，及び混在文字列を含む）の埋込み変数に取り出すデータが，その埋込み変数の定義長より短い場合は，左詰めに挿入され，余りの部分に空白が設定されます。
7. 検索結果の列の値又は CALL 文の出力?パラメタの値がナル値の場合，対応する埋込み変数の値は保証しません。
8. INTO 句に指定した埋込み変数が既定文字集合の文字データ型でかつ，検索結果の列又は CALL 文の出力?パラメタの文字集合が異なる文字データ型の場合，自動的に埋込み変数の文字集合に変換します。
9. USING 句に指定した埋込み変数が既定文字集合の文字データ型で，かつ入力?パラメタの文字集合と異なる場合，自動的に入力?パラメタの文字集合に変換します。

(6) 留意事項

1. PREPARE 文で前処理した一つの SQL に対して，EXECUTE 文は，何回でも実行できます。
2. SQL 記述領域の設定内容については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(7) 使用例

1. PREPARE 文で前処理した SQL（SQL 文識別子：PRESQL）を実行します。ただし，PRESQL の?パラメタに値を与える埋込み変数（HATENA）を指定します。

```
EXECUTE PRESQL USING :HATENA
```

2. PREPARE 文で前処理した SQL（SQL 文識別子：PRESQL）を実行します。ただし，PRESQL の?パラメタに値を与える情報を格納した SQL 記述領域（SQLDA）を指定します。

```
EXECUTE PRESQL USING DESCRIPTOR SQLDA
```

3. PREPARE 文で前処理した SQL（SQL 文識別子：PRESQL）を実行します。

   CALL PROC1（?，?，?）

   ただし，手続き PROC1 の第 1SQL パラメタのパラメタモードを IN，第 2SQL パラメタのパラメタモードを INOUT，第 3SQL パラメタのパラメタモードを OUT とします。また，第 1SQL パラメタに対しては埋込み変数 XPARAM1，第 2SQL パラメタに対しては埋込み変数 XPARAM2，第 3SQL パラメタに対しては埋込み変数 XPARAM3 を指定します。

```
EXECUTE PRESQL
    INTO  :XPARAM2,:XPARAM3
    USING :XPARAM1,:XPARAM2
```

4.20　EXECUTE 文　形式 2（配列を使用した SQL の実行）

4.20.1　EXECUTE 文 形式 2 の形式と規則

(1)　機能

PREPARE 文で前処理した SQL を，配列を使用して複数行分一括して実行します。

(2)　使用権限

なし。

(3)　形式 2 <前処理した SQL（INSERT 文，UPDATE 文，又は DELETE 文）の複数行分又は複数回分の一括実行>

```
EXECUTE  {SQL文識別子｜拡張文名}
         {USING ：埋込み変数配列〔：標識変数配列〕
                    〔，：埋込み変数配列〔：標識変数配列〕〕…
          ｜USING DESCRIPTOR 〔：〕SQL記述領域名
                    〔CHARACTER_SET〔：〕文字集合名記述領域名〕}
        BY ：埋込み変数〔ROWS〕
```

(4)　オペランド

(a)　{SQL 文識別子｜拡張文名}

SQL 文識別子

PREPARE 文で前処理した SQL に付けられた SQL 文識別子を指定します。

拡張文名

PREPARE 文で前処理した SQL 文を識別する拡張文名を指定します。

拡張文名については，「拡張文名」を参照してください。

(b)　USING ：埋込み変数配列〔：標識変数配列〕〔，埋込み変数配列〔：標識変数配列〕〕…

埋込み変数配列

PREPARE 文で前処理した，SQL に含まれる入力？パラメタの値を埋込み変数で与える場合に，入力？パラメタに与える値を持つ配列形式の埋込み変数を指定します。

USING 句で指定する埋込み変数配列の数と，EXECUTE 文で実行する SQL に含まれる入力？パラメタの数を同じにしてください。なお，埋込み変数配列と入力？パラメタは，それぞれの並びの順に先頭から対応付けられます。

USING 句で指定する埋込み変数配列のデータ型は，対応する入力？パラメタに対して許されるデータ型にしてください。

標識変数配列

埋込み変数の値が，ナル値かどうかを示す配列形式の標識変数を指定します。

標識変数配列は，SMALLINT のデータ型を持つ配列形式の埋込み変数として，埋込み SQL 宣言節で宣言してください。

標識変数配列を省略した場合は，埋込み変数配列の値はナル値以外と仮定されます。

(c) USING DESCRIPTOR 〔：〕SQL 記述領域名〔CHARACTER_SET 〔：〕文字集合名記述領域名〕

SQL 記述領域名

PREPARE 文で前処理した，SQL の入力?パラメタの値を SQL 記述領域を用いて与えるときに，その入力?パラメタの値を与えるための変数を記述した，SQL 記述領域の名称を指定します。

文字集合名記述領域名

PREPARE 文で前処理した，SQL の入力?パラメタの値を SQL 記述領域を用いて与えるときに，その入力?パラメタの値を与えるための変数の文字集合名を記述した，文字集合名記述領域の名称を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(d) BY ：埋込み変数〔ROWS〕

埋込み変数

PREPARE 文で前処理した SQL 文が INSERT 文の場合，処理（挿入）する行数を設定します。
PREPARE 文で前処理した SQL 文が UPDATE 文又は DELETE 文の場合，処理（更新又は削除）を実行する回数を設定した埋込み変数を指定します。

SMALLINT 型の埋込み変数を指定してください。

指定できる値の範囲を次に示します。

- オペランドに”USING ：埋込み変数配列〔：標識変数配列〕〔,：埋込み変数配列〔：標識変数配列]〕…”を指定する場合

  値の範囲は，1～4,096 です。
- オペランドに”USING DESCRIPTOR 〔：〕SQL 記述領域名”を指定する場合

  値の範囲は，1～30,000 です。

0，及び負の値は設定できません。範囲外の値を設定した場合は，実行時にエラーとなります。

ROW は指定してもしなくても意味は変わりません。

(5) 共通規則

1. EXECUTE 文で実行する SQL は，PREPARE 文で前処理しておく必要があります。

2. EXECUTE 文の形式 2 は，PREPARE 文で前処理した SQL 文が次に示す INSERT 文，UPDATE 文，又は DELETE 文の場合にだけ使用できます。

- INSERT INTO 〔認可識別子.〕表識別子 〔(列名〔，列名〕…)〕
  {VALUES（挿入値〔，挿入値〕…）
  ｜ 問合せ式本体 }
  〔WITH ROLLBACK〕
- INSERT INTO 〔認可識別子.〕表識別子（ROW）
  {VALUES（：埋込み変数配列〔：標識変数配列〕）
  ｜問合せ式本体 }
  〔WITH ROLLBACK〕
- UPDATE 〔認可識別子.〕表識別子 〔〔AS〕相関名〕
  〔使用インデクスの SQL 最適化指定〕
  SET｛更新対象＝更新値
  ｜（更新対象，更新対象〔，更新対象〕）＝行副問合せ｝
  〔，｛更新対象＝更新値
  ｜（更新対象，更新対象〔，更新対象〕…）＝行副問合せ｝〕…
  〔WHERE 探索条件〕
  〔WITH ROLLBACK〕
- UPDATE 〔認可識別子.〕表識別子 〔〔AS〕相関名〕
  〔使用インデクスの SQL 最適化指定〕
  SET ROW ＝行更新値
  〔WHERE 探索条件〕
  〔WITH ROLLBACK〕
- DELETE FROM 〔認可識別子.〕表識別子
  〔〔AS〕相関名〕
  〔使用インデクスの SQL 最適化指定〕
  〔WHERE 探索条件〕
  〔WITH ROLLBACK〕

PREPARE 文で前処理した SQL 文が上記の形式でない場合，実行時にエラーとなります。

3. PREPARE 文と，その PREPARE 文で前処理した SQL を実行する EXECUTE 文は，同じトランザクション内で実行してください。

4. SQL 文識別子は，埋込み変数と同様に，コンパイル単位のファイル内で有効な名前であり，同じ SQL 文識別子に対する複数の SQL を，複数のモジュールにわたって使用できません。

5. オペランドに”USING ：埋込み変数配列〔：標識変数配列〕〔,：埋込み変数配列〔：標識変数配列〕〕…”を指定する場合，次の規則に従ってください。

- 埋込み変数配列，及び標識変数配列の要素数は，1～4,096 の範囲にしてください。範囲外の値を指定した場合はエラーとなります。
- 埋込み変数配列，及び標識変数配列の要素数は，”BY ：埋込み変数〔ROWS〕”で指定する行数の最大値以上になるようにしてください。

6. オペランドに”USING DESCRIPTOR 〔：〕SQL 記述領域名”を指定する場合，次の規則に従ってください。
   - SQL 記述領域名で指定した，SQL 記述領域の SQLDATA が指す領域に，入力？パラメタの値を設定する場合は，マニュアル「HiRDB UAP 開発ガイド」を参照してください。このとき，配列の要素数は，”BY ：埋込み変数〔ROWS〕”で指定する行数の最大値以上になるようにしてください。
   - SQL 記述領域名で指定した SQL 記述領域の SQLSYS 領域に，データ型に応じた値を設定してください。

     ・可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）の場合は，文字列の長さを格納する領域と境界調整によって生じる要素間のギャップも含めた 1 要素分の長さを設定してください。

     (例)

     C言語では，次のような VARCHAR 型の配列変数の場合，SQLSYS に設定する値は，sizeof(vchr[0])となります。

     ```
     struct {
           short len;
           char str[257];
     } vchr[128];
     ```

     ・ほかのデータ型の場合は，0 を設定してください。
7. BLOB 型，最大長が 32,001 バイト以上の BINARY，及び抽象データ型は指定できません。
8. USING 句に指定した埋込み変数配列が既定文字集合の文字データ型で，かつ入力?パラメタの文字集合と異なる場合，自動的に入力?パラメタの文字集合に変換します。

(6) 留意事項

1. PREPARE 文で前処理した一つの SQL に対して，EXECUTE 文は何回でも実行できます。
2. EXECUTE 文形式 2 では，”BY ：埋込み変数”で指定する行数分の処理をするため，埋込み変数配列，標識変数配列，又は入力？パラメタの値を設定する，SQLDATA が指す領域には，必ず指定する行数分の値を設定しておいてください。

(7) 使用例

1. C 言語の配列変数に設定した 50 行分のデータを，在庫表に一括して挿入します。

```
EXEC SQL BEGIN DECLARE SECTION;
  short   XINSERT_NUM;
  long    XSCODE[50];
  short   ISCODE[50];
  char    XSNAME[50][17];
```

```
  short   ISNAME[50];
EXEC SQL END DECLARE SECTION;

EXEC SQL
  PREPARE PRESQL FROM
       'INSERT INTO ZAIKO(SCODE, SNAME) VALUES (?, ?)' ;

                  :
    各変数配列の要素に値を設定
                  :

XINSERT_NUM = 50;
EXEC SQL
  EXECUTE PRESQL USING :XSCODE:ISCODE, :XSNAME:ISNAME
      BY :XINSERT_NUM ROWS;
```

2. C 言語の配列変数に設定した商品コード（SCODE）及び在庫量（ZSURYO）の値別に，在庫量（ZSURYO）を次の表で示す値に更新します。

表 4-6　表に格納された商品コードと在庫量（更新前）

商品コード	在庫量の更新値
'101M'	40
'101L'	70
'201M'	15
'202M'	28
'302S'	7

表 4-7　更新対象となる行の商品コードと在庫量（埋込み変数配列に設定）

商品コード	在庫量の更新値
'101M'	35
'101L'	62
'201M'	13
'202M'	10
'302S'	6

表 4-8　表に格納された商品コードと在庫量（更新後）

商品コード	在庫量の更新値
'101M'	35
'101L'	62
'201M'	13
'202M'	10

商品コード	在庫量の更新値
‘302S’	6

```
EXEC SQL BEGIN DECLARE SECTION;
      short   XUPDATE_NUM;
      char    XSCODE[5][5];
      short   ISCODE[5];
      long    XZSURYO[5];
      short   IZSURYO[5];
EXEC SQL END DECLARE SECTION;
EXEC SQL
PREPARE PRESQL FROM
       ‘UPDATE ZAIKO SET ZSURYO = ? WHERE SCODE = ?’ ;
    ・・・各変数配列の要素に値設定・・・
        XSCODEに{’101M’,’101L’,’201M’,’202M’,’302S’}を設定
        XZSURYOに{35,62,13,10,6}を設定
XUPDATE_NUM = 5;
EXEC SQL
EXECUTE PRESQL USING :XZSURYO:IZSURYO, :XSCODE:ISCODE
BY :XUPDATE_NUM
```

4.21　EXECUTE IMMEDIATE 文（SQL の前処理と実行）

4.21.1　EXECUTE IMMEDIATE 文の形式と規則

(1)　機能

文字列で与えられた SQL を，前処理して実行します。

(2)　使用権限

なし。

(3)　形式

```
EXECUTE IMMEDIATE {’文字列‘|：埋込み変数}
 〔{INTO ：埋込み変数〔：標識変数〕
            〔, ：埋込み変数〔：標識変数〕〕…
    |INTO DESCRIPTOR 〔：〕SQL記述領域名
                    〔CHARACTER_SET〔：〕文字集合名記述領域名〕}〕
 〔{USING ：埋込み変数〔：標識変数〕
              〔, ：埋込み変数〔：標識変数〕〕…
    |USING DESCRIPTOR 〔：〕SQL記述領域名
                    〔CHARACTER_SET〔：〕文字集合名記述領域名〕}〕
```

(4)　オペランド

(a)　{’文字列’ |：埋込み変数}

文字列

実行する SQL の文字列を直接文字定数として指定します。

実行する SQL を文字定数として指定する場合で，SQL 中に 1 個のアポストロフィを表すには，2 個のアポストロフィを続けて指定してください。

埋込み変数

実行する SQL の文字列を埋込み変数で指定します。

埋込み変数を指定する場合は，前に：（コロン）を付けます。

(b)　INTO：埋込み変数〔：標識変数〕〔,：埋込み変数〔：標識変数〕〕…

埋込み変数

1 行 SELECT 文の検索結果を受け取る場合，又は CALL 文に出力?パラメタが含まれる場合，その値を埋込み変数で受け取るときに，その検索結果の列の値，又は出力?パラメタの値を受け取る埋込み変数を指定します。

標識変数

埋込み変数に返される検索結果の列の値がナル値かどうかを示す値が返される標識変数を指定します。

標識変数は，SMALLINT のデータ型を持つ埋込み変数として，埋込み SQL 宣言節で宣言してください。

標識変数を省略した場合は，ナル値を受け取れません。

(c) INTO DESCRIPTOR〔：〕SQL 記述領域名〔CHARACTER_SET〔：〕文字集合名記述領域名〕

SQL 記述領域名

1 行 SELECT 文の検索結果，又は CALL 文の出力?パラメタの値を SQL 記述領域を用いて受け取るときに，その検索結果の列の値，又は出力?パラメタの値を受け取るための変数を記述した，SQL 記述領域の名称を指定します。

文字集合名記述領域名

1 行 SELECT 文の検索結果，又は CALL 文の出力?パラメタの値を SQL 記述領域を用いて受け取るときに，その検索結果の列の値，又は出力?パラメタの値を受け取るための変数の文字集合名を記述した，文字集合名記述領域の名称を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(d) USING：埋込み変数〔：標識変数〕〔,：埋込み変数〔：標識変数〕〕…

埋込み変数

SQL に入力?パラメタが含まれる場合，その値を埋込み変数で与えるときに，その入力?パラメタに与える値を持つ埋込み変数を指定します。

USING 句で指定する埋込み変数の数と，EXECUTE IMMEDIATE 文で実行する SQL に含まれる入力?パラメタの数を同じにしてください。なお，埋込み変数と入力?パラメタは，それぞれ並びの順に先頭から対応付けられます。

USING 句で指定する埋込み変数のデータ型は，対応する入力?パラメタに対して許されるデータ型にしてください。

標識変数

埋込み変数の値がナル値かどうかを示す標識変数を指定します。

標識変数は，SMALLINT のデータ型を持つ埋込み変数として，埋込み SQL 宣言節で宣言してください。

標識変数を省略した場合は，埋込み変数の値はナル値以外と仮定されます。

(e) USING DESCRIPTOR〔：〕SQL 記述領域名〔CHARACTER_SET〔：〕文字集合名記述領域名〕

SQL 記述領域名

入力？パラメタの値を SQL 記述領域を用いて与えるときに，その入力?パラメタの値を与えるための変数を記述した，SQL 記述領域の名称を指定します。

文字集合名記述領域名

入力?パラメタの値を SQL 記述領域を用いて与えるときに，その入力?パラメタの値を与えるための変数の文字集合名を記述した，文字集合名記述領域の名称を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. 実行する SQL 文字列中に，SQL 先頭子，及び SQL 終了子は指定できません。
2. 実行する SQL の最大長は，2,000,000 バイトです。ただし，SQL の文字列を文字定数で指定した場合は，UAP 記述言語の文字定数の最大長です。
3. 埋込み変数の型は，次に示す構造体です。

```
struct {
      long  xxxxxxx;      /* SQL文の有効長    */
      char  yyyyyyy[n];   /* SQL文格納エリア */
} zzzzzzz;
```

（凡例）

xxxxxxx は，文字配列 yyyyyyy 中に格納した文字列の有効長を示します。

1 ≦ （xxxxxxx の値） ≦ 2000000

なお，文字列の有効長に文字列の終わりを示す‘￥ 0’は含みません。

n は，任意です。

4. INTO 句で指定する埋込み変数の数と，検索結果の列の数を同じにしてください。実行する SQL が 1 行 SELECT 文の場合，埋込み変数の数と，列の数が同じでないときは，SQL 連絡領域の SQLWARN3 領域に警告フラグ”W”が設定されます。なお，埋込み変数と検索結果の列，又は埋込み変数と出力?パラメタは，それぞれ並びの順に先頭から対応付けられます。
5. INTO 句で指定する埋込み変数のデータ型は，対応する列，出力パラメタのデータ型，又は変換できるデータ型にしてください。
6. INTO 句で指定する固定長文字列（各国文字列，及び混在文字列を含む）の埋込み変数に取り出すデータが，その埋込み変数の定義長より短い場合は，左詰めに挿入され，余りの部分に空白が設定されます。
7. 検索結果の列の値又は CALL 文の出力?パラメタの値がナル値の場合，対応する埋込み変数の値は保証しません。
8. INTO 句に指定した埋込み変数が既定文字集合の文字データ型でかつ，検索結果の列又は CALL 文の出力?パラメタの文字集合が異なる文字データ型の場合，自動的に埋込み変数の文字集合に変換します。
9. USING 句に指定した埋込み変数が既定文字集合の文字データ型で，かつ入力?パラメタの文字集合と異なる場合，自動的に入力?パラメタの文字集合に変換します。

(6) 留意事項

1. EXECUTE IMMEDIATE 文は，次に示す SQL を実行する場合と同じです。文字列で指定した SQL を繰り返して実行する場合は，一度 PREPARE 文で前処理して，EXECUTE 文で繰り返し実行することをお勧めします。

   - PREPARE SQL 文識別子 FROM｛’文字列‘|：埋込み変数｝
   - EXECUTE SQL 文識別子

   EXECUTE IMMEDIATE 文で前処理・実行できる SQL を次に示します。

   - 操作系 SQL

     ASSIGN LIST 文，CALL 文，DELETE 文，準備可能動的 DELETE 文：位置付け，DROP LIST 文，INSERT 文，PURGE TABLE 文，1 行 SELECT 文，UPDATE 文，準備可能動的 UPDATE 文：位置付け
   - 制御系 SQL

     LOCK TABLE 文
   - 定義系 SQL

     ALTER INDEX，ALTER PROCEDURE，ALTER ROUTINE，ALTER TABLE，ALTER TRIGGER，COMMENT，CREATE AUDIT，CREATE CONNECTION SECURITY，CREATE FUNCTION，CREATE INDEX，CREATE PROCEDURE，CREATE SCHEMA，CREATE SEQUENCE，CREATE TABLE，CREATE TRIGGER，CREATE TYPE，CREATE VIEW，DROP AUDIT，DROP CONNECTION SECURITY，DROP DATA TYPE，DROP FUNCTION，DROP INDEX，DROP PROCEDURE，DROP SCHEMA，DROP SEQUENCE，DROP TABLE，DROP TRIGGER，DROP VIEW，GRANT，REVOKE

(7) 使用例

1. 文字列として与えた SQL’PURGE TABLE ZAIKO’を前処理して実行します。

```
EXECUTE IMMEDIATE ‘PURGE TABLE ZAIKO’
```

2. 埋込み変数（：ZAIX）として定義した SQL を前処理して実行します。

```
EXECUTE IMMEDIATE :ZAIX
```

3. 文字列として与えた SQL’SELECT SNAME FROM ZAIKO WHERE SCODE ＝?’を前処理して実行し，検索結果を埋込み変数（:XSNAME)，標識変数(:ISNAME)に読み込みます。このとき，?パラメタに与える値の情報を格納した埋込み変数（:XSCODE)，標識変数(:ISCODE)を指定します。

```
EXECUTE IMMEDIATE ’ SELECT SNAME FROM ZAIKO WHERE SCODE = ?’
      INTO :XSNAME:ISNAME
      USING :XSCODE:ISCODE
```

4.22　FETCH 文　形式 1（データの取り出し）

4.22.1　FETCH 文 形式 1 の形式と規則

(1)　機能

取り出す行を示すカーソルの位置を次の行に進め，その 1 行の列の値を INTO 句で指定した埋込み変数に読み込みます。

(2)　使用権限

なし。

(3)　形式 1 <検索結果中の 1 行を変数に読み込む>

```
FETCH  {カーソル名|拡張カーソル名}  INTO
   {:埋込み変数〔:標識変数〕
    |〔文ラベル.〕SQL変数名
    |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
    |〔文ラベル.〕SQL変数名..属性名〔..属性名〕…
    |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名..属性名〔..属性名〕…}
      〔,  {:埋込み変数〔:標識変数〕
            |〔文ラベル.〕SQL変数名
            |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
            |〔文ラベル.〕SQL変数名..属性名〔..属性名〕…
            |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
                 ..属性名〔..属性名〕…}〕…
```

(4)　オペランド

(a)　{カーソル名|拡張カーソル名}

カーソル名

検索結果を取り出すカーソルの名称を指定します。

拡張カーソル名

検索結果を取り出すカーソルを識別する拡張カーソル名を指定します。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(b)　:埋込み変数〔:標識変数〕

埋込み変数

ナル値以外の列の値を読み込むための埋込み変数を指定します。

ナル値を含む列の値を受け取る場合は，埋込み変数と，標識変数を指定します。

標識変数

埋込み変数に読み込まれる列の値がナル値かどうかを示す値が返される標識変数を指定します。

(c) 〔文ラベル.〕SQL 変数名

SQL 手続き中で列の値を受け取るために，SQL 変数を指定します。

(d) 〔〔認可識別子.〕ルーチン識別子.〕SQL パラメタ名

SQL 手続き中で列の値を受け取るために，SQL パラメタを指定します。

パブリック手続き定義の SQL 手続き文中で認可識別子を指定する場合は，認可識別子に PUBLIC を指定してください。

(e) 〔文ラベル.〕SQL 変数名..属性名〔..属性名〕…

SQL 手続き中で，列中の属性の値を受け取るために指定します。

(f) 〔〔認可識別子.〕ルーチン識別子.〕SQL パラメタ名..属性名〔..属性名〕…

SQL 手続き中で，列中の属性の値を受け取るために指定します。

パブリック手続き定義の SQL 手続き文中で認可識別子を指定する場合は，認可識別子に PUBLIC を指定してください。

(5) 共通規則

1. FETCH 文で指定するカーソルは，ALLOCATE CURSOR 文 形式 2（結果集合カーソルの割当て）で割り当てた場合以外は，OPEN 文で開いておいてください。
2. 検索結果の列の個数（カーソル宣言，又は ALLOCATE CURSOR 文に指定した SELECT 文で指定した選択式の個数）と，FETCH 文の INTO 句で指定した埋込み変数，SQL 変数，又は SQL パラメタの個数は同じにしてください。個数が異なる場合は，個数が少ない方に合わせて列の値を埋込み変数に読み込みます。このとき，SQL 連絡領域の SQLWARN3 領域に警告フラグ（W）が設定されます。
3. INTO 句で指定する埋込み変数のデータ型は，対応する検索項目のデータ型，又は変換できるデータ型にしてください。
4. 埋込み変数が既定文字集合の文字データ型で，かつ検索結果がその埋込み変数の文字集合と異なる文字データ型の場合，自動的に埋込み変数の文字集合に変換します。
5. 固定長文字列（各国文字列，及び混在文字列を含む）の埋込み変数に取り出すデータが，検索項目の長さより短い場合は，左詰めに挿入され，余りの部分に空白が設定されます。
6. 取り出す行がない場合，次のリターンコードが設定されます。
   - SQL 連絡領域の SQLCODE 領域に 100
   - SQLCODE 変数に 100
   - SQLSTATE 変数に’02000’

ただし，リストを介した検索でリスト作成時にあった行が削除された場合，又は属性値が削除，更新された場合，それぞれ 110,110,’R2000’ が設定されます。

(6) 留意事項

1. 検索結果の列の値がナル値の場合は，対応する埋込み変数の値は保証しません。
2. カーソル名は，埋込み変数名と同様に，コンパイル単位のファイル内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のファイルにわたって使用できません。

(7) 使用例

カーソル（CR1）による在庫表（ZAIKO）の検索結果を商品コード（SCODE）列，商品名（SNAME）列，色（COL）列，単価（TANKA）列，在庫量（ZSURYO）列に対応する埋込み変数，及び標識変数に読み込みます。

＜埋込み変数を指定した場合＞

```
FETCH CR1
    INTO :XSCODE,:XSNAME,:XCOL,
        :XTANKA,:XZSURYO
```

＜埋込み変数，標識変数を指定した場合＞

```
FETCH CR1
    INTO :XSCODE:ISCODE,
         :XSNAME:ISNAME,
         :XCOL:ICOL,
         :XTANKA:ITANKA,
         :XZSURYO:IZSURYO
```

4.23　FETCH 文　形式 2（データの取り出し）

4.23.1　FETCH 文 形式 2 の形式と規則

(1)　機能

検索結果中の 1 行，又は複数行を，SQL 記述領域に指定した受け取り領域に読み込みます。

(2)　使用権限

なし。

(3)　形式 2 <検索結果中の 1 行，又は複数行を，SQL 記述領域に指定した受け取り領域に読み込む>

```
FETCH　{カーソル名｜拡張カーソル名}
     USING DESCRIPTOR　〔：〕SQL記述領域名
                    〔CHARACTER_SET〔：〕文字集合名記述領域名〕
          〔BY　：埋込み変数〔ROWS〕〕
```

(4)　オペランド

(a)　{カーソル名｜拡張カーソル名}

カーソル名

検索結果を取り出すカーソルの名称を指定します。

拡張カーソル名

検索結果を取り出すカーソルを識別する拡張カーソル名を指定します。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(b)　〔：〕SQL 記述領域名

検索結果を受け取るために必要な情報を設定した SQL 記述領域の名称を指定します。

(c)　〔CHARACTER_SET〔：〕文字集合名記述領域名〕

検索結果の文字集合名を受け取るために必要な情報を設定した文字集合名記述領域の名称を指定します。

(d)　〔BY　：埋込み変数〔ROWS〕〕

SQL 記述領域に設定された配列を使用して，FETCH 用の領域の大きさを要素数で設定した埋込み変数を指定します。SMALLINT 型の埋込み変数を指定してください。設定値は 1～30,000 の範囲にしてくださ

い。0 及び負の値を設定した場合はエラーとなります。また，バージョン 05-03 より前のクライアントライブラリを使用した場合には，動作が保証されません。

(5) 共通規則

1. FETCH 文で指定するカーソルは，ALLOCATE CURSOR 文 形式 2（結果集合カーソルの割当て）で割り当てた場合以外は，OPEN 文で開いておいてください。
2. UAP が FETCH 文を実行するときに必要な情報は，SQL 記述領域名で指定した SQL 記述領域中に設定しておいてください。SQL 記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。
3. 取り出す行がない場合，次のリターンコードが設定されます。
   - SQL 連絡領域の SQLCODE 領域に 100
   - SQLCODE 変数に 100
   - SQLSTATE 変数に’02000’

   ただし，リストを介した検索でリスト作成時にあった行が削除された場合，又は属性値が削除，更新された場合，それぞれ 110,110,’R2000’が設定されます。
4. SQL 記述領域で指定する受取り領域のデータ型は，対応する検索項目のデータ型と変換できるデータ型です。
5. 検索結果が文字データ型で，検索結果の文字集合が文字集合名記述領域に指定した文字集合と異なる場合，自動的に文字集合名記述領域に指定した文字集合に変換します。
6. BY ：埋込み変数〔ROWS〕を指定する場合，SQL 記述領域の SQLSYS にデータ型に対応した値を設定しておく必要があります。
   - 可変長文字列（VARCHAR，NVARCHAR，又は MVARCHAR）の場合

     文字列の長さを格納する領域と，境界調整によって生じる要素間のギャップも含めた 1 要素分の値を設定してください。

     例えば，次のような VARCHAR 型の配列変数の場合は，SQLSYS に設定する値は sizeof(vchr[0]) となります。

```
struct {
    short   len;
    char    str[257];
} vchr[128];
```

   - ほかのデータ型の場合

     0 を設定してください。

(6) 留意事項

1. カーソル名は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって使用できません。

(7) 使用例

カーソル（CR2）による在庫表の検索結果を，SQL 記述領域に指定した受け取り領域に読み込みます。

```
FETCH CR2
    USING DESCRIPTOR SQLDA
```

4.24　FETCH 文　形式 3（データの取り出し）

4.24.1　FETCH 文 形式 3 の形式と規則

(1)　機能

検索結果中の複数行を，INTO 句で指定した埋込み変数に一度に読み込みます。

(2)　使用権限

なし。

(3)　形式 3 <検索結果中の複数行を，INTO 句で指定した埋込み変数に一度に読み込む>

```
FETCH　{カーソル名｜拡張カーソル名}　 INTO　：埋込み変数配列〔：標識変数配列〕
                    〔，　：埋込み変数配列〔：標識変数配列〕〕…
```

(4)　オペランド

(a)　{カーソル名｜拡張カーソル名}

カーソル名

検索結果を取り出すカーソルの名称を指定します。

拡張カーソル名

検索結果を取り出すカーソルを識別する拡張カーソル名を指定します。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(b)　：埋込み変数配列〔：標識変数配列〕〔，：埋込み変数配列〔：標識変数配列〕〕…

埋込み変数配列

ナル値以外の列の値を読み込むための配列変数（配列形式で宣言した埋込み変数）を指定します。ナル値を含む列の値を受け取る場合は，埋込み変数と標識変数を指定します。

標識変数配列

埋込み変数配列に読み込まれる列の値がナル値かどうかを示す値が返される標識変数（配列形式で宣言した標識変数）を指定します。

(5) 共通規則

1. FETCH 文で指定するカーソルは，ALLOCATE CURSOR 文 形式 2（結果集合カーソルの割当て）で割り当てた場合以外は，OPEN 文で開いておいてください。
2. 埋込み変数配列，及び標識変数配列の配列の要素数は，すべて同じ大きさにしてください。異なっている場合，取り出す行数は指定された配列の中で最も小さい配列の要素数に合わせます。
3. 実際に取り出した行の累積行数は，SQL 連絡領域の SQLERRD2（C言語の場合），又は SQLERRD(3)（COBOL 言語の場合）に設定されます。
4. 配列を使用した FETCH は，手続きの中では使えません。
5. 配列を使用した FETCH では，LOB データは扱えません。
6. 配列を使用した FETCH では，ブロック転送の指定は無効です。
7. 取り出し中に取り出す行がなくなった場合，次のリターンコードが設定されます。この場合，なくなる直前までの行のデータは返されます。
   - SQL 連絡領域の SQLCODE 領域に 100
   - SQLCODE 変数に 100
   - SQLSTATE 変数に’02000’

   また，リストを介した検索で，リスト作成時にあった行が削除された場合，又は属性値が削除，更新された場合，それを示すリターンコードは設定されません。この場合，無視して検索が継続されます。
8. 取り出したどれかの行で警告する必要がある事象が発生した場合，SQL 連絡領域の SQLWARN に警告情報が設定されます。
9. 取り出した行のどれかの行でエラーが発生した場合，その行までのデータは返されます。
10. 配列を使用した FETCH は，添字なし繰返し列の検索には使用できません。

(6) 留意事項

1. 検索結果の列の値がナル値の場合は，それに対応する埋込み変数配列の要素の値は保証しません。
2. 配列でない変数と配列型の変数は混在できません。
3. SQL 記述領域は指定できません。
4. カーソルは FETCH をした最後の行に位置づけられます。カーソルを使用した更新を実行すると，FETCH をした最後の行が更新されます。

(7) 使用例

カーソル（CR3）による在庫表の検索結果を，C言語の配列変数に読み込みます。

```
EXEC  SQL  BEGIN  DECLARE  SECTION;
  long    XSCODE[50];
  short   ISCODE[50];
  char    XSNAME[50][17];
```

```
  short   ISNAME[50];
EXEC  SQL  END  DECLARE  SECTION;

EXEC  SQL  FETCH  CR3
    INTO   :XSCODE  :ISCODE,
             :XSNAME  :INAME;
```

4.25　FREE LOCATOR 文（位置付け子の無効化）

4.25.1　FREE LOCATOR 文の形式と規則

(1)　機能

位置付け子と割り当てられたデータの関連を削除し，その位置付け子を無効にします。

(2)　使用権限

なし。

(3)　形式

```
FREE LOCATOR 位置付け子参照〔, 位置付け子参照〕…
位置付け子参照：：＝：埋込み変数
```

(4)　オペランド

(a)　位置付け子参照：：＝：埋込み変数

無効にする位置付け子の埋込み変数を指定します。

(5)　共通規則

1. 無効な位置付け子の値を持つ埋込み変数を指定した場合はエラーとなります。また，複数の埋込み変数を指定し，その中に無効な位置付け子の値を持つ埋込み変数が含まれていた場合はエラーとなりますが，有効な位置付け子はすべて無効となります。

4.26　INSERT 文　形式 1（行挿入）

4.26.1　INSERT 文 形式 1 の形式と規則

(1)　機能

表に，行を列単位で挿入します。直接，値を指定して一つの行の挿入ができます。また，問合せ式本体を使用して，一つ，又は複数の行の挿入もできます。

(2)　使用権限（形式 1）

表に対する INSERT 権限を持つユーザが，その表に行を挿入できます。

ただし，INSERT 文中に問合せ指定を指定する場合は，その問合せ指定の表に対して SELECT 権限が必要です。

(3)　形式 1<列単位で表に行を挿入する>

```
INSERT INTO 〔認可識別子.〕表識別子
       {〔（列名〔, 列名〕…）〕
           {VALUES（挿入値〔, 挿入値〕…）
         |問合せ式本体}
         |DEFAULT VALUES}
       〔WITH ROLLBACK〕
       〔WRITE IMMEDIATE〕

問合せ式本体：：＝｛問合せ指定
                   |（問合せ式本体）
                   |問合せ式本体 {UNION|EXCEPT} 〔ALL〕
                     {問合せ指定|（問合せ式本体）} }
問合せ指定   ：：＝SELECT〔 { ALL | DISTINCT} 〕 {選択式〔, 選択式〕…|＊} 表式
表式         ：：＝FROM  表参照〔, 表参照〕…
                     〔WHERE 探索条件〕
                     〔GROUP BY 値式〔, 値式〕…〕
                     〔HAVING 探索条件〕
```

(4)　オペランド

(a)　〔認可識別子.〕表識別子

認可識別子

表の所有者の認可識別子を指定します。

認可識別子に，"MASTER"は指定できません。

表識別子

行を挿入する表の名称を指定します。

(b) 〔(列名〔，列名〕…)〕

データを挿入する列の名称の並びを指定します。

列名の並びを省略した場合は，表定義で表を定義したときの列の指定順序に従って，すべての列（予備列を除く）を指定したことになります。予備列には，予備列の定義長分の 0x00 が自動的に挿入されます。

列名についての規則を次に示します。

1. 読み込み専用のビュー表は，行の挿入，更新，及び削除ができません。読み込み専用のビュー表については，「共通規則」を参照してください。
2. ビュー表への行の挿入を指定した場合，そのビュー表の基の実表に行を挿入します。ビュー表の基の実表の列でビュー表の列に対応しない列はナル値になります。したがって，FIX 属性の実表から定義したビュー表の場合，ビュー表の基の実表の列でビュー表の列に対応しない列があるときは行は挿入できません。
3. 指定されていない列の値は列の規定値になります。DEFAULT 句がある場合は，指定した既定値となります。DEFAULT 句の指定がなく，WITH DEFAULT の指定がある場合は，WITH DEFAULT の既定値となります。DEFAULT 句，及び WITH DEFAULT のどちらの指定もない場合，NULL が既定値となります。予備列の場合は予備列の定義長分の 0x00 になります。
4. 指定されていない繰返し列の要素数は 0 になります。
5. 列名には添字を指定できません。
6. 列名には予備列を指定できません。

(c) VALUES（挿入値〔，挿入値〕…）

挿入値

列名で指定した各列に対応した挿入値を指定します。指定できる項目を次に示します。

- 値式
- NULL（ナル値を表します）
- DEFAULT
- ARRAY ［要素の値〔，要素の値〕…］※

注※

要素の値には，次の項目を指定できます。

- 値式
- NULL（ナル値を表します）
- DEFAULT

挿入値についての規則を次に示します。

1. 挿入値は，列名の指定順序に従って指定します。

2. 挿入値にナル値を入れる場合は，NULL と指定します。
3. 挿入値，又は挿入値の要素の値に DEFAULT を指定した場合，挿入の対象となる列の既定値を挿入します。DEFAULT 句がある場合は，指定した既定値となります。DEFAULT 句の指定がなく，WITH DEFAULT の指定がある場合は，WITH DEFAULT の既定値となります。DEFAULT 句，及び WITH DEFAULT のどちらの指定もない場合，NULL が既定値となります。
4. 挿入値に対応する列に SYSTEM GENERATED を指定している場合，指定した挿入値は無視され，DATE 型の場合は現在の日付（CURRENT_DATE），TIME 型の場合は現在の時刻（CURRENT_TIME）を挿入します。
5. PREPARE 文で前処理する場合は，埋込み変数，標識変数を指定できません。
6. 埋込み変数は，対応する列の列構造と同じ構造の埋込み変数を指定します。
7. ?パラメタに値を指定するための埋込み変数の場合も，対応する列と同じ構造の埋込み変数を指定します。
8. ARRAY ［要素の値〔，要素の値〕…］は，対応する列が繰返し列の場合にだけ指定できます。要素の値は，最大 30,000 個指定できます。ただし，挿入する列の最大要素数以下にしてください。要素の値の埋込み変数（標識変数），？パラメタは，単純構造にしてください。
9. 挿入値に列名又は集合関数を含む値式は指定できません。
10. 挿入値に WRITE 指定及び GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。

(d) 問合せ式本体

```
::= {問合せ指定
      | （問合せ式本体）
      |問合せ式本体 {UNION|EXCEPT} 〔ALL〕
        {問合せ指定 | （問合せ式本体）} }
```

挿入するデータを取り出す問合せ式本体を指定します。

問合せ式本体については，「問合せ式」を参照してください。

(e) 問合せ指定：：= SELECT〔{ ALL | DISTINCT}〕

```
{選択式〔，選択式〕…|＊} 表式
```

問合せ指定については，「問合せ指定」を参照してください。

取り出したデータを挿入する列が繰返し列の場合，その列に対応する問合せ指定の選択式には，添字なしで繰返し列を指定します。

(f) 表式

```
::＝FROM　表参照〔，表参照〕…
  〔WHERE 探索条件〕
```

```
〔GROUP BY 値式〔，値式〕…〕
〔HAVING 探索条件〕
```

表式については，「表式」，表参照については，「表参照」，探索条件については，「探索条件」を参照してください。

(g) DEFAULT VALUES

挿入する行のすべての列に既定値を挿入します。

DEFAULT VALUES は，次の形式を指定する場合と同じ意味になります。

```
VALUES(DEFAULT,DEFAULT,…)
```

上記の DEFAULT の数は，挿入対象の表の列数と同じです。

(h) 〔WITH ROLLBACK〕

挿入の対象となる表が，ほかのユーザで使用されているときは，トランザクションを取り消して無効にする場合，指定します。

WITH ROLLBACK を省略した場合は，挿入の対象となる表が，ほかのユーザで使用されているとき，使用中のユーザのトランザクションが終了してから実行します。

(i) 〔WRITE IMMEDIATE〕

表オプションに WITHOUT ROLLBACK の指定がある表に対する更新処理で，更新完了時にシステムログを書き出す場合，このオプションを指定してください。WITHOUT ROLLBACK 指定のない表に対して，このオプションを指定しても無効になります。

更新完了時にシステムログを書き出す効果については，マニュアル「HiRDB UAP 開発ガイド」の採番業務で使用する表を参照してください。

(5) 共通規則

1. 埋込み変数，SQL 変数，又は SQL パラメタのデータ型は対応する列のデータ型，又は変換できるデータ型にしてください。
2. INSERT 文で?パラメタが指定できるのは，PREPARE 文で前処理された場合だけです。?パラメタに対して与える値は，前処理した PREPARE 文に対応する EXECUTE 文の USING 句の埋込み変数で指定します。
3. 埋込み変数，標識変数は，PREPARE 文で前処理される INSERT 文，及び SQL 手続き中では使用できません。

   SQL 手続き中では，SQL 変数，又は SQL パラメタを使用します。Java 手続き中の指定値については，マニュアル「HiRDB UAP 開発ガイド」の JDBC ドライバ又は SQLJ を参照してください。
4. 挿入する 1 行分の列の個数は，列名で指定した列の個数と同じにしてください。

また，それらの値は，列のデータ型，又は変換できるデータ型にしてください（ただし，データを挿入する列が各国文字データ型で，挿入値に文字列定数を指定した場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードはチェックされないで，文字データの長さだけがチェックされます）。複数の行を挿入する場合，問合せ式本体で検索する列の個数と列名で指定した列の個数を同じにしてください。また，対応する列のデータ型は変換できるデータ型にしてください（ただし，データを挿入する列が各国文字データ型で，問合せ式本体の選択式に文字列定数を指定した場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードはチェックされないで，文字データの長さだけがチェックされます）。

5. 固定小数点，又は浮動小数点のデータを次に示すデータ型の列に挿入する場合は，端数（小数）部分は切り捨てられます。

   - INTEGER
   - SMALLINT

   また，固定小数点のデータを DECIMAL 型の列に挿入する場合は，列の位取りより下位のけた部分が切り捨てられます。

6. 表を定義したときに指定した長さ以上の文字データは，挿入できません。
7. 列に定義されているデータ型の範囲外の数データは，挿入できません。
8. 固定長文字列（各国文字列，及び混在文字列を含む）の列に挿入するデータが，列の長さより短い場合は，左詰めに挿入され，余りの部分に空白が設定されます。
9. データ型が BLOB の列にデータを挿入する場合，挿入値には，埋込み変数，?パラメタ，SQL 変数，SQL パラメタ，スカラ関数 SUBSTR，関数呼出し，又は NULL が指定できます。
10. 抽象データ型の列にデータを挿入する場合，挿入値には埋込み変数及び？パラメタは指定できません。
11. 問合せ式本体の選択式にコンポネント指定は指定できません。
12. 抽象データ型の列にデータを挿入する場合，挿入値には，表定義時に LOB 属性格納用 RD エリア名を指定していない BLOB 属性を含む抽象データ型の値は指定できません。
13. 問合せ式本体の選択式には，WRITE 指定及び GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。
14. WITHOUT ROLLBACK を指定した表に対して，INSERT 文を実行する場合，インデクスの定義有無によって行排他の解除タイミングが異なります。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
15. 共用表にデータを挿入する場合，事前にその表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表にデータ挿入した場合，エラーとなります。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。
16. 文字データ型の列にデータを挿入する場合，挿入対象の列と，対応する挿入値の文字集合を同じにしてください。ただし，挿入値が埋込み変数（既定文字集合），?パラメタ，又は文字列定数の場合，自動的に挿入対象の列の文字集合に変換します。

(6) 参照制約に関する規則

1. 被参照表，参照表に行を挿入する場合の規則は「CREATE TABLE（表定義）」の参照動作の説明を参照してください。

(7) 使用例

1. 在庫表（ZAIKO）のすべての列に，埋込み変数に読み込まれた値の行を挿入します。

```
INSERT INTO ZAIKO
    VALUES(:XSCODE,:XSNAME,:XCOL,
           :XTANKA,:XZSURYO)
```

2. 在庫表（ZAIKO）の商品コード（SCODE）列，商品名（SNAME）列，在庫量（ZSURYO）列の各列に，埋込み変数に読み込まれた値の行を挿入します。

```
INSERT INTO ZAIKO
    (SCODE,SNAME,ZSURYO)
    VALUES(:XSCODE,:XSNAME,:XZSURYO)
```

3. 在庫表（ZAIKO）に在庫表と同じ列定義情報を持つ在庫表２（ZAIKO2）のすべての表データを挿入します。

```
INSERT INTO ZAIKO
    SELECT * FROM ZAIKO2
```

4. 在庫表（ZAIKO）の商品コード（SCODE），商品名（SNAME），色（COL）の各列に，612S，ズボン，白のデータを挿入します。

```
INSERT INTO ZAIKO(SCODE,SNAME,COL)
    VALUES( '612S',N'ズボン',N'白')
```

5. 受注表（JUTYU）の各列に，02561，TT001，302S，50，現在の日付（CURRENT_DATE），現在の時刻（CURRENT_TIME）を挿入します。

```
INSERT INTO JUTYU
    VALUES( '02561','TT001','302S',50,CURRENT_DATE,CURRENT_TIME)
```

4.27　INSERT 文　形式 2（行挿入）

4.27.1　INSERT 文 形式 2 の形式と規則

(1)　機能

行全体を一つのデータとみなして，FIX 属性の表に，行を行単位で挿入します。直接，値を指定して一つの行の挿入ができます。また，問合せ式本体を使用して，一つ，又は複数の行の挿入もできます。

(2)　使用権限（形式 2）

表に対する INSERT 権限を持つユーザが，その表に行を挿入できます。

ただし，INSERT 文中に問合せ指定を指定する場合は，その問合せ指定の表に対して SELECT 権限が必要です。

(3)　形式 2<FIX 属性の表に，一つの行全体を一つのデータとみなして行単位で行を挿入>

```
INSERT INTO 〔認可識別子.〕表識別子 (ROW)
     {VALUES (行挿入値)
       | 問合せ式本体 }
     〔WITH ROLLBACK〕
     〔WRITE IMMEDIATE〕
問合せ式本体：：＝ {問合せ指定
                    | (問合せ式本体)
                    | 問合せ式本体 {UNION | EXCEPT} 〔ALL〕
                      {問合せ指定 | (問合せ式本体) } }
問合せ指定   ：：＝SELECT〔 { ALL  |  DISTINCT} 〕 {選択式〔, 選択式〕… | *} 表式
表式         ：：＝FROM  表参照〔, 表参照〕…
                    〔WHERE 探索条件〕
                    〔GROUP BY 値式〔, 値式〕…〕
                    〔HAVING 探索条件〕
```

(4)　オペランド

ROW 及び VALUES 以外の説明については，「INSERT 文　形式 1（行挿入）」を参照してください。

(a)　(ROW)

行単位でデータを挿入する場合に指定します。ROW を指定する場合の規則を次に示します。

1. FIX 属性の実表に対してだけ指定できます。ROW は行全体（予備列を含む）意味し，これを指定することで行全体を一つのデータとして，一つの領域から挿入します。挿入するデータのデータ型は，各列のデータ型に関係なく，ROW 型（ROW 型に対しては，CHAR（n）〔n は行長〕に対応する変数，

又は同じ長さの構造体を指定できます。ただし，構造体中に境界調整による空きがあってはいけません）にしてください。また，データ長は行長（各列のデータ長の総和）にしてください。

2. UAP が動作するプラットフォームと HiRDB サーバが動作するプラットフォームのエンディアンを同じにしてください。異なるエンディアン間では，ROW は使用できません。例えば，Windows の UAP で ROW を使用する場合は，HiRDB サーバも同じエンディアンの Windows 版を使用してください。

(b) {VALUES（行挿入値）｜問合せ式本体｝

行挿入値

ROW に対応した行の挿入値を指定します。指定できる項目を次に示します。

- ：埋込み変数〔：標識変数〕
- ?パラメタ
- SQL 変数，又は SQL パラメタ

(c) 問合せ式本体

```
::= {問合せ指定
      | （問合せ式本体）
      |問合せ式本体 {UNION|EXCEPT} 〔ALL〕
        {問合せ指定|（問合せ式本体）} }
```

挿入するデータを取り出す問合せ式本体を指定します。

問合せ式本体については，「問合せ式」を参照してください。

(d) 問合せ指定

```
::=SELECT〔 { ALL | DISTINCT} 〕
  {選択式〔, 選択式〕…|*} 表式
```

問合せ指定については，「問合せ指定」を参照してください。

(e) 表式

```
::=FROM 表参照〔, 表参照〕…
  〔WHERE 探索条件〕
  〔GROUP BY 値式〔, 値式〕…〕
  〔HAVING 探索条件〕
```

表式については，「表式」，表参照については，「表参照」，探索条件については，「探索条件」を参照してください。

(5) 留意事項

1. 挿入する表の列のデータ型が DECIMAL，又は各国文字列の場合に，対応する部分だけ行挿入値の内容をチェックします。
2. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の日付データ型の部分は 4 バイトであり，X’YYYYMMDD’の形式にしてください。行単位（ROW 指定）インタフェースを使用し，日付データを既定の文字列表現で受け渡しする場合，列定義時に日付データ型ではなく，CHAR（10）として列を定義してください。さらに，日付演算は，スカラ関数 DATE を使用し，日付データ型に変換した後に指定してください。
3. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の時刻データ型の部分は 3 バイトであり，X’hhmmss’の形式にしてください。行単位（ROW 指定）インタフェースを使用し，時刻データを既定の文字列表現で受け渡しする場合，列定義時に時刻データ型ではなく，CHAR（8）として列を定義してください。さらに，時刻演算は，スカラ関数 TIME を使用し，時刻データ型に変換した後に指定してください。
4. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の時刻印データ型の部分は，（7 ＋ p/2）バイトであり，X’YYYYMMDDhhmmss〔nn…n〕’の形式にしてください。行単位（ROW 指定）インタフェースを使用し，時刻印データを既定の文字列表現で受け渡しをする場合，列定義時に時刻印データ型ではなく，長さが 19，22，24，又は 26 バイトの CHAR として列を定義してください。
5. 挿入する表の列に SYSTEM GENERATED を指定している場合，対応する部分のデータは無視され，DATE 型の場合は現在の日付（CURRENT_DATE），TIME 型の場合は現在の時刻（CURRENT_TIME）を挿入します。
6. 挿入する表に予備列が定義されている場合，対応する部分のデータは無視され，予備列の定義長分の 0x00 を挿入します。

(6) 共通規則

1. SELECT 文を指定した場合の共通規則は，「INSERT 文　形式 1（行挿入）」の共通規則を参照してください。
2. WITHOUT ROLLBACK を指定した表に対して，INSERT 文を実行する場合，インデクスの定義有無によって行排他の解除タイミングが異なります。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
3. 共用表にデータを挿入する場合，事前にその表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表にデータ挿入した場合，エラーとなります。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。
4. 挿入する表に既定文字集合以外の文字データ型の列が存在する場合，行単位のデータ挿入はできません。

(7) 参照制約に関する規則

1. SELECT 文を指定した場合の参照制約に関する規則は，「INSERT 文　形式 1（行挿入）」の参照制約に関する規則を参照してください。
2. 参照表に行を挿入する場合，外部キー構成列の値が被参照表の主キー構成列の値に含まれるかどうかを確認するため，被参照表を検索します。このとき，被参照表の検索時のデータ保証レベルは共用モードになります。そのため，参照表への行の挿入時，ほかのトランザクションによって被参照表に対する操作が行われている場合は，そのトランザクションが決着されるまで参照表への行の挿入は待ち状態になります。

(8) 使用例

在庫表（ZAIKO）に埋込み変数（XROW）に読み込まれた 1 行分の値をまとめて挿入します。

```
INSERT INTO ZAIKO(ROW)
    VALUES(:XROW)
```

4.28　INSERT 文　形式 3，形式 4（配列を使用した行挿入）

4.28.1　INSERT 文 形式 3，形式 4 の形式と規則

(1)　機能

配列形式の埋込み変数を指定して複数の行の挿入ができます。

形式 3

　表に，行を列単位で複数行挿入します。

形式 4

　行全体を一つのデータとみなして，FIX 属性の表に，行を行単位で複数行挿入します。

(2)　使用権限

表に対する INSERT 権限を持つユーザが，その表に行を挿入できます。

ただし，INSERT 文中に問合せ指定を指定する場合は，その問合せ指定の表に対して SELECT 権限が必要です。

(3)　形式 3 <埋込み変数配列を指定して，表に列単位で複数行を挿入>

```
FOR ：埋込み変数
INSERT INTO 〔認可識別子.〕表識別子
       〔（列名〔, 列名〕…）〕
           {VALUES（挿入値〔, 挿入値〕…）
          |問合せ式本体 }
       〔WITH ROLLBACK〕
       〔WRITE IMMEDIATE〕

問合せ式本体：：＝｛問合せ指定
                   |（問合せ式本体）
                   |問合せ式本体 {UNION|EXCEPT} 〔ALL〕
                     {問合せ指定|（問合せ式本体）} }
問合せ指定　：：＝SELECT〔 { ALL | DISTINCT} 〕 {選択式〔, 選択式〕…|＊} 表式
表式　　：：＝FROM　表参照〔, 表参照〕…
                   〔WHERE 探索条件〕
                   〔GROUP BY 値式〔, 値式〕…〕
                   〔HAVING 探索条件〕
```

(4)　形式 4 <埋込み変数配列を指定して，FIX 属性の表に行単位で複数行を挿入>

```
FOR ：埋込み変数
INSERT INTO 〔認可識別子.〕表識別子（ROW）
```

```
      {VALUES（：埋込み変数配列〔：標識変数配列〕）
     | 問合せ式本体 }
     〔WITH ROLLBACK〕
     〔WRITE IMMEDIATE〕

問合せ式本体：：＝｛問合せ指定
                    |（問合せ式本体）
                    |問合せ式本体 {UNION | EXCEPT} 〔ALL〕
                      {問合せ指定 |（問合せ式本体）} }
問合せ指定   ：：＝SELECT〔 { ALL | DISTINCT} 〕 {選択式〔, 選択式〕… | ＊} 表式
表式         ：：＝FROM  表参照〔, 表参照〕…
                    〔WHERE 探索条件〕
                    〔GROUP BY 値式〔, 値式〕…〕
                    〔HAVING 探索条件〕
```

(5) オペランド

FOR，VALUES，及び ROW 以外の説明については，「INSERT 文　形式 1（行挿入）」，及び「INSERT 文　形式 2（行挿入）」を参照してください。

(a) <形式 3 のオペランド>

VALUES（挿入値〔, 挿入値〕…）

挿入値

列名で指定した各列に対応した挿入値を指定します。指定できる項目を次に示します。

- ：埋込み変数配列〔：標識変数配列〕
- 値式
- NULL（ナル値を表します）
- DEFAULT

ただし，挿入値には配列形式でない埋込み変数は指定できません。

(b) <形式 4 のオペランド>

(ROW)

行単位でデータを挿入する場合に指定します。ROW を指定する場合の規則を次に示します。

1. FIX 属性の実表に対してだけ指定できます。ROW は行全体（予備列も含む）を意味し，これを指定することで行全体を一つのデータとして，一つの領域から挿入します。挿入するデータのデータ型は，各列のデータ型に関係なく，ROW 型（ROW 型に対しては，CHAR（n）〔n は行長〕に対応する変数，又は同じ長さの構造体を指定できます。ただし，構造体中に境界調整による空きがあってはいけません）にしてください。また，データ長は行長（各列のデータ長の総和）にしてください。
2. UAP が動作するプラットフォームと HiRDB サーバが動作するプラットフォームのエンディアンを同じにしてください。異なるエンディアン間では，ROW は使用できません。例えば，Windows の UAP で ROW を使用する場合は，HiRDB サーバも同じエンディアンの Windows 版を使用してください。

(c) <形式 3 及び形式 4 で共通のオペランド>

FOR ：埋込み変数

埋込み変数配列を使用して，挿入する行数を設定した埋込み変数を指定します。SMALLINT 型の埋込み変数を指定してください。設定値は，1～4,096 の範囲で，かつ埋込み変数配列，及び標識変数配列の要素数以下にしてください。0 及び負の値は設定できません。範囲外の値を設定した場合は実行時にエラーとなります。

埋込み変数配列

配列形式で宣言した埋込み変数を指定します。NULL 値以外の値を挿入するための配列変数を指定してください。

それぞれの行に挿入する値を，埋込み変数配列の各要素に設定してください。挿入する値に NULL 値を含む場合は，埋込み変数配列と標識変数配列を両方指定します。

標識変数配列

配列形式で宣言した標識変数を指定します。埋込み変数配列の各要素に挿入する値が NULL 値でないことを示す値を，標識変数配列の対応する要素に設定してください。設定する値については，「標識変数の値の設定」を参照してください。FIX 属性の実表には，NULL 値を挿入できない点に注意してください。

(6) 共通規則

(a) 形式 3 の共通規則

1. 埋込み変数配列のデータ型は，対応する列のデータ型，又は変換できるデータ型にしてください。
2. 挿入する 1 行分の列の個数は，列名で指定した列の個数と同じにしてください。また，それらの値は，列のデータ型，又は変換できるデータ型にしてください。
3. 配列を使用した INSERT 文では，BLOB 型，最大長が 32,001 バイト以上の BINARY 型，及び抽象データ型は扱えません。
4. 配列を使用した INSERT 文では，繰返し列への挿入はできません。
5. 固定小数点，又は浮動小数点のデータを，次に示すデータ型の列に挿入する場合，端数（小数）部分は切り捨てられます。
   - INTEGER
   - SMALLINT

   また，固定小数点のデータを DECIMAL 型の列に挿入する場合には，列の位取りより下位のけた部分が切り捨てられます。
6. 表を定義したときに指定した長さ以上の文字データは，挿入できません。
7. 列に定義されているデータ型の範囲外の数データは，挿入できません。
8. 固定長文字列（各国文字列，及び混在文字列を含む）の列に挿入するデータが，列の長さより短い場合は，左詰めに挿入され，余りの部分に空白が設定されます。

9. 文字データ型の列にデータを挿入する場合，挿入対象の列と，対応する挿入値の文字集合を同じにしてください。ただし，挿入値が埋込み変数配列（既定文字集合）の場合，自動的に挿入対象の列の文字集合に変換します。

(b)　形式 4 の共通規則

1. 挿入対象の表に，既定文字集合以外の文字データ型の列が存在する場合，行単位のデータ更新はできません。

(c)　形式 3 及び形式 4 の共通規則

1. FOR 句以外に埋込み変数配列を一つ以上指定してください。指定しない場合はエラーとなります。
2. FOR 句以外に配列形式でない埋込み変数を指定するとエラーとなります。
3. 埋込み変数配列，及び標識変数配列の要素数は，1～4,096 の範囲にしてください。範囲外の値を指定した場合はエラーとなります。また，”FOR ：埋込み変数”で指定する行数の最大値以上になるようにしてください。
4. INSERT 文形式 3 は，埋込み変数配列，及び標識変数配列を含むため，PREPARE 文で前処理できません。動的に実行する場合については，「EXECUTE 文　形式 2（配列を使用した SQL の実行）」を参照してください。
5. 配列を使用した INSERT 文は，手続きの中では使用できません。
6. 挿入するどれかの行で警告する必要のある事象が発生した場合，SQL 連絡領域の SQLWARN に警告情報が設定されます。
7. 挿入するどれかの行でエラーが発生した場合，ロールバックされます。
8. WITHOUT ROLLBACK を指定した表に対して，INSERT 文を実行する場合，インデクスの定義有無によって行排他の解除タイミングが異なります。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
9. 共用表にデータを挿入する場合，事前にその表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表にデータ挿入した場合，エラーとなります。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。

(7)　参照制約に関する規則

1. 埋込み変数配列で挿入する場合，又は行単位で挿入する場合の参照制約に関する規則は，「INSERT 文　形式 1（行挿入）」の参照制約に関する規則を参照してください。
2. 参照表に行を挿入する場合，外部キー構成列の値が被参照表の主キー構成列の値に含まれるかどうかを確認するため，被参照表を検索します。このとき，被参照表の検索時のデータ保証レベルは共用モードになります。そのため，参照表への行の挿入時，ほかのトランザクションによって被参照表に対する操作が行われている場合は，そのトランザクションが決着されるまで参照表への行の挿入は待ち状態になります。

(8) 使用例

1. C 言語の配列変数に設定した 50 行分のデータを，在庫表に一括して挿入します。

```
EXEC SQL BEGIN DECLARE SECTION;
  short   XINSERT_NUM;
  long    XSCODE[50];
  short   ISCODE[50];
  char    XSNAME[50][17];
  short   ISNAME[50];
EXEC SQL END DECLARE SECTION;
            :
各変数配列の要素に値を設定
            :
XINSERT_NUM = 50;
EXEC SQL FOR :XINSERT_NUM
    INSERT INTO ZAIKO(SCODE,SNAME)
      VALUES (:XSCODE:ISCODE,:XSNAME:ISNAME);
```

2. C 言語の配列変数に設定した行全体の値を，在庫表（ZAIKO）に 50 行分まとめて挿入します。

```
EXEC SQL BEGIN DECLARE SECTION;
  short   XINSERT_NUM;
  char    XROWS[50][31];
EXEC  SQL  END  DECLARE  SECTION;
              :
各変数配列の要素に値を設定
              :
XINSERT_NUM = 50;
EXEC  SQL  FOR :XINSERT_NUM
    INSERT INTO ZAIKO(ROW) VALUES(:XROWS);
```

4.29　OPEN 文　形式 1（カーソルオープン）

4.29.1　OPEN 文 形式 1 の形式と規則

(1)　機能

カーソルを開きます。DECLARE CURSOR で宣言したカーソル，又は ALLOCATE CURSOR 文で割り当てたカーソルを，検索結果の先頭の行の直前に位置づけて，検索結果を取り出せる状態にします。

形式 1 では，埋込み変数によって，?パラメタに値を与えてカーソルを開きます。

(2)　使用権限

SELECT 権限を持つユーザ

カーソル宣言中や ALLOCATE CURSOR 文に，指定したすべての表に対して SELECT 権限を持っている必要があります。

(3)　形式 1 ＜（埋込み変数によって，?パラメタに値を与えて）カーソルを開く場合＞

```
OPEN ｛カーソル名｜拡張カーソル名｝　〔USING ：埋込み変数〔，：埋込み変数〕…〕
```

(4)　オペランド

(a)　｛カーソル名｜拡張カーソル名｝

カーソル名

開くカーソルの名称を指定します。

拡張カーソル名

開くカーソルを識別する拡張カーソル名を指定します。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(b)　USING ：埋込み変数〔，：埋込み変数〕…

DECLARE CURSOR 中の SELECT 文で指定した埋込み変数を別の埋込み変数に変更したい場合は，変更後の埋込み変数を指定します。

PREPARE 文で前処理した SELECT 文中に指定した?パラメタに値を与える場合は，値を与える埋込み変数を指定します。

DECLARE CURSOR 中の SELECT 文で指定した埋込み変数，又は?パラメタの値は，カーソルを閉じるまでこの SQL 実行時の値が有効です。値を変更したい場合は，一度カーソルを閉じてから，再度開いてください。

USING 句で指定した埋込み変数は指定した順序に，カーソル宣言中の SELECT 文で指定した埋込み変数と置き換えられます。

USING 句で指定した埋込み変数は指定した順序に，前処理した SELECT 文で指定した?パラメタに値を与えます。

(5) 留意事項

1. 一度開いたカーソルを再度開く場合は，いったんカーソルを閉じてから，再度開いてください。
2. 内部的に CLOSE 文を発行してカーソルを閉じる契機については，「CLOSE 文（カーソルクローズ)」の共通規則 1 を参照してください。
3. FETCH 文を実行する場合は，OPEN 文でカーソルを開いてから，そのカーソルに対する FETCH 文を実行してください。
4. USING 句で埋込み変数を指定する場合は，埋込み変数に値を設定してから OPEN 文を実行してください。
5. USING 句に指定した埋込み変数が既定文字集合の文字データ型で，かつ SELECT 文で指定した？パラメタ又は埋込み変数に関連する項目と異なる文字集合の文字データ型の場合，自動的に SELECT 文で指定した？パラメタ又は埋込み変数の文字集合に変換します。
6. OPEN 文実行時はカーソルの位置づけをし，対応する FETCH 文実行時に排他制御をします。
7. カーソル名は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって使用できません。
8. 手続き中では，USING 句は指定できません。
9. 同一の表に対して，複数のホールダブルカーソルを開くことはできません。

(6) 使用例

在庫表（ZAIKO）の検索結果を取り出すために，カーソル（CR1）を開きます。

```
OPEN CR1
```

4.30　OPEN 文　形式 2（カーソルオープン）

4.30.1　OPEN 文 形式 2 の形式と規則

(1)　機能

カーソルを開きます。DECLARE CURSOR で宣言したカーソル，又は ALLOCATE CURSOR 文で割り当てたカーソルを，検索結果の先頭の行の直前に位置づけて，検索結果を取り出せる状態にします。

形式 2 では，SQL 記述領域によって?パラメタに値を与えてカーソルを開きます。

(2)　使用権限

SELECT 権限を持つユーザ

カーソル宣言中や ALLOCATE CURSOR 文に，指定した SELECT 文に含まれるすべての表に対して SELECT 権限を持っている必要があります。

(3)　形式 2 <SQL 記述領域によって?パラメタに値を与えてカーソルを開く場合>

```
OPEN ｛カーソル名｜拡張カーソル名｝ USING DESCRIPTOR 〔：〕SQL記述領域名
                    〔CHARACTER SET 〔：〕文字集合名記述領域名〕
```

(4)　オペランド

(a)　{カーソル名｜拡張カーソル名}

カーソル名

開くカーソルの名称を指定します。

拡張カーソル名

開くカーソルを識別する拡張カーソル名を指定します。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(b)　〔：〕SQL 記述領域名

SQL 記述領域名

PREPARE 文で前処理した，SELECT 文中で指定した?パラメタの値を SQL 記述領域を用いて与えるときに，その入力?パラメタの値を与えるための変数を記述した SQL 記述領域の名称を指定します。

SQL 記述領域の SQLVAR 配列で指定した変数は，指定した順序に，前処理した SELECT 文で指定した?パラメタに値を与えます。

(c) 〔CHARACTER SET〔：〕文字集合名記述領域名〕

文字集合名記述領域名

PREPARE 文で前処理した，SELECT 文中で指定した?パラメタの値を SQL 記述領域を用いて与えるときに，その入力?パラメタの値を与えるための変数の文字集合名を記述した，文字集合名記述領域の名称を指定します。

(5) 留意事項

1. 一度開いたカーソルを再度開く場合は，いったんカーソルを閉じてから，再度開いてください。
2. 内部的に CLOSE 文を発行してカーソルを閉じる契機については，「CLOSE 文（カーソルクローズ）」の共通規則 1 を参照してください。
3. FETCH 文を実行する場合は，OPEN 文でカーソルを開いてから，そのカーソルに対する FETCH 文を実行してください。
4. OPEN 文を実行するまでに，UAP は SQL 記述領域及び文字集合名記述領域に必要な情報を設定してください。SQL 記述領域及び文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。
5. カーソル名は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって使用できません。
6. 同一の表に対して，複数のホールダブルカーソルを開くことはできません。

(6) 使用例

在庫表の検索結果を取り出すために，カーソル（CR2）を開きます。ただし，PREPARE 文で前処理した SELECT 文中で指定した?パラメタの値を与えるための情報（SQLDA）を指定します。

```
OPEN CR2
    USING DESCRIPTOR SQLDA
```

4.31　PREPARE 文（SQL の前処理）

4.31.1　PREPARE 文の形式と規則

(1)　機能

文字列で与えられた SQL を実行するための前処理をして，その SQL に SQL 文識別子，又は拡張文名を付けます。また，OUTPUT，INPUT を指定することで，DESCRIBE〔OUTPUT〕文，DESCRIBE INPUT 文で得られる検索情報，入出力情報を取得することもできます。

(2)　使用権限

なし。

(3)　形式

```
PREPARE　{SQL文識別子｜拡張文名} FROM {’文字列 ‘｜ :埋込み変数}
                        〔WITH　{SQLNAME｜〔ALL〕TYPE}
                        〔,　{SQLNAME｜〔ALL〕TYPE} 〕OPTION〕
                        〔OUTPUT〔:〕SQL記述領域名〔〔:〕列名記述領域名〕
                                                    〔TYPE〔:〕型名記述領域名〕〕
                                  〔CHARACTER_SET　〔:〕文字集合名記述領域名〕〕
                        〔INPUT　〔:〕SQL記述領域名〔〔:〕列名記述領域名〕
                                  〔CHARACTER_SET　〔:〕文字集合名記述領域名〕〕
```

(4)　オペランド

(a)　{SQL 文識別子｜拡張文名}

SQL 文識別子

前処理する SQL を識別するために SQL 文に付けた名称を指定します。

SQL 文識別子については，「名前の指定」を参照してください。

HiRDB の予約語も使用できますが，使用する場合，予約語と同じ名称でも引用符（”）で囲まないでください。ただし，’SELECT’，及び’WITH’は使用できません。

拡張文名

ALLOCATE CURSOR 文でカーソルを割り当てる場合に，前処理する SQL を識別するために SQL 文に付けた拡張文名を指定します。

拡張文名については，「拡張文名」を参照してください。

(b) {'文字列' | :埋込み変数}

文字列

前処理する SQL の文字列を直接文字定数として指定します。

前処理する文字列中に，SQL 先頭子，及び SQL 終了子は指定できません。

前処理する SQL を文字定数として指定する場合で，SQL 中に 1 個のアポストロフィを表すには，2 個のアポストロフィを続けて指定します。

前処理する SQL の文字列の最大長は，2,000,000 バイトです。ただし，埋込み型で，前処理する SQL を直接文字定数で指定した場合は，ホスト言語の文字定数の最大長です。

埋込み変数

可変長文字型の埋込み変数を指定します。

既定文字集合以外の文字集合は指定できません。

(c) 〔WITH {SQLNAME | 〔ALL〕TYPE} 〔, {SQLNAME | 〔ALL〕TYPE}〕 OPTION〕

SQLNAME

DESCRIBE 文，DESCRIBE TYPE 文で列名記述領域名を指定して，検索項目の列名情報，ユーザ定義型の属性名を受け取る場合に指定します。OUTPUT 又は INPUT 句で，列名記述領域名を指定した場合は，SQLNAME を省略できます。

〔ALL〕TYPE

DESCRIBE 文で型名記述領域を指定して，検索項目の型名情報を受け取る場合に指定します。OUTPUT 句で型名記述領域名を指定した場合は，TYPE を省略できます。ALL TYPE は，DESCRIBE TYPE 文でユーザ定義型の定義情報を受け取る場合に指定します。OUTPUT 句で型名記述領域名を指定しても，ALL TYPE は省略できません。

(d) 〔OUTPUT 〔:〕SQL 記述領域名〔〔:〕列名記述領域名〕〔TYPE 〔:〕型名記述領域名〕〕 〔CHARACTER_SET 〔:〕文字集合名記述領域名〕〕

SQL 記述領域名

SQL の検索項目情報（前処理した SQL が SELECT 文の場合），又は出力?パラメタ情報（前処理した SQL が CALL 文の場合）を受け取る SQL 記述領域の名称を指定します。

SQL 記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

列名記述領域名

検索項目の名称，又はルーチンのパラメタ名を受け取る列名記述領域を指定します。

列名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

型名記述領域名

検索項目のユーザ定義型名を受け取る型名記述領域を指定します。

型名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

文字集合名記述領域名

検索項目情報（前処理した SQL が SELECT 文の場合），又は出力?パラメタ情報（前処理した SQL が CALL 文の場合）の文字集合名を受け取る文字集合名記述領域を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(e) 〔INPUT 〔:〕SQL 記述領域名〔〔:〕列名記述領域名〕〕〔CHARACTER_SET 〔:〕文字集合名記述領域名〕〕

SQL 記述領域名

入力?パラメタ情報を受け取る SQL 記述領域の名称を指定します。

SQL 記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

列名記述領域名

検索項目の名称，又はルーチンのパラメタ名を受け取る列名記述領域を指定します。

列名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

文字集合名記述領域名

入力?パラメタ情報の文字集合名を受け取る文字集合名記述領域を指定します。

文字集合名記述領域については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 共通規則

1. 埋込み変数の型は，次に示す構造体です。

```
struct {
    long  xxxxxxx;     /* SQL文の有効長    */
    char  yyyyyyy[n];  /* SQL文格納エリア */
} zzzzzzz;
```

（凡例）

xxxxxxx は，文字配列 yyyyyyy 中に格納した文字列の有効長を示します。

1 ≦ （xxxxxxx の値） ≦ 2000000

なお，文字列の有効長に文字列の終わりを示す 0（ゼロ）は含みません。

n は，任意です。

2. SQLNAME は重複して指定できません。また，〔ALL〕TYPE も重複して指定できません。
3. UAP は，PREPARE 文を実行する前に，SQL 記述領域に SQLVAR の数（SQLN 領域）を設定してください。
4. SQLDATA と SQLIND は DESCRIBE 文実行時，又は INPUT，OUTPUT を指定した PREPARE 文実行時にクリアされるので，DESCRIBE 文を使用する，又は INPUT，OUTPUT を指定して PREPARE 文を使用する場合は，その実行後に値を設定してください。
5. 列名記述領域名は，検索項目の名称，又はルーチンのパラメタ名を受け取る場合だけ指定してください。ただし，ルーチンのパラメタ名は，CALL 文の引数に単独で？パラメタを指定した場合にだけ受け

取ることができます。？パラメタを含む値式を指定した場合は，列名記述領域の名称の長さが 0 になります。

6. 型名記述領域名は，検索項目のユーザ定義型名を受け取る場合だけ指定してください。

7. 文字集合名記述領域名は，文字集合名を受け取る場合だけ指定してください。

(6) 留意事項

1. 前処理した結果は，そのトランザクション内だけで有効です。したがって，前処理した SQL に対する DESCRIBE 文，EXECUTE 文，OPEN 文，FETCH 文及び CLOSE 文は同じトランザクション内で実行してください。ただし，前処理した SQL がホールダブルカーソルの場合は次のようになります。

   - 該当するトランザクションで前処理し，コミットした場合
     前処理した結果は，DISCONNECT 文を実行するまで有効です。
   - 該当するトランザクションで前処理し，ロールバックした場合
     該当するトランザクション内でだけ有効です。

2. PREPARE 文で前処理できる SQL は，PREPARE 文で前処理しておく必要があります。PREPARE 文で前処理できる SQL を次に示します。

   - 操作系 SQL
     ASSIGN LIST 文（EXECUTE 文で実行します）
     CALL 文（EXECUTE 文で実行します）
     DELETE 文（EXECUTE 文で実行します）
     準備可能動的 DELETE 文：位置付け（EXECUTE 文で実行します）
     DROP LIST 文（EXECUTE 文で実行します）
     INSERT 文（EXECUTE 文で実行します）
     PURGE TABLE 文（EXECUTE 文で実行します）
     1 行 SELECT 文（EXECUTE 文で実行します）
     動的 SELECT 文（OPEN 文，FETCH 文，及び CLOSE 文で実行します）
     UPDATE 文（EXECUTE 文で実行します）
     準備可能動的 UPDATE 文：位置付け（EXECUTE 文で実行します）
     代入文（EXECUTE 文で実行します）
   - 制御系 SQL
     LOCK TABLE 文（EXECUTE 文で実行します）
     SET SESSION AUTHORIZATION 文（EXECUTE 文で実行します）
   - 定義系 SQL
     ALTER INDEX，ALTER PROCEDURE，ALTER ROUTINE，ALTER TABLE，ALTER TRIGGER，COMMENT，CREATE AUDIT，CREATE CONNECTION SECURITY，CREATE FUNCTION，CREATE INDEX，CREATE PROCEDURE，CREATE SCHEMA，CREATE SEQUENCE，CREATE TABLE，CREATE TRIGGER，CREATE TYPE，CREATE

VIEW，DROP AUDIT，DROP CONNECTION SECURITY，DROP DATA TYPE，DROP FUNCTION，DROP INDEX，DROP PROCEDURE，DROP SCHEMA，DROP SEQUENCE，DROP TABLE，DROP TRIGGER，DROP VIEW，GRANT，REVOKE
（定義系 SQL は，すべて EXECUTE 文で実行します）

3. SQL 文識別子は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じ SQL 文識別子に対する複数の SQL を，複数のモジュールにわたって使用できません。
4. PREPARE 文で前処理した動的 SELECT 文の実行中（OPEN 文を実行してから CLOSE 文を実行するまでの間）に，その動的 SELECT 文の FROM 句に指定した表を別の SQL 文で更新しないでください。
5. 指定した SQL 文識別子，又は拡張文名が既にほかの SQL 文を識別している場合は，暗黙的に DEALLOCATE PREPARE 文が実行され，前に識別していた SQL 文は無効になります。その後，指定した SQL 文識別子，又は拡張文名はこの PREPARE 文で前処理した SQL 文を識別します。ただし，暗黙的に実行された DEALLOCATE PREPARE 文でエラーが発生した場合は，前に識別していた SQL 文を識別したままとなります。
6. PREPARE 文で OUTPUT を指定した場合，DESCRIBE〔OUTPUT〕を実行した場合と同様に扱われます。また，PREPARE 文で INPUT を指定した場合，DESCRIBE〔INPUT〕を実行した場合と同様に扱われます。OUTPUT，INPUT については，「DESCRIBE 文　形式 1（検索情報，入出力情報の受け取り）」及び「DESCRIBE 文　形式 2（検索情報，入出力情報の受け取り）」を参照してください。
7. ユーザ定義型の所有者名の長さが 9 文字以上の場合，型名記述領域を使用できません。

(7)　使用例

1. 文字列で与えられた SQL’SELECT * FROM ZAIKO’を実行するために前処理します。前処理後の SQL 文字列に付ける SQL 識別子の名称は,’PRESQL’とします。

```
PREPARE PRESQL FROM
     ‘SELECT * FROM ZAIKO’
```

2. 埋込み変数（XSQL）中に指定した SQL 文字列を前処理します。前処理後の SQL 文字列に付ける SQL 識別子の名称は,’PRESQL’とします。

```
PREPARE PRESQL FROM :XSQL
```

4.32　PURGE TABLE 文（全行削除）

4.32.1　PURGE TABLE 文の形式と規則

(1)　機能

実表中の行をすべて削除します。

(2)　使用権限

表に対する DELETE 権限を持つユーザが，その表の行を削除できます。

(3)　形式

```
{PURGE | TRUNCATE}  TABLE〔認可識別子.〕表識別子
              〔IN （RDエリア名指定）〕
              〔 {WITH ROLLBACK | NO WAIT} 〕
```

(4)　オペランド

(a)　〔認可識別子.〕表識別子

認可識別子

表を所有するユーザの認可識別子を指定します。

認可識別子に"MASTER"は指定できません。認可識別子を省略した場合については，「名前の修飾」を参照してください。

表識別子

行をすべて削除する実表の名称を指定します。

(b)　〔IN （RD エリア名指定)〕

IN

アクセス対象の RD エリアを指定します。なお，表識別子に指定した表が一時表の場合は指定できません。

RD エリア名指定：：＝定数

表識別子に指定した表を格納している RD エリアのうち，アクセスする RD エリアの名称を VARCHAR 型，CHAR 型，MVARCHAR 型，又は MCHAR 型の定数で指定します。複数の RD エリア名を指定する場合はコンマ（,）で区切って指定してください。RD エリア名は重複して指定できません。重複して指定した場合はエラーとなります。定数で指定する RD エリア名に許される文字については，「名

前の指定」を参照してください。また，定数で指定した RD エリア名の前後の空白は無視されます。RD エリア名を引用符（"）で囲んだ場合は，引用符（"）の外側の空白だけを無視します。

また，表識別子に指定した表が分割されている場合，定義しているプライマリーキー，クラスタキー，又はインデクスは表の分割数に合わせてください。

インナレプリカ機能を適用している RD エリアを指定する場合，オリジナル RD エリア名を指定してください。レプリカ RD エリアを対象とする場合は，カレント切り替えコマンド（pddbchg コマンド），又はクライアント環境定義の PDDBACCS オペランドでアクセス対象 RD エリアをレプリカ RD エリアに切り替えてください。

(c) 〔{ WITH ROLLBACK | NO WAIT }〕

WITH ROLLBACK，又は NO WAIT を省略した場合は，行を削除する表がほかのユーザのトランザクションによって使用されていたら，そのトランザクションが終了してから実行します。

システム定義の pd_check_pending オペランドに USE を指定した場合，行を削除する表を被参照表とする参照表がほかのユーザのトランザクションによって使用されていたときは，トランザクションが終了してから参照表を検査保留状態に設定します。

WITH ROLLBACK

行を削除する表がほかのユーザで使用されていたら，トランザクションを取り消して無効にします。

システム定義の pd_check_pending オペランドに USE を指定した場合，行をすべて削除する表を被参照表とする参照表がほかのユーザのトランザクションに使用されているときは，トランザクションを取り消して無効にします。

NO WAIT

行を削除する表がほかのユーザで使用されていたら，トランザクションを取り消さないで，この SQL を無効にします。

システム定義の pd_check_pending オペランドに USE を指定した場合，行をすべて削除する表を被参照表とする参照表がほかのユーザのトランザクションによって使用されているときは，トランザクションを取り消さないで，この SQL を無効にします。

(5) 共通規則

1. PURGE TABLE 文が正常に実行された場合は，処理完了と同時に COMMIT されます。
2. LOB 列又は LOB 属性を格納するユーザ LOB 用 RD エリアが更新凍結状態の場合，その LOB 列又は LOB 属性がある表に対して PURGE TABLE 文は実行できません（実行すると凍結済みエラーとなります）。
3. 改竄防止表に対して PURGE TABLE 文は実行できません。
4. 共用表に対して PURGE TABLE 文を実行した場合，共用表の EXCLUSIVE 指定の LOCK 文相当の排他が掛かります。HiRDB/パラレルサーバの場合，全バックエンドサーバで掛かります。

(6) 検査保留状態に関する規則

1. システム定義の pd_check_pending オペランドに USE を指定した場合，操作対象表が被参照表のときは，操作対象表を参照する参照表を検査保留状態に設定します。
2. 操作対象表が検査保留状態である場合，操作対象表の検査保留状態を解除します。ただし，次の条件のどれかを満たす場合，ディクショナリ表中の検査保留状態は解除されません。そのため，整合性チェックユティリティを使用し，ディクショナリ表中の検査保留状態を解除してください。
   - システム定義の pd_check_pending オペランドに NOUSE を指定している。
   - インナレプリカ機能を使用している。
   - RD エリア名指定の PURGE TABLE 文を実行している。

(7) 留意事項

1. PURGE TABLE 文はビュー表に対して指定できません。
2. PURGE TABLE 文は，OLTP 下の X/Open に従った UAP から指定できません。また，OLTP 下の UAP から手続きを呼び出す場合，PURGE TABLE 文を使用した手続きは実行できません。
3. PURGE TABLE 文は，トリガの動作中は実行できません。
4. システム定義の pd_check_pending オペランドに USE を指定するか，又は指定を省略した場合，コマンド又はユティリティと PURGE TABLE 文を同時に実行できないことがあります。詳細については，マニュアル「HiRDB システム定義」の pd_check_pending オペランドの説明を参照してください。
5. RD エリア名指定の PURGE TABLE 文は抽象データ型を定義した表に対して指定できません。
6. RD エリア名指定の PURGE TABLE 文で，指定した RD エリアの数が，表に定義した分割数と同じ場合，RD エリア名指定のない PURGE TABLE 文として動作します。

(8) 使用例

在庫表（ZAIKO）のすべての行を削除します。

```
PURGE TABLE ZAIKO
TRUNCATE TABLE ZAIKO
```

4.33　1 行 SELECT 文（1 行検索）

4.33.1　1 行 SELECT 文の形式と規則

(1)　機能

表のデータを検索します。

表から 1 行だけデータを取り出す場合は，カーソルを使用しないでデータを取り出す 1 行 SELECT 文を指定します。

1 行 SELECT 文は，問合せ指定の SELECT 句とオペランドは同じですが，問合せ指定での SELECT 句のように集合に対する操作をする文ではありません。

また，1 行 SELECT 文は検索した結果を受け取る領域についての指定をする INTO 句を含んでいます。

(2)　使用権限

問合せ指定の権限と同じです。

(3)　形式<1 行以下のデータを，指定された埋込み変数に取り出し>

```
SELECT〔 {ALL｜DISTINCT} 〕 {選択式〔, 選択式〕…｜＊}
   〔INTO {：埋込み変数〔：標識変数〕
            |〔文ラベル.〕SQL変数名
            |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
            |〔文ラベル.〕SQL変数名..属性名〔..属性名〕…
            |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
                  ..属性名〔..属性名〕…}
        〔,  {：埋込み変数〔：標識変数〕
              |〔文ラベル.〕SQL変数名
              |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
              |〔文ラベル.〕SQL変数名..属性名〔..属性名〕…
              |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
                    ..属性名〔..属性名〕…} 〕…〕
   〈表式〉
      FROM  表参照〔, 表参照〕…
            〔WHERE 探索条件〕
            〔GROUP BY 値式〔, 値式〕…〕
            〔HAVING 探索条件〕
   〈排他オプション〉
     〔〔 {WITH {SHARE｜EXCLUSIVE} LOCK
        ｜WITHOUT LOCK〔 {WAIT｜NOWAIT} 〕} 〕〕
     〔 {WITH ROLLBACK｜NO WAIT} 〕
        〔FOR UPDATE 〔OF 列名〔, 列名〕…〕〔NOWAIT〕〕
```

(4) オペランド

SELECT 句については「問合せ指定」を，表式については「表式」を，排他オプションについては「排他オプション」をそれぞれ参照してください。

(a) INTO 句

INTO 句は，SELECT 文を単独で UAP，又は手続き中に直接記述する場合は，必ず指定してください。

ただし，次の箇所には指定できません。

- PREPARE 文で前処理する SQL 中の SELECT 文
- EXECUTE IMMEDIATE 文で前処理・実行する SQL 中の SELECT 文
- カーソル宣言中の SELECT 文

(b) ：埋込み変数〔：標識変数〕

埋込み変数

一つの行の列の値を読み込む埋込み変数を指定します。

標識変数

埋込み変数に読み込まれる列の値がナル値の可能性がある場合に指定します。

(c) 〔文ラベル.〕SQL 変数名

SQL 手続き中で列の値を受け取るために，SQL 変数を指定します。Java 手続き中の指定値については，マニュアル「HiRDB UAP 開発ガイド」の JDBC ドライバ又は SQLJ を参照してください。

(d) 〔〔認可識別子.〕ルーチン識別子.〕SQL パラメタ名

SQL 手続き中で列の値を受け取るために，SQL パラメタを指定します。Java 手続き中の指定値については，マニュアル「HiRDB UAP 開発ガイド」の JDBC ドライバ又は SQLJ を参照してください。

パブリック手続き定義の SQL 手続き文中で認可識別子を指定する場合は，認可識別子に PUBLIC を指定してください。

(e) 〔文ラベル.〕SQL 変数名..属性名〔..属性名〕…

SQL 手続き中で，列中の属性の値を受け取るために指定します。

(f) 〔〔認可識別子.〕ルーチン識別子.〕SQL パラメタ名..属性名〔..属性名〕…

SQL 手続き中で，列中の属性の値を受け取るために指定します。

パブリック手続き定義の SQL 手続き文中で認可識別子を指定する場合は，認可識別子に PUBLIC を指定してください。

(g) 〔FOR UPDATE 〔OF 列名〔，列名〕…〕〔NOWAIT〕〕

FOR UPDATE〔OF 列名〔，列名〕…〕を FOR UPDATE 句といいます。

FOR UPDATE 〔OF 列名〔，列名〕…〕

1 行 SELECT 文で取り出したデータを使用して，行の更新，削除，又は追加をする場合に指定します。1 行 SELECT 文で取り出したデータを使用した行の更新，削除，又は追加もしない場合は，オペランドを省略してください。SQL 中の排他オプションを省略した場合，排他オプションは pd_isolation_level の指定値，PDISLLVL の指定値，又は SQL コンパイルオプションで指定するデータ保証レベルの指定値によって決まりますが，PDFORUPDATEEXLOCK に YES，又は SQL コンパイルオプションのデータ保証レベルの後に FOR UPDATE EXCLUSIVE を指定することで，この指定のあるカーソルに対する排他オプションは WITH EXCLUSIVE LOCK が仮定されます。詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

〔OF 列名〔，列名〕…〕

一般的には指定する必要がありません。

1 行 SELECT 文の選択式で指定していない列でも指定できます。ただし，同じ列を 2 回以上指定できません。また，予備列は指定できません。

指定する列名は，AS 列名に指定した列名ではなく，最も外側の問合せ指定の FROM 句に指定した表の列を指定します。

〔NOWAIT〕

FOR UPDATE 句を指定し，かつ排他オプションに WITH EXCLUSIVE LOCK NO WAIT を指定した場合と同じ動作をします。ただし，排他オプションを指定した場合，NOWAIT は指定できません。

排他オプションに WITH EXCLUSIVE LOCK NO WAIT を指定した場合の動作については，「排他オプション」を参照してください。

(5) 共通規則

1. 検索結果が 1 行以下の場合は，カーソルを使用しないで INTO 句を指定して検索できます。ただし，検索結果が 2 行以上の場合，検索できません。
   - 検索結果の列の個数と，INTO 句で指定した埋込み変数の個数は同じにしてください。列の個数と，埋込み変数の個数が同じでない場合は，SQL 連絡領域の SQLWARN3 領域に警告フラグ”W”が設定されます。
   - INTO 句で指定する埋込み変数のデータ型は，対応する列のデータ型，又は変換できるデータ型にしてください。
   - 埋込み変数が既定文字集合の文字データ型で，かつ検索結果がその埋込み変数と異なる文字集合の文字データ型の場合，自動的に埋込み変数の文字集合に変換します。
   - 固定長文字列（各国文字列，及び混在文字列を含む）の埋込み変数に取り出すデータが，検索項目の長さより短い場合は，左詰めに挿入され，余りの部分に空白が設定されます。
   - 検索結果の列の値がナル値の場合，対応する埋込み変数の値は保証しません。
   - 検索結果の列の値がナル値の場合は，対応する標識変数を指定してください。

2. 取り出す行がない場合，次のリターンコードが設定されます。
   - SQL 連絡領域の SQLCODE 領域に 100
   - SQLCODE 変数に 100
   - SQLSTATE 変数に’02000’
3. 「カーソル指定　形式 1」，及び「問合せ指定」の規則に従って，UNION〔ALL〕，EXCEPT〔ALL〕，ORDER BY 句，及び LIMIT 句を指定できます。

   UNION〔ALL〕，及び EXCEPT〔ALL〕を指定した場合の INTO 句は，最初の SELECT 句の次に一度だけ指定してください。
4. 1 行 SELECT 文又は排他オプション中に次のどれかの指定を含む場合は，FOR UPDATE 句の指定はできません。

   (5-a) UNION〔ALL〕，又は EXCEPT〔ALL〕

   (5-b) 最も外側の問合せ指定の FROM 句に指定した表を，副問合せの FROM 句に指定

   (5-c) 最も外側の問合せ指定での表の結合

   (5-d) 最も外側の問合せ指定での FROM 句の導出表

   (5-e) 最も外側の問合せ指定での SELECT DISTINCT

   (5-f) 最も外側の問合せ指定での GROUP BY 句

   (5-g) 最も外側の問合せ指定での HAVING 句

   (5-h) 最も外側の問合せ指定に対する集合関数

   (5-i) 最も外側の問合せ指定でのウィンドウ関数

   (5-j) 最も外側の問合せ指定の FROM 句中で，次に示すどれかのビュー表を指定
   - ビュー定義文で上記(5-a)～(5-i)を指定して定義したビュー表
   - ビュー定義文で最も外側の問合せ指定の SELECT 句に，列指定以外の値式を指定して定義したビュー表
   - ビュー定義文中で，READ ONLY を指定して定義したビュー表

   (5-k) WITHOUT LOCK NOWAIT

(6)　使用例

使用例については，「DECLARE CURSOR　形式 1（カーソル宣言）」を参照してください。

4.34　動的 SELECT 文　形式 1（動的検索）

4.34.1　動的 SELECT 文 形式 1 の形式と規則

(1)　機能

次に示す検索ができます。

- 一つ以上の表からデータを検索します。
- 動的に SELECT 文を実行する場合に指定します。
- 動的 SELECT 文は，PREPARE 文で前処理します。検索するときは，DECLARE CURSOR によってカーソルを宣言してから，そのカーソルを使用して検索結果を 1 行ずつ取り出します。

(2)　使用権限

「カーソル指定　形式 1」を参照してください。

(3)　形式

```
〈カーソル指定　形式1〉
  〈問合せ式〉
    〈問合せ指定〉
      {SELECT〔 {ALL｜DISTINCT} 〕 {選択式〔, 選択式〕…｜*}
      〈表式〉
     FROM 表参照〔, 表参照〕…
              〔WHERE 探索条件〕
              〔GROUP BY 値式〔, 値式〕…〕
              〔HAVING 探索条件〕
       ｜ 問合せ式 }
      〔ORDER BY {列指定｜ソート項目指定番号} 〔 {ASC｜DESC} 〕
            〔, {列指定｜ソート項目指定番号} 〔 {ASC｜DESC} 〕〕… 〕
      〔LIMIT { 〔オフセット行数,〕 {リミット行数｜ALL}
              ｜ {リミット行数｜ALL} 〔OFFSET オフセット行数〕} 〕
〈排他オプション〉
  〔〔 {WITH {SHARE｜EXCLUSIVE} LOCK
       ｜WITHOUT LOCK 〔 {WAIT｜NOWAIT} 〕} 〕
  〔 {WITH ROLLBACK｜NO WAIT} 〕〕
       〔FOR {UPDATE 〔OF列名〔, 列名〕…〕 〔NOWAIT〕 ｜READ ONLY} 〕
       〔UNTIL DISCONNECT〕
```

(4)　オペランド

次の項目については，それぞれの箇所を参照してください。

- カーソル指定　形式 1：「カーソル指定　形式 1」
- 問合せ式：「問合せ式」

- 問合せ指定：「問合せ指定」
- 表式：「表式」
- 探索条件：「探索条件」
- 排他オプション：「排他オプション」

(a) FOR {UPDATE〔OF 列名〔，列名〕…〕〔NOWAIT〕 | READ ONLY}

FOR UPDATE 〔OF 列名〔，列名〕…〕を FOR UPDATE 句といいます。

FOR UPDATE

カーソルを使用して検索中の表に対して，そのカーソルを使用した行の更新，又は削除をして，更にほかのカーソル，又は直接探索条件を指定して行を更新，追加，又は削除する場合に指定します。

FOR UPDATE を省略した場合，そのカーソルを使用して検索中の表に対して行の更新，追加，又は削除はできません。

カーソル指定，又は排他オプション中に次のどれかの指定を含む場合は，FOR UPDATE 句の指定はできません。

(4-a) UNION〔ALL〕，又は EXCEPT〔ALL〕

(4-b) 最も外側の問合せ指定の FROM 句に指定した表を，副問合せの FROM 句に指定

(4-c) 最も外側の問合せ指定での表の結合

(4-d) 最も外側の問合せ指定での FROM 句の導出表

(4-e) 最も外側の問合せ指定での SELECT DISTINCT

(4-f) 最も外側の問合せ指定での GROUP BY 句

(4-g) 最も外側の問合せ指定での HAVING 句

(4-h) 最も外側の問合せ指定に対する集合関数

(4-i) 最も外側の問合せ指定に対するウィンドウ関数

(4-j) 最も外側の問合せ指定の FROM 句中で，次に示すどれかのビュー表を指定

- ビュー定義文で上記(4-a)～(4-i)を指定して定義したビュー表
- ビュー定義文で最も外側の問合せ指定の SELECT 句に，列指定以外の値式を指定して定義したビュー表
- ビュー定義文中で，READ ONLY を指定して定義したビュー表

(4-k) WITHOUT LOCK NOWAIT

(4-l) WITH 句を指定した問合せ式本体の最も外側の問合せ指定の FROM 句に問合せ名を指定

OF 列名〔，列名〕…

カーソルを使用して検索中の表に対して，そのカーソルを使用した検索行の更新だけをする場合に，更新する列を指定します。

SELECT 文の選択式で指定していない列でも指定できます。ただし，同じ列を 2 回以上指定できません。また，予備列は指定できません。

カーソルを使用して検索中の表に対して，そのカーソル及びほかのカーソルを使用した行の更新，削除がなく，カーソルを使用しない行の更新，削除，及び追加もしない場合には，オペランドを省略してください。

指定する列名は，AS 列名に指定した列名ではなく，最も外側の問合せ指定の FROM 句に指定した表の列を指定します。

〔NOWAIT〕

排他オプションに WITH EXCLUSIVE LOCK NO WAIT を指定した場合と同じ動作をします。ただし，排他オプションを指定した場合，FOR UPDATE 句に NOWAIT は指定できません。

排他オプションに WITH EXCLUSIVE LOCK NO WAIT を指定した場合の動作については，「排他オプション」を参照してください。

FOR READ ONLY

カーソルを使用して検索中に，ほかのカーソル，又は直接探索条件を指定して更新する場合に指定します。検索中の更新が検索結果に影響を与えないようにする場合に FOR READ ONLY を指定します。

FOR READ ONLY 句を指定した場合，次の制限があります。

(4-a) 選択式中に，結果が次のデータ型になるスカラ演算，関数呼出し，及びコンポネント指定は指定できません。

- BLOB
- 最大長が 32,001 バイト以上の BINARY
- BOOLEAN
- 抽象データ型

(4-b) 選択式中の WRITE 指定の出力 BLOB 値には，列指定だけ指定できます。

(4-c) GET_JAVA_STORED_ROUTINE_SOURCE 指定は指定できません。

(b) 〔UNTIL DISCONNECT〕

ホールダブルカーソルを使用する場合に指定します。ホールダブルカーソルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

ホールダブルカーソルについての規則を次に示します。

1. 次の場合，ホールダブルカーソルは使用できません。
   - ホールダブルカーソルに未対応のプラグイン※を使用した抽象データ型の列を指定した場合
   - ホールダブルカーソルに未対応のプラグイン※を使用した関数呼出しを指定した場合
   - ホールダブルカーソルに未対応のプラグイン※を使用した関数呼出しを指定して導出した，名前付きの導出表に対する問合せを指定した場合
   - ON COMMIT DELETE ROWS を指定して定義した一時表に対する問合せ

2. ホールダブルカーソルが開いている場合，定義系 SQL は実行できません。また，ホールダブルカーソルが閉じている場合，定義系 SQL を実行するとホールダブルカーソルを使用している前処理は無効になります。
3. ホールダブルカーソルを使用した SELECT 文に対して OPEN 文を実行後，その SELECT 文中で使用している表に対して PURGE TABLE 文を実行すると，カーソルは閉じた状態になります。
4. ホールダブルカーソルを使用した SELECT 文に対して OPEN 文を実行してから DISCONNECT するまでの間に，その SELECT 文中で使用している表に対して，ほかのユーザが定義系 SQL 文を発行すると，定義系 SQL は排他待ちの状態になります。また，ホールダブルカーソルを使用した SELECT 文に対する前処理が有効な間に，その SELECT 文中で使用している表に対して，ほかのユーザが定義系 SQL 文を発行すると，定義系 SQL は排他待ちの状態になります。

注※

プラグインのホールダブルカーソルへの対応状況については，ディクショナリ表 SQL_PLUGINS の PLUGIN_HOLDABLE 列で確認できます。ディクショナリ表 SQL_PLUGINS については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(5) 参照制約に関する規則

1. ホールダブルカーソルを使用して，外部キーの定義されている表を検索している場合，検索中の表が検査保留状態となったときには，カーソルは閉じた状態になります。

(6) 留意事項

1. FOR UPDATE を指定すると作業表が作成されるのでカーソルを使用して検索中の表に対して，行の更新，追加，又は削除をしない場合は，このオペランドを省略してください。
2. SQL 最適化オプションの更新 SQL の作業表作成抑止を適用して，更にインデクスキー値無排他機能を使用すると，FOR UPDATE 又は FOR UPDATE OF を指定しないカーソルの使用中に，行の更新，追加，又は削除ができます。

(7) 使用例

使用例については，「DECLARE CURSOR　形式 1（カーソル宣言）」，及び「ALLOCATE CURSOR 文 形式 1（文カーソル割当て）」を参照してください。

4.35　動的 SELECT 文　形式 2（動的検索）

4.35.1　動的 SELECT 文 形式 2 の形式と規則

(1)　機能

リストを介して表を検索します。

(2)　使用権限

自分の所有するリストを介して表を検索できます。

(3)　形式 2 ＜リストを介した表のデータの検索＞

```
 〈カーソル指定　形式2〉
SELECT　{ {値式｜WRITE指定｜GET_JAVA_STORED_ROUTINE_SOURCE指定}
                  〔〔AS〕 列名〕
           〔,　{値式｜WRITE指定｜GET_JAVA_STORED_ROUTINE_SOURCE指定}
                     〔〔AS〕 列名〕〕…
            ｜＊}
    FROM LIST リスト名
 〈排他オプション〉
  〔〔 {WITH {SHARE｜EXCLUSIVE} LOCK
       ｜WITHOUT LOCK〔 {WAIT｜NOWAIT} 〕} 〕
  〔 {WITH ROLLBACK｜NO WAIT} 〕
```

(4)　オペランド

カーソル指定については「カーソル指定　形式 2」を，排他オプションについては「排他オプション」を参照してください。

(a)　〔排他オプション〕

リストを介して検索する場合，検索する実表に対する排他制御について指定します。

(5)　共通規則

1. リスト作成時にあった行が検索時にない場合には，SQL コード+110 を返します。この場合，検索は続行します。
2. 同一ユーザが，複数同時に HiRDB と接続してリストを操作できません。

(6)　留意事項

1. リストを作成後，基表の行を削除した場合，検索時にその行は検索されません。

2. リストを作成後，基表の行を更新した場合，更新後のデータが検索されます。
3. リストを作成後，基表に対して，行を削除後に行を挿入した場合，挿入した行が検索されることがあります。
4. リストを介した表の検索をする SQL を PREPARE 文で前処理した後，OPEN 文を実行するまでの間に，同一リスト名の ASSIGN LIST 文を実行しないようにしてください。

4.36　UPDATE 文　形式 1（データ更新）

4.36.1　UPDATE 文 形式 1 の形式と規則

(1)　機能

表内の，指定した探索条件を満足する行，又はカーソルが指している行の指定した列の値を更新します。

(2)　使用権限

表に対する UPDATE 権限を持つユーザが，その表の行の値を更新できます。

ただし，探索条件中に副問合せを指定する場合は，その副問合せ指定の表に対して SELECT 権限が必要です。

(3)　形式 1 <列単位で表内の行を更新>

```
UPDATE 〔認可識別子.〕表識別子
            〔IN （RDエリア名指定）〕〔〔AS〕 相関名〕
            〔使用インデクスのSQL最適化指定〕
            {SET {更新対象＝更新値
                   |（更新対象，更新対象〔，更新対象〕）＝行副問合せ}
                  〔，{更新対象＝更新値
                         |（更新対象，更新対象〔，更新対象〕…）＝行副問合せ}〕…
             |ADD繰返し列名 ［{添字|＊}］
                        ＝{ARRAY ［要素の値〔，要素の値〕…］
                            |？パラメタ|：埋込み変数〔：標識変数〕}
                       〔，繰返し列名 ［{添字|＊}］
                              ＝{ARRAY ［要素の値〔，要素の値〕…］
                                  |？パラメタ|：埋込み変数〔：標識変数〕}〕…
             |DELETE繰返し列名 ［{添字|＊}］
                          〔，繰返し列名 ［{添字|＊}］〕…}
〔WHERE {探索条件|CURRENT OF {カーソル名|拡張カーソル名}}〕
〔WITH ROLLBACK〕
〔WRITE IMMEDIATE〕

更新対象：：＝{列名|コンポネント指定|列名［{添字|＊}］}
```

(4)　オペランド

(a)　〔認可識別子.〕表識別子

更新する表を指定します。

規則を次に示します。

1. 読み込み専用のビュー表は行の更新ができません。読み込み専用のビュー表については，「CREATE〔PUBLIC〕VIEW（ビュー定義，パブリックビュー定義）」の共通規則を参照してください。
2. ビュー表の列の更新を指定した場合は，そのビュー表の列に対応する基の実表を更新します。
3. 表名の有効範囲は UPDATE 文全体です。
4. SET 句，又は ADD 句に指定した副問合せ中に，更新する表の列を外への参照で指定し，かつ副問合せの選択式に次に示す属性の値式を指定した場合，副問合せの FROM 句の表名と同じ名称は指定できません。
   - BLOB
   - 最大長が 32,001 バイト以上の BINARY
   - 繰返し列
   - 抽象データ型

   ただし，副問合せの FROM 句の表名にビュー表を指定した場合は，そのビュー表定義の導出問合せ式中で指定したすべての表名が対象になります。

認可識別子

表の所有者の認可識別子を指定します。認可識別子に”MASTER”は指定できません。

表識別子

更新する表の名称を指定します。

(b) 〔IN （RD エリア名指定）〕

IN

アクセス対象の RD エリアを指定します。表識別子に指定した表が一時表の場合は指定できません。

RD エリア名指定：：＝値指定

表識別子に指定した表を格納している RD エリアのうち，アクセスする RD エリアの名称を VARCHAR 型，CHAR 型，MVARCHAR 型，又は MCHAR 型の値指定で指定します。複数の RD エリア名を指定する場合はコンマ（,）で区切って指定してください。RD エリア名は重複して指定できません。重複して指定した場合はエラーとなります。値指定で指定する RD エリア名に許される文字については，「名前の指定」を参照してください。また，値指定で指定した RD エリア名の前後の空白は無視されます。RD エリア名を引用符（"）で囲んだ場合は，引用符（"）の外側の空白だけを無視します。

カーソル名又は拡張カーソル名の指定がある場合は，カーソル宣言で宣言したカーソルで指定した RD エリアと同じ RD エリアの集合を指定してください（順不同）。指定しない場合はエラーとなります。

インナレプリカ機能を適用している RD エリアを指定する場合，オリジナル RD エリア名を指定してください。レプリカ RD エリアを対象とする場合は，カレント切り替えコマンド（pddbchg コマンド），又はクライアント環境定義の PDDBACCS オペランドでアクセス対象 RD エリアをレプリカ RD エリアに切り替えてください。

(c) 〔AS〕 相関名

更新対象の表に対して相関名を使用する場合に指定します。

相関名の有効範囲は UPDATE 文全体です。更新対象の表識別子は有効範囲を持ちません。

(d) 使用インデクスの SQL 最適化指定

使用インデクスの SQL 最適化指定については，「SQL 最適化指定」を参照してください。

(e) SET 更新対象＝更新値

列の値，又は抽象データ型の属性の値を更新する場合に指定します。

列名

更新する列の名称を指定します。

コンポネント指定

更新する抽象データ型の属性を指定します。

列名 ［{添字｜＊}］

列名

要素を更新する繰返し列を指定します。

［{添字｜＊}］

添字には，更新する要素の位置を指定します。＊は，要素の最後を更新する場合に指定します。

＊を指定した場合，更新対象となっている繰返し列の要素が 0 のときは，＊の指定は無効となります。

更新値

更新値の列の値として次に示す項目が指定できます。

- 列名
- コンポネント指定
- 定数
- 値式（四則演算，連結演算を含みます）
- スカラ副問合せ
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- NULL（ナル値を表します）
- DEFAULT（更新対象の列の既定値を表します）
- ：埋込み変数［：標識変数］

- ?パラメタ
- SQL 変数，又は SQL パラメタ
- ARRAY ［要素の値［，要素の値］…］※

注※

要素の値には，次の項目を指定できます。

- 繰返し列以外の列名
- 添字付き繰返し列
- 定数
- 値式（四則演算，連結演算を含みます）
- スカラ副問合せ
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- NULL（ナル値を表します）
- DEFAULT
- ：埋込み変数［：標識変数］
- ?パラメタ
- SQL 変数，又は SQL パラメタ

更新値の規則

1. 更新値として列名を指定する場合は，更新対象になる列，又は属性と同じデータ型，又は変換できるデータ型の列名を指定します。
2. 更新値としてスカラ副問合せを指定する場合，スカラ副問合せで得られる列のデータ型は，更新対象の列又は属性と同じデータ型，又は変換できるデータ型にしてください。
3. 更新対象のデータ型が文字データ型の場合，更新対象と更新値の文字集合は同じにしてください。ただし，更新値として埋込み変数（既定文字集合），?パラメタ，又は文字列定数を指定した場合は，自動的に更新対象の文字集合に変換します。
4. 更新対象に添字のない繰返し列を指定する場合，更新値の列名，又はスカラ副問合せの選択式には，添字のない繰返し列の列名を指定してください。
5. 更新対象に添字のある繰返し列を指定する場合，更新値の列名，又はスカラ副問合せの選択式に添字のない繰返し列の列名は指定できません。
6. 埋込み変数，標識変数は，PREPARE 文で前処理する UPDATE 文，又は手続き中には指定できません。手続き中では，SQL 変数，又は SQL パラメタを使用します。

7. 埋込み変数（標識変数），?パラメタ，SQL 変数，又は SQL パラメタを指定する場合，埋込み変数(?パラメタの場合は，それに値を与えるために指定する埋込み変数)，SQL 変数，又は SQL パラメタのデータ型は，更新する列又は属性のデータ型，又は変換できるデータ型にしてください。

   また，更新対象となる列が繰返し列の場合，更新値の埋込み変数（標識変数），？パラメタは繰返し構造にしてください。

8. 標識変数を指定した場合は，標識変数の値が負の場合，埋込み変数の値がナル値と解釈され，対応する列にナル値が設定されます。標識変数の値が 0，又は正の場合，埋込み変数の値が対応する列に設定されます。

9. ?パラメタは，PREPARE 文で前処理をする UPDATE 文の場合だけ指定できます。

   ?パラメタに与える値は，前処理をした PREPARE 文に対応する EXECUTE 文の USING 句の埋込み変数で指定します。

10. 更新値は，更新対象になる列又は属性と変換，比較できるデータ型にしてください。

    ただし，更新対象になる列又は属性が各国文字データ型で，更新値として文字列定数を指定した場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードはチェックされないで，文字データの長さだけがチェックされます。

11. 更新する列が添字のない繰返し列の場合，更新値には，定数，値式，USER 値関数，CURRENT_DATE 値関数，CURRENT_TIME 値関数，CURRENT_TIMESTAMP 値関数，SQL 変数，又は SQL パラメタは指定できません。

12. ARRAY ［要素の値［，要素の値］…］は，更新対象になる列が繰返し列の場合にだけ指定できます。

13. ARRAY ［要素の値［，要素の値］…］の中の要素の値は，最大 30,000 個指定できます。ただし，更新する列の最大要素数以下にしてください。

14. 更新値として DEFAULT を指定する場合の既定値は，次のようになります。

    ・更新対象の列に DEFAULT 句の指定がある場合は，指定した既定値となります。

    ・DEFAULT 句の指定がなく，WITH DEFAULT の指定がある場合は，WITH DEFAULT の既定値となります。

    ・DEFAULT 句，及び WITH DEFAULT のどちらの指定もない場合，NULL が既定値となります。

要素の値の規則

規則を次に示します。

1. 埋込み変数，標識変数は，PREPARE 文で前処理する UPDATE 文，及び手続き中には指定できません。手続き中では，SQL 変数又は SQL パラメタを使用します。

2. 埋込み変数（標識変数），？パラメタ，SQL 変数，又は SQL パラメタを指定する場合，埋込み変数(？パラメタの場合は，それに値を与えるために指定する埋込み変数)，SQL 変数，又は SQL パラメタのデータ型は，更新する列のデータ型，又は変換できるデータ型にしてください。また，埋込み変数（標識変数），？パラメタの構造は単純構造にしてください。

3. 標識変数を指定した場合は，標識変数の値が負のとき，埋込み変数の値がナル値と解釈され，対応する列にナル値が設定されます。標識変数の値が 0 又は正の場合，埋込み変数の値が対応する列に設定されます。

4. ？パラメタは，PREPARE 文で前処理をする UPDATE 文の場合にだけ指定できます。？パラメタに与える値は，前処理をした PREPARE 文に対応する EXECUTE 文の USING 句の埋込み変数で指定します。

5. 要素の値には，更新対象になる列と変換・比較できるデータ型にしてください。ただし，更新対象になる列が各国文字データ型で，更新値として文字列定数を指定した場合，文字列定数を各国文字列定数とみなします。文字列定数を各国文字列定数とみなした場合，文字コードのチェックはしないで，文字データの長さだけチェックします。

6. 要素の値としてスカラ副問合せを指定する場合は，スカラ副問合せの選択式に添字のない繰返し列を指定できません。

7. 要素の値として DEFAULT を指定する場合，既定値はナル値となります。

連結演算を使用して BLOB 型，又は定義長が 32,001 バイト以上の BINARY 型の列を更新する場合の規則

規則を次に示します。

1. 連結演算の第 1 演算項には，列指定が指定できます。第 2 演算項には，埋込み変数，？パラメタ，SQL 変数，及び SQL パラメタを指定できます。

2. 更新値として BLOB 型，又は定義長が 32,001 バイト以上の BINARY 型の連結演算を指定する場合，更新対象と連結演算の第 1 演算項には，必ず同じ列を指定してください。

3. 連結演算の結果に対して，連結演算は指定できません。

4. BLOB 型と連結できるデータ型は，BLOB 型だけです。数データ，文字データ，各国文字データ，又は混在文字データとは連結できません。

5. BINARY 型と連結できるデータ型は，BINARY 型だけです。数データ，文字データ，各国文字データ，及び混在文字データとは連結できません。

6. 連結演算の結果は，第 1 演算項又は第 2 演算項の値の非ナル値制約の有無に関係なく，ナル値を許します。

7. 連結演算の結果の実長が，BLOB 型，又は BINARY 型の最大長（2,147,483,647 バイト）を超えた場合はエラーになります。

8. 更新対象の表に次の定義がある場合，連結演算を使用した更新はエラーになります。
   - 更新対象の表に UPDATE トリガを定義している
   - 連結演算を使用して更新する列を検査制約の探索条件に指定している

(f) SET （更新対象，更新対象〔，更新対象〕…）＝行副問合せ

行副問合せの結果で，複数の列の値を更新する場合に指定します。なお，更新対象は 2 個以上指定してください。

列名

更新する列の名称を指定します。

コンポネント指定

更新する抽象データ型の属性を指定します。

列名 ［{添字｜＊}］

列名

要素を更新する繰返し列を指定します。

［{添字｜＊}］

添字には，更新する要素の位置を指定します。＊は，要素の最後を更新する場合に指定します。

＊を指定した場合，更新対象となっている繰返し列の要素が 0 のときは，＊の指定は無効となります。

行副問合せ

更新するデータを取り出す行副問合せを指定します。行副問合せについては，「副問合せ」を参照してください。

行副問合せについての規則を次に示します。

1. 更新対象の数と，行副問合せの選択式の数は同じにしてください。
2. 行副問合せで得られる列のデータ型は，更新対象の列又は属性と同じデータ型，又は変換できるデータ型にしてください。
3. 更新対象のデータ型が文字データ型の場合，更新対象の文字集合と行副問合せで得られる列の文字集合は同じにしてください。
4. 更新対象に添字のない繰返し列を指定する場合，行副問合せの選択式の列は添字のない繰返し列の列名を指定してください。
5. 更新対象に添字のある繰返し列を指定する場合，行副問合せの選択式の列に添字のない繰返し列の列名は指定できません。

SET 句共通規則

1. SET 句で添字を指定して繰返し列の要素を更新する場合，同じ SET 句内で同じ列の同じ要素に対する更新は，1 回だけ指定できます。
2. SET 句で添字を指定しないで繰返し列を更新する場合，同じ SET 句内で添字を指定してその繰返し列を更新できません。
3. 添字に＊を指定して繰返し列を更新する場合，同じ SET 句内でその繰返し列を更新できません。
4. 更新する列に予備列は指定できません。

(g) ADD 繰返し列名 ［{添字｜＊}］＝ {ARRAY［要素の値〔，要素の値〕…］｜？パラメタ｜：埋込み変数〔：標識変数〕}

繰返し列の要素を追加する場合に指定します。

繰返し列名 ［{添字｜＊}］

繰返し列名

要素を追加する繰返し列を指定します。

［{添字｜＊}］

添字には，追加する要素の位置を指定します。＊は，最後に要素を追加する場合に指定します。

ARRAY ［要素の値〔，要素の値〕…］

要素の値として，次に示す項目が指定できます。

- 繰返し列以外の列名
- 添字付き繰返し列
- 定数
- 値式（四則演算，連結演算を含みます）
- スカラ副問合せ
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- NULL（ナル値を表します）
- DEFAULT
- ：埋込み変数〔：標識変数〕
- ?パラメタ
- SQL 変数，又は SQL パラメタ

要素の値の規則については，SET 更新対象＝更新値の「要素の値の規則」を参照してください。

?パラメタ

?パラメタのデータ型は，更新する列のデータ型，又は変換できるデータ型にしてください。また，?パラメタの構造は繰返し構造にしてください。

?パラメタは，PREPARE 文で前処理をする UPDATE 文の場合にだけ指定できます。?パラメタに与える値は，前処理をした PREPARE 文に対応する EXECUTE 文の USING 句の埋込み変数で指定します。

：埋込み変数〔：標識変数〕

埋込み変数，標識変数は，PREPARE 文で前処理する UPDATE 文，又は手続き中には指定できません。

埋込み変数（標識変数）のデータ型は，更新する列のデータ型，又は変換できるデータ型にしてください。また，埋込み変数（標識変数）の構造は，繰返し構造にしてください。

標識変数を指定した場合は，標識変数の値が負のとき，埋込み変数の値がナル値と解釈され，対応する列にナル値が設定されます。標識変数の値が 0 又は正の場合，埋込み変数の値が対応する列に設定されます。

ADD 句共通規則

1. 一つの ADD 句には，一つの列に対する要素の追加は，1 回だけ指定できます。
2. 要素を追加する場合，追加後の要素数が最大要素数を超えないようにしてください。
3. ARRAY ［要素の値〔，要素の値〕…］の中の要素の値は，最大 30,000 個指定できます。

4. 添字で指定した場所より後ろの要素は，新しく追加した要素の後ろに移動します。添字に，要素追加の対象となっている繰返し列の要素の数よりも 2 以上大きい数を指定した場合は，その繰返し列の要素の数が（指定した要素）-1 になるまでナル値を追加して，その後ろに要素の値を追加します。
5. 追加対象列に複数列インデクスが定義されている場合，そのインデクスを構成するすべての繰返し列に，一つの ADD 句で同じ数の要素を同じ要素の位置で指定して追加してください。
6. WHERE 句に CURRENT OF カーソル名を指定して，カーソルを用いた要素の追加をする場合，ADD 句の要素の値には列名及び値式を指定できません。
7. 要素の値として DEFAULT を指定する場合，既定値はナル値になります。

(h) DELETE 繰返し列名 ［{添字｜*}］

繰返し列の要素を削除する場合に指定します。

繰返し列名

要素を削除する繰返し列を指定します。

［{添字｜*}］

添字には，削除する要素の位置を指定します。*は，最後の要素を削除する場合に指定します。

DELETE 句共通規則

1. DELETE 句で添字を指定して要素を削除する場合は，同じ DELETE 句内で同じ列の同じ要素に対する削除は 1 回だけ指定できます。
2. 添字に，要素の削除対象になっている列の要素の数より大きい数を指定できません。
3. 要素に*を指定して繰返し列の要素を削除する場合は，同じ DELETE 句内でその列の要素を同時に指定できません。
4. 削除された要素よりも後ろの要素は，一つずつ繰り上がります。
5. 削除対象の列に複数列インデクスが定義されている場合，そのインデクスを構成するすべての繰返し列に，一つの DELETE 句で同じ数の要素を同じ要素の位置で指定して削除してください。

(i) 〔WHERE {探索条件｜ CURRENT OF {カーソル名｜拡張カーソル名}}〕

省略すると，指定した表のすべての行が更新されます。

探索条件

更新の対象となる行を選択する条件を指定します。

この探索条件を満足するすべての行を更新します。

カーソル名

更新する行を指すカーソルの名称を指定します。

PREPARE 文で前処理する場合は，指定できません。

カーソル名で指定するカーソルは，カーソル宣言で宣言されたカーソルにしてください。カーソル宣言の FOR UPDATE 句に，UPDATE 文で値を更新する列名を指定しておいてください。

カーソル名を指定する場合は，カーソル宣言で指定したカーソルが更新できなければなりません。更新できるカーソルについては，「DECLARE CURSOR　形式 1（カーソル宣言）」の共通規則 4.を参照してください。

カーソル名で指定するカーソルは，OPEN 文で開き，FETCH 文で更新する行に位置づけてください。

カーソル名で指定するカーソルの位置は，UPDATE 文実行後も同じです。更新した行よりも後の行を更新する場合は，そのカーソルに対して FETCH 文を実行して，カーソルを移動させます。

拡張カーソル名

更新する行を指すカーソルを識別する拡張カーソル名を指定します。

PREPARE 文で前処理する場合は，指定できません。

ALLOCATE CURSOR 文で割り当てられたカーソルを識別している拡張カーソル名を指定してください。ただし，結果集合カーソルは指定できません。

拡張カーソル名を指定する場合は，FOR UPDATE 句の指定のある問合せに対する拡張カーソルを指定しなければなりません。FOR UPDATE 句については，「動的 SELECT 文　形式 1（動的検索）」のオペランド規則の FOR UPDATE 句を参照してください。

拡張カーソル名が識別するカーソルは，開いた状態であり，かつ更新する行に位置づけられている必要があります。

拡張カーソル名が識別するカーソルの位置は，UPDATE 文実行後も同じです。更新した行よりも後の行を更新する場合は，そのカーソルに対して FETCH 文を実行して，カーソルを移動させます。

拡張カーソル名については，「拡張カーソル名」を参照してください。

(j)　〔WITH ROLLBACK〕

更新の対象となる表が，ほかのユーザで使用されているとき，トランザクションを取り消して無効にする場合，指定します。

WITH ROLLBACK を省略した場合は，更新する表がほかのユーザで使用されているとき，使用中のユーザのトランザクションが終了してから実行します。

(k)　〔WRITE IMMEDIATE〕

表オプションに WITHOUT ROLLBACK の指定がある表に対する更新処理で，更新完了時にシステムログを書き出す場合，このオプションを指定してください。WITHOUT ROLLBACK 指定のない表に対して，このオプションを指定しても無効になります。

更新完了時にシステムログを書き出す効果については，マニュアル「HiRDB UAP 開発ガイド」の採番業務で使用する表を参照してください。

(5) 共通規則

1. INTEGER，又は SMALLINT のデータ型の列を，固定小数点，又は浮動小数点数のデータで更新する場合は，端数（小数）部分は更新前に切り捨てられます。また，DECIMAL 型の列を固定小数点のデータで更新する場合，列の位取りより下位のけた部分が，更新前に切り捨てられます。
2. 更新する列の値として，表の定義時に指定した長さ以上の文字データ，BLOB データ，又は BINARY データは入力できません。
3. 更新する列の値として，定義域の範囲外の数データは入力できません。
4. 固定長文字列（各国文字列，及び混在文字列を含む）の列を更新するデータが列の長さより短い場合は，左詰めに挿入され，余りの部分に余白が設定されます。
5. データ型が BLOB の列又は属性を更新する場合，更新値には列指定，コンポネント指定，埋込み変数，？パラメタ，SQL 変数，SQL パラメタ，連結演算，スカラ関数 SUBSTR，関数呼出し，副問合せ，又は NULL が指定できます。
6. データ型が BLOB の列又は属性を更新する場合，データベース中に新しいデータを書き込んだ後，既存のデータを削除します。このため，データを更新する LOB 用 RD エリアには，新しいデータを書き込めるだけの空き領域が必要になります。空き領域が確保できない場合は，RD エリア満杯エラーになります。
7. 抽象データ型の列又は属性を更新する場合，更新値には埋込み変数及び？パラメタは指定できません。
8. 抽象データ型の列又は属性を更新する場合，更新値には表定義時に LOB 属性格納用 RD エリア名を指定していない BLOB 属性を含む抽象データ型の値は指定できません。
9. 要素を更新，削除する場合，指定した添字がその列の現在要素数より大きいとき，又は添字として＊を指定して現在要素数が 0 のときは，更新，削除する要素がないため，その行に対する更新，削除指定は無視されます。この場合，SQLCA の SQLWARN7 に’W’が設定されます。
10. SET 句，ADD 句，及び DELETE 句は，一つの SQL 文中にそれぞれ 1 回だけ指定できます。
11. SET 句，ADD 句，及び DELETE 句内の項目は，それぞれ最大 30,000 個指定できます。
12. SET 句，ADD 句，及び DELETE 句は，指定した順に左から実行されます。
13. LOB 列又は LOB 属性を格納するユーザ LOB 用 RD エリアが更新凍結状態の場合，その LOB 列又は LOB 属性は更新できません（更新しようとすると凍結済みエラーとなります）。
14. 改竄防止表に対して UPDATE 文は実行できません。ただし更新可能列は更新できます。

    改竄防止表で UPDATE ONLY FROM NULL 指定した列の値の更新可否を次に示します。

更新前の列の値	更新後の列の値	更新可否
ナル値	ナル値	○
ナル値	非ナル値	○
非ナル値	ナル値	×
非ナル値	非ナル値※	×

（凡例）

○：更新できます。

×：更新できません。

注

繰返し列の場合は，ナル値（要素数が 0 の値）から添え字指定無しでの列単位更新だけ実行できます。

注※

更新前の値と同値を含みます。

15. WITHOUT ROLLBACK を指定した表のインデクス構成列を更新対象列とする場合，インデクス構成列に対する更新値が同値更新以外の場合は実行できません。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
16. 更新対象表が共用表で，かつ更新対象列にインデクスを定義している場合，更新前にその共用表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表のインデクスを定義した列の値を更新した場合，エラーとなります。ただし，インデクスを定義している列の値に変更がない場合は，LOCK 文を実行する必要はありません。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。
17. 更新対象表が次の条件をすべて満たす場合，その列を DEFAULT で更新するには，更新前にその共用表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで更新した場合，エラーとなります。
    - 更新対象表が共用表である
    - 更新対象列が時刻印データ型である
    - 表定義時に既定値として CURRENT_TIMESTAMP USING BES を指定している
18. 探索条件にカーソル名，又は拡張カーソル名を指定した場合，更新対象表を FROM 句に指定した副問合せを，SET 句中に指定できません。
19. RD エリア名指定を指定した場合，分割数が表の分割数と異なるインデクスは利用できません。RD エリア名指定を指定する問合せのためにインデクスを定義する場合は，分割数が表の分割数と等しいインデクスを定義してください。

(6) 参照制約に関する規則

1. 被参照表の主キーの更新，参照表の外部キーの更新をする場合の規則については，「CREATE TABLE（表定義）」の参照動作の説明を参照してください。
2. 参照表の外部キー構成列の値を更新する場合，その更新値が被参照表の主キー構成列の値に含まれるかどうかを確認するため，被参照表を検索します。このとき，被参照表の検索時のデータ保証レベルは共用モードになります。そのため，参照表への更新時，ほかのトランザクションによって被参照表に対する操作が行われている場合，そのトランザクションが決着されるまでその更新は待ち状態になります。
3. 制約動作が RESTRICT で定義された被参照表の行を削除する場合，削除対象になる行中の主キー構成列の値が参照表の外部キー構成列の値に含まれるかどうかを確認するため，参照表を検索します。この

とき，参照表の検索時のデータ保証レベルは共用モードになります。そのため，制約動作が RESTRICT で定義された被参照表の行を削除するときに，ほかのトランザクションによって参照表に対する操作が行われていると，そのトランザクションが決着するまでその行削除は待ち状態になります。

4. 次の条件が重なった場合，参照制約での被参照表と参照表間でデータの不整合が発生することがあります。また，制約動作が RESIRICT 又は CASCADE のどちらかの場合でも発生することがあります。参照制約に関する規則については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。
   - 参照表の行を削除するトランザクションと，被参照表の行を更新，又は削除するトランザクションが異なる
   - 上記二つのトランザクションが同時に実行される
   - 参照表で削除する行の主キー構成列の値と，被参照表で更新，又は削除する行の外部キー構成列の値と同じである
   - 参照表の行を削除するトランザクションをコミット，被参照表の行を更新，又は削除するトランザクションをロールバックする

(7) 留意事項

1. カーソル名は，埋込み変数名と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって使用できません。

(8) 使用例

1. 在庫表（ZAIKO）の商品コード（SCODE）列が 302S の商品の在庫量（ZSURYO）列を 100 に変更します。

```
UPDATE ZAIKO
    SET ZSURYO = 100
    WHERE SCODE = ‘302S’
```

2. 在庫表（ZAIKO）の商品コード（SCODE）列で，最後の文字が S の商品の，単価（TANKA）を 2 割引きにします。

```
UPDATE ZAIKO
    SET TANKA=TANKA*0.8
    WHERE SCODE LIKE’%S’
```

3. 在庫表（ZAIKO）の単価（TANKA）列，在庫量（ZSURYO）列を，埋込み変数に読み込まれた値に変更します。

```
UPDATE ZAIKO
    SET TANKA=:XTANKA,ZSURYO=:XZSURYO
```

4. 在庫表（ZAIKO）の商品コード（SCODE）列が 302S の商品の在庫量（ZSURYO）を，在庫表と同じ列定義情報を持つ在庫表 2（ZAIKO2）の商品コード（SCODE）列が 302S の商品の在庫量（XSURYO）に変更します。

```
UPDATE ZAIKO
    SET ZSURYO=
        (SELECT ZSURYO FROM ZAIKO2 WHERE SCODE=’302S’)
    WHERE SCODE=’302S’
```

5. 在庫表（ZAIKO）の商品コード（SCODE）列が 302S の商品の在庫量（ZSURYO）と単価（TANKA）列を，在庫表と同じ列定義情報を持つ在庫表 2（ZAIKO2）の商品コード（SCODE）列が 302S の商品の在庫量（XSURYO）と単価（TANKA）に変更します。

```
UPDATE ZAIKO
    SET (TANKA,ZSURYO)=
        (SELECT TANKA,ZSURYO FROM ZAIKO2 WHERE SCODE=’302S’)
    WHERE SCODE=’302S’
```

4.37　UPDATE 文　形式 2（データ更新）

4.37.1　UPDATE 文 形式 2 の形式と規則

(1)　機能

表内の，指定した探索条件を満足する行，又はカーソルが指している行の値を行単位で更新します。

(2)　使用権限

表に対する UPDATE 権限を持つユーザが，その表の行の値を更新できます。

ただし，探索条件中に副問合せを指定する場合は，その副問合せ指定の表に対して SELECT 権限が必要です。

(3)　形式 2 <FIX 指定の表内の行を行単位で更新>

```
UPDATE 〔認可識別子.〕表識別子
          〔IN （RDエリア名指定）〕 〔〔AS〕 相関名〕
          〔使用インデクスのSQL最適化指定〕
           SET ROW＝行更新値
          〔WHERE｛探索条件｜CURRENT OF ｛カーソル名｜拡張カーソル名｝｝〕
          〔WITH ROLLBACK〕
          〔WRITE IMMEDIATE〕
```

(4)　オペランド

ROW 指定以外のオペランド，及びオペランド規則は，UPDATE 文の形式 1 と同じです。ROW を指定する場合の規則を次に示します。

1. 行単位の更新は，FIX 属性の実表に対してだけ使用できます。ROW は行全体（予備列を含む）を意味し，これを指定すると行全体を一つのデータとして，一つの領域のデータで更新します。更新するデータのデータ型は，各列のデータ型に関係なく，ROW 型（ROW 型に対しては，CHAR（n）〔n は行長〕に対する変数，又は同じ長さの構造体を指定できます。ただし，構造体中に境界調整による空きがあってはいけません）になります。また，データ長は，行長（各列のデータ長の総和）になります。
2. UAP が動作するプラットフォームと HiRDB サーバが動作するプラットフォームのエンディアンを同じにしてください。異なるエンディアン間では，ROW は使用できません。例えば，Windows の UAP で ROW を使用する場合は，HiRDB サーバも同じエンディアンの Windows 版を使用してください。

(a)　SET 句

列の値を更新します。

(b) SET ROW ＝行更新値

ROW

行単位でデータを更新する場合に指定します。

行更新値

ROW に対応した行更新値として，次に示す項目が指定できます。

- ：埋込み変数〔：標識変数〕
- ?パラメタ
- SQL 変数，又は SQL パラメタ

(5) 共通規則

1. 行単位での UPDATE 文はすべての列の値を更新するため，少なくとも 1 列は更新不可となる改竄防止表に対して行単位での UPDATE 文は実行できません。
2. 既定文字集合以外の文字データ型の列を含む表に対して，行単位での UPDATE 文は実行できません。
3. WITHOUT ROLLBACK を指定し，かつインデクスを定義した表を表識別子に指定する場合，インデクス構成列に対する更新値が同値更新以外の場合は実行できません。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
4. 更新対象表が共用表で，かつどれかの列にインデクスを定義している場合，更新前にその共用表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表を行単位（ROW 指定）で更新した場合，エラーとなります。ただし，インデクスを定義している列の値に変更がない場合は LOCK 文を実行する必要はありません。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。
5. RD エリア名指定を指定した場合，分割数が表の分割数と異なるインデクスは利用できません。RD エリア名指定を指定する問合せのためにインデクスを定義する場合は，分割数が表の分割数と等しいインデクスを定義してください。

(6) 参照制約に関する規則

1. 被参照表，参照表に対して行単位で更新する場合の規則については，「CREATE TABLE（表定義）」の参照動作の説明を参照してください。
2. 制約動作が RESTRICT で定義された被参照表の主キー構成列の値を更新する場合，その更新値が参照表の外部キー構成列の値に含まれるかどうかを確認するため，参照表を検索します。このとき，参照表の検索時のデータ保証レベルは共用モードになります。そのため，制約動作が RESTRICT で定義された被参照表の行を削除するときに，ほかのトランザクションによって参照表に対する操作が行われていると，そのトランザクションが決着するまでその行削除は待ち状態になります。
3. 次の条件が重なった場合，参照制約での被参照表と参照表間でデータの不整合が発生することがあります。また，制約動作が RESIRICT 又は CASCADE のどちらかの場合でも発生することがあります。参

照制約に関する規則については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

- 参照表の行を削除するトランザクションと，被参照表の行を更新，又は削除するトランザクションが異なる
- 上記二つのトランザクションが同時に実行される
- 参照表で削除する行の主キー構成列の値と，被参照表で更新，又は削除する行の外部キー構成列の値と同じである
- 参照表の行を削除するトランザクションをコミット，被参照表の行を更新，又は削除するトランザクションをロールバックする

4. 次の条件が重なった場合，被参照表を操作するトランザクションと参照表を操作するトランザクション間でデッドロックが発生することがあります。また，制約動作が RESIRICT，CASCADE のどれかの場合でも発生することがあります。被参照表と参照間のデッドロックについては，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。
   - 参照表の行を更新するトランザクションと，被参照表の行を更新するトランザクションが異なる
   - 上記二つのトランザクションが同時に実行される
   - 参照表で更新する行の外部キー構成列の値と，被参照表で削除する行の主キー構成列が同じである

(7) 留意事項

1. 更新する表の列のデータ型が DECIMAL，又は各国文字列の場合に，対応する部分だけ行更新の内容をチェックします。
2. カーソル名は，埋込み変数と同様に，コンパイル単位のモジュール内で有効な名前であり，同じカーソルに対する複数の SQL を，複数のモジュールにわたって指定できません。
3. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の日付データ型の部分は 4 バイトであり，X’YYYYMMDD’の形式にしてください。

   行単位（ROW 指定）インタフェースを使用し，日付データを既定の文字列表現で受け渡しする場合，列定義時に日付データ型ではなく，CHAR（10）として列を定義してください。

   さらに，日付演算は，スカラ関数 DATE を使用し，日付データ型に変換した後に指定してください。
4. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の時刻データ型の部分は 3 バイトであり，X’hhmmss’の形式にしてください。

   行単位（ROW 指定）インタフェースを使用し，時刻データを既定の文字列表現で受け渡しする場合，列定義時に時刻データ型ではなく，CHAR（8）として列を定義してください。さらに，時刻演算は，スカラ関数 TIME を使用し，時刻データ型に変換した後に指定してください。
5. 行単位（ROW 指定）の検索，又は更新をする場合，ROW に対する埋込み変数，SQL 変数，又は SQL パラメタ中の時刻印データ型の部分は（7+p/2）バイトであり，X’YYYYMMDDhhmmss［nn…n］’の形式にしてください。

行単位（ROW 指定）インタフェースを使用し，時刻印データを文字列表現で受け渡しする場合，列定義時に時刻印データ型ではなく，長さが 19，22，24，又は 26 バイトの CHAR として列を定義してください。

6. 更新する表に予備列が定義されている場合，対応する部分のデータは無視され，予備列の定義長分の 0x00 で更新します。

(8) 使用例

在庫表（ZAIKO）のカーソル（CR1）が指定した行のデータを埋込み変数（XROW）の内容でまとめて更新します。

```
UPDATE ZAIKO
    SET ROW = :XROW
    WHERE CURRENT OF CR1
```

4.38　UPDATE 文　形式 3，形式 4（配列を使用した行更新）

4.38.1　UPDATE 文 形式 3，形式 4 の形式と規則

(1)　機能

配列形式の埋込み変数を指定して，複数の更新処理を実行できます。

形式 3

　表内の，指定した探索条件を満足する行の値を列単位で複数回更新します。

形式 4

　FIX 指定の表内の，指定した探索条件を満足する行の値を行単位で複数回更新します。

(2)　使用権限

表に対する UPDATE 権限を持つユーザが，その表の列の値を更新できます。

ただし，探索条件中に副問合せを指定する場合は，その副問合せ指定の表に対して SELECT 権限が必要です。

(3)　形式 3 <埋込み変数配列を指定して，列単位で表内の行を複数回更新>

```
FOR :埋込み変数
UPDATE 〔認可識別子.〕表識別子
            〔IN （RDエリア名指定）〕 〔〔AS〕 相関名〕
            〔使用インデクスのSQL最適化指定〕
           SET {更新対象＝更新値
                   |（更新対象，更新対象〔，更新対象〕）＝行副問合せ}
                  〔，{更新対象＝更新値
                        |（更新対象，更新対象〔，更新対象〕…）＝行副問合せ}〕…
〔WHERE 探索条件〕
〔WITH ROLLBACK〕
〔WRITE IMMEDIATE〕

更新対象：：＝｛列名｜コンポネント指定｜列名［｛添字｜＊｝］｝
```

(4)　形式 4 <埋込み変数配列を指定して，FIX 指定の表内の行を行単位で複数回更新>

```
FOR ：埋込み変数
UPDATE 〔認可識別子.〕表識別子
            〔IN （RDエリア名指定）〕 〔〔AS〕 相関名〕
            〔使用インデクスのSQL最適化指定〕
           SET ROW＝行更新値
          〔WHERE 探索条件〕
```

```
        〔WITH ROLLBACK〕
        〔WRITE IMMEDIATE〕
```

(5) オペランド

FOR，IN（RD エリア名指定），SET 句，探索条件以外のオペランド，及びオペランド規則については，「UPDATE 文　形式 1（データ更新）」及び「UPDATE 文　形式 2（データ更新）」を参照してください。

(a) 形式 3 のオペランド及びオペランド規則

SET 句

列の値を更新します。

更新値

更新値の列の値として，次に示す項目が指定できます。

- ：埋込み変数配列〔：標識変数配列〕
- 列名
- 定数
- 値式（四則演算，連結演算を含みます）
- スカラ副問合せ
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- NULL（ナル値を表します）
- DEFAULT（更新対象の列の既定値を表します）
- ?パラメタ

ただし，更新値には配列形式でない埋込み変数は指定できません。

(b) 形式 4 のオペランド及びオペランド規則

SET 句

列の値を更新します。

ROW

行単位でデータを更新する場合に指定します。

行更新値

ROW に対応した行更新値として，次に示す項目が指定できます。

：埋込み変数配列〔：標識変数配列〕

ROW を指定する場合の規則を次に示します。

1. 行単位の更新は，FIX 属性の実表に対してだけ使用できます。ROW は行全体（予備列を含む）を意味し，これを指定すると行全体を一つのデータとして，一つの領域のデータで更新します。更新するデータのデータ型は，各列のデータ型に関係なく，ROW 型になります。ROW 型に対しては，CHAR（n）〔n は行長〕に対する変数，又は同じ長さの構造体を指定できます。ただし，構造体中に境界調整による空きがあってはいけません。また，データ長は，行長（各列のデータ長の総和）になります。
2. UAP が動作するプラットフォームと HiRDB サーバが動作するプラットフォームのエンディアンを同じにしてください。異なるエンディアン間では，ROW は使用できません。例えば，Windows の UAP で ROW を使用する場合は，HiRDB サーバも同じエンディアンの Windows 版を使用してください。

(c) 形式 3 及び形式 4 に共通するオペランド及びオペランド規則

FOR ：埋込み変数

埋込み変数配列を使用して更新を行う回数を設定した埋込み変数を指定します。SMALLINT 型の埋込み変数を指定してください。設定値は 1 から 4,096 までの範囲で，かつ埋込み変数配列，及び標識変数配列の要素数以下にしてください。0 及び負の値は設定できません。範囲外の値を設定した場合は実行時にエラーとなります。

埋込み変数配列

配列形式で宣言した埋込み変数です。NULL 値以外の更新値を指定するための配列変数を指定します。それぞれの行を更新する値を埋込み変数配列の各要素に設定してください。更新する値に NULL 値を含む場合は，埋込み変数配列と標識変数配列を両方指定します。

標識変数配列

配列形式で宣言した標識変数です。埋込み変数配列の各要素を更新する値が，NULL 値かどうかを示す値を，標識変数配列の対応する要素に設定してください。設定する値については，「標識変数の値の設定」を参照してください。

〔IN （RD エリア名指定)〕

RD エリア名指定を埋込み変数で行う場合は，埋込み変数配列にしてください。配列でない埋込み変数を指定した場合はエラーとなります。そのほかのオペランド規則については，形式 1 を参照してください。

〔 WHERE 探索条件 〕

省略すると，指定した表のすべての行が更新されます。

探索条件

更新の対象となる行を選択する条件を指定します。この探索条件を満足するすべての行を更新します。探索条件に埋込み変数を使用する場合，配列形式でない埋込み変数は指定できません。

(6) 共通規則

(a) 形式 3 の規則

1. 埋込み変数配列のデータ型は対応する列のデータ型，又は変換できるデータ型にしてください。
2. 配列を使用した UPDATE では，BLOB 型，最大長が 32,001 バイト以上の BINARY 型及び抽象データ型は扱えません。
3. 配列を使用した UPDATE では，繰返し列の複数要素の更新はできません。
4. INTEGER，又は SMALLINT のデータ型の列を，固定小数点，又は浮動小数点数のデータで更新する場合は，端数（小数）部分は更新前に切り捨てられます。また，DECIMAL データ型の列を固定小数点のデータで更新する場合には，列の位取りから下位のけたの部分が更新前に切り捨てられます。
5. 更新する列の値として，表の定義時に指定した長さ以上の文字データ又は BINARY データは入力できません。
6. 更新する列の値として，定義域の範囲外の数データは入力できません。
7. 固定長文字列（各国文字列，混在文字列を含む）の列を更新するデータが列の長さより短い場合は，左詰めに挿入され，余りの部分に余白が設定されます。
8. SET 句は，一つの SQL 文中にそれぞれ 1 回だけ指定できます。
9. SET 句内の項目は，それぞれ最大 30,000 個指定できます。
10. 改竄防止表に対して UPDATE 文は実行できません。ただし，更新可能列は更新できます。

    改竄防止表で UPDATE ONLY FROM NULL 指定した列の値の更新可否を次に示します。

更新前の列の値	更新後の列の値	更新可否
ナル値	ナル値	○
ナル値	非ナル値	○
非ナル値	ナル値	×
非ナル値	非ナル値※	×

（凡例）

○：更新できます。

×：更新できません。

注

繰返し列の場合は，ナル値（要素数が 0 の値）から添え字指定無しでの列単位更新だけ実行できます。

改竄防止表で，指定した探索条件を満たす行のうち，UPDATE ONLY FROM NULL を指定した列の値に非ナル値の行がある場合，エラーになります。

注※

更新前の値と同値を含みます。

11. 更新対象列を含む表に WITHOUT ROLLBACK を指定し，かつ更新対象列にインデクスを定義している場合にエラーとなります。
12. WITHOUT ROLLBACK を指定した表のインデクス構成列を更新対象列とする場合，インデクス構成列に対する更新値が同値更新以外のときは実行できません。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
13. 更新対象表が共用表で，かつ更新対象列にインデクスを定義している場合，更新前にその共用表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表のインデクスを定義した列の値を更新した場合，エラーとなります。ただし，インデクスを定義している列の値に変更がない場合は LOCK 文を実行する必要はありません。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。
14. 更新対象表が次の条件をすべて満たす場合，その列を DEFAULT で更新するには更新前にその共用表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで更新した場合，エラーとなります。
    - 更新対象表が共用表である
    - 更新対象列が時刻印データ型である
    - 表定義時に既定値として CURRENT_TIMESTAMP USING BES を指定している

(b) 形式 4 の規則

1. WITHOUT ROLLBACK を指定し，かつインデクスを定義した表を表識別子に指定する場合，インデクス構成列に対する更新値が同値更新以外の場合は実行できません。詳細は「CREATE TABLE（表定義）」の WITHOUT ROLLBACK の規則を参照してください。
2. 更新対象表が共用表で，かつどれかの列にインデクスを定義している場合，更新前にその共用表に対する LOCK 文を排他モードで実行してください。LOCK 文を実行しないで共用表を行単位（ROW 指定）で更新した場合，エラーとなります。ただし，インデクスを定義している列の値に変更がない場合は LOCK 文を実行する必要はありません。共用表に対する更新については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。共用表に対して LOCK 文を実行した場合の排他制御の対象については「LOCK 文（表の排他制御）」の留意事項を参照してください。
3. 行単位での UPDATE 文はすべての列の値を更新するため，1 列でも更新不可である改竄防止表に対して，行単位での UPDATE 文は実行できません。
4. 既定文字集合以外の文字データ型の列を含む表に対しては，行単位での UPDATE 文は実行できません。

(c) 形式 3 及び形式 4 の共通規則

1. FOR 句以外に埋込み変数配列を一つ以上指定してください。指定しない場合はエラーとなります。
2. FOR 句以外に配列形式でない埋込み変数を指定するとエラーとなります。
3. 埋込み変数配列，及び標識変数配列の要素数は 1 から 4,096 までの範囲にしてください。範囲外の値を指定した場合はエラーとなります。また，”FOR ：埋込み変数”で指定する要素数の最大値以上になるようにしてください。

4. 各埋込み変数配列，及び標識変数配列で 1 回の更新処理によって評価される要素は同じ要素番号の要素となります。
5. 配列の先頭の要素から順に評価されます。
6. 複数回更新を行う場合の更新対象は，前回の配列要素を使用して更新が行われた後の更新対象となります。
7. SQL 連絡領域の SQLERRD[2]領域には，重複して更新された行数も含めた更新行数の総和が設定されます。
8. UPDATE 文形式 3 は埋込み変数配列，及び標識変数配列を含むため，PREPARE 文で前処理することはできません。動的に実行する場合については，「EXECUTE 文　形式 2（配列を使用した SQL の実行）」を参照してください。
9. 配列を使用した UPDATE は手続きの中では使えません。
10. 更新するどれかの行で警告する必要のある事象が発生した場合，SQL 連絡領域の SQLWARN に警告情報が設定されます。
11. 更新するどれかの行でエラーが発生した場合，ロールバックされます。
12. UPDATE 文は改竄防止表に対して指定できません。ただし，更新可能列を定義している場合は形式 3 及び形式 4 の規則に従ってください。
13. RD エリア名指定を指定した場合，分割数が表の分割数と異なるインデクスは利用できません。RD エリア名指定を指定した問合せのためにインデクスを定義する場合は，分割数が表の分割数と等しいインデクスを定義してください。

(7)　参照制約に関する規則

1. 被参照表の主キーの更新，参照表の外部キーの更新をする場合，又は被参照表，参照表に対して行単位で更新する場合の規則については，「CREATE TABLE（表定義）」の参照動作の説明を参照してください。
2. 制約動作が RESTRICT で定義された被参照表の主キー構成列の値を更新する場合，その更新値が参照表の外部キー構成列の値に含まれるかどうかを確認するため，参照表を検索します。このとき，参照表の検索時のデータ保証レベルは共用モードになります。そのため，制約動作が RESTRICT で定義された被参照表の行を削除するときに，ほかのトランザクションによって参照表に対する操作が行われていると，そのトランザクションが決着するまでその行削除は待ち状態になります。
3. 次の条件が重なった場合，参照制約での被参照表と参照表間でデータの不整合が発生することがあります。また，制約動作が RESIRICT 又は CASCADE のどちらかの場合でも発生することがあります。
   - 参照表の行を削除するトランザクションと，被参照表の行を更新，又は削除するトランザクションが異なる
   - 上記二つのトランザクションが同時に実行される
   - 参照表で削除する行の主キー構成列の値と，被参照表で更新，又は削除する行の外部キー構成列の値と同じである

- 参照表の行を削除するトランザクションをコミット，被参照表の行を更新，又は削除するトランザクションをロールバックする

4. 次の条件が重なった場合，被参照表を操作するトランザクションと参照表を操作するトランザクション間でデッドロックが発生することがあります。また，制約動作が RESIRICT，CASCADE のどれかの場合でも発生することがあります。
   - 参照表の行を更新するトランザクションと，被参照表の行を更新するトランザクションが異なる
   - 上記二つのトランザクションが同時に実行される
   - 参照表で更新する行の外部キー構成列の値と，被参照表で削除する行の主キー構成列が同じである

(8) 使用例

1. UPDATE 文形式 1 を使用して，C言語の配列変数に設定した商品コード（SCODE）及び在庫量（ZSURYO）の値別に，在庫量（ZSURYO）を次の表で示す値に更新します。

表 4-9 表に格納された商品コードと在庫量（更新前）

商品コード	在庫量の更新値
‘101M’	40
‘101L’	70
‘201M’	15
‘202M’	28
‘302S’	7

表 4-10 更新対象となる行の商品コードと在庫量（埋込み変数配列に設定）

商品コード	在庫量の更新値
‘101M’	35
‘101L’	62
‘201M’	13
‘202M’	10
‘302S’	6

表 4-11 表に格納された商品コードと在庫量（更新後）

商品コード	在庫量の更新値
‘101M’	35
‘101L’	62
‘201M’	13
‘202M’	10

商品コード	在庫量の更新値
‘302S’	6

```
EXEC SQL BEGIN DECLARE SECTION;
      short   XUPDATE_NUM;
      char    XSCODE[5][5];
      short   ISCODE[5];
      long    XZSURYO[5];
      short   IZSURYO[5];
EXEC SQL END DECLARE SECTION;
     ・・・各変数配列の要素に値設定・・・
        XSCODEに{’101M’,’101L’,’201M’,’202M’,’302S’}を設定
        XZSURYOに{35,62,13,10,6}を設定
XUPDATE_NUM = 5;
EXEC SQL FOR :XUPDATE_NUM
UPDATE ZAIKO SET ZSURYO = :XZSURYO:IZSURYO
WHERE SCODE = :XSCODE:ISCODE;
```

2. UPDATE 文形式 2 を使用して，C言語の配列変数に設定した商品コード（SCODE）の値別に，在庫表（ZAIKO）の行全体を，埋込み変数配列(XROW)の内容でまとめて更新します。

```
XUPDATE_NUM = 5;
EXEC SQL FOR :XUPDATE_NUM
UPDATE ZAIKO SET ROW = :XROW
WHERE SCODE = :XSCODE:ISCODE;
```

4.39　準備可能動的 UPDATE 文：位置付け 形式 1（前処理可能なカーソルを使用したデータ更新）

4.39.1　準備可能動的 UPDATE 文：位置付け 形式 1 の形式と規則

(1)　機能

表内のカーソルが指している行の指定した列を更新します。PREPARE 文で前処理してから EXECUTE 文で実行，又は EXECUTE IMMEDIATE 文で前処理と実行を一度に行う場合に使用します。

(2)　使用権限

表に対する UPDATE 権限を持つユーザが，その表の列の値を更新できます。

(3)　形式 1 <列単位で表内の行をカーソルを使用して更新（前処理可能）>

```
UPDATE 〔〔認可識別子.〕表識別子
            〔IN （RDエリア名指定）〕 〔〔AS〕 相関名〕
            〔使用インデクスのSQL最適化指定〕〕
            {SET {更新対象＝更新値
                    |（更新対象，更新対象〔，更新対象〕）＝行副問合せ}
                   〔，{更新対象＝更新値
                         |（更新対象，更新対象〔，更新対象〕…）＝行副問合せ}〕…
             |ADD繰返し列名 ［{添字|＊}］
                       ＝{ARRAY ［要素の値〔，要素の値〕…］
                          |？パラメタ}
                      〔，繰返し列名 ［{添字|＊}］
                            ＝{ARRAY ［要素の値〔，要素の値〕…］
                               |？パラメタ}〕…
             |DELETE繰返し列名 ［{添字|＊}］
                        〔，繰返し列名 ［{添字|＊}］〕…}
WHERE CURRENT OF GLOBAL カーソル名
 〔WITH ROLLBACK〕
 〔WRITE IMMEDIATE〕

更新対象：：＝{列名|コンポネント指定|列名［{添字|＊}］}
```

(4)　オペランド

SET 句の更新値と ARRAY ［要素の値〔，要素の値〕 …］，WHERE CURRENT OF GLOBAL カーソル名以外のオペランド，及びオペランド規則については，「UPDATE 文　形式 1（データ更新）」を参照してください。

(a) SET 句

更新値

更新値の列の値として次に示す項目が指定できます。

- 列名
- コンポネント指定
- 定数
- 値式（四則演算，連結演算を含む）
- スカラ副問合せ
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- NULL（ナル値を表す）
- DEFAULT（更新対象の列の既定値を表す）
- ?パラメタ
- ARRAY ［要素の値［，要素の値］…］※

注※

要素の値には，次の項目を指定できます。

- 繰返し列以外の列名
- 添字付き繰返し列
- 定数
- 値式（四則演算，連結演算を含みます）
- スカラ副問合せ
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- NULL（ナル値を表します）
- DEFAULT
- ?パラメタ

更新値の規則

1. 埋込み変数，標識変数，SQL 変数，及び SQL パラメタを含む更新値は指定できません。

2. そのほかの更新値の規則については，UPDATE 文 形式 1 の更新値の規則が適用されます。

要素の値の規則

1. 埋込み変数，標識変数，SQL 変数，及び SQL パラメタを含む要素の値は指定できません。

2. そのほかの要素の値の規則については，UPDATE 文 形式 1 の要素の値の規則が適用されます。

ARRAY ［要素の値〔，要素の値〕…］

要素の値として，次に示す項目が指定できます。

- 繰返し列以外の列名
- 添字付き繰返し列
- 定数
- 値式（四則演算，連結演算を含む）
- スカラ副問合せ
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- NULL（ナル値を表す）
- DEFAULT
- ?パラメタ

要素の値の規則については，「UPDATE 文　形式 1（データ更新）」の SET 句の「要素の値の規則」を参照してください。

(b)　WHERE CURRENT OF GLOBAL カーソル名

GLOBAL

カーソル名の有効範囲として GLOBAL を指定します。

カーソル名

更新する行を指すカーソルの名称を指定します。

カーソル名で指定するカーソルは，ALLOCATE CURSOR 文で指定した拡張カーソル名が識別するカーソルです。ALLOCATE CURSOR 文で指定した拡張カーソル名の値を指定してください。ただし，結果集合カーソルは指定できません。

実行時には，カーソル名で指定するカーソルは，開いた状態であり，かつ更新する行に位置づけられている必要があります。

カーソル名で指定するカーソルの位置は，UPDATE 文実行後も同じです。更新した行よりも後の行を更新する場合は，そのカーソルに対して FETCH 文を実行して，カーソルを移動させてください。

拡張カーソル名を指定する場合は，FOR UPDATE 句の指定のある問合せに対する拡張カーソルを指定しなければなりません。FOR UPDATE 句については，「動的 SELECT 文　形式 1（動的検索）」のオペランド規則の FOR UPDATE 句を参照してください。

(5)　共通規則

1. PREPARE 文で前処理してから，EXECUTE 文で実行，又は EXECUTE IMMEDIATE 文で前処理と実行を一度に行います。
2. 表識別子を省略する場合は，前処理する前に，ALLOCATE CURSOR 文によって動的 SELECT 文にカーソルが割り当てられている必要があります。このとき，カーソルを割り当てた動的 SELECT 文に指定している検索対象の表を仮定します。表識別子を指定する場合には，前処理する前に，動的 SELECT 文にカーソルが割り当てられている必要はありません。
3. そのほかの共通規則については，UPDATE 文 形式 1 の共通規則が適用されます。

(6)　参照制約に関する規則

1. UPDATE 文 形式 1 の参照制約に関する規則が適用されます。

(7)　使用例

1. 在庫表（ZAIKO）内のカーソル（cr（有効範囲：GLOBAL，値：’CR1’））が指している行の単価（TANKA）を 1 割引に更新する SQL を動的に実行します。

```
<埋込み変数selに任意の名称を設定>
PREPARE GLOBAL :sel FROM 'SELECT * FROM ZAIKO FOR UPDATE'
＜埋込み変数crに’CR1’を設定＞
ALLOCATE GLOBAL :cr CURSOR FOR GLOBAL :sel
PREPARE PRE1 FOR
     'UPDATE SET TANKA = ＜単価の 1 割引の値＞ WHERE CURRENT OF GLOBAL CR1'
OPEN GLOBAL :cr
FETCH GLOBAL :cr INTO ＜各列を取り出す変数名＞
EXECUTE PRE1
CLOSE GLOBAL :cr
DEALLOCATE PREPARE GLOBAL :sel
```

4.40　準備可能動的 UPDATE 文：位置付け 形式 2（前処理可能なカーソルを使用したデータ更新）

4.40.1　準備可能動的 UPDATE 文：位置付け 形式 2 の形式と規則

(1)　機能

FIX 指定の表内のカーソルが指している行の指定した列を更新します。PREPARE 文で前処理してから EXECUTE 文で実行，又は EXECUTE IMMEDIATE 文で前処理と実行を一度に行う場合に使用します。

(2)　使用権限

表に対する UPDATE 権限を持つユーザが，その表の列の値を更新できます。

(3)　形式 2 <FIX 指定の表内の行をカーソルを使用して行単位で更新（前処理可能）>

```
UPDATE 〔〔認可識別子.〕表識別子
          〔IN （RDエリア名指定）〕 〔〔AS〕 相関名〕
          〔使用インデクスのSQL最適化指定〕〕
           SET ROW＝行更新値
         WHERE CURRENT OF GLOBAL カーソル名
          〔WITH ROLLBACK〕
          〔WRITE IMMEDIATE〕
```

(4)　オペランド

SET 句の行更新値，及び WHERE CURRENT OF GLOBAL カーソル名以外のオペランド，及びオペランド規則については，「UPDATE 文　形式 2（データ更新）」を参照してください。

また，WHERE CURRENT OF GLOBAL カーソル名のオペランド，及びオペランド規則については，「準備可能動的 UPDATE 文：位置付け 形式 1（前処理可能なカーソルを使用したデータ更新）」を参照してください。

(a)　SET 句

行更新値

ROW に対応した行更新値として，次に示す項目が指定できます。

- ?パラメタ

(5) 共通規則

1. PREPARE 文で前処理してから，EXECUTE 文で実行，又は EXECUTE IMMEDIATE 文で前処理と実行を一度に行います。
2. 表識別子を省略する場合は，前処理する前に，ALLOCATE CURSOR 文によって動的 SELECT 文にカーソルが割り当てられている必要があります。このとき，カーソルを割り当てた動的 SELECT 文に指定している検索対象の表を仮定します。表識別子を指定する場合には，前処理する前に，動的 SELECT 文にカーソルが割り当てられている必要はありません。
3. そのほかの共通規則については，UPDATE 文 形式 2 の共通規則が適用されます。

(6) 参照制約に関する規則

1. UPDATE 文 形式 2 の参照制約に関する規則が適用されます。

(7) 留意事項

1. UPDATE 文 形式 2 の留意事項が適用されます。

(8) 使用例

1. 在庫表（ZAIKO）内のカーソル（cr（値：’CR1’））が指している行のデータを埋込み変数（XROW）の内容でまとめて更新する SQL を動的に実行します。

```
PREPARE GLOBAL :sel FOR  ‘SELECT * FROM ZAIKO FOR UPDATE’
＜埋込み変数crに’CR1’を設定＞
ALLOCATE GLOBAL :cr CURSOR FOR GLOBAL :sel
PREPARE PRE1 FOR
     ‘UPDATE SET ROW = ? WHERE CURRENT OF GLOBAL CR1’
OPEN GLOBAL :cr
FETCH GLOBAL :cr INTO ＜各列を取り出す変数名＞
EXECUTE PRE1 USING :XROW
CLOSE GLOBAL :cr
DEALLOCATE PREPARE GLOBAL :sel
```

4.41　代入文　形式 1（SQL 変数，又は SQL パラメタへの値の代入）

4.41.1　代入文 形式 1 の形式と規則

(1)　機能

SQL 変数，又は SQL パラメタに値を代入します。

(2)　使用権限

なし。

(3)　形式　<SQL 変数，又は SQL パラメタへの代入>

```
SET 代入先 ＝ 代入値

代入先：：＝ {〔文ラベル.〕SQL変数名
             |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
             |〔文ラベル.〕SQL変数名..属性名〔..属性名〕…
             |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名..属性名〔..属性名〕…}
代入値：：＝ {値式 | NULL | DEFAULT}
```

(4)　オペランド

(a)　代入先

```
：：＝ {〔文ラベル.〕SQL変数名
       |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名
       |〔文ラベル.〕SQL変数名..属性名〔..属性名〕…
       |〔〔認可識別子.〕ルーチン識別子.〕SQLパラメタ名..属性名〔..属性名〕…}
```

値を代入する SQL 変数，SQL パラメタ，又は SQL 変数の属性名，SQL パラメタの属性名を指定します。

パブリック手続き定義，又はパブリック関数定義の SQL 手続き文中で認可識別子を指定する場合は，認可識別子に PUBLIC を指定してください。

(b)　代入値：：＝ {値式 | NULL | DEFAULT}

代入する値を指定します。

(5)　共通規則

1. 代入先には，SQL ルーチンで指定した入出力モード（パラメタモード）が IN の SQL パラメタ名を指定できません。また，関数の SQL パラメタ名も指定できません。

2. 代入先のデータ型は，代入値のデータ型と互換性がなければなりません。
3. 代入先と代入値のデータ型が異なる場合には型変換され，同じ場合にはそのまま代入値が代入先に代入されます。
4. 代入値が文字データ型の場合，代入先と代入値の文字集合は同じでなければなりません。ただし，代入値が文字列定数の場合は，代入先と代入値の文字集合が異なっていても，自動的に代入先の文字集合に変換してから代入します。
5. 代入値に指定する値式中に，副問合せは指定できません。
6. 代入値に DEFAULT を指定した場合，代入の対象となる SQL 変数の既定値を代入します。SQL 変数の既定値の宣言は「複合文（複数文実行）」を参照してください。代入先に SQL パラメタを指定し，代入値に DEFAULT を指定した場合，代入先の SQL パラメタにナル値を代入します。トリガ SQL 文中に指定した代入文（形式 1 ）の代入先に新旧値相関名で修飾された列名を指定し，代入値に DEFAULT を指定した場合，代入先の列の既定値を代入します。ただし，代入先の新旧値相関名で修飾された列の既定値が USING BES を指定した CURRENT_TIMESTAMP の場合は，代入値に DEFAULT を指定できません。トリガ SQL 文については「CREATE TRIGGER（トリガ定義）」を参照してください。

(6) 留意事項

1. 代入文の形式 1 は，SQL ルーチン中にだけ指定できます。SQL ルーチン以外で代入文を指定する場合は，代入文の形式 2 を指定してください。

4.42　代入文　形式 2（埋込み変数，又は？パラメタへの値の代入）

4.42.1　代入文 形式 2 の形式と規則

(1)　機能

埋込み変数，又は？パラメタに値を代入します。

(2)　使用権限

なし。

(3)　形式　＜埋込み変数，又は？パラメタへの代入＞

```
SET　代入先 = 代入値
代入先：：={：埋込み変数〔：標識変数〕| ？パラメタ}
代入値：：={：埋込み変数〔：標識変数〕ASデータ型
              |？パラメタASデータ型
              |LENGTH（値式）
              |SUBSTR（値式1，値式2 〔，値式3〕）
              |POSITION（値式1 IN値式2 〔FROM値式3〕）}
```

(4)　オペランド

(a)　代入先：：＝ {：埋込み変数〔：標識変数〕 | ？パラメタ}

値を代入する埋込み変数，又は？パラメタを指定します。

(b)　代入値

```
：：＝{：埋込み変数〔：標識変数〕 ASデータ型
       |？パラメタASデータ型
       |LENGTH（値式）
       |SUBSTR（値式1，値式2 〔，値式3〕）
       |POSITION（値式1 IN値式2 〔FROM値式3〕）}
```

代入する値を指定します。

：埋込み変数〔：標識変数〕 AS データ型

指定できるデータ型は BLOB 型，又は BINARY 型だけです。AS 句で埋込み変数のデータ型を指定してください。このとき，埋込み変数で与えるデータの実長（位置付け子の場合は位置付け子に割り当てられたデータの実長）が AS 句で指定したデータ型の最大長より大きい場合はエラーとなります。

？パラメタ AS データ型

指定できるデータ型は BLOB 型，又は BINARY 型だけです。AS 句で？パラメタのデータ型を指定してください。このとき，？パラメタに与えるデータの実長（位置付け子の場合は位置付け子に割り当てられたデータの実長）が AS 句で指定したデータ型の最大長より大きい場合はエラーとなります。

LENGTH（値式）

SUBSTR（値式 1，値式 2，〔値式 3〕）

POSITION（値式 1 IN 値式 2 〔FROM 値式 3〕）

スカラ関数 LENGTH の値式，スカラ関数 SUBSTR の値式 1，及びスカラ関数 POSITION の値式 2 に指定できるデータ型は BLOB 型，又は BINARY 型だけです。指定できるものは埋込み変数，？パラメタだけです。そのほかの指定方法は各スカラ関数の規則に従ってください。

(5) 共通規則

1. 代入先のデータ型は，代入値のデータ型と互換性がなければなりません。
2. 代入先と代入値のデータ型が異なる場合には型変換され，同じ場合にはそのまま代入値が代入先に代入されます。
3. 代入値に指定する値式中に，以下のものは指定できません。
   - 列指定
   - コンポネント指定
   - スカラ副問合せ

(6) 留意事項

1. 代入文の形式 2 は SQL ルーチン中には指定できません。SQL ルーチン中に代入文を指定する場合は，代入文の形式 1 を指定してください。

(7) 使用例

BLOB 位置付け子の埋込み変数（XLOC）に割り当てられた BLOB データの一部を BLOB 型の埋込み変数（XDATA）に代入します。

```
SET :XDATA = SUBSTR(:XLOC AS BLOB(1M), 100, 1024)
```

5 制御系 SQL

この章では，制御系 SQL の構文形式，及び文法について説明します。

5.1　全般規定

5.1.1　制御系 SQL の全般規定

(1)　制御系 SQL の種類と機能

制御系 SQL は，HiRDB との接続・切り離し，及び表の排他制御に使用する SQL です。

制御系 SQL の種類と機能を次の表に示します。

表 5-1　制御系 SQL の種類と機能

種　類	機　能
CALL COMMAND 文（コマンド又はユティリティの実行）	HiRDB のコマンド・ユティリティを実行します。
COMMIT 文（トランザクションの正常終了）	現在のトランザクションを正常終了させて，同期点を設定し 1 コミットメント単位を生成します。そのトランザクションが更新したデータベースの内容を有効にします。
CONNECT 文（HiRDB との接続）	HiRDB に認可識別子，及びパスワードを連絡して，UAP が HiRDB を使用できる状態にします。
DISCONNECT 文（HiRDB との切り離し）	現在のトランザクションを正常終了させて，同期点を設定し 1 コミットメント単位を生成します。その後，UAP を HiRDB から切り離します。
LOCK 文（表の排他制御）	指定した表に排他制御をします。
ROLLBACK 文（トランザクションの取り消し）	現在のトランザクションを取り消して，そのトランザクション内でのデータベースの更新を無効にします。
SET SESSION AUTHORIZATION 文（実行ユーザの変更）	HiRDB に認可識別子，及びパスワードを連絡して，接続中のユーザを変更します。

(2)　制御系 SQL 指定時の留意事項

トランザクションは，論理的な仕事の単位であり，回復や同時実行の基本単位です。同時実行性向上のためには，一つのトランザクションの期間をできるだけ短くすることが望ましいです。特に大量データの更新をする場合には，排他期間，排他リソース数，ログ量などに留意する必要があります。そして，場合によってはトランザクションを分割することも必要です。

(3)　OLTP 下の X/Open に従った UAP での留意事項

OLTP 下の X/Open に従った UAP では，次に示す SQL 文は使用できません。

- COMMIT 文

- CONNECT 文
- DISCONNECT 文
- ROLLBACK 文

5.2 CALL COMMAND 文（コマンド又はユティリティの実行）

5.2.1 CALL COMMAND 文の形式と規則

(1) 機能

HiRDB のコマンド又はユティリティを実行し，実行結果（標準出力，標準エラー出力，戻り値）を取得できます。

(2) 使用権限

システム共通定義の pd_sql_command_exec_users オペランドで指定されている認可識別子を持つユーザ

(3) 形式

```
  CALL COMMAND {：埋込み変数１｜？パラメタ１｜定数１}
        〔WITH {：埋込み変数２｜？パラメタ２｜定数２}〔，{：埋込み変数２｜？パラメタ２｜定
数２}〕…〕
        〔INPUT {：埋込み変数３｜？パラメタ３｜定数３}〕
        〔OUTPUT TO {：埋込み変数４　：標識変数１｜？パラメタ４}〕
        〔ERROR TO {：埋込み変数５　：標識変数２｜？パラメタ５}〕
        〔RETURN CODE TO {：埋込み変数６｜？パラメタ６}〕
        〔ENVIRONMENT {：埋込み変数７｜？パラメタ７｜定数４}〕
        〔SERVER {：埋込み変数８｜？パラメタ８｜定数５}〕
```

(4) オペランド

(a) ：埋込み変数１｜？パラメタ１｜定数１

実行するコマンドやユティリティの名称を格納した埋込み変数，？パラメタ又は定数を指定します。コマンド又はユティリティの名称に，相対パスや絶対パスは指定できません。埋込み変数及び？パラメタのデータ型は，最大長が 30 バイト以下の可変長文字列にしてください。ただし，埋込み変数及び？パラメタに文字集合名 UTF16 を指定した場合，最大長が 60 バイト以下の可変長文字列になります。定数の長さは，30 バイト以下にしてください。

(b) WITH {：埋込み変数２｜？パラメタ２｜定数２}〔，{：埋込み変数２｜？パラメタ２｜定数２}〕…

コマンドやユティリティに渡す引数を格納した埋込み変数，？パラメタ又は定数を指定します。引数の並びが長いために，一つの埋込み変数，？パラメタ又は定数で指定できない場合は，複数の埋込み変数，？パラメタ又は定数を用いて指定してください。この場合は，指定した順序で文字列を結合します。複数の引数を与えたい場合は，セミコロンで区切って記述してください。セミコロン自身を引数に指定する場合は，2 個の連続するセミコロンを指定してください。埋込み変数及び？パラメタのデータ型は，最大長が

32,000 バイト以下の可変長文字列にしてください。定数の長さは，32,000 バイト以下にしてください。引数の中にパスを含む場合は，絶対パスで指定してください。

(c) INPUT {：埋込み変数３｜？パラメタ３｜定数３}

実行するコマンドやユティリティに渡す標準入力の内容を格納した埋込み変数，？パラメタ又は定数を指定します。埋込み変数及び？パラメタのデータ型は，最大長が 32,000 バイト以下の可変長文字列にしてください。定数の長さは，32,000 バイト以下にしてください。パスワードの入力を要求するコマンドやユティリティに対して，INPUT 句を用いてパスワードを与えることはできません。

(d) OUTPUT TO {：埋込み変数４　：標識変数１｜？パラメタ４}

実行したコマンドやユティリティの標準出力の内容を格納する埋込み変数又は？パラメタを指定します。埋込み変数及び？パラメタのデータ型は，最大長が 2 ギガバイト以下のバイナリデータ列（BLOB）です。標準出力が埋込み変数又は？パラメタの最大長を超える場合は，出力の先頭から最大長までの情報が格納され，それ以降の情報は切り捨てられます。この場合，標識変数には出力を切り捨てたことを示す標準出力の長さが格納されます。埋込み変数を使用する場合は，必ず標識変数を指定してください。

(e) ERROR TO {：埋込み変数５　：標識変数２｜？パラメタ５}

実行したコマンドやユティリティの標準エラー出力の内容を格納する埋込み変数又は？パラメタを指定します。埋込み変数及び？パラメタのデータ型は，最大長が 2 ギガバイト以下のバイナリデータ列（BLOB）です。標準エラー出力が埋込み変数又は？パラメタの最大長を超える場合は，出力の先頭から最大長までの情報が格納され，それ以降の情報は切り捨てられます。この場合，標識変数には出力を切り捨てたことを示す標準エラー出力の長さが格納されます。埋込み変数を使用する場合は，必ず標識変数を指定してください。

(f) RETURN CODE TO {：埋込み変数６｜？パラメタ６}

実行したコマンドやユティリティの戻り値を格納する埋込み変数又は？パラメタを指定します。埋込み変数及び？パラメタのデータ型は，整数型です。

(g) ENVIRONMENT {：埋込み変数７｜？パラメタ７｜定数４}

コマンドやユティリティを実行する時のクライアント環境定義を格納した埋込み変数，？パラメタ又は定数を指定します。埋込み変数及び？パラメタのデータ型は，最大長が 32,000 バイト以下の可変長文字列にしてください。定数の長さは，32,000 バイト以下にしてください。

クライアント環境定義は，「クライアント環境定義名＝値」の形式で指定してください。複数のクライアント環境定義を指定したい場合は，セミコロンで区切って記述してください。セミコロン自身を引数に指定する場合は，2 個の連続するセミコロンを指定してください。

指定できるクライアント環境定義の詳細は，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

なお，ENVIRONMENT で指定できない環境変数を変更したい場合は，システム共通定義に putenv 形式で設定するか，又は HiRDB 管理者の環境変数を変更してください。

(h) SERVER {：埋込み変数８｜？パラメタ８｜定数５}

コマンドやユティリティを実行するサーバ名を格納した埋込み変数，？パラメタ又は定数を指定します。サーバ名とは，システム共通定義の pdstart オペランドの-s オプションで指定されているサーバ名のことです。埋込み変数及び？パラメタのデータ型は，最大長が８バイト以下の可変長文字列にしてください。ただし，埋込み変数及び？パラメタに文字集合名 UTF16 を指定した場合，最大長が 16 バイト以下の可変長文字列になります。定数の長さは，８バイト以下にしてください。HiRDB/パラレルサーバでシステムマネジャユニットを指定する場合は，MGR を指定してください。SERVER 句が省略された場合，HiRDB/パラレルサーバでは MGR，HiRDB/シングルサーバでは SDS のサーバ名が仮定されます。

(5) 共通規則

1. 実行したコマンドやユティリティによってトランザクションが発生する場合，SQL を実行したトランザクションとは別のものになります。
2. SQL を実行したトランザクションと，実行したコマンドやユティリティによって発生したトランザクションとの間に，排他待ち又はデッドロックが発生する可能性があります。
3. 応答を要求するコマンドやユティリティを実行する場合，そのコマンドやユティリティを正しく終了させることができない標準入力の内容を与えると，システムが無応答になる場合がありますので注意してください。

   （例）

   HiRDB ファイルシステム（/hirdb/ios/db0）のバックアップ（pdfbkup）を実行する場合

   コマンドライン上で pdfbkup コマンドを実行すると，バックアップの続行又は終了を確認するため，「G」又は「T」の入力と，改行（CR）の入力が要求されます。

```
% pdfbkup /hirdb/ios/db0 /hirdb/ios/db0.backup
1605756 19:27:29 SQA2          KFPI21514-Q HiRDB file system area
/hirdb/ios/db0 backup to /hirdb/ios/db0.backup. [G:continue, T:terminate]
```

   次のような CALL COMMAND 文で，改行を含まない標準入力を pdfbkup コマンドに渡すと，システムは無応答になってしまいます。

```
EXEC SQL CALL COMMAND ’ pdfbkup’  WITH ’ /hirdb/ios/db0;/hirdb/ios/db0.backup’  INPUT
’ G’ ;
```

   pkfbkupコマンドを正しく実行させるためには，INPUT句には下記のように改行を含めた標準入力を指定してください。

```
EXEC SQL BEGIN DECLARE SECTION;
        char input_data[10];
EXEC SQL END DECLARE SECTION;
sprintf(input_data, ” G¥n” );
EXEC SQL CALL COMMAND ’ pdfbkup’  WITH ’ /hirdb/ios/db0;
/hirdb/ios/db0.backup’  INPUT :input_data;
```

   システムが無反応になった場合は，HiRDB のプロセス（HiRDB/パラレルサーバの場合は FES，HiRDB/シングルサーバの場合は SDS），及びコマンドやユティリティのプロセスを pdcancel コマンドで終了させてください。

(6) 注意事項

1. コマンドやユティリティのタイムアウトが必要な場合には，クライアント環境定義のPDCALCMDWAITTIME オペランドを指定してください。コマンド又はユティリティの実行中にタイムアウトした場合，制御用コマンド（pdcmdexec）のプロセス，及び実行中のコマンド又はユティリティのプロセスを pdkill コマンド（UNIX 版の場合は OS の kill コマンド）で終了してください。
2. 実行するプラットフォームの制限によって，コマンドやユティリティの実行に失敗する場合があります。次の表に，コマンド又はユティリティを実行できなかった場合に取得する情報と情報の格納先を示します。

表 5-2　コマンド又はユティリティを実行できなかった場合に取得する情報と情報の格納先

取得する情報	取得する情報の格納先
• エラーが発生したシステム関数名※ • エラーコード※	埋込み変数５又は？パラメタ５
OS のシステム関数の終了コード	埋込み変数６又は？パラメタ６

注※

エラーが発生したシステム関数及びエラーコードは，func=aa....aa,errno=bb....bb の形式で出力されます（errno は，エラーの状態を表す外部整数変数です）。

aa....aa：エラーが発生したシステム関数

bb....bb：エラーコード（エラーコードを取得できない場合，何も格納しません）

対策

エラーコードを調査し，errno.h 又はユーザが使用する OS のマニュアルを参照して，エラーの原因を取り除き，再度実行してください。

OS のシステム関数の実行に失敗する原因の一つとして，コマンドやユティリティに指定した引数が，OS で定義されているコマンドライン引数長の限界値を超えたことが考えられます。この問題を解決するためには，OS をチューニングしてコマンドライン引数長の限界値を大きくしてください。

(7) 使用例

1. HiRDB システムの状態表示（pdfls）を使用して，HiRDB ファイルシステム領域（/hirdb/ios/rdfiles）内に存在する HiRDB ファイルの一覧を取得します。実行結果は OUTPUT TO 句に指定した埋込み変数（output）に代入されます。コマンドライン上から‘pdfls -H /hirdb/ios/rdfiles’を実行する場合と同じ結果が得られます。

```
EXEC SQL BEGIN DECLARE SECTION;
        SQL TYPE IS BLOB(1M) OUTPUT;
EXEC SQL END DECLARE SECTION;
EXEC SQL CALL COMMAND ' pdfls' WITH ' -H;/hirdb/ios/rdfiles' OUTPUT TO :OUTPUT;
```

2. RD エリアの閉塞コマンド（pdhold）を使用して，RD エリア（RU01，RU02 及び RU03）を閉塞します。実行結果は OUTPUT TO 句に指定した埋込み変数（output）に代入されます。コマンドライン上から‘pdhold -r RU01,RU02,RU03’を実行する場合と同じ結果が得られます。

```
EXEC SQL BEGIN DECLARE SECTION;
        SQL TYPE IS BLOB(1M) OUTPUT;
EXEC SQL END DECLARE SECTION;
EXEC SQL CALL COMMAND ’ pdhold’  WITH ’ -r;RU01,RU02,RU03’  OUTPUT TO :OUTPUT;
```

WITH 句に複数の定数を指定したい場合は，次のように記述することもできます。

```
EXEC SQL BEGIN DECLARE SECTION;
        SQL TYPE IS BLOB(1M) OUTPUT;
EXEC SQL END DECLARE SECTION;
EXEC SQL CALL COMMAND ’ pdhold’  WITH ’ -r;RU01’,’ ,RU02,RU03’  OUTPUT TO :OUTPUT;
```

WITH 句に定数及び埋込み変数を指定したい場合は，次のように記述することもできます。

```
EXEC SQL BEGIN DECLARE SECTION;
        SQL TYPE IS BLOB(1M) OUTPUT;
        char RDAREAS[100];
EXEC SQL END DECLARE SECTION;
sprintf(RDAREAS, ” RU01,RU02,RU03” );
EXEC SQL CALL COMMAND ’ pdhold’  WITH ’ -r;’ ,:RDAREAS OUTPUT TO :OUTPUT;
```

5.3 COMMIT 文（トランザクションの正常終了）

5.3.1 COMMIT 文の形式と規則

(1) 機能

現在のトランザクションを正常終了させ，同期点を設定し 1 コミットメント単位を生成します。そのトランザクションが更新したデータベースの内容を有効にします。

(2) 使用権限

なし。

(3) 形式

```
COMMIT 〔WORK〕〔RELEASE〕
```

(4) オペランド

(a) WORK

指定しても，指定しなくてもトランザクションの正常終了の機能は変わりません。JIS 規格と互換のためサポートしています。

(b) RELEASE

トランザクションを正常終了させた後に UAP を HiRDB から切り離す場合に指定します。

(5) 共通規則

1. COMMIT 文を実行すると，ホールダブルカーソル以外の開いているカーソルはすべて閉じられます。
2. ホールダブルカーソルで使用している排他資源以外は解放されます。
3. COMMIT 文を実行すると，有効な位置付け子はすべて無効となります。
4. RELEASE を省略した COMMIT 文を実行すると，ON COMMIT DELETE ROWS を指定して定義した一時表の行データは削除されます。RELEASE を指定した COMMIT 文を実行すると，ON COMMIT の指定に関係なく，一時表の行データは削除されます。

(6) 留意事項

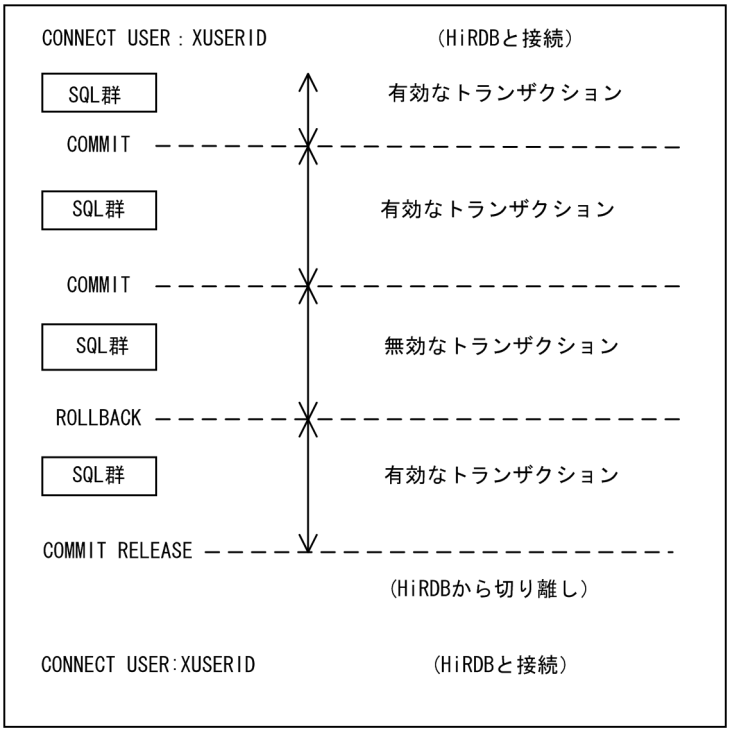
1. COMMIT 文は，OLTP 下の X/Open に従った UAP から指定できません。また，OLTP 下の UAP から手続きを呼び出す場合，COMMIT 文を使用した手続きは実行できません。

2. 手続き中から COMMIT 文を実行する場合，RELEASE オペランドは指定できません。

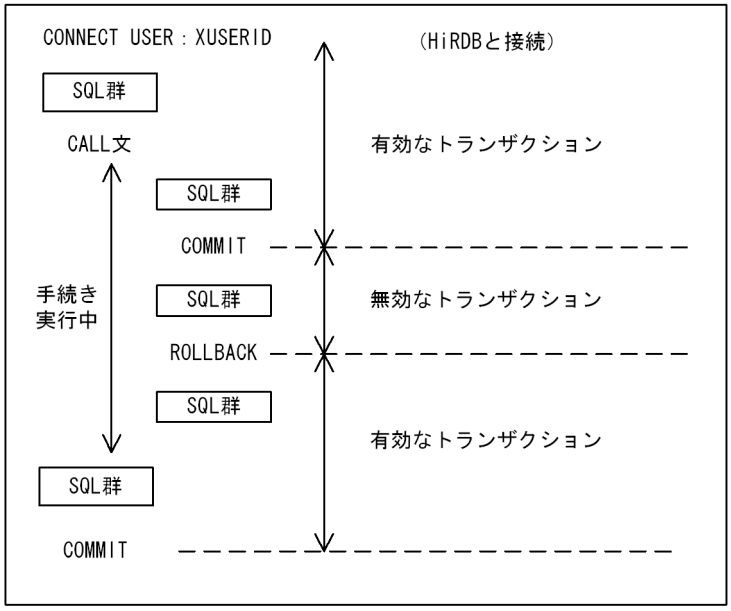
3. COMMIT 文は，トリガの動作中は実行できません。

(7) 使用例

1. COMMIT 文を使用してトランザクションを終了させ，再び HiRDB と接続するために RELEASE オペランド指定の COMMIT 文を使用します。



2. 手続き内，又は手続き終了後に COMMIT 文を使用してトランザクションを終了させます。



5.4 CONNECT 文（HiRDB との接続）

5.4.1 CONNECT 文の形式と規則

(1) 機能

HiRDB に認可識別子，及びパスワードを連絡して，UAP が HiRDB を使用できる状態にします。これを HiRDB との接続といいます。

(2) 使用権限

DBA 権限又は CONNECT 権限を持つユーザ

HiRDB との接続ができます。

(3) 形式

```
CONNECT 〔 {USER   ：埋込み変数1 〔USING   ：埋込み変数2〕
              | ：埋込み変数1  〔IDENTIFIED BY  ：埋込み変数2〕 } 〕
```

(4) オペランド

(a) 埋込み変数 1

認可識別子を値として持つ埋込み変数を指定します。

認可識別子の大文字，小文字を区別して扱う場合，認可識別子を引用符（"）で囲んだ文字列を埋込み変数に設定してください。

UAP の記述言語が C 言語の場合，33 バイト以下の固定長文字列のデータ型で，文字列はナル文字で終わらせてください。ナル文字で終わらないと，（エリア長－1）の文字列を認可識別子として使用します。ただし，埋め込み変数としてポインタを使用する場合は，文字列を必ずナル文字で終わらせてください。

UAP の記述言語が COBOL の場合，このオペランドは 32 バイト以下の固定長文字列のデータ型で，認可識別子がエリア長より短い場合は右端に空白詰めしてください。文字列をナル文字で終わらせる必要はありません。

既定文字集合以外の文字集合は指定できません。

CONNECT だけを指定し，USER 以降のオペランドをすべて省略すると，環境変数 PDUSER に指定した認可識別子とパスワードを使用して，HiRDB と接続します。

大文字，小文字を区別して扱う場合，引用符（"）で囲んでください。

(b) 埋込み変数 2

パスワードを値として持つ埋込み変数を指定します。

パスワードの大文字，小文字を区別して扱う場合，パスワードを引用符（''）で囲んだ文字列を埋込み変数に設定してください。

UAP の記述言語が C 言語の場合，33 バイト以下の固定長文字列のデータ型で，文字列はナル文字で終わらせてください。ナル文字で終わらないと，（エリア長-1）の文字列をパスワードとして使用します。

ただし，埋込み変数としてポインタを使用した場合は，最大エリア長として 31 バイトを仮定するため，指定できるパスワード長が引用符を含めて 30 バイトに制限されます。引用符を含めて 31 バイト以上のパスワードを指定する場合はポインタを使用しないでください。ポインタを使用する場合は，文字列をナル文字で終わらせてください。31 バイト目までにナル文字がないと，30 バイト目までの文字列をパスワードとして使用します。

なお，UAP の記述言語が COBOL，又は OOCOBOL の場合，このオペランドは 32 バイト以下の固定長文字列のデータ型で，パスワードがエリア長より短い場合は右端に空白詰めしてください。文字列をナル文字で終わらせる必要はありません。

既定文字集合以外の文字集合は指定できません。

パスワードのないユーザを使用する場合は，USING 句，IDENTIFIED BY 句を省略するか，埋込み変数に任意の文字列を指定してください。

(5) 共通規則

1. HiRDB と接続した場合は，DISCONNECT 文によって一度 HiRDB からの切り離しをしなければ，再び HiRDB と接続できません。

(6) 留意事項

1. CONNECT 文は，OLTP 下の X/Open に従った UAP から指定できません。

(7) 使用例

HiRDB を使用するユーザの認可識別子（埋込み変数 USER1），パスワード（埋込み変数 PSWD1）を連絡して HiRDB と接続します。

```
    CONNECT USER :USER1 USING :PSWD1
```

5.5 DISCONNECT 文（HiRDB との切り離し）

5.5.1 DISCONNECT 文の形式と規則

(1) 機能

現在のトランザクションを正常終了させ，同期点を設定し 1 コミットメント単位を生成します。その後，UAP を HiRDB から切り離します。

(2) 使用権限

なし。

(3) 形式

```
DISCONNECT
```

(4) 共通規則

1. DISCONNECT 文を実行すると，HiRDB が COMMIT 文を実行した後に，HiRDB から UAP を切り離します。このときホールダブルカーソルもすべて閉じられます。
2. DISCONNECT 文を実行しないで UAP が終了すると，その UAP のトランザクションを ROLLBACK します。
3. DISCONNECT 文を実行すると，一時表の行データは削除されます。

(5) 留意事項

1. DISCONNECT 文は，OLTP 下の X/Open に従った UAP から指定できません。

(6) 使用例

UAP を HiRDB から切り離します。

```
DISCONNECT
```

5.6 LOCK 文（表の排他制御）

5.6.1 LOCK 文の形式と規則

(1) 機能

指定した表に排他制御をします。HiRDB が自動的に排他制御する行，又はキー値単位より大きな単位で排他制御をするため，行，又はキー値単位の場合に比べて排他制御によるオーバヘッドを削減できます。

(2) 使用権限

表に対する SELECT 権限を持つユーザが，その表に共用モードの排他を掛けられます。

表に対する INSERT 権限，UPDATE 権限，又は DELETE 権限を持つユーザが，その表に排他モードの排他を掛けられます。

(3) 形式

```
LOCK TABLE 〔認可識別子.〕表識別子
              〔, 〔認可識別子.〕表識別子〕…
              〔IN {SHARE | EXCLUSIVE }  MODE〕
              〔UNTIL DISCONNECT〕
              〔 {WITH ROLLBACK | NO WAIT | NOWAIT} 〕
```

(4) オペランド

(a) 〔認可識別子.〕表識別子〔, 〔認可識別子.〕表識別子〕…

認可識別子

排他制御をする表の所有者の認可識別子を指定します。

表識別子

排他制御をする表の名称を指定します。

表識別子にビュー表を指定した場合は，ビュー表の基になる実表に対して排他制御をします。この場合，ビュー表には排他制御はしません。

表識別子に一時表を指定した場合，指定は無視されます。

表名は最大 128 個指定できます。また，表名は重複してもかまいません。

(b) 〔IN SHARE MODE〕

排他制御をする表のデータを，ほかのユーザが参照することは許すが，更新することは許さない（共用モード）場合に指定します。このオペランドを指定した LOCK 文発行後の表アクセス時に，共用モードの行，及びキー値の排他資源が削減されます。

(c) 〔IN EXCLUSIVE MODE〕

排他制御をする表のデータを，ほかのユーザが参照することも，更新することも許さない（排他モード）場合に指定します。ただし，ほかのユーザが排他オプション WITHOUT LOCK NOWAIT を指定して検索した場合を除きます。このオペランドを指定した LOCK 文発行後の表アクセス時に，共用モードの行，キー値，及び排他モードの行，キー値の排他資源が削減されます。

(d) 〔UNTIL DISCONNECT〕

DISCONNECT するまで排他制御をする場合に指定します。省略した場合，トランザクションの終了まで排他制御をします。

(e) {WITH ROLLBACK | NO WAIT | NOWAIT}

ほかのユーザと排他が競合した場合，待たないでエラーを受け取るときに指定します。省略した場合，排他が競合したときは，排他が解除されるまで，又はシステム定義の pd_lck_wait_timeout オペランドで指定した時間まで待ちます。

WITH ROLLBACK

排他制御をする表をほかのユーザが使用している場合，トランザクションを取り消して無効にするときに指定します。

NO WAIT

排他制御をする表が他ユーザで使用されているとき，トランザクションを取り消さないでこの SQL 文を無効にする場合，指定します。

NOWAIT

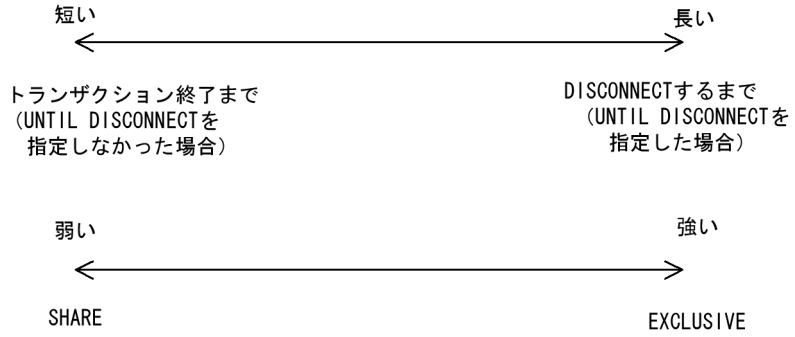
NO WAIT を指定した場合と同じです。

(5) 共通規則

1. 排他制御をする実表が重複する場合は，重複排除されます。一つの LOCK 文で排他制御できる実表の合計は，最大 128 個です。
2. データディクショナリ表には，排他制御はできません。
3. 排他制御をしているとき，更に LOCK 文を実行した場合，排他期間は長い方へ，又排他レベルは強い方へ遷移します。このため，排他期間を短くする，又は排他レベルを下げる指定をした LOCK 文を実行しても，排他期間及び排他レベルは変更されません。

   排他期間の長さと排他レベルの強弱を次の図に示します。

図 5-1　排他期間の長さと排他レベルの強弱



4. LOCK 文発行後に PURGE TABLE 文を実行すると，排他レベルが EXCLUSIVE に遷移します。この場合，PURGE TABLE 文実行前にトランザクションを終了させると，PURGE TABLE 文を実行しても EXCLUSIVE には遷移しません。
5. UNTIL DISCONNECT を指定して排他制御をしている表が，DROP TABLE 文などで削除された場合，その表に対する排他制御は HiRDB システムによって自動的に解除されます。

(6)　留意事項

1. 通常，HiRDB が排他制御をするので，排他制御の単位を変える場合だけ LOCK 文を使用してください。
2. UNTIL DISCONNECT を指定した場合，クライアント環境定義の PDXAMODE の指定によって，排他制御の期間が異なります。PDXAMODE 及び排他制御については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。
3. 共用表に対して排他モード（IN EXCLUSIVE MODE）で LOCK 文を実行した場合，共用表のほかに次の RD エリアも排他制御の対象となります。
   - 共用表を格納している RD エリア
   - 共用表にインデクスが定義されている場合，そのインデクスを格納している RD エリア

   共用表に対する排他制御については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(7)　使用例

同じトランザクション中で，在庫表（ZAIKO）を全件検索するため，この表に共用モードの排他制御をします。

```
    LOCK TABLE ZAIKO IN SHARE MODE
```

5.7 ROLLBACK 文（トランザクションの取り消し）

5.7.1 ROLLBACK 文の形式と規則

(1) 機能

現在のトランザクションを取り消して，そのトランザクション内でのデータベースの更新を無効にします。

(2) 使用権限

なし。

(3) 形式

```
ROLLBACK 〔WORK〕〔RELEASE〕
```

(4) オペランド

(a) WORK

指定しても，指定しなくてもトランザクションの取り消しの機能は変わりません。JIS 規格と互換のためサポートしています。

(b) RELEASE

トランザクションを取り消して，データベースの更新を無効にした後に，UAP を HiRDB から切り離す場合に指定します。

(5) 共通規則

1. ROLLBACK 文を実行すると，開いているカーソルはすべて閉じられます。
2. 現行トランザクション中での排他制御は解除されます。ただし，UNTIL DISCONNECT 指定の LOCK TABLE 文による排他制御は解除されません。
3. 定義系 SQL は，ロールバックの対象にはなりません。
4. ホールダブルカーソル指定の動的 SELECT 文の前処理については，現行トランザクション中に前処理された場合は無効とし，それ以外は有効とします。
5. ROLLBACK 文を実行すると，有効な位置付け子はすべて無効となります。
6. RELEASE を省略した ROLLBACK 文を実行すると，ON COMMIT DELETE ROWS を指定して定義した一時表の行データは削除されます。RELEASE を指定した ROLLBACK 文を実行すると，ON COMMIT の指定に関係なく，一時表の行データは削除されます。

(6) 留意事項

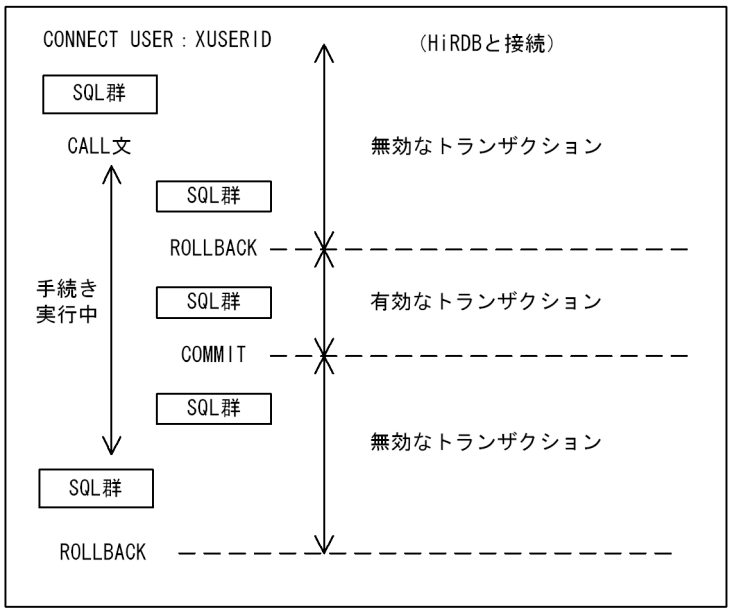
1. ROLLBACK 文は，OLTP 下の X/Open に従った UAP から指定できません。また，OLTP 下の UAP から手続きを呼び出す場合，ROLLBACK 文を使用した手続きは実行できません。
2. 手続き中から ROLLBACK 文を実行する場合，RELEASE オペランドは指定できません。
3. ROLLBACK 文は，トリガの動作中は実行できません。

(7) 使用例

1. 現在のトランザクションを取り消します。

```
ROLLBACK
```

2. 手続き内，又は手続き終了後にトランザクションを取り消します。



5.8 SET SESSION AUTHORIZATION 文（実行ユーザの変更）

5.8.1 SET SESSION AUTHORIZATION 文の形式と規則

(1) 機能

HiRDB に認可識別子，及びパスワードを連絡して，接続中のユーザを変更します。

(2) 使用権限

DBA 権限，又は CONNECT 権限を持つユーザ

接続中のユーザを変更できます。

(3) 形式

```
SET SESSION AUTHORIZATION {:埋込み変数1|？パラメタ1}
      〔 { USING | IDENTIFIED BY } {:埋込み変数2|？パラメタ2 }〕
```

(4) オペランド

(a) 埋込み変数 1 ｜？パラメタ 1

認可識別子を値として持つ埋込み変数，又は？パラメタを指定します。

認可識別子の大文字，小文字を区別して扱う場合，認可識別子を引用符（"）で囲んだ文字列を埋込み変数又は？パラメタに設定してください。

UAP の記述言語が C 言語の場合，33 バイト以下の固定長文字列のデータ型で，文字列はナル文字で終わらせてください。ナル文字で終わらないと，(エリア長－1)の文字列を認可識別子として使用します。ただし，埋め込み変数としてポインタを使用する場合は，文字列を必ずナル文字で終わらせてください。

UAP の記述言語が COBOL の場合，32 バイト以下の固定長文字列のデータ型で指定してください。認可識別子がエリア長より短い場合は文字列の右端に空白詰めしてください。文字列をナル文字で終わらせる必要はありません。

埋込み変数 1 には既定文字集合以外の文字集合は指定できません。？パラメタ 1 に文字集合名 UTF16 を指定した場合，64 バイト以下の固定長文字列のデータ型で指定してください。

(b) 埋込み変数 2 ｜？パラメタ 2

パスワードを値として持つ埋込み変数，又は？パラメタを指定します。

パスワードの大文字，小文字を区別して扱う場合，パスワードを引用符（''）で囲んだ文字列を埋込み変数又は？パラメタに設定してください。

UAP の記述言語がC言語の場合，33 バイト以下の固定長文字列のデータ型で，文字列はナル文字で終わらせてください。ナル文字で終わらないと，(エリア長−1)の文字列をパスワードとして使用します。

ただし，埋込み変数としてポインタを使用した場合は，エリア長として 31 バイトを仮定するため，指定できるパスワード長が引用符を含めて 30 バイトに制限されます。引用符を含めて 31 バイト以上のパスワードを指定する場合はポインタを使用しないでください。ポインタを使用する場合は，文字列をナル文字で終わらせてください。31 バイト目までにナル文字がないと，30 バイト目までの文字列をパスワードとして使用します。

UAP の記述言語が COBOL の場合，32 バイト以下の固定長文字列のデータ型で指定してください。パスワードがエリア長より短い場合は文字列の右端に空白詰めしてください。文字列をナル文字で終わらせる必要はありません。

埋込み変数 2 には既定文字集合以外の文字集合は指定できません。？パラメタ 2 に文字集合名 UTF16 を指定した場合，64 バイト以下の固定長文字列のデータ型で指定してください。

(5) 共通規則

1. トランザクションの開始前，又はトランザクション内で最初に実行しなければユーザを変更できません。
2. PREPARE 文で前処理した結果はすべて無効になります。
3. ホールダブルカーソルが開いている場合には，ユーザを変更できません。
4. LOCK TABLE 文で UNTIL DISCONNECT 指定の排他を掛けている場合は，ユーザを変更できません。
5. OLTP 下の X/Open に従った UAP で SET SESSION AUTHORIZATION を指定する場合は，tx_begin 又は xa_begin 関数コール直後に実行するように指定します。
6. 動的 SQL の SET SESSION AUTHORIZATION を実行した場合，処理完了時に COMMIT されます。
7. ストアドプロシジャ中では実行できません。
8. SET SESSION AUTHORIZATION 文を実行すると，一時表の行データは削除されます。
9. EXECUTE 文を実行するとき，？パラメタ 1 に設定する変数のデータ長は，最大 32 まで指定できますが，PREPARE 文で前処理した SET SESSION AUTHORIZATION 文に対して DESCRIBE INPUT 文を実行すると，？パラメタ 1 のデータ長には 30 が設定されます。
10. EXECUTE 文を実行するとき，？パラメタ 2 に設定する変数のデータ長は，最大 32 まで指定できますが，PREPARE 文で前処理した SET SESSION AUTHORIZATION 文に対して DESCRIBE INPUT 文を実行すると，？パラメタ 2 のデータ長には 30 が設定されます。

6 埋込み言語文法

この章では，埋込み言語の構文形式，及び文法について説明します。

6.1　全般規則

6.1.1　埋込み言語の全般規定

(1)　埋込み言語の種類と機能

埋込み言語は，埋込み型の UAP を作成する場合にプログラム用 SQL と一緒に使用して，埋込み変数の宣言，及びリターンコードによる処理の宣言をする SQL です。

埋込み言語の種類と機能を次の表に示します。

表 6-1　埋込み言語の種類と機能

種　類	機　能
BEGIN DECLARE SECTION（埋込み SQL 開始宣言）	埋込み変数宣言節の始まりを示します。埋込み変数宣言節には SQL 中で使用する埋込み変数，及び標識変数を指定します。
END DECLARE SECTION（埋込み SQL 終了宣言）	埋込み変数宣言節の終わりを示します。
ALLOCATE CONNECTION HANDLE（接続ハンドルの割り当て）	複数接続機能を使用した環境で，UAP が使用する接続ハンドルを割り当てます。
FREE CONNECTION HANDLE（接続ハンドルの解放）	ALLOCATE CONNECTION HANDLE で割り当てた接続ハンドルを解放します。
DECLARE CONNECTION HANDLE SET（使用する接続ハンドルの宣言）	複数接続機能を使用した環境で，UAP 中の SQL が使用する接続ハンドルを宣言します。
DECLARE CONNECTION HANDLE UNSET（使用する接続ハンドルの全解除）	この文以前に DECLARE CONNECTION HANDLE SET で指定した接続ハンドルの，使用の宣言をすべて解除します。
GET CONNECTION HANDLE（接続ハンドル取得）	X/Open XA インタフェース環境下で複数接続機能を使用する場合，UAP が使用する接続ハンドルを割り当てます。
COPY（登録集原文の引き込み）	登録集原文をソースプログラム中に引き込みます。
GET DIAGNOSTICS（診断情報取得）	直前に実行した SQL 文が CREATE PROCEDURE，CREATE FUNCTION，CREATE TYPE，ALTER PROCEDURE，ALTER ROUTINE，ALTER TRIGGER，CREATE TRIGGER，CALL 文，又は WITH 句指定のある動的 SELECT 文，若しくはカーソル宣言の場合に，そのエラー情報，及び診断情報を診断領域から取得します。
COMMAND EXECUTE（UAP からのコマンド実行）	UAP 中から，HiRDB のコマンド，及び OS のコマンドを実行します。
SQL 先頭子	SQL の始まりを示します。
SQL 終了子	SQL の終わりを示します。
WHENEVER（埋込み例外宣言）	SQL の実行後に HiRDB が SQL 連絡領域に設定したリターンコードによって，UAP の処理を宣言します。

種　類	機　能
SQLCODE 変数	SQL の実行後に HiRDB から返されるリターンコードを受け取ります。
SQLSTATE 変数	SQL の実行後に HiRDB から返されるリターンコードを受け取ります。
PDCNCTHDL 型変数の宣言	複数接続機能を使用した環境で，使用する接続情報を持つハンドルを宣言します。
INSTALL JAR	HiRDB サーバに JAR ファイルを登録します。
REPLACE JAR	HiRDB サーバに JAR ファイルを再登録します。
REMOVE JAR	HiRDB サーバの JAR ファイルを削除します。
INSTALL CLIB	HiRDB サーバに C ライブラリファイルを新規登録します。
REPLACE CLIB	HiRDB サーバに C ライブラリファイルを再登録します。
REMOVE CLIB	HiRDB サーバの C ライブラリファイルを削除します。
DECLARE AUDIT INFO SET	HiRDB サーバにアクセスするアプリケーションのアカウント情報などユーザ任意接続情報を設定します。

6.2 BEGIN DECLARE SECTION（埋込み SQL 開始宣言）

6.2.1 BEGIN DECLARE SECTION の形式と規則

(1) 機能

埋込み SQL 宣言節の始まりを示します。埋込み SQL 宣言節には，SQL 中で使用する埋込み変数，及び標識変数を指定します。

(2) 形式

```
BEGIN DECLARE SECTION
```

(3) 共通規則

1. 埋込み SQL 宣言節の終わりは，END DECLARE SECTION（埋込み SQL 終了宣言）を指定してください。
2. SQL 中で使用する埋込み変数，及び標識変数は，埋込み SQL 宣言節で宣言します。
3. 埋込み型の UAP には，0 個以上の埋込み SQL 宣言節を指定できます。
4. 埋込み SQL 宣言節には，変数の宣言だけ指定できます。ただし，変数の宣言を含まない埋込み SQL 宣言節は指定できます。

(4) 使用例

SQL 中で使用する埋込み変数を宣言します。

```
<C言語の場合>

  EXEC SQL BEGIN DECLARE SECTION;
      char  XSCODE[5];
      char  XSNAME[17];
      char  XCOL[3];
      long  XTANKA;
      long  XZSURYO;
  EXEC SQL END DECLARE SECTION;

<COBOL言語の場合>

      EXEC SQL
          BEGIN DECLARE SECTION
      END-EXEC.
  77 XSCODE      PIC X(4).
  77 XSNAME      PIC X(16).
  77 XCOL        PIC X(2).
  77 XTANKA      PIC S9(9) COMP.
```

```
 77 XZSURYO     PIC S9(9) COMP.
     EXEC SQL
         END DECLARE SECTION
     END-EXEC.
```

6.3 END DECLARE SECTION（埋込み SQL 終了宣言）

6.3.1 END DECLARE SECTION の形式と規則

(1) 機能

埋込み SQL 宣言節の終わりを示します。

(2) 形式

```
END DECLARE SECTION
```

(3) 共通規則

1. 埋込み SQL 宣言節の始まりは，BEGIN DECLARE SECTION（埋込み SQL 開始宣言）を指定してください。
2. SQL 中で使用する埋込み変数，及び標識変数は，埋込み SQL 宣言節で宣言します。

(4) 使用例

使用例については，「BEGIN DECLARE SECTION（埋込み SQL 開始宣言）」を参照してください。

6.4 ALLOCATE CONNECTION HANDLE（接続ハンドルの割り当て）

6.4.1 ALLOCATE CONNECTION HANDLE の形式と規則

(1) 機能

複数接続機能を使用した環境で，UAP が使用する接続ハンドルを割り当てます。

(2) 形式

```
ALLOCATE CONNECTION HANDLE ：PDCNCTHDL型変数,
                                ：リターンコード受け取り変数
                                〔, {：接続PDHOST変数, ：接続PDNAMEPORT変数
                                    |：環境変数グループ名変数}〕
```

(3) オペランド

(a) ：PDCNCTHDL 型変数

PDCNCTHDL 型として宣言した埋込み変数を指定します。

(b) ：リターンコード受け取り変数

INT 型として宣言した埋込み変数を指定します。

リターンコード受け取り変数に返される値を次に示します。

C 言語の場合

正常に割り当てられた場合：

p_rdb_RC_NORM

接続ハンドルの値が不正な場合：

p_rdb_RC_ERRPARM

メモリ不足が発生した場合：

p_rdb_RC_MEMERR

これらの値は，pdberrno.h に定義しています。

COBOL 言語の場合

正常に割り当てられた場合：

P-RDB-RC-NORM

接続ハンドルの値が不正な場合：

P-RDB-RC-ERRPARM

メモリ不足が発生した場合：

P-RDB-RC-MEMERR

これらの値は，PDBSQLCAMTH.CBL に定義しています。

(c) ：接続 PDHOST 変数

接続先のホスト名を CHARACTER 型（領域長 511 バイト）の埋込み変数で指定します。ホスト名はクライアント環境定義の PDHOST オペランドに指定する形式で指定してください。

既定文字集合以外の文字集合は指定できません。

(d) ：接続 PDNAMEPORT 変数

接続先のポート番号を SMALLINT 型の埋込み変数で指定します。ポート番号はクライアント環境定義の PDNAMEPORT オペランドに指定する形式で指定してください。

ただし，接続先のポート番号が 65535 の場合は，クライアント環境定義の PDHOST と PDNAMEPORT，又は環境変数グループ名変数を使用して，接続先を指定してください。

(e) ：環境変数グループ名変数

CHAR 型（領域長 256 バイト）として宣言した埋込み変数を指定します。

既定文字集合以外の文字集合は指定できません。

UNIX 環境では，環境変数を記述した通常ファイルのファイル名を絶対パス名（ナル文字を含めて 256 バイトまで）で指定します。

Windows 環境では，環境変数登録ツールで登録したグループ名（ナル文字を含めて 31 バイトまで），又は環境変数グループファイル名を絶対パス名（ナル文字を含めて 256 バイトまで）で指定します。「ドライブ名:¥」から始まっていた場合は，すべて環境変数グループファイル名指定とみなします。Windows 環境での環境変数グループファイルは Windows の ini ファイルの仕様に準じます。Windows 環境での環境変数グループファイル指定の場合，パス名に空白などを含むロングパス指定もできますが，パス名の前後を引用符（"）で囲まないでください。

環境変数グループについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4) 共通規則

1. ALLOCATE CONNECTION HANDLE は，CONNECT 文より先に発行してください。
2. 使用する埋込み変数は，埋込み SQL 宣言節で宣言してください。
3. 接続ハンドルの割り当てに失敗した場合，その障害コードをリターンコード受け取り変数に設定します。
4. 接続 PDHOST 変数及び接続 PDNAMEPORT 変数は，両方指定するか，又は両方省略するかのどちらかにしてください。省略した場合は，クライアント環境定義の指定値でデータベースとの接続処理をします。

5. 接続 PDHOST 変数に設定するホスト名の最後は，ナル文字にしてください。

(5) 留意事項

1. 割り当てた接続ハンドルは，DISCONNECT 文の発行時には解放されません。接続ハンドルを解放する場合は，FREE CONNECTION HANDLE を発行してください。
2. COBOL 言語の場合，接続 PDHOST 変数，及び環境変数グループ名変数に値を設定するときは，設定値の最後をナル文字にしてください。
3. 接続 PDHOST 変数と接続 PDNAMEPORT 変数を指定した場合でも，クライアント環境定義で高速接続又は FES ホストダイレクト接続を指定しているときは，クライアント環境定義で指定されている接続形態で HiRDB サーバに接続します。接続形態の選択方法については，マニュアル「HiRDB UAP 開発ガイド」の「HiRDB サーバと接続するための環境変数と接続形態との関係」を参照してください。

(6) 使用例

1. PDCNCTHDL 型変数の使用例を次に示します。

(C 言語の場合)

```
EXEC SQL BEGIN DECLARE SECTION;
  PDCNCTHDL   CnctHdl;
  long        AlchdlRtn;
EXEC SQL END DECLARE SECTION;
EXEC SQL ALLOCATE CONNECTION HANDLE :CnctHdl,
                                    :AlchdlRtn;
```

(COBOL 言語の場合)

```
DATA DIVISION.
WORKING-STORAGE SECTION.
  EXEC SQL
    BEGIN DECLARE SECTION
  END-EXEC.
01 CNCTHDL     SQL TYPE IS PDCNCTHDL.
01 ALCHDLRTN   PIC S9(9) COMP.
  EXEC SQL
    END DECLARE SECTION
  END-EXEC.
        :
PROCEDURE DIVISION.
        :
  EXEC SQL
    ALLOCATE CONNECTION HANDLE :CNCTHDL,
                               :ALCHDLRTN;
  END-EXEC.
```

2. 接続 PDHOST 変数，及び接続 PDNAMEPORT 変数の使用例を次に示します。

(C 言語の場合)

```
EXEC SQL BEGIN DECLARE SECTION;
  PDCNCTHDL   CnctHdl;
  long        AlchdlRtn;
```

```
  char        CnctHost[31];
  short       CnctPort;
EXEC SQL END DECLARE SECTION;
strcpy(CnctHost,"HOST01");
EXEC SQL ALLOCATE CONNECTION HANDLE :CnctHdl,
                                    :AlchdlRtn,
                                    :CnctHost,
                                    :CnctPort;
```

(COBOL 言語の場合)

```
DATA DIVISION.
WORKING-STORAGE SECTION.
  EXEC SQL
    BEGIN DECLARE SECTION
  END-EXEC.
01 CNCTHDL     SQL TYPE IS PDCNCTHDL.
01 ALCHDLRTN   PIC S9(9) COMP.
01 CNCTHOST    PIC X(31).
01 CNCTPORT    PIC S9(4) COMP.
  EXEC SQL
    END DECLARE SECTION
  END-EXEC.
        :
PROCEDURE DIVISION.
        :
  MOVE 'HOST01' & X'00' TO CNCTHOST.
  EXEC SQL
    ALLOCATE CONNECTION HANDLE :CNCTHDL,
                               :ALCHDLRTN,
                               :CNCTHOST,
                               :CNCTPORT;
  END-EXEC.
```

3. 環境変数グループ名の使用例を次に示します。

(C 言語の場合)

```
EXEC SQL BEGIN DECLARE SECTION;
  PDCNCTHDL   CnctHdl;
  long        AlchdlRtn;
  char        GroupName[31];
EXEC SQL END DECLARE SECTION;
strcpy(GroupName,"HRD01");
EXEC SQL ALLOCATE CONNECTION HANDLE :CnctHdl,
                                    :AlchdlRtn,
                                    :GroupName;
```

(COBOL 言語の場合)

```
DATA DIVISION.
WORKING-STORAGE SECTION.
  EXEC SQL
    BEGIN DECLARE SECTION
  END-EXEC.
01 CNCTHDL     SQL TYPE IS PDCNCTHDL.
01 ALCHDLRTN   PIC S9(9) COMP.
01 GROUPNAME   PIC X(31).
```

```
  EXEC SQL
    END DECLARE SECTION
  END-EXEC.
        :
PROCEDURE DIVISION.
        :
  MOVE 'HRD01' & X'00' TO GROUPNAME.
  EXEC SQL
    ALLOCATE CONNECTION HANDLE :CNCTHDL,
                               :ALCHDLRTN,
                               :GROUPNAME
  END-EXEC.
```

6.5 FREE CONNECTION HANDLE（接続ハンドルの解放）

6.5.1 FREE CONNECTION HANDLE の形式と規則

(1) 機能

ALLOCATE CONNECTION HANDLE で割り当てた接続ハンドルを解放します。

(2) 形式

```
FREE CONNECTION HANDLE ：PDCNCTHDL型変数,
                                    ：リターンコード受け取り変数
```

(3) オペランド

(a) ：PDCNCTHDL 型変数

ALLOCATE CONNECTION HANDLE で指定した PDCNCTHDL 型変数を指定します。

(b) ：リターンコード受け取り変数

ALLOCATE CONNECTION HANDLE で指定したリターンコード受け取り変数を指定します。

リターンコード受け取り変数に返される値を次に示します。

C 言語の場合

正常に割り当てられた場合：

p_rdb_RC_NORM

接続ハンドルの値が不正な場合：

p_rdb_RC_ERRPARM

接続ハンドルが現在使用中の場合：

p_rdb_RC_SIMERR

これらの値は，pdberrno.h に定義しています。

COBOL 言語の場合

正常に割り当てられた場合：

P-RDB-RC-NORM

接続ハンドルの値が不正な場合：

P-RDB-RC-ERRPARM

接続ハンドルが現在使用中の場合：

P-RDB-RC-SIMERR

これらの値は，SQLCAMTH.CBL に定義しています。

(4) 共通規則

1. FREE CONNECTION HANDLE は，DISCONNECT 文の後に発行してください。
2. 使用する埋込み変数は，埋込み SQL 宣言節で宣言してください。
3. 接続ハンドルの解放に失敗した場合，その障害コードをリターンコード受け取り変数に設定します。

(5) 留意事項

1. 予約した接続ハンドルは，DISCONNECT 文の発行時には解放されません。接続ハンドルを解放する場合は，FREE CONNECTION HANDLE を発行してください。
2. OLTP 下の X/Open に従った UAP では，GET CONNECTION HANDLE で取得した接続ハンドルは OLTP が解放します。このため，FREE CONNECTION HANDLE を発行して，接続ハンドルを解放しないでください。FREE CONNECTION HANDLE を発行して接続ハンドルを解放した場合，その後 UAP が不正な動作をして，異常終了するおそれがあります。

(6) 使用例

ALLOCATE CONNECTION HANDLE の例で割り当てた接続ハンドルを解放します。

(C 言語の場合)

```
EXEC SQL BEGIN DECLARE SECTION;
  PDCNCTHDL   CnctHdl;
  long        FrchdlRtn;
EXEC SQL END DECLARE SECTION;
EXEC SQL FREE CONNECTION HANDLE :CnctHdl,
                                :FrchdlRtn;
```

(COBOL 言語の場合)

```
DATA DIVISION.
WORKING-STORAGE SECTION.
  EXEC SQL
    BEGIN DECLARE SECTION
  END-EXEC.
01 CNCTHDL     SQL TYPE IS PDCNCTHDL.
01 FRCHDLRTN   PIC S9(9) COMP.
  EXEC SQL
    END DECLARE SECTION
  END-EXEC.
        :
PROCEDURE DIVISION.
        :
```

```
EXEC SQL
  FREE CONNECTION HANDLE :CNCTHDL,
                         :FRCHDLRTN;
END-EXEC.
```

6.6 DECLARE CONNECTION HANDLE SET（使用する接続ハンドルの宣言）

6.6.1 DECLARE CONNECTION HANDLE SET の形式と規則

(1) 機能

複数接続機能を使用した環境で，UAP 中の SQL，又は SQL 連絡領域が使用する接続ハンドルを宣言します。

(2) 形式

```
DECLARE CONNECTION HANDLE SET ：PDCNCTHDL型変数
```

(3) オペランド

(a) ：PDCNCTHDL 型変数

PDCNCTHDL 型として宣言した埋込み変数を指定します。

(4) 共通規則

1. DECLARE CONNECTION HANDLE SET で指定した接続ハンドルの有効範囲は，ソースプログラム上での出現位置によって決定します。ある DECLARE CONNECTION HANDLE SET で指定した接続ハンドルの変数は，ソースプログラム上の，次に出現する DECLARE CONNECTION HANDLE SET 又は DECLARE CONNECTION HANDLE UNSET までの，すべての SQL に対して有効となります。
2. 使用する接続ハンドルの宣言は，同じ UAP 上で複数回できます。

(5) 留意事項

1. C 言語の場合，DECLARE CONNECTION HANDLE SET の記述以前に発行した SQL 文に対しては，その SQL 文が単一接続環境で発行されたものとして処理します。

   COBOL 言語の場合，複数接続機能を使用した UAP では，DECLARE CONNECTION HANDLE SET の記述以前に SQL 文は記述できません。
2. DECLARE CONNECTION HANDLE SET で宣言するのは，接続ハンドルの変数の名称であり，値そのものではありません。
3. 複数接続機能を使用した SQL 文の実行結果を格納している SQL 連絡領域を，ほかのモジュールから参照する場合は，そのモジュールで DECLARE CONNECTION HANDLE SET を実行しておく必要があります。

(6) 使用例

PDCNCTHDL 型変数が CnctHdl の接続ハンドルを宣言します。

```
(C言語の場合)
  EXEC SQL BEGIN DECLARE SECTION;
      PDCNCTHDL CnctHdl;
  EXEC SQL END DECLARE SECTION;

  EXEC SQL DECLARE CONNECTION HANDLE SET :CnctHdl;

(COBOL言語の場合)
  EXEC SQL BEGIN DECLARE SECTION END-EXEC
      01 CNCTHDL SQL TYPE IS PDCNCTHDL.
  EXEC SQL END DECLARE SECTION END-EXEC

  EXEC SQL DECLARE CONNECTION HANDLE SET :CNCTHDL END-EXEC
```

6.7 DECLARE CONNECTION HANDLE UNSET（使用する接続ハンドルの全解除）

6.7.1 DECLARE CONNECTION HANDLE UNSET の形式と規則

(1) 機能

この文以前に DECLARE CONNECTION HANDLE SET で指定した接続ハンドルの，使用の宣言をすべて解除します。

(2) 形式

```
DECLARE CONNECTION HANDLE UNSET
```

(3) 共通規則

1. DECLARE CONNECTION HANDLE UNSET で指定した接続ハンドルの有効範囲は，ソースプログラム上での出現位置によって決定します。ある DECLARE CONNECTION HANDLE UNSET で指定した接続ハンドルを解除した場合，ソースプログラム上の次に出現する DECLARE CONNECTION HANDLE SET までの，すべての SQL に対して接続ハンドルの使用を解除します。
2. DECLARE CONNECTION HANDLE UNSET は，COBOL 言語では使用できません。

(4) 留意事項

1. DECLARE CONNECTION HANDLE UNSET 以降に発行された SQL 文に対しては，複数接続機能の処理はしません。

(5) 使用例

宣言した接続ハンドルをすべて解除します。

```
DECLARE CONNECTION HANDLE UNSET;
```

6.8 GET CONNECTION HANDLE（接続ハンドル取得）

6.8.1 GET CONNECTION HANDLE の形式と規則

(1) 機能

X/Open XA インタフェース環境下で複数接続機能を使用する場合，UAP が使用する接続ハンドルを割り当てます。

(2) 形式

```
GET CONNECTION HANDLE ：PDCNCTHDL型変数,
                       ：リターンコード受取変数,
                       ：環境変数グループ識別子変数
```

(3) オペランド

(a) ：PDCNCTHDL 型変数

PDCNCTHDL 型として宣言した埋込み変数を指定します。

(b) ：リターンコード受け取り変数

INT 型として宣言した埋込み変数を指定します。

リターンコード受け取り変数に返される値を次に示します。

C 言語の場合

正常に取得された場合：

p_rdb_RC_NORM

何らかの障害が発生した場合：

各障害に対応したエラーコード

これらの値は，pdberrno.h に定義しています。

COBOL 言語の場合

正常に取得された場合：

P-RDB-RC-NORM

これらの値は，SQLCAMTH.CBL に定義しています。

(c) ：環境変数グループ識別子変数

CHAR 型（5 バイト固定）として宣言した埋込み変数を指定します。

既定文字集合以外の文字集合は指定できません。

環境変数グループ識別子変数には，xa_open 関数用文字列に指定した環境変数グループ識別子（ナル文字を含めて 5 バイト固定）を指定してください。

(4) 共通規則

1. 使用する埋込み変数の宣言は，埋込み変数宣言部で行います。
2. 接続ハンドルの取得に失敗した場合，その障害コードをリターンコード受け取り変数に設定します。

(5) 留意事項

1. ここで取得した接続ハンドルで SQL を発行する場合は，DECLARE CONNECTION HANDLE SET で使用する接続ハンドルの宣言をしてください。
2. COBOL 言語の場合，環境変数グループ識別子変数に値を設定するときは，設定値の最後をナル文字にしてください。
3. OLTP 下の X/Open に従った UAP では，GET CONNECTION HANDLE で取得した接続ハンドルは OLTP が解放します。このため，FREE CONNECTION HANDLE を発行して，接続ハンドルを解放しないでください。FREE CONNECTION HANDLE を発行して接続ハンドルを解放した場合，その後 UAP が不正な動作をして，異常終了するおそれがあります。

(6) 使用例

PDCNCTHDL 型変数が CnctHdl，リターンコード受け取り変数が GetchdlRtn，環境変数グループ識別子変数が GroupName の接続ハンドルを割り当てます。

(C 言語の場合)

```
EXEC SQL BEGIN DECLARE SECTION;
  PDCNCTHDL   CnctHdl;
  long        GetchdlRtn;
  char        GroupId[5];
EXEC SQL END DECLARE SECTION;
strcpy(GroupId,"HR01") ;
EXEC SQL GET CONNECTION HANDLE  :CnctHdl,
                                :GetchdlRtn,
                                :GroupId;
```

(COBOL 言語の場合)

```
DATA DIVISION.
WORKING-STORAGE SECTION.
  EXEC SQL
    BEGIN DECLARE SECTION
  END-EXEC.
01 CNCTHDL     SQL TYPE IS PDCNCTHDL.
01 GETCHDLRTN  PIC S9(9) COMP.
```

```
01 GROUPID     PIC X(5).
  EXEC SQL
    END DECLARE SECTION
  END-EXEC.
        :
PROCEDURE DIVISION.
        :
  MOVE 'HR01' & X'00' TO GROUPID.
  EXEC SQL
    GET CONNECTION HANDLE :CNCTHDL,
                          :GETCHDLRTN,
                          :GROUPID;
  END-EXEC.
```

(7) 注意事項

HiRDB サーバが複数接続機能をサポートしていない場合，一つのトランザクションから同一の HiRDB に対して，複数のコネクションハンドルを使って SQL を実行できません。実行した場合，HiRDB サーバが Pac2354 のコードでアボートし，トランザクションがロールバックされます。よって，同一トランザクション内で複数のコネクションハンドルを使った SQL 文は同時に実行できません。

接続先がそれぞれ異なる HiRDB であれば，同一トランザクション内で複数のコネクションハンドルを使用できます。

誤りの例を次に示します。

```
《HiRDB1をRM01とRM02で登録している状態》

tx_begin()
strcpy(grpnm, "RM01")
GET CONNECTION HANDLE :hCnct,:rc,:grpnm;
DECLARE CONNECTION HANDLE SET :hCnct;

SQL実行

strcpy(grpnm, "RM02")
GET CONNECTION HANDLE :hCnct,:rc,:grpnm;
DECLARE CONNECTION HANDLE SET :hCnct;

SQL実行

tx_commit()
```

6.9 COPY（登録集原文の引き込み）

6.9.1 COPY の形式と規則

(1) 機能

埋込み変数及び標識変数の宣言，又は SQL 文を含む登録集原文をソース中に引き込みます。

(2) 形式

```
COPY 原文名
```

(3) オペランド

(a) 原文名

登録集原文，又はヘッダファイルが登録されているファイルの名称（サフィックスを除く）を 30 文字以下の文字列で指定します。

(4) 共通規則

1. SQL 先頭子と SQL 終了子で囲んでください。また，SQL 終了子の後は，行末まで何も記述しないでください。
2. 登録集原文，又はヘッダファイルが登録されているディレクトリは，次に示す順位で検索するため，次に示すどれかのファイルに登録しておいてください。

   ディレクトリの検索順位

   - 環境変数で登録したディレクトリ
     （COBOL，及び OOCOBOL の場合は PDCBLLIB，C，及び C++の場合は PDCLIB）
   - カレントディレクトリ

   ファイルの検索順位（COBOL，及び OOCOBOL の場合）

   - 環境変数 PDCBLFIX で登録したサフィックスが付いたファイル
   - ファイル名.cbl
   - ファイル名.CBL
   - ファイル名.cob

   ファイルの検索順位（C，及び C++の場合）

   - ファイル名.h
3. COPY で引き込んだ原文中に SQL の COPY は指定できません。

4. COBOL，及び OOCOBOL の場合，登録集原文は固定形式正書法で記述してください。

(5) 留意事項

1. 埋込み変数及び標識変数の宣言，又は SQL 文を含む登録集原文をソース中に引き込む場合は，SQL の COPY 文を使用します。この場合，次の文，及び命令は使用できません。
   - COBOL 言語及び OOCOBOL 言語の，COPY 文及び INCLUDE 文
   - C 言語，及び C++言語の#include 命令

     ただし，C 言語の#include 命令については，例外的に使用できる場合があります。詳細はマニュアル「HiRDB UAP 開発ガイド」の「SQL の記述規則」を参照してください。
2. 埋込み変数及び標識変数の宣言，又は SQL 文のどれも含まない登録集原文若しくはヘッダファイルをソース中に引き込む場合は，次に示す UAP 記述言語の機能を使用します。
   - COBOL 言語及び OOCOBOL 言語の，COPY 文及び INCLUDE 文
   - C 言語，及び C++言語の#include 命令

(6) 使用例

登録集原文（ファイル名：SAMPLE）をソースプログラム中に引き込みます。

```
<COBOL言語の場合>

  EXEC SQL
      COPY SAMPLE
  END-EXEC.
```

6.10 GET DIAGNOSTICS（診断情報取得）

6.10.1 GET DIAGNOSTICS の形式と規則

(1) 機能

直前に実行した SQL 文が次のどれかの場合，そのエラー情報及び診断情報を，診断領域から取得します。

- 定義系 SQL
- 操作系 SQL
- 制御系 SQL
- 埋込み言語文法（INSTALL JAR，REPLACE JAR，REMOVE JAR，INSTALL CLIB，REPLACE CLIB，REMOVE CLIB）
- ルーチン制御 SQL

ルーチンの SQL オブジェクトの再作成時，この診断情報に，正常に再作成したルーチンの情報が含まれます。

(2) 形式

```
GET DIAGNOSTICS
     {:埋込み変数＝文情報項目名
          〔, :埋込み変数＝文情報項目名〕…
       |EXCEPTION 条件番号
          :埋込み変数＝条件情報項目名
              〔, :埋込み変数＝条件情報項目名〕…}

文情報項目名：：＝ {NUMBER|MORE}
条件情報項目名：：＝ {RETURNED_SQLCODE
                     |ERROR_POSITION
                     |ERROR_SQL_NO
                     |ERROR_SQL
                     |ROUTINE_TYPE
                     |ROUTINE_SCHEMA
                     |ROUTINE_NAME
                     |TRIGGER_SCHEMA
                     |TRIGGER_NAME
                     |CONSTRAINT_SCHEMA
                     |CONSTRAINT_NAME
                     |MESSAGE_TEXT
                     |QUERY_NAME
                     |CONDITION_IDENTIFIER}
```

(3) オペランド

(a) 文情報項目名

NUMBER

診断領域中の診断情報の数を取得する場合に指定します。データ型は SMALLINT を指定してください。

MORE

診断情報の数が，診断領域に格納できる数を超えたかどうかの情報を取得する場合に指定します。

データ型は CHAR，長さは 1 バイトを指定してください。

診断領域にすべての診断情報が設定されている場合は N，そうでない場合は Y が格納されます。

(b) 条件番号

何番目の診断情報を取得するかを示す値を埋込み変数で指定します。

(c) 条件情報項目名

RETURNED_SQLCODE

リターンコード（SQLCODE）の値を取得する場合に指定します。データ型は INTEGER を指定してください。

ERROR_POSITION

構文エラーが発生した場合の，SQL 中のエラーの位置を取得するときに指定します。

データ型は INTEGER を指定してください。構文エラー以外のエラーが発生した場合は，0 が設定されます。

ERROR_SQL_NO

次のどれかの SQL を実行した場合，そのルーチン中でエラーとなった SQL 手続き文を示す番号を取得したいときに指定します。

- CREATE PROCEDURE
- CREATE FUNCTION
- CREATE TYPE
- CREATE TRIGGER
- ALTER PROCEDURE
- ALTER ROUTINE
- ALTER TRIGGER
- CALL 文
- トリガを引き起こす操作系 SQL（ただし，取得できる情報はトリガの動作についてだけです）

番号は，手続き中の SQL 手続き文の開始位置の順序で，各 SQL 手続き文に 0 から順に付けたものです。データ型は INTEGER を指定してください。

ERROR_SQL

診断情報の SQL の種別を示す文字列を取得したいときに指定します。

データ型は VARCHAR，長さは 32 バイト（文字集合名 UTF16 を指定した場合は 64 バイト）を指定してください。

設定される文字列を次の表に示します。

表 6-2　ERROR_SQL に設定される文字列

分類（SQL 種別）		文字列設定の有無				設定される文字列
		ルーチン		左記以外		
		定義	実行	前処理	実行	
定義系 SQL	ALTER INDEX	×	×	○	○	ALTER INDEX
	ALTER PROCEDURE	×	×	○	○	ALTER PROCEDURE
	ALTER ROUTINE	×	×	○	○	ALTER ROUTINE
	ALTER TABLE	×	×	○	○	ALTER TABLE
	ALTER TRIGGER	×	×	○	○	ALTER TRIGGER
	COMMENT（表に注釈を付ける）	×	×	○	○	COMMENT TABLE
	COMMENT（列に注釈を付ける）	×	×	○	○	COMMENT COLUMN
	CREATE AUDIT	×	×	○	○	CREATE AUDIT
	CREATE CONNECTION SECURITY	×	×	○	○	CREATE CONNECTION SECURITY
	CREATE FUNCTION	×	×	○	○	CREATE FUNCTION
	CREATE PUBLIC FUNCTION	×	×	○	○	CREATE FUNCTION
	CREATE INDEX	×	×	○	○	CREATE INDEX
	CREATE PROCEDURE	×	×	○	○	CREATE PROCEDURE
	CREATE PUBLIC PROCEDURE	×	×	○	○	CREATE PROCEDURE
	CREATE SCHEMA	×	×	○	○	CREATE SCHEMA
	CREATE SEQUENCE	×	×	○	○	CREATE SEQUENCE
	CREATE TABLE	×	×	○	○	CREATE TABLE
	CREATE TRIGGER	×	×	○	○	CREATE TRIGGER
	CREATE TYPE	×	×	○	○	CREATE TYPE
	CREATE VIEW	×	×	○	○	CREATE VIEW
	CREATE PUBLIC VIEW	×	×	○	○	CREATE VIEW
	DROP AUDIT	×	×	○	○	DROP AUDIT

分類（SQL 種別）		文字列設定の有無				設定される文字列
		ルーチン		左記以外		
		定義	実行	前処理	実行	
	DROP CONNECTION SECURITY	×	×	○	○	DROP CONNECTION SECURITY
	DROP DATA TYPE	×	×	○	○	DROP DATA TYPE
	DROP FUNCTION	×	×	○	○	DROP FUNCTION
	DROP PUBLIC FUNCTION	×	×	○	○	DROP FUNCTION
	DROP INDEX	×	×	○	○	DROP INDEX
	DROP PROCEDURE	×	×	○	○	DROP PROCEDURE
	DROP PUBLIC PROCEDURE	×	×	○	○	DROP PROCEDURE
	DROP SCHEMA	×	×	○	○	DROP SCHEMA
	DROP SEQUENCE	×	×	○	○	DROP SEQUENCE
	DROP TABLE	×	×	○	○	DROP TABLE
	DROP TRIGGER	×	×	○	○	DROP TRIGGER
	DROP VIEW	×	×	○	○	DROP VIEW
	DROP PUBLIC VIEW	×	×	○	○	DROP VIEW
	GRANT CONNECT	×	×	○	○	GRANT CONNECT
	GRANT DBA	×	×	○	○	GRANT DBA
	GRANT RDAREA	×	×	○	○	GRANT RDAREA
	GRANT SCHEMA	×	×	○	○	GRANT SCHEMA
	GRANT SCHEMA OPERATION	×	×	○	○	GRANT SCHEMA
	GRANT アクセス権限	×	×	○	○	GRANT ACCESS
	GRANT AUDIT	×	×	○	○	GRANT AUDIT
	REVOKE CONNECT	×	×	○	○	REVOKE CONNECT
	REVOKE DBA	×	×	○	○	REVOKE DBA
	REVOKE RDAREA	×	×	○	○	REVOKE RDAREA
	REVOKE SCHEMA	×	×	○	○	REVOKE SCHEMA
	REVOKE SCHEMA OPERATION	×	×	○	○	REVOKE SCHEMA
	REVOKE アクセス権限	×	×	○	○	REVOKE ACCESS
操作系 SQL	ALLOCATE CURSOR 文	×	×	×	○	ALLOCATE CURSOR
	ASSIGN LIST 文	×	×	○	○	ASSIGN LIST

分類（SQL 種別）		文字列設定の有無				設定される文字列
		ルーチン		左記以外		
		定義	実行	前処理	実行	
	CALL 文	○	○	○	○	CALL
	CLOSE 文	○	○	×	×	CLOSE
		×	×	○	○	（カーソルに指定した SQL 種別の，この項目を参照してください）
	DEALLOCATE PREPARE 文	×	×	×	○	－
	DECLARE CURSOR	○	×	×	×	DECLARE CURSOR
		×	×	○	○	（動的 SELECT 文の，この項目を参照してください）
	DELETE 文	○	○	○	○	DELETE
	準備可能動的 DELETE 文：位置付け	×	×	○	○	DELETE
	DESCRIBE 文	×	×	○	○	（DESCRIBE 文に指定した SQL 種別の，この項目を参照してください）
	DESCRIBE CURSOR 文	×	×	○	○	（DESCRIBE CURSOR 文に指定した SQL 種別の，この項目を参照してください）
	DESCRIBE TYPE 文	×	×	○	○	（DESCRIBE TYPE 文に指定した SQL 種別の，この項目を参照してください）
	DROP LIST 文	×	×	○	○	DROP LIST
	EXECUTE 文	×	×	○	○	（EXECUTE 文で実行した SQL 種別の，この項目を参照してください）
	EXECUTE IMMEDIATE 文	×	×	○	○	（EXECUTE IMMEDIATE 文で実行した SQL 種別の，この項目を参照してください）
	FETCH 文	○	○	×	×	FETCH
		×	×	○	○	（カーソルに指定した SQL 種別の，この項目を参照してください）
	FREE LOCATOR 文	×	×	○	○	FREE LOCATOR
	INSERT 文	○	○	○	○	INSERT
	OPEN 文	○	○	×	×	OPEN
		×	×	○	○	（カーソルに指定した SQL 種別の，この項目を参照してください）

分類（SQL 種別）		文字列設定の有無				設定される文字列
		ルーチン		左記以外		
		定義	実行	前処理	実行	
	PREPARE 文	○	○	○	○	（PREPARE 文に指定した SQL 種別の，この項目を参照してください）
	PURGE TABLE 文	○	○	○	○	PURGE TABLE
	1 行 SELECT 文	○	○	○	○	SELECT
	動的 SELECT 文	×	×	○	○	SELECT
	UPDATE 文	○	○	○	○	UPDATE
	準備可能動的 UPDATE 文：位置付け	×	×	○	○	UPDATE
	代入文	○	○	○	○	SET
制御系 SQL	CALL COMMAND 文	○	○	○	○	CALL COMMAND
	COMMIT 文	○	○	○	○	COMMIT
	CONNECT 文	×	×	×	×	－
	DISCONNECT 文	×	×	×	×	－
	LOCK 文	○	○	○	○	LOCK TABLE
	ROLLBACK 文	○	○	○	○	ROLLBACK
	SET SESSION AUTHORIZATION 文	×	×	○	○	SET SESSION AUTHORIZATION
埋込み言語文法	BEGIN DECLARE SECTION	×	×	×	×	－
	END DECLARE SECTION	×	×	×	×	（該当しません）
	ALLOCATE CONNECTION HANDLE	×	×	×	×	－
	FREE CONNECTION HANDLE	×	×	×	×	－
	DECLARE CONNECTION HANDLE SET	×	×	×	×	（該当しません）
	DECLARE CONNECTION HANDLE UNSET	×	×	×	×	（該当しません）
	GET CONNECTION HANDLE	×	×	×	×	－
	COPY	×	×	×	×	（該当しません）
	GET DIAGNOSTICS	×	×	×	×	－
	COMMAND EXECUTE	×	×	×	×	－
	SQL 先頭子	×	×	×	×	（該当しません）

分類（SQL 種別）		文字列設定の有無				設定される文字列
		ルーチン		左記以外		
		定義	実行	前処理	実行	
	SQL 終了子	×	×	×	×	（該当しません）
	WHENEVER	×	×	×	×	−
	SQLCODE 変数	×	×	×	×	（該当しません）
	SQLSTATE 変数	×	×	×	×	（該当しません）
	PDCNCTHDL 型変数の宣言	×	×	×	×	（該当しません）
	INSTALL JAR	×	×	○	○	INSTALL JAR
	REPLACE JAR	×	×	○	○	REPLACE JAR
	REMOVE JAR	×	×	○	○	REMOVE JAR
	INSTALL CLIB	×	×	○	○	INSTALL CLIB
	REPLACE CLIB	×	×	○	○	REPLACE CLIB
	REMOVE CLIB	×	×	○	○	REMOVE CLIB
	DECLARE AUDIT INFO SET	×	×	×	×	（該当しません）
ルーチン制御 SQL	SQL 変数宣言	○	○	×	×	DECLARE
	カーソル宣言	○	×	×	×	DECLARE CURSOR
		○	○	×	×	SELECT
	条件宣言	○	×	×	×	DECLARE CONDITION
	ハンドラ宣言	○	×	×	×	DECLARE HANDLER
	SQL 手続き文	○	○	×	×	（SQL 手続き文に指定した SQL 種別の，この項目を参照してください）
	複合文	○	○	×	×	BEGIN
	IF 文	○	○	×	×	IF
	LEAVE 文	○	○	×	×	LEAVE
	RETURN 文	○	○	×	×	RETURN
	WHILE 文	○	○	×	×	WHILE
	FOR 文	○	○	×	×	FOR
	WRITE LINE 文	○	○	×	×	WRITE LINE
	SIGNAL 文	○	○	×	×	SIGNAL
	RESIGNAL 文	○	○	×	×	RESIGNAL

分類（SQL 種別）	文字列設定の有無				設定される文字列
	ルーチン		左記以外		
	定義	実行	前処理	実行	
上記に分類できない場合	○	○	○	○	半角空白

（凡例）

○：「設定される文字列」欄に記載されている文字列が設定されます。

×：文字列は設定されません。

－：設定される文字列はありません。

括弧内の表記は，注釈を示します。

ROUTINE_TYPE

エラーとなった関数又は手続きの種別を取得したい場合に指定します。データ型は CHAR，長さは 1 バイト（文字集合名 UTF16 を指定した場合は 2 バイト）を指定してください。種別が手続きの場合は P，関数の場合は F が設定されます。

ROUTINE_SCHEMA

エラーとなった関数又は手続きの認可識別子を取得したい場合に指定します。データ型は VARCHAR，長さは 30 バイト（文字集合名 UTF16 を指定した場合は 60 バイト）を指定してください。

以下のエラーの場合は認可識別子に”PUBLIC”が設定されます。

- パブリック関数，パブリック手続きの定義，削除時のエラー
- パブリック手続きの CALL 文実行時のエラー

ROUTINE_NAME

エラーとなった関数又は手続きの識別子を取得したい場合に指定します。データ型は VARCHAR，長さは 30 バイト（文字集合名 UTF16 を指定した場合は 60 バイト）を指定してください。

TRIGGER_SCHEMA

エラーとなったトリガの認可識別子を取得したい場合に指定します。データ型は VARCHAR，長さは 30 バイト（文字集合名 UTF16 を指定した場合は 60 バイト）を指定してください。

TRIGGER_NAME

エラーとなったトリガの認可識別子を取得したい場合に指定します。データ型は VARCHAR，長さは 30 バイト（文字集合名 UTF16 を指定した場合は 60 バイト）を指定してください。

CONSTRAINT_SCHEMA

検査制約，参照制約で制約違反エラーとなった制約の認可識別子を取得したい場合に指定します。データ型は VARCHAR，長さは 30 バイト（文字集合名 UTF16 を指定した場合は 60 バイト）を指定してください。

CONSTRAINT _NAME

検査制約，参照制約で制約違反エラーとなった制約の制約名を取得したい場合に指定します。データ型は VARCHAR，長さは 30 バイト（文字集合名 UTF16 を指定した場合は 60 バイト）を指定してください。

MESSAGE_TEXT

メッセージテキストを取得したい場合に指定します。データ型は VARCHAR，長さは 254 バイト（文字集合名 UTF16 を指定した場合は 508 バイト）を指定してください。

QUERY_NAME

WITH 句指定のある動的 SELECT 文，及びカーソル宣言を実行した場合，エラーが発生した問合せ指定の問合せ名を取得したい場合に指定します。データ型は VARCHAR，長さは 30 バイト（文字集合名 UTF16 を指定した場合は 60 バイト）を指定してください。

CONDITION_IDENTIFIER

SQL 手続き中又はトリガ中で実行された，SIGNAL 文，又は RESIGNAL 文に指定した条件名，又は SQLSTATE 値を取得したい場合に指定します。データ型は VARCHAR，長さは 30 バイト（文字集合名 UTF16 を指定した場合は 60 バイト）を指定してください。

SQLSTATE 値を指定した場合は，'SQLSTATE:XXXXX'（XXXXX は指定した SQLSTATE 値）が設定されます。

(4) 共通規則

1. GET DIAGNOSTICS は，動的に実行できません。
2. 直前に実行した SQL の結果だけ取得できます。取得できる SQL については，表「ERROR_SQL に設定される文字列」を参照してください。
3. 条件番号を指定する埋込み変数のデータ型は，SMALLINT を指定してください。
4. 文情報項目，及び条件情報項目を受け取る埋込み変数のデータ型は，それぞれの項目のデータ型と同じ型を指定してください。
5. 条件番号には，0 以下の値，又は診断領域中のエラーの数を超える値は指定できません。
6. SQL 解析前に発生したエラーは GET DIAGNOSTICS では診断情報が取得できない場合があります。このとき，GET DIAGNOSTICS を実行すると 0 件が返ってきます。エラー情報はエラー発生時の SQLCA を参照してください。
7. 条件情報項目名の説明の中で，設定する情報を明記していない SQL については，1 バイトの空白，又は 0 が設定されます。
8. 関数実行時に発生したエラーに対して，GET DIAGNOSTICS では次の診断情報が取得できない場合があります。
   - ROUTINE_TYPE
   - ROUTINE_SCHEMA
   - ROUTINE_NAME

(5) 使用例

直前に実行した CREATE FUNCTION の次の診断情報を取得します。

- リターンコード（データ型が INTEGER の埋込み変数 XSQLCODE）

- 構文エラーが発生した場合の SQL 中のエラーの位置（データ型が INTEGER の埋込み変数 XPOSITION）
- ルーチン中でエラーとなった SQL 手続き文を示す番号（データ型が INTEGER の埋込み変数 XSQL_NO）
- ルーチン中でエラーとなった SQL 手続き文の種別を示す文字列（データ型が VARCHAR(32)の埋込み変数 XSQL）
- メッセージテキスト（データ型が VARCHAR(254)の埋込み変数 XMESSAGE）

```
CREATE FUNCTION NEXT_DAY(INDATE DATE,曜日 NCHAR)
  RETURNS DATE
  BEGIN
    DECLARE SDOW,TDOW INTEGER;
    SET TDOW=(CASE 曜日 WHEN N'日' THEN 0
      WHEN N'月' THEN 1
      WHEN N'火' THEN 2
      WHEN N'水' THEN 3
      WHEN N'木' THEN 4
      WHEN N'金' THEN 5
      ELSE 6 END);
    SET SDOW=MOD(DAYS(INDATE),7);
    RETURN (INDATE + (CASE WHEN TDOW>SDOW THEN TDOW-SDOW
      ELSE 7+TDOW-SDOW END) DAYS);
  END
GET DIAGNOSTICS EXCEPTION 1
    :XSQLCODE=RETURN_SQLCODE,
    :XPOSITION=ERROR_POSITION,
    :XSQL_NO=ERROR_SQL_NO,
    :XSQL=ERROR_SQL,
    :XMESSAGE=MESSAGE_TEXT
```

6.11 COMMAND EXECUTE（UAP からのコマンド実行）

6.11.1 COMMAND EXECUTE の形式と規則

(1) 機能

UAP 中から，HiRDB のコマンド，及び OS のコマンドを実行します。

COMMAND EXECUTE を実行する場合，HiRDB サーバに HiRDB Control Manager - Agent をインストールしておく必要があります。これは，HiRDB Control Manager - Agent がコマンドを実行するためです。

(2) 形式

```
COMMAND EXECUTE ：コマンドライン変数,
                 ：リターンコード受け取り変数,
                 ：実行結果受け取り領域長変数,
                 ：実行結果長受け取り変数,
                 ：実行結果受け取り変数,
                 ：実行コマンドリターンコード受け取り変数,
                 ：環境変数グループ名変数
```

(3) オペランド

(a) ：コマンドライン変数

コマンドライン変数には，HiRDB サーバで実行するコマンドのコマンドラインを設定します。

CHAR 型（領域長 30,000 バイト以内）として宣言した埋込み変数を指定してください。また，コマンドラインの最後には，必ずナル文字を指定してください。

既定文字集合以外の文字集合は指定できません。

コマンドライン変数に複数のコマンドは指定しないでください。指定した場合，動作は保証されません。

(b) ：リターンコード受け取り変数

リターンコード受け取り変数には，COMMAND EXECUTE 実行時のリターンコードが設定されます。INT 型として宣言した埋込み変数を指定してください。

リターンコード受け取り変数には，次の値が設定されます。なお，エラーの場合には，実行結果受け取り変数に詳細情報が設定されます。

p_rdb_RC_NORM：
　HiRDB サーバで正常にコマンドが実行された場合

p_rdb_RC_ERRPARM：
　引数が不正な場合

p_rdb_PROTO：
　通信エラーの場合

p_rdb_RC_NOTF：
　環境変数グループがない場合

p_rdb_RC_TIMEOUT：
　タイムアウトした場合

p_rdb_RC_SQLERR：
　そのほかのエラーの場合

(c)　：実行結果受け取り領域長変数

実行結果受け取り領域長変数には，実行結果受け取り変数の領域長を設定します。INT 型として宣言した埋込み変数を指定してください。

実行結果受け取り領域長は，2 ギガバイト以内で設定してください。

(d)　：実行結果長受け取り変数

実行結果長受け取り変数には，実行結果受け取り変数への出力長が設定されます。INT 型として宣言した埋込み変数を指定してください。

(e)　：実行結果受け取り変数

実行結果受け取り変数には，実行結果受け取り用に確保した領域のアドレスを設定します。PDOUTBUF 型として宣言した埋込み変数を指定してください。

COMMAND EXECUTE 実行後，実行結果受け取り変数には，次の値が設定されます。ただし，（実行結果受け取り領域長変数の指定値－1）以降のデータは切り捨てられます。また，実行結果の最後には，1 バイトのナル文字が設定されます。

リターンコード受け取り変数に p_rdb_RC_NORM が設定されている場合：
　HiRDB サーバで実行したコマンドラインの実行結果（標準出力と標準エラー出力）が設定されます。

リターンコード受け取り変数に p_rdb_RC_NORM 以外が設定されている場合：
　障害コードに対応する詳細メッセージが設定されます。

(f)　：実行コマンドリターンコード受け取り変数

実行コマンドリターンコード受け取り変数には，HiRDB サーバで実行したコマンドラインのリターンコードが設定されます。INT 型として宣言した埋込み変数を指定してください。

COMMAND EXECUTE が正常終了（リターンコード受け取り変数に p_rdb_RC_NORM が設定されている場合）したときだけ，実行コマンドリターンコード受け取り変数に有効な値が設定されます。

なお，実行したコマンドが標準出力又は標準エラー出力に情報を出力しない場合は，実行コマンドリターンコード受け取り変数には 0 が設定されます。

(g) ：環境変数グループ名変数

環境変数グループ名変数には，次の値を指定します。

UNIX 版の場合：

クライアント環境定義を記述した通常ファイルの名称（ナル文字を含めて 256 バイト以内）を，絶対パス名で指定します。

Windows 版の場合：

HiRDB クライアント環境変数登録ツールで登録したグループ名（ナル文字を含めて 31 バイト以内）を指定します。

CHAR 型（領域長 256 バイト以内）として宣言した埋込み変数を指定してください。

既定文字集合以外の文字集合は指定できません。

環境変数グループを使用しない場合は，1 バイト目にナル文字を設定してください。

環境変数グループについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4) 共通規則

1. 使用する埋込み変数は，埋込み変数宣言節で宣言してください。
2. COMMAND EXECUTE を実行する場合，次のクライアント環境定義を設定しておく必要があります。
   - PDASTHOST（HiRDB Control Manager - Agent のホスト名）
   - PDASTPORT（HiRDB Control Manager - Agent のポート番号）
   - PDSYSTEMID（HiRDB Control Manager - Agent の HiRDB 識別子）
   - PDASTUSER（HiRDB サーバでコマンドを実行する認可識別子）

   クライアント環境定義については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。
3. UAP が HiRDB と接続中でも COMMAND EXECUTE は実行できます。ただし，コマンドが終了するまで UAP 側に制御は戻ってこないので，デッドロックが発生しないように注意してください。
4. 応答要求があるコマンドは指定しないでください。HiRDB Control Manager - Agent では，応答の入力要求は受け付けないため，コマンドがエラーで終了します。
5. 実行するコマンドの入力ファイルは，あらかじめ HiRDB サーバに用意しておいてください。

6. コマンドの実行が長時間になる場合，コマンドが終了するまでUAPに制御は戻ってきません。この場合，クライアント環境定義PDCMDWAITTIMEを指定しておくと，HiRDBクライアント側の長時間待ちを回避できます。

   なお，クライアント側がタイムアウトとなった場合，HiRDB Control Manager - Agentのプロセス，又は実行中のコマンドを，OSのkillコマンド（Windows版の場合はpdkillコマンド）で取り消してください。

7. HiRDBサーバ接続中に，コマンドの実行が長時間になる場合，クライアント環境定義PDSWAITTIME及びPDSWATCHTIMEによってHiRDBサーバ側でタイムアウトを検知して，接続が終了することがあるので注意してください。

(5) 留意事項

1. COMMAND EXECUTE実行時は，次のクライアント環境定義だけ有効となり，それ以外のものについては無効となります。

   PDCLTPATH，PDASTHOST，PDASTUSER，PDUSER，PDASTPORT，
   PDCMDWAITTIME，PDCMDTRACE，PDSYSTEMID，PDCLTAPNAME

2. COBOL言語の場合，COMMAND EXECUTEは実行できません。

(6) 使用例

UAP中から，pdlsコマンドを実行します。なお，HiRDB/シングルサーバで実行しているものとします。

```
EXEC SQL BEGIN DECLARE SECTION;
    char         CmdLine[30000];
    int          ReturnCode;
    int          OutBufLen;
    int          OutDataLen;
    int          DataLength;
    PDOUTBUF     OutBuf ;
    int          CmdRetCode;
    char         EnvGroup[256];
EXEC SQL END DECLARE SECTION;

strcpy(CmdLine,"c:¥HiRDB_S¥bin¥pdls -d trn");
OutBuf = malloc(30000) ;
OutBufLen = 30000 ;
EnvGroup[0] = '¥0';

EXEC SQL COMMAND EXECUTE    :CmdLine,
                            :ReturnCode,
                            :OutBufLen,
                            :DataLength,
                            :OutBuf,
                            :CmdRetCode,
                            :EnvGroup ;

if (ReturnCode == p_rdb_RC_NORM)
{
      if (CmdRetCode == 0)
```

```
        {
            printf("%s実行成功¥n",CmdLine) ;
            printf("実行結果：%s¥n",OutBuf) ;
        }
        else
        {
            printf("%s実行失敗¥n",CmdLine) ;
            printf("実行結果：%s¥n",OutBuf) ;
        }
}
else
{
            printf("ReturnCode=%d¥n",ReturnCode) ;
            printf("エラー詳細情報：%s¥n",OutBuf) ;
}
```

6.12　SQL 先頭子

6.12.1　SQL 先頭子の形式と規則

(1)　機能

SQL の始まりを示します。

(2)　形式

```
EXEC SQL
```

(3)　共通規則

1. SQL は，一つの SQL ごとに SQL 先頭子と SQL 終了子で囲んでください。
2. SQL の記述規則については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4)　使用例

OPEN 文を記述します。

```
<C言語の場合>

  EXEC SQL
      OPEN CR1;

<COBOL言語の場合>

  EXEC SQL
      OPEN CR1
  END-EXEC.
```

6.13　SQL 終了子

6.13.1　SQL 終了子の形式と規則

(1)　機能

SQL の終わりを示します。

(2)　形式

C 言語の場合

```
;
```

COBOL 言語の場合

```
END-EXEC
```

(3)　共通規則

1. SQL は，一つの SQL ごとに SQL 先頭子と SQL 終了子で囲んでください。
2. SQL の記述規則については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4)　使用例

使用例については，「SQL 先頭子」を参照してください。

6.14 WHENEVER（埋込み例外宣言）

6.14.1 WHENEVER の形式と規則

(1) 機能

SQL の実行後，HiRDB が SQL 連絡領域に設定したリターンコード（SQLCODE）によって，UAP の処理を宣言します。

(2) 形式

```
WHENEVER
   {SQLERROR | SQLWARNING | NOT FOUND}
   {CONTINUE |  {GO TO | GOTO}  〔:〕ホスト識別子
              | 〔DO〕PERFORM 〔:〕ホスト識別子
              | DO  {break | continue | '命令文'} }
```

(3) オペランド

(a) SQLERROR

ユーザの誤りや HiRDB の異常によって SQL が正常に実行されなかったとき（SQL 連絡領域の SQLCODE 領域，及び SQLCODE 変数に負の値が返されたとき）の処理を指示する場合に指定します。

(b) SQLWARNING

SQL は正常に実行されたが，ユーザに警告する状態を検知したとき（SQL 連絡領域の SQLWARN0 領域に W が返されたとき，又は SQL 連絡領域の SQLCODE 領域及び SQLCODE 変数に 100 以外の正の値が返されたとき）の処理を指示する場合に指定します。

(c) NOT FOUND

表の検索で検索結果の検索する行がなくなったとき（SQL 連絡領域の SQLCODE 領域に 100，SQLCODE 変数に 100，及び SQLSTATE 変数に'02000''が返されたとき）の処理を指示する場合に指定します。

(d) CONTINUE

UAP の実行を続行させる場合に指定します。

(e) {GO TO | GOTO} 〔:〕ホスト識別子

UAP の実行を分岐させる場合，次に示すホスト識別子によって分岐先を指定します。

- ラベル（C 言語の場合）

- 節名，又は段落名（COBOL 言語の場合）

(f) 〔DO〕PERFORM〔：〕ホスト識別子

指定した手続きを実行させる場合，次に示すホスト識別子によって実行させる手続きを指定します。オブジェクトのメソッドは指定できません。

- 関数名（C 言語の場合）
- 節名，又は段落名（COBOL 言語の場合）

(g) DO {break | continue | '命令文'}

UAP の実行を分岐，又は任意の命令文を実行します。continue，及び'命令文'は，C 言語，及び C++言語の場合に使用できます。

DO break
: break 文を実行します。

DO continue
: continue 文を実行します。

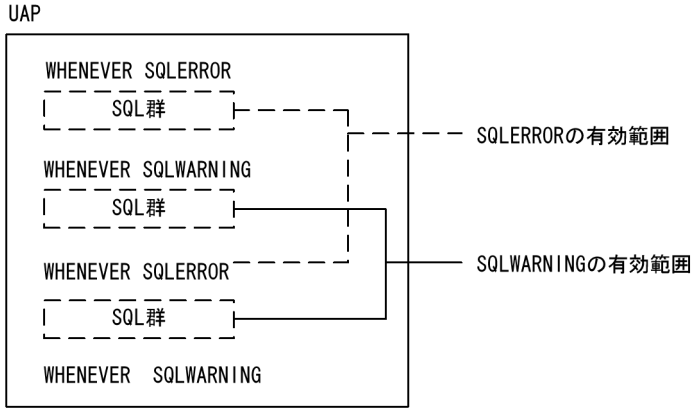
DO '命令文'
: 文字列として記述したホスト言語の任意の命令文（引数を指定した関数呼出しなど）を実行します。

(4) 共通規則

1. 埋込み例外宣言を指定しないと，すべてのリターンコードに対して CONTINUE が仮定されます。
2. 埋込み例外宣言は，同じ UAP 中に複数個指定できます。
3. SQLERROR の状態で，ROLLBACK 文以外の SQL は実行できません。
4. 手続き実行後の制御は，特異状態が発生した SQL の次の命令へ戻ります。
5. 埋込み例外宣言で指定した処理の有効範囲は，ソースプログラム上の位置によって決まります。すなわち，ある一つの埋込み例外宣言で指定された処理は，ソースプログラム上の次の同じ特異状態の処理を指定した埋込み例外宣言までの間にあるすべての SQL の実行結果で有効になります。

   埋込み例外宣言で指定した処理の有効範囲を，次の図に示します。

図 6-1　埋込み例外宣言で指定した処理の有効範囲



6. SQLERROR，SQLWARNING，NOT FOUND の有効範囲が重複した場合は，次に示す優先順位で SQL が処理されます。

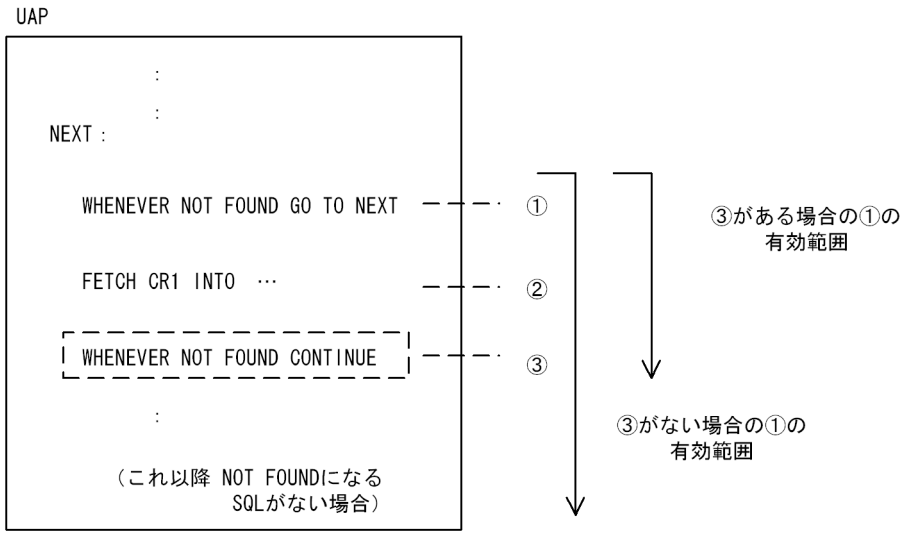
   SQLERROR→NOT FOUND→SQLWARNING

(5)　留意事項

1. WHENEVER 文で宣言した処置が途中の SQL から不要になる場合，同じ特異状態（SQLERROR など）に対する処置として，CONTINUE（処理を続行する）を指定した WHENEVER 文を，不要にする位置に記述してください。

   WHENEVER 文の記述例を，次の図に示します。

図 6-2　WHENEVER 文の記述例（その 1）

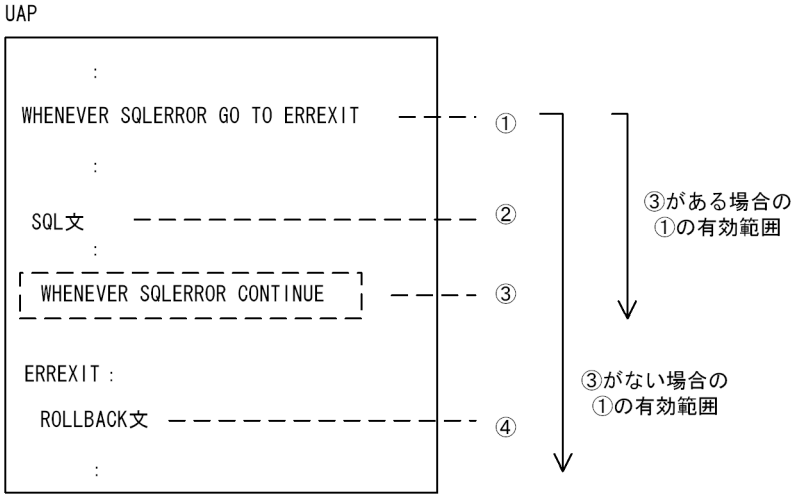


（説明）
③以降のSQL文で，NOT FOUNDになるSQLがない場合，③以降のSQL文でNOT FOUNDになった場合の処理は不要なので，③以降のWHENEVER文で処置をCONTINUEにします。

2. SQLERROR 指定の WHENEVER 文によって，ある SQL の実行エラーで分岐した先で，ROLLBACK 文を実行する場合，無限ループしないように注意してください。

   WHENEVER 文の記述例を，次の図に示します。

図 6-3　WHENEVER 文の記述例（その 2）



（説明）

③のWHENEVER文がなければ，④のROLLBACK文についても①のWHENEVER文が有効となり，ROLLBACK中にエラーが発生した場合に，無限ループします。このような場合，ROLLBACK文の前にCONTINUEを指定した③のWHENEVER文を指定する配慮が必要です。

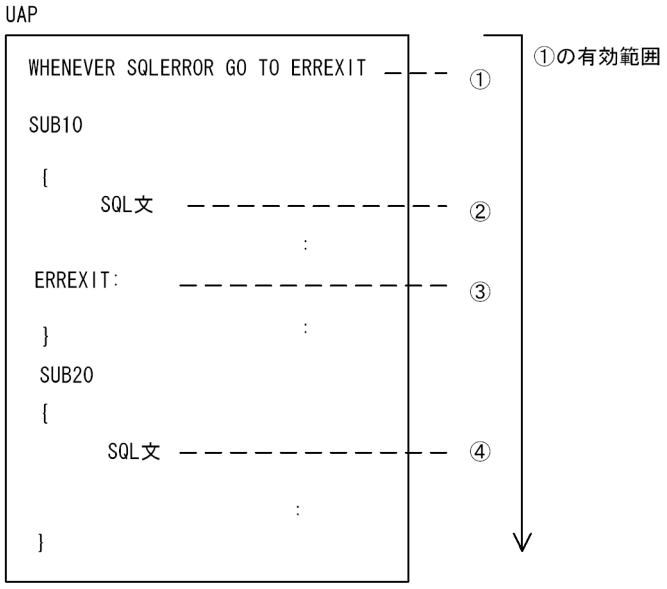
3. 埋込み例外宣言は，UAP の複数の関数にわたって有効です。

WHENEVER 文で指定した GOTO 文の分岐先は，SQL 文と同じ関数内に指定してください。関数外への分岐先を指定すると，コンパイル時にエラーになります。

また，必要に応じて，関数ごとに WHENEVER 文を宣言し直してください。

WHENEVER 文の記述例を，次の図に示します。

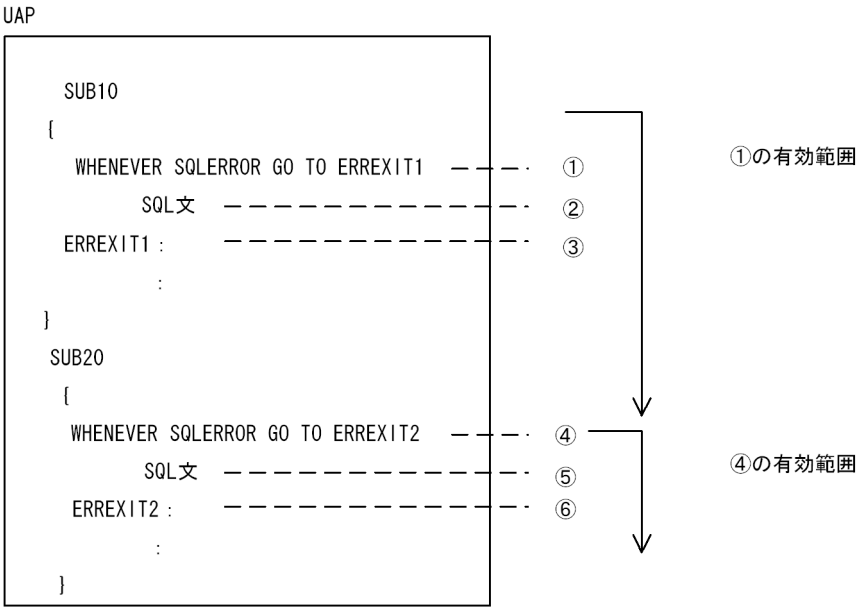
図 6-4　WHENEVER 文の記述例（その 3）



（説明）

②でSQLERRORが発生すると，①で指定した分岐先③は②と同じ関数内なので③に制御が分岐します。④でSQLERRORが発生すると，①で指定した分岐先③は④と同じ関数内にないので，コンパイルエラーになります。

図 6-5　WHENEVER 文の記述例（その 4）

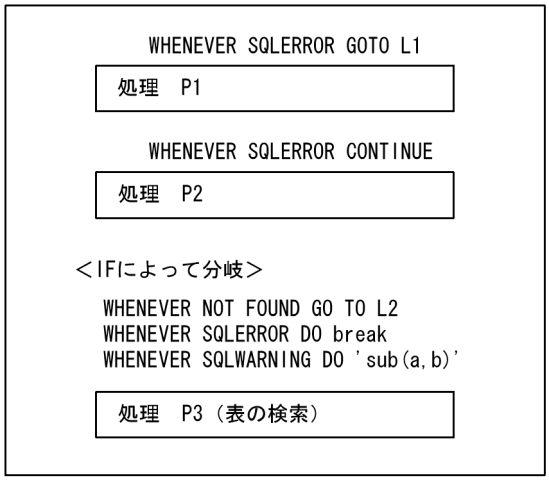


（説明）
②でSQLERRORが起きたときの分岐先③と，⑤でSQLERRORが起きたときの分岐先⑥は，
それぞれSQL文で同じ関数なので分岐できます。

4. WHENEVER 文は，FREE CONNECTION HANDLE の前で宣言しないでください。

(6)　使用例

- 処理 P1 中の SQL を実行した後，SQLCODE に負の値が返された場合，L1 へ分岐します。
- 処理 P2 中の SQL を実行した後，SQLCODE に負の値が返された場合，IF 文を使用して分岐先を決めるため，WHENEVER 文による制御を取り消します。
- 処理 P3 中の表の検索で，検索する行がなくなった場合は L2 へ分岐します。
- 処理 p3 中の表の検索で SQLCODE に負の値が返された場合，break 文を実行します。
- 処理 p3 中の表の検索でユーザに警告する状態を検知した場合，関数を呼び出します。



6.15 SQLCODE 変数

SQL を実行すると HiRDB によってリターンコード（SQLCODE）が設定されます。SQLCODE 変数に返される値の内容は，SQL 連絡領域の SQLCODE 領域の内容と同じです。

SQLCODE 変数は，プリプロセス実行時にシステムが宣言文をソースプログラム中に埋め込むため，UAP での宣言は必要ありません。データ型は，C 言語では signed long int，COBOL 言語では S9 （ 9） COMPUTATIONAL で宣言しています。

参照する場合は，変数名称 SQLCODE を指定してください。

また，複数接続機能を使用した環境では，DECLARE CONNECTION HANDLE SET 文で SQLCODE が使用する接続ハンドルを宣言しておく必要があります。

6.16 SQLSTATE 変数

SQL を実行すると HiRDB によってリターンコード（SQLSTATE）が設定されます。SQLSTATE 変数は，2 けたのクラス，及び 3 けたのサブクラスで構成されている 5 けたの文字列です。

SQLSTATE 変数は，プリプロセス実行時にシステムが宣言文をソースプログラム中に埋め込むため，UAP での宣言は必要ありません。データ型は，C 言語では char[5]，COBOL 言語では PIC X（5）で宣言しています。

参照する場合は，変数名称 SQLSTATE を指定してください。

また，複数接続機能を使用した環境では，DECLARE CONNECTION HANDLE SET 文で SQLSTATE が使用する接続ハンドルを宣言しておく必要があります。

設定される SQLSTATE の値は，クライアント環境変数 PDSTANDARDSQLSTATE 及びシステム共通定義 pd_standard_sqlstate の設定によって変わります。SQLSTATE については，マニュアル「HiRDB メッセージ」を参照してください。

クラスの意味とサブクラスの関係を次に示します。

クラス	サブクラス	設定条件
00	000	正常終了
01	nnn	正常終了（ただし，警告あり）
02	000	データがない
40	nnn	異常終了（トランザクションはロールバックした）
R2	000	データがない（ただし，リストを使用した検索で，リスト作成時には存在した行が返らなかった場合）
mm	nnn	異常終了

注

mm 及び nnn の意味を次に示します。

mm：マニュアル「HiRDB メッセージ」で示す SQLSTATE のクラスが設定されます。

nnn：マニュアル「HiRDB メッセージ」で示す SQLSTATE のサブクラスが設定されます。

ただし，mm 及び nnn は HiRDB の機能拡張によって値が変わる場合があります。

6.17 PDCNCTHDL 型変数の宣言

機能

複数接続機能を使用した環境で，使用する接続ハンドル型の変数を宣言します。

6.17.1 C 言語の場合

(1) 形式 1

```
PDCNCTHDL 接続ハンドル変数名
```

(2) オペランド

(a) 接続ハンドル変数名

使用する接続ハンドルの名前を指定します。

(3) 共通規則

1. PDCNCTHDL 型変数は，複数接続機能を使用した環境で SQL を実行する場合に宣言します。

(4) 留意事項

1. PDCNCTHDL 型変数は，埋込み SQL 宣言節で宣言します。
2. PDCNCTHDL 型変数は，配列として定義できません。

(5) 使用例

CnctHdl という名称の PDCNCTHDL 型変数を宣言します。

```
EXEC SQL BEGIN DECLARE SECTION;
    PDCNCTHDL    CnctHdl;
EXEC SQL END DECLARE SECTION;
```

6.17.2 COBOL 語の場合

(1) 形式 2

```
接続ハンドル変数名 SQL TYPE IS PDCNCTHDL.
```

(2) オペランド

(a) 接続ハンドル変数名

使用する接続ハンドルの名前を指定します。

(3) 共通規則

1. PDCNCTHDL 型変数は，複数接続機能を使用した環境で SQL を実行する場合に宣言します。

(4) 留意事項

1. PDCNCTHDL 型変数は，埋込み SQL 宣言節で宣言します。
2. PDCNCTHDL 型変数は，OCCURS 句を使用できません。

(5) 使用例

CONNECTHDL という名称の PDCNCTHDL 型変数を宣言します。

```
EXEC SQL BEGIN DECLARE SECTION END-EXEC
  01 CONNECTHDL SQL TYPE IS PDCNCTHDL.
EXEC SQL END DECLARE SECTION END-EXEC
```

6.18 INSTALL JAR（JAR ファイルの登録）

6.18.1 INSTALL JAR の形式と規則

(1) 機能

HiRDB サーバに JAR ファイルを登録します。接続している認可識別子に対応する JAR ファイル格納ディレクトリに登録してください。

(2) 形式

```
INSTALL JAR {：埋込み変数|'文字列'}
```

(3) オペランド

(a) {：埋込み変数|'文字列'}

登録する JAR ファイルの名称を指定します。JAR ファイルは，絶対パス名又は相対パス名で指定してください。

：埋込み変数

JAR ファイルの名称を VARCHAR 型の埋込み変数で指定します。

既定文字集合以外の文字集合は指定できません。

'文字列'

JAR ファイルの名称を文字列定数で指定します。

(4) 共通規則

1. INSTALL JAR を実行する前に，HiRDB サーバに接続しておく必要があります。
2. エラーコードは SQLCODE に返されます。
3. ほかのサーバマシンの JAR ファイルは指定できません。
4. ワイルドカードは使用できません。
5. 既に同じ名称の JAR ファイルが登録されている場合，上書きはしないでエラーとします。
6. INSTALL JAR は，トランザクションの開始前に実行してください。

(5) 使用例

JAR ファイル（c:¥work¥sampleproc.jar）を埋込み変数で指定して登録します。

```
EXEC SQL BEGIN DECLARE SECTION ;
struct {
    short  len ;
    char   str[256] ;
} filename ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL CONNECT ;
strcpy(filename.str,"c:¥work¥sampleproc.jar") ;
filename.len = strlen(filename.str) ;
EXEC SQL INSTALL JAR :filename ;
```

6.19 REPLACE JAR（JAR ファイルの再登録）

6.19.1 REPLACE JAR の形式と規則

(1) 機能

HiRDB サーバに JAR ファイルを再登録します。接続している認可識別子に対応する JAR ファイル格納ディレクトリに再登録してください。

(2) 形式

```
REPLACE JAR {：埋込み変数|'文字列'}
```

(3) オペランド

(a) {：埋込み変数|'文字列'}

再登録する JAR ファイルの名称を指定します。JAR ファイルは，絶対パス名又は相対パス名で指定してください。

：埋込み変数

JAR ファイルの名称を VARCHAR 型の埋込み変数で指定します。

既定文字集合以外の文字集合は指定できません。

'文字列'

JAR ファイルの名称を文字列定数で指定します。

(4) 共通規則

1. REPLACE JAR を実行する前に，HiRDB サーバに接続しておく必要があります。
2. エラーコードは SQLCODE に返されます。
3. ほかのサーバマシンの JAR ファイルは指定できません。
4. ワイルドカードは使用できません。
5. 既に同じ名称の JAR ファイルが登録されている場合は上書きします。
6. REPLACE JAR は，トランザクションの開始前に実行してください。

(5) 使用例

JAR ファイル（c:¥work¥sampleproc.jar）を埋込み変数で指定して再登録します。

```
EXEC SQL BEGIN DECLARE SECTION ;
struct {
    short  len ;
    char   str[256] ;
} filename ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL CONNECT ;
strcpy(filename.str,"c:¥work¥sampleproc.jar") ;
filename.len = strlen(filename.str) ;
EXEC SQL REPLACE JAR :filename ;
```

6.20 REMOVE JAR（JAR ファイルの削除）

6.20.1 REMOVE JAR の形式と規則

(1) 機能

HiRDB サーバの JAR ファイルを削除します。接続している認可識別子に対応する JAR ファイル格納ディレクトリから削除してください。

(2) 形式

```
REMOVE JAR {：埋込み変数|'文字列'}
```

(3) オペランド

(a) {：埋込み変数|'文字列'}

削除する JAR ファイルの名称を指定します。JAR ファイルは，絶対パス名又は相対パス名では指定できません。

：埋込み変数

JAR ファイルの名称を VARCHAR 型の埋込み変数で指定します。

既定文字集合以外の文字集合は指定できません。

'文字列'

JAR ファイルの名称を文字列定数で指定します。

(4) 共通規則

1. REMOVE JAR を実行する前に，HiRDB サーバに接続しておく必要があります。
2. エラーコードは SQLCODE に返されます。
3. ほかのサーバマシンの JAR ファイルは指定できません。
4. ワイルドカードは使用できません。
5. REMOVE JAR は，トランザクションの開始前に実行してください。

(5) 使用例

JAR ファイル（sampleproc.jar）を埋込み変数で指定して削除します。

```
EXEC SQL BEGIN DECLARE SECTION ;
struct {
```

```
    short  len ;
    char   str[256] ;
} filename ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL CONNECT ;
strcpy(filename.str,"sampleproc.jar") ;
filename.len = strlen(filename.str) ;
EXEC SQL REMOVE JAR :filename ;
```

6.21　INSTALL CLIB（外部Ｃライブラリファイルの新規登録）

6.21.1　INSTALL CLIB の形式と規則

(1)　機能

外部Ｃストアドルーチンを実行するために，UAP 実行マシンにある，外部Ｃストアドルーチンを実装した外部Ｃライブラリファイルを HiRDB サーバに新規登録します。

(2)　形式

```
INSTALL CLIB　{：埋込み変数｜'文字列'}
```

(3)　オペランド

(a)　{：埋込み変数｜'文字列'}

登録する外部Ｃライブラリファイルの名称を指定します。外部Ｃライブラリファイルの名称は，絶対パス名又は相対パス名で指定してください。

：埋込み変数

外部Ｃライブラリファイルの名称を VARCHAR 型の埋込み変数で指定します。

既定文字集合以外の文字集合は指定できません。

'文字列'

外部Ｃライブラリファイルの名称を文字列定数で指定します。

(4)　共通規則

1. INSTALL CLIB を実行する前に，HiRDB サーバに接続しておく必要があります。
2. エラーコードは SQLCODE に返されます。
3. ほかのサーバマシンの外部Ｃライブラリファイルは指定できません。
4. ワイルドカードは使用できません。
5. 既に同じ名称の外部Ｃライブラリファイルが登録されている場合，上書きはしないでエラーとします。
6. INSTALL CLIB は，トランザクションの開始前に実行してください。
7. 外部Ｃライブラリファイルの登録先と同じプラットフォームで，外部Ｃライブラリファイルを作成してください。

(5) 使用例

外部Ｃライブラリファイル（c:¥work¥sampleproc.dll）を埋込み変数で指定して新規登録します。

```
EXEC SQL BEGIN DECLARE SECTION ;
struct {
    short  len ;
    char   str[256] ;
} filename ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL CONNECT ;
strcpy(filename.str,"c:¥¥work¥¥sampleproc.dll") ;
filename.len = strlen(filename.str) ;
EXEC SQL INSTALL CLIB :filename ;
```

6.22 REPLACE CLIB（外部Ｃライブラリファイルの再登録）

6.22.1 REPLACE CLIB の形式と規則

(1) 機能

外部Ｃストアドルーチンを実行するために，UAP 実行マシンにある，外部Ｃストアドルーチンを実装した外部Ｃライブラリファイルを HiRDB サーバに再登録します。

(2) 形式

```
REPLACE CLIB {：埋込み変数|'文字列'}
```

(3) オペランド

(a) {：埋込み変数｜'文字列'}

再登録する外部Ｃライブラリファイルの名称を指定します。外部Ｃライブラリファイルの名称は，絶対パス名又は相対パス名で指定してください。

：埋込み変数

外部Ｃライブラリファイルの名称を VARCHAR 型の埋込み変数で指定します。

既定文字集合以外の文字集合は指定できません。

'文字列'

外部Ｃライブラリファイルの名称を文字列定数で指定します。

(4) 共通規則

1. REPLACE CLIB を実行する前に，HiRDB サーバに接続しておく必要があります。
2. エラーコードは SQLCODE に返されます。
3. ほかのサーバマシンの外部Ｃライブラリファイルは指定できません。
4. ワイルドカードは使用できません。
5. 既に同じ名称の外部Ｃライブラリファイルが登録されている場合は上書きします。
6. HiRDB サーバに外部Ｃライブラリファイルが登録されていない場合，外部Ｃライブラリファイルを HiRDB サーバに登録します。
7. REPLACE CLIB は，トランザクションの開始前に実行してください。
8. 外部Ｃライブラリファイルの再登録先と同じプラットフォームで，外部Ｃライブラリファイルを作成してください。

9. トランザクションの実行中は，Cライブラリファイルの再登録を行わないでください。

(5) 使用例

外部Cライブラリファイル（c:¥work¥sampleproc.dll）を埋込み変数で指定して再登録します。

```
EXEC SQL BEGIN DECLARE SECTION ;
struct {
    short  len ;
    char   str[256] ;
} filename ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL CONNECT ;
strcpy(filename.str,"c:¥¥work¥¥sampleproc.dll") ;
filename.len = strlen(filename.str) ;
EXEC SQL REPLACE CLIB :filename ;
```

6.23　REMOVE CLIB（外部 C ライブラリファイルの削除）

6.23.1　REMOVE CLIB の形式と規則

(1)　機能

HiRDB サーバに登録されている，外部 C ストアドルーチンを実装した外部 C ライブラリファイルを削除します。

(2)　形式

```
REMOVE CLIB　{：埋込み変数｜'文字列'}
```

(3)　オペランド

(a)　{：埋込み変数｜'文字列'}

削除する外部 C ライブラリファイルの名称を指定します。ディレクトリパスは指定しないで，ファイル名だけを指定してください。

：埋込み変数

外部 C ライブラリファイルの名称を VARCHAR 型の埋込み変数で指定します。

既定文字集合以外の文字集合は指定できません。

'文字列'

外部 C ライブラリファイルの名称を文字列定数で指定します。

(4)　共通規則

1. REMOVE CLIB を実行する前に，HiRDB サーバに接続しておく必要があります。
2. エラーコードは SQLCODE に返されます。
3. ワイルドカードは使用できません。
4. REMOVE CLIB は，トランザクションの開始前に実行してください。
5. トランザクションの実行中は，Cライブラリファイルの削除を行わないでください。

(5)　使用例

外部 C ライブラリファイル（sampleproc.dll）を埋込み変数で指定して削除します。

```
EXEC SQL BEGIN DECLARE SECTION ;
struct {
```

```
    short  len ;
    char   str[256] ;
} filename ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL CONNECT ;
strcpy(filename.str,"sampleproc.dll") ;
filename.len = strlen(filename.str) ;
EXEC SQL REMOVE CLIB:filename ;
```

6.24 DECLARE AUDIT INFO SET（ユーザ任意接続情報の設定）

6.24.1 DECLARE AUDIT INFO SET の形式と規則

(1) 機能

HiRDB サーバにアクセスするアプリケーションのアカウント情報など，ユーザ任意の接続情報を設定します。設定したユーザ任意接続情報は解除するまで有効となり，SQL 実行時に HiRDB サーバの監査証跡に出力されます。

(2) 形式

```
DECLARE AUDIT INFO SET  POS=：埋込み変数,
                            INF=：埋込み変数〔：標識変数〕
```

(3) オペランド

(a) POS=：埋込み変数

情報を設定したいユーザ付加情報の番号を INTEGER 型の埋込み変数で指定します。各ユーザ付加情報に該当する番号は次のとおりです。

- 1：ユーザ付加情報 1
- 2：ユーザ付加情報 2
- 3：ユーザ付加情報 3

(b) INF=：埋込み変数〔：標識変数〕

ユーザ任意接続情報を VARCHAR 型（領域長 100 バイト以内）の埋込み変数で指定します。設定済みの情報を解除する場合は，ナル値を指定してください。ユーザ任意接続情報にはユーザ付加情報 1～3 があり，それぞれ監査証跡に出力されます。

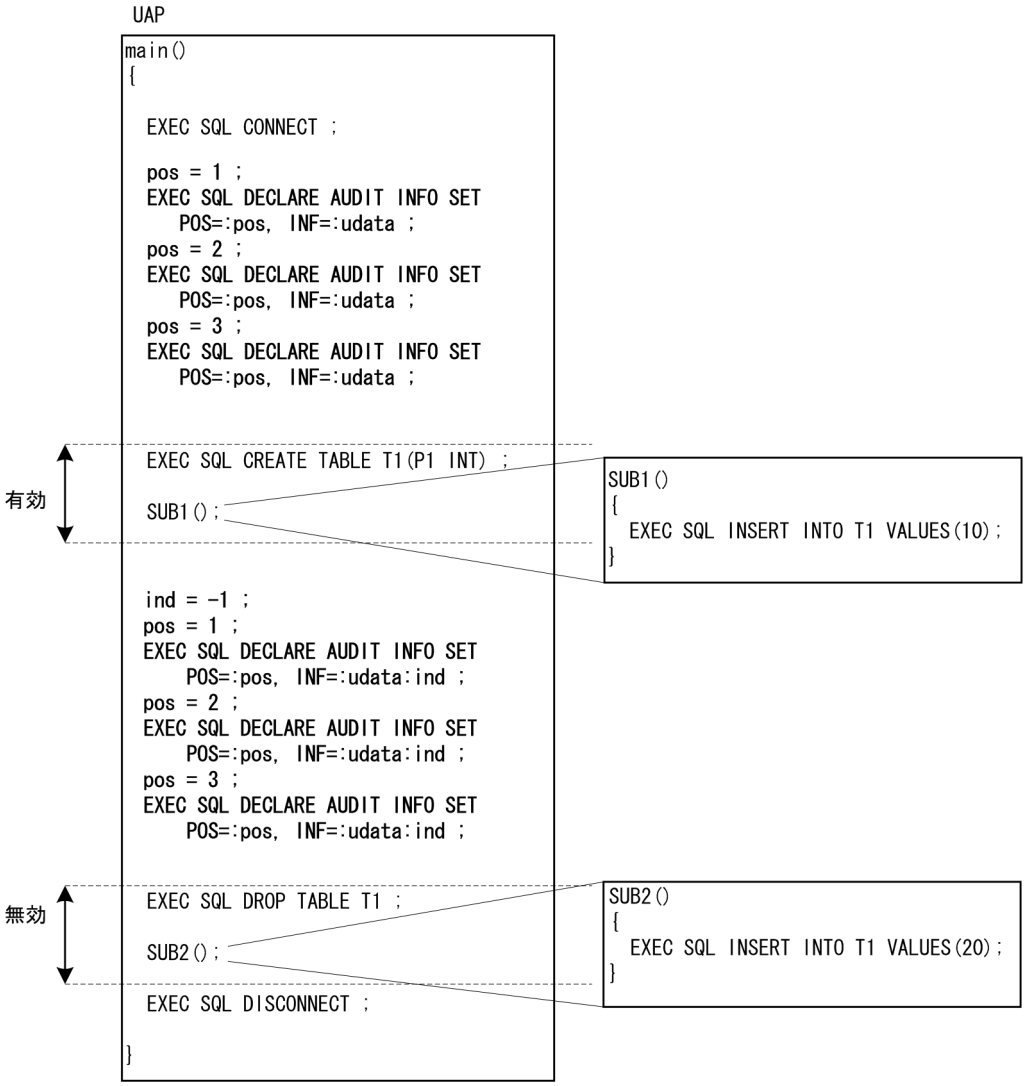
既定文字集合以外の文字集合は指定できません。

(4) 共通規則

1. 使用する埋込み変数は，埋込み変数宣言節で宣言してください。
2. エラーコードは SQLCODE に返されます。
3. ユーザ任意接続情報が既に設定されている状態で DECLARE AUDIT INFO SET を再度実行した場合は，設定済みのユーザ任意接続情報が更新されます。

4. 設定したユーザ任意接続情報は，DECLARE AUDIT INFO SET でナル値を指定して解除するまで有効です。DECLARE AUDIT INFO SET で設定したユーザ任意接続情報の有効範囲を，次の図に示します。

図 6-6　DECLARE AUDIT INFO SET で設定したユーザ任意接続の有効範囲



5. CONNECT 実行前に DECLARE AUDIT INFO SET でユーザ任意接続情報を設定した場合は，CONNECT 時にその情報が引き継がれます。

6. ユーザ任意接続情報が設定されている状態で DISCONNECT を実行しても，ユーザ任意接続情報は解除されません。解除する場合は，INF にナル値を指定して DECLARE AUDIT INFO SET を実行してください。

7. 複数接続時は，接続ハンドル単位でユーザ任意接続情報を設定してください。

(5)　留意事項

1. CONNECT 文の実行時には，ユーザ任意接続情報は監査証跡に出力されません。

(6) 使用例

(例1) ユーザ任意接続情報（ユーザ付加情報1～3）を埋込み変数で指定して設定します。

```
EXEC SQL BEGIN DECLARE SECTION ;
SQL TYPE IS VARCHAR(100) udata ;
long pos ;
EXEC SQL END DECLARE SECTION ;

strcpy(udata.str,"userA") ;
udata.len = strlen(udata.str) ;
pos = 1;
EXEC SQL DECLARE AUDIT INFO SET POS=:pos, INF=:udata ;
strcpy(udata.str," user data B ") ;
udata.len = strlen(udata.str) ;
pos = 2;
EXEC SQL DECLARE AUDIT INFO SET POS=:pos, INF=:udata ;
strcpy(udata.str," user data C ") ;
udata.len = strlen(udata.str) ;
pos = 3;
EXEC SQL DECLARE AUDIT INFO SET POS=:pos, INF=:udata ;
```

(例2) ユーザ任意接続情報（ユーザ付加情報1～3）の設定内容を解除します。

```
EXEC SQL BEGIN DECLARE SECTION ;
SQL TYPE IS VARCHAR(100) udata ;
long pos ;
short ind ;
EXEC SQL END DECLARE SECTION ;

ind = -1;
pos = 1;
EXEC SQL DECLARE AUDIT INFO SET POS=:pos, INF=:udata:ind ;
pos = 2;
EXEC SQL DECLARE AUDIT INFO SET POS=:pos, INF=:udata:ind ;
pos = 3;
EXEC SQL DECLARE AUDIT INFO SET POS=:pos, INF=:udata:ind ;
```

7 ルーチン制御 SQL

この章では，ルーチン制御 SQL の構文形式，及び文法について説明します。

7.1　全般規則

7.1.1　ルーチン制御 SQL の全般規定

(1)　ルーチン制御 SQL の種類と機能

ルーチン制御 SQL は，ルーチン定義の SQL 手続き文中に指定できる SQL です。

ルーチン制御 SQL の種類と機能を次の表に示します。

表 7-1　ルーチン制御 SQL の種類と機能

種　類	機　能
代入文　形式 1（SQL 変数，又は SQL パラメタへの値の代入）	SQL 変数又は SQL パラメタに値を代入します。※
複合文（複数文実行）	複数の SQL 文をまとめて一つの SQL 文として実行します。
IF 文（条件分岐による実行）	条件に合った SQL 文を実行します。
LEAVE 文（文の途中終了）	複合文又は WHILE 文の途中から抜けて，それらの文の実行を終了します。
RETURN 文（関数の戻り値の返却）	関数の戻り値を返却します。
WHILE 文（文の繰り返し）	SQL 文の実行を繰り返します。
FOR 文（各行に対する文の繰返し）	表の各行に対して SQL 文の実行を繰り返します。
WRITE LINE 文（ファイルへの文字列出力）	ファイルへ文字列を出力します。
SIGNAL 文（エラーの通知）	エラーを通知します。
RESIGNAL 文（エラーの再通知）	エラーを再通知します。

注※　SQL 手続き文中の代入文については「代入文　形式 1（SQL 変数，又は SQL パラメタへの値の代入）」を参照してください。

表「ルーチン制御 SQL の種類と機能」に示したルーチン制御 SQL 以外に，ルーチン中に SQL 手続き文として指定できる SQL 文を次に示します。ただし，関数本体中には指定できません。

- CALL 文（手続きの呼出し）
- CLOSE 文（カーソルクローズ）
- DECLARE CURSOR（カーソル宣言）
- DELETE 文（行削除）
- FETCH 文（データの取り出し）
- INSERT 文（行挿入）

- OPEN 文（カーソルオープン）
- PURGE TABLE 文（全行削除）
- 1 行 SELECT 文（1 行検索）
- UPDATE 文（データ更新）
- COMMIT 文（トランザクションの正常終了）
- LOCK 文（表の排他制御）
- ROLLBACK 文（トランザクションの取り消し）

7.2 複合文（複数文実行）

7.2.1 複合文の形式と規則

(1) 機能

複数の SQL 文をまとめて，一つの SQL 文（複合文）として実行します。

(2) 形式

```
 〔開始ラベル：〕
BEGIN
 〔 {SQL変数宣言;｜カーソル宣言;｜条件宣言;｜ハンドラ宣言;} 〕…
 〔SQL手続き文;〕…

END 〔終了ラベル〕
SQL変数宣言：：＝DECLARE SQL変数名〔, SQL変数名〕…データ型 〔DEFAULT句〕
DEFAULT句：：＝DEFAULT 〔既定値〕
条件宣言：：＝DECLARE 条件名 CONDITION 〔FOR SQLCODE値〕
ハンドラ宣言：：＝DECLARE ハンドラ種別
                    HANDLER FOR 条件値〔, 条件値〕… ハンドラ動作
ハンドラ種別：：＝ {CONTINUE｜EXIT}
条件値：：＝ {SQLERROR｜NOT FOUND｜条件名｜SQLCODE値}
ハンドラ動作：：＝SQL手続き文
SQLCODE値：：＝SQLCODE 〔VALUE〕 整数定数
```

(3) オペランド

(a) 〔開始ラベル〕

複合文の文ラベルを指定します。

(b) BEGIN

複合文の開始を指定します。

(c) SQL 変数宣言：：＝ DECLARE SQL 変数名〔, SQL 変数名〕…データ型 〔DEFAULT 句〕

複合文中で使用する SQL 変数を宣言します。SQL 変数が割り当てられたとき，初期値として SQL 変数の既定値が設定されます。SQL 変数の既定値は DEFAULT 句で指定します。DEFAULT 句を省略した場合，SQL 変数の既定値はナル値になります。

SQL 変数名

宣言する SQL 変数の名称を指定します。

データ型

宣言する SQL 変数のデータ型を指定します。

SQL 変数宣言の規則

1. SQL 変数宣言中で宣言する SQL 変数の名称には，ルーチンのパラメタ名と同じ名称は指定できません。
2. 同じ複合文中に直接含まれる SQL 変数宣言中で宣言する SQL 変数の名称は，重複して指定できません。
3. 複合文中で宣言した SQL 変数は，その複合文の最初に割り当てられ，最後に解放されます。
4. SQL 変数の有効範囲は，その SQL 変数が宣言されている複合文の中，及び同じ複合文の中に宣言されているハンドラ宣言のハンドラ動作の中になります。また，その複合文の中の SQL 手続き文に複合文を指定した場合は，内側の複合文の中でも有効です。
5. 複合文の中の SQL 手続き文に複合文を指定した場合，外側の複合文の中で宣言した SQL 変数と，内側の複合文の中で宣言した SQL 変数が同じ名称のときは，内側の複合文の中では内側で宣言した SQL 変数が有効となります。内側の複合文が終了したら，外側で宣言した SQL 変数が有効となります。
6. SQL 変数宣言中で宣言する SQL 変数のデータ型には，BOOLEAN は指定できません。

DEFAULT 句：：＝ DEFAULT 〔既定値〕

DEFAULT 句の規則については，「CREATE TABLE（表定義）」の DEFAULT 句の規則を参照してください。

既定値として CURRENT_TIMESTAMP〔(小数秒精度)〕 USING BES 又は CURRENT TIMESTAMP〔(小数秒精度)〕 USING BES は指定できません。

(d) カーソル宣言

複合文中で使用するカーソルを宣言します。

カーソル宣言の規則

1. 同じ複合文中に直接含まれる，カーソル宣言中で宣言するカーソルの名称は，重複して指定できません。
2. 複合文中で宣言したカーソルは，その複合文の最初に割り当てられ，最後に解放されます。ただし，WITH RETURN 指定をして宣言したカーソルは，解放されません。
3. カーソルの有効範囲は，そのカーソルが宣言されている複合文の中，及び同じ複合文の中に宣言されているハンドラ宣言のハンドラ動作の中になります。また，その複合文の中の SQL 手続き文に複合文を指定した場合は，内側の複合文の中でも有効です。
4. 複合文の中の SQL 手続き文に複合文を指定した場合，外側の複合文の中で宣言したカーソルと，内側の複合文の中で宣言したカーソルが同じ名称のときは，内側の複合文の中では内側で宣言したカーソルが有効となります。内側の複合文が終了したら，外側で宣言したカーソルが有効となります。

(e) 条件宣言：：＝ DECLARE 条件名 CONDITION 〔FOR SQLCODE 値〕

ハンドラ宣言，SIGNAL 文，又は RESIGNAL 文の中で使用する条件名と，それに対応する SQLCODE の値を宣言します。

条件名

宣言する条件の名称を指定します。

FOR SQLCODE 値

宣言する条件に対応づける SQLCODE の値を指定します。

条件宣言の規則

1. 同じ複合文中に直接含まれるほかの条件宣言に，同じ条件名を重複して指定できません。
2. 条件名の有効範囲は，その条件名が宣言されている複合文の中，及び同じ複合文の中に宣言されているハンドラ宣言のハンドラ動作の中になります。また，その複合文の中の SQL 手続き文に複合文を指定した場合は，内側の複合文の中でも有効です。
3. 複合文の中の SQL 手続き文に複合文を指定した場合，外側の複合文の中で宣言した条件名と同じ条件名を使用した条件宣言は，内側の複合文の中で宣言できません。
4. 同じ複合文中に直接含まれる条件宣言を複数指定する場合，同じ SQLCODE 値は指定できません。
5. SIGNAL 文，又は RESIGNAL 文の中で使用する条件名を宣言する場合は，"FOR SQLCODE 値"を省略してください。指定した場合はエラーとなります。

(f) ハンドラ宣言：：＝ DECLARE ハンドラ種別 HANDLER FOR 条件値〔，条件値〕… ハンドラ動作

複合文中で例外処理をするためのハンドラを宣言します。

複合文中の SQL 文を実行した結果の SQLCODE 値，SIGNAL 文，又は RESIGNAL 文の条件名がハンドラ宣言で指定した条件値と一致すると，ハンドラが制御を受け取り，ハンドラ動作を実行します。

ハンドラ種別：：＝ {CONTINUE ｜ EXIT}

CONTINUE

ハンドラ動作を実行した後に，例外を発生した SQL 手続き文の次の SQL 手続き文に制御を移します。ただし，例外を発生した SQL 手続き文がルーチン制御 SQL の IF 文，又は WHILE 文の場合，END IF，又は END WHILE〔終了ラベル〕の次の SQL 手続き文に制御を移します。

EXIT

ハンドラ動作を実行した後に，ハンドラ宣言を指定した複合文の最後に制御を移します。

条件値：：＝ {SQLERROR ｜ NOT FOUND ｜条件名｜ SQLCODE 値}

ハンドラが有効になる条件を指定します。

SQLERROR

SQLERROR が発生した場合に，ハンドラを呼び出すときに指定します。SQLERROR は，SQLCODE<0 となる場合に対応しています。

NOT FOUND

NOT FOUND が発生した場合に，ハンドラを呼び出すときに指定します。NOT FOUND は，SQLCODE=100 となる場合に対応しています。

条件名

ハンドラが呼び出される条件となる条件名を指定します。

条件名は，条件宣言であらかじめ定義されていて，かつこのハンドラ宣言を有効範囲として含んでいる必要があります。

条件宣言の中で条件名に対応する SQLCODE 値が定義されている場合は，SQLCODE がその値と一致したときにハンドラが呼び出されます。条件宣言の中で条件名に対応する SQLCODE 値が定義されていない場合は，その条件名を指定した SIGNAL 文，又は RESIGNAL 文を実行したときだけハンドラが呼び出されます。

SQLCODE 値

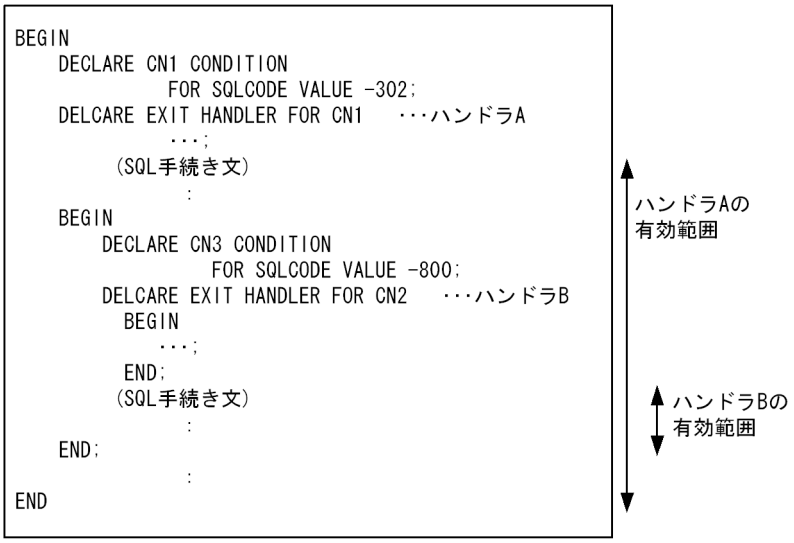
ハンドラが呼び出される条件となる SQLCODE の値を指定します。

ハンドラ動作：：= SQL 手続き文

ハンドラが呼び出されたときに実行する SQL 手続き文を指定します。

ハンドラ宣言の規則

1. ハンドラは，そのハンドラが宣言されている複合文の中の SQL 手続き文を有効範囲とします。その複合文の中の SQL 手続き文に複合文を指定した場合は，内側の複合文全体で有効となります。ただし，そのハンドラが宣言されている複合文の中の，ハンドラ宣言中の SQL 手続き文は無効となります。例を次に示します。



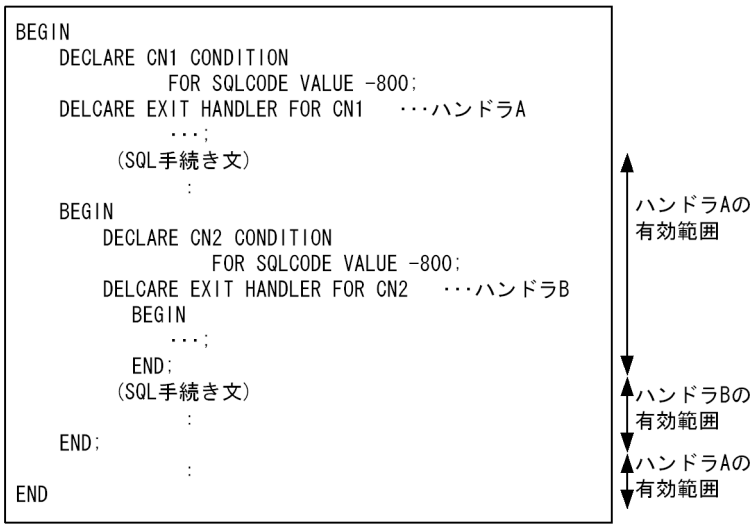
2. ハンドラ宣言の中で，条件値に SQLERROR，又は NOT FOUND のどちらかを指定した場合，同時に SQLCODE 値，又は条件名を指定できません。

3. ハンドラ宣言の中で，同じ条件値は重複して指定できません。また，SQLCODE 値と同じ SQLCODE を表す条件名も指定できません。

4. 同じ複合文の中に含まれるほかのハンドラ宣言に，同じ条件を表す条件値は指定できません。

5. 条件値に SQLERROR，又は NOT FOUND を指定したハンドラ宣言を一般ハンドラ宣言，それ以外は特定ハンドラ宣言とします。同じ複合文の中に，同じ SQL の実行状態（異常終了，警告あり正常終了，又はデータなし）を示す条件値を指定した一般ハンドラ宣言と特定ハンドラ宣言が定義されている場合，特定ハンドラ宣言で指定した SQLCODE 値に対しては，特定ハンドラ宣言だけが有効になります。

6. 複合文の中の SQL 手続き文に複合文を指定した場合に，外側の複合文で宣言したハンドラ A と内側の複合文の中で宣言したハンドラ B とで，同じ SQLCODE，又は条件名を指定したときは，内側の複合文の中では内側のハンドラ B が有効となります。内側の複合文が終了したら，再び外側のハンドラ A が有効となります。例を次に示します。



7. ハンドラ動作が正常終了でない（SQLCODE=0 以外）場合，条件を満たすほかのハンドラがあれば呼び出します。

8. ハンドラ動作開始直後は，SQLCODE=0 となります。

(g) SQL 手続き文

複合文中で実行する SQL 手続き文を指定します。

(h) END〔終了ラベル〕

複合文の終了を指定します。終了ラベルには，文ラベルを指定します。

(i) SQLCODE 値：：＝ SQLCODE 〔VALUE〕 整数定数

SQLCODE の値を整数定数で指定します。

SQLCODE の値として，正常終了を表す値である 0 は指定できません。SQLCODE 値に指定する整数定数を次の表に示します。

表 7-2　SQLCODE 値に指定する整数定数

SQL 文の実行状態	SQLCODE の値
正常終了（警告あり）	＞ 0(≠100，110)
データがない	100
異常終了	＜ 0

HiRDB で発生する可能性がある，SQLCODE に対応するメッセージを次の表に示します。

表 7-3　HiRDB で発生する可能性がある，SQLCODE に対応するメッセージ

SQLCODE	対応するメッセージ ID
-yyy	KFPA11yyy
-1yyy	KFPA19yyy
-3yyy	KFPA18yyy
yyy	KFPA12yyy
+3yyy	KFPA13yyy

(4)　共通規則

1. 最も外側の SQL 手続き文に複合文を指定して，開始ラベルを省略した場合は，そのルーチンのルーチン識別子が文ラベルとして仮定されます。また，複合文の中の SQL 手続き文に複合文を指定した場合，内側の複合文に対する開始ラベルを省略したときは，文ラベルなしとして扱われます。
2. 終了ラベルを指定する場合は，開始ラベルと同じ名称の文ラベルを指定してください。
3. 文ラベルの有効範囲は，文ラベルを指定した複合文の開始から終了までの間です。その複合文中に含まれる，ほかの文の文ラベル，及びループ変数名と同じ名称の文ラベルは指定できません。ただし，複合文中にハンドラ宣言がある場合，そのハンドラ宣言中は除きます。同一名称の文ラベルの指定可否の例を次に示します。

```
AAA: BEGIN  ..........................................1
  DECLARE CN1 CONDITION FOR SQLCODE VALUE -800;
  DECLARE EXIT HANDLER FOR CN1
  AAA: BEGIN  ........................................2
     :
  END AAA;
  AAA: BEGIN  ........................................3
    DECLARE CN2 CONDITION FOR SQLCODE VALUE -800;
    DECLARE EXIT HANDLER FOR CN2
      :
  END AAA;
    :
END AAA
```

〔説明〕

2 は，1 と同一名称ですが，ハンドラ宣言中のため指定できます。

3 は，1 と同一名称であり，ハンドラ宣言中でないため指定できません。

4. 指定した SQL 手続き文は，指定した順序で実行されます。

5. SQL 手続きの実行中にエラーが発生した場合に，トランザクションが無効になるのは，そのエラーが暗黙的ロールバックありのときだけです。トリガ動作の SQL 手続き文の実行中にエラーが発生した場合は，必ず暗黙的ロールバックとなります。

6. SQL 手続き文の実行でエラーが発生した場合は，次の規則に従います。

   - 暗黙的ロールバックなしのエラーが発生した場合

     条件を満たすハンドラがあれば，そのハンドラの例外処理を実行します。条件を満たすハンドラがなければ，その時点で SQL ルーチンの実行を終了し，エラーを返します。それ以降の SQL 手続き文は実行されません。

   - 暗黙的ロールバックありのエラーが発生した場合

     条件を満たすハンドラがあっても，例外処理はしないで，その時点で SQL ルーチン，又はトリガの実行を終了し，エラーを返します。それ以降の SQL 手続き文は実行されません。

7. 複合文，及び FOR 文のネスト数は，最大 255 です。

8. SQL 変数名，カーソル名，及び条件名に，次の名称を指定する場合は，引用符（"）で名称を囲んでください。

   - CONDITION
   - EXIT
   - HANDLER

(5) 留意事項

1. 複合文は，SQL ルーチン中，及びトリガ中に指定できます。

(6) 使用例

1. 在庫表（ZAIKO）の数量が，1,000 個以上の商品の単価を 3 割引きに，0 の場合はその行を削除し，そのほかの場合は単価を 1 割引きにする手続き（PROC1）を定義します。

```
CREATE PROCEDURE PROC1(OUT OUTDATA INT)
BEGIN
  DECLARE CR1 CURSOR FOR SELECT SURYO FROM ZAIKO ;
  OPEN CR1 ;
  WHILE SQLCODE=0 DO
    FETCH CR1 INTO OUTDATA ;
    IF SQLCODE=0 THEN
      IF OUTDATA>=1000 THEN
        UPDATE ZAIKO SET TANKA = (1-0.3)*TANKA WHERE CURRENT OF CR1 ;
      ELSE IF OUTDATA=0 THEN
        DELETE FROM ZAIKO WHERE CURRENT OF CR1 ;
      ELSE
        UPDATE ZAIKO SET TANKA = (1-0.1)*TANKA WHERE CURRENT OF CR1 ;
      END IF ;
    END IF ;
```

```
  END;
END
```

2. 在庫表（ZAIKO）に対し，指定した商品コードの数量を更新する SQL 手続き（PROC2）を定義します。

   - 指定した数量が 0 以下である場合（手続き内で定義する条件 illegal_value になる場合）

     SIGNAL 文でエラーを発生させ，例外処理で出力パラメタにメッセージを設定します。また，SQL 手続きの実行は終了します。

   - 指定した商品コードのデータがない場合（NOT FOUND になる場合）

     例外処理で出力パラメタにメッセージを設定します。また，SQL 手続き文の実行は終了します。

   - 非ナル値制約ありの列を NULL 値で更新しようとした場合（SQLCODE=-210 になる場合）

     例外処理で出力パラメタにメッセージを設定します。また，SQL 手続き文の実行は継続します。

```
CREATE PROCEDURE PROC2(IN USCODE CHAR(4), IN USURYO INT,
                       OUT MSG MVARCHAR(255))
BEGIN
  DECLARE PSURYO INT;
  DECLARE illegal_value CONDITION ;
  DECLARE EXIT HANDLER FOR illegal_value
    SET MSG=M'数量として無効な値です。';
  DECLARE EXIT HANDLER FOR NOT FOUND
    SET MSG=M'指定された商品コードは登録されていません。';
  DECLARE CONTINUE HANDLER FOR SQLCODE VALUE -210
    SET MSG=M'非ナル値制約ありの列をNULL値で更新しようとしましたが,
              無視します。';
  SET MSG ='';
  IF USURYO<0 THEN
    SIGNAL illegal_value;
  ELSE
    UPDATE ZAIKO SET ZSURYO=USURYO WHERE ZSCODE=USCODE;
    SET MSG=MSG||M'処理が完了しました。 '
    SELECT ZSURYO INTO PSURYO FROM ZAIKO WHERE ZSCODE=USCODE;
    SET MSG=MSG||M'現在の数量：'||NUMEDIT(PSURYO,'<999999');
  END IF;
END
```

3. 在庫表（ZAIKO）に対し，新規の商品データを登録する SQL 手続き（PROC3）を定義します。

   なお，商品コード列（SCODE）は主キーであるものと仮定します。挿入するデータの商品コードが，既に登録されているデータの商品コードと重複する場合（SQLCODE=-803 の場合），例外処理でロールバックをさせて，出力パラメタにメッセージを設定します。また，SQL 手続き文の実行は終了します。

```
CREATE PROCEDURE PROC3(IN  USCODE CHAR(4) , IN  USNAME NCHAR(8),
                       IN  UCOL   NCHAR(1), IN  UTANKA INT,
                       OUT MSG    MVARCHAR(255))
BEGIN
  DECLARE EXIT HANDLER FOR SQLCODE VALUE -803
    BEGIN
      ROLLBACK;
      SET MSG=M'重複キー違反のため，ロールバックしました。';
    END;
  INSERT INTO ZAIKO VALUES(USCODE,USNAME,UCOL,UTANKA,0);
```

```
  SET MSG=M'登録が完了しました。';
END
```

7.3 IF 文（条件分岐による実行）

7.3.1 IF 文の形式と規則

(1) 機能

条件に合った SQL 文を実行します。

(2) 形式

```
IF 探索条件 THEN SQL手続き文;〔SQL手続き文;〕…
 〔ELSEIF 探索条件 THEN SQL手続き文;〔SQL手続き文;〕…〕
 〔ELSE SQL手続き文;〔SQL手続き文;〕…〕
END IF
```

(3) オペランド

(a) IF 探索条件 THEN SQL 手続き文;〔SQL 手続き文;〕…

探索条件

THEN 句に指定する SQL 手続き文を実行する条件を指定します。

SQL 手続き文

IF 句に指定した条件を満たす場合に実行する SQL 文を指定します。

(b) 〔ELSEIF 探索条件 THEN SQL 手続き文;〔SQL 手続き文;〕…〕

探索条件

THEN 句に指定する SQL 手続き文を実行する条件を指定します。

SQL 手続き文

IF 句の条件を満たさないで，ELSEIF 句に指定した条件を満たす場合に実行する SQL 文を指定します。

(c) 〔ELSE SQL 手続き文;〔SQL 手続き文;〕…〕

SQL 手続き文

IF 句，及び ELSEIF 句の条件を満たさない場合に実行する SQL 文を指定します。

(d) END IF

IF 文の終了を指定します。

(4) 共通規則

1. 指定した SQL 手続き文は，指定した順序で実行されます。SQL 手続き文の実行でエラーが発生した場合，それ以降の SQL 手続き文は実行されません。
2. 探索条件中に副問合せを指定できません。

(5) 留意事項

1. IF 文は，SQL ルーチン中に指定できます。

7.4 LEAVE 文（文の途中終了）

7.4.1 LEAVE 文の形式と規則

(1) 機能

複合文，WHILE 文，又は FOR 文の途中から抜けて，それらの文の実行を終了します。

(2) 形式

```
LEAVE 〔文ラベル〕
```

(3) オペランド

(a) 文ラベル

途中で抜けて実行を終了させる複合文，WHILE 文，又は FOR 文の文ラベルを指定します。

文ラベルを省略した場合は，文ラベルを省略した LEAVE 文を囲む最も内側の複合文，WHILE 文，又は FOR 文を途中終了します。

(4) 共通規則

1. 文ラベルには，LEAVE 文を含む文（複合文，WHILE 文，又は FOR 文）の開始ラベルを指定します。
2. ハンドラ動作を抜ける LEAVE 文は，ハンドラ動作中に指定できません。

(5) 留意事項

1. LEAVE 文は，SQL ルーチン中に指定できます。

7.5 RETURN 文（関数の戻り値の返却）

7.5.1 RETURN 文の形式と規則

(1) 機能

関数の戻り値を返却します。

(2) 形式

```
RETURN 戻り値
戻り値：：＝ {値式 | NULL}
```

(3) オペランド

(a) 戻り値：：＝ {値式 | NULL}

戻り値として，値式又は NULL を指定します。

(4) 共通規則

1. 戻り値は，関数定義時の RETURNS 句で指定されたデータ型にデータ変換されます。
2. RETURN 文は，手続き中には指定できません。
3. 戻り値に指定する値式中に，副問合せは指定できません。

(5) 留意事項

1. RETURN 文は，SQL 関数中に指定できます。

7.6 WHILE 文（文の繰り返し）

7.6.1 WHILE 文の形式と規則

(1) 機能

SQL 文の実行を繰り返します。

(2) 形式

```
〔開始ラベル：〕
WHILE 探索条件 DO
  SQL手続き文;〔SQL手続き文;〕…
END 〔WHILE〕 〔終了ラベル〕
```

(3) オペランド

(a) 〔開始ラベル〕

WHILE 文の文ラベルを指定します。

(b) 探索条件

SQL 手続き文の実行を繰り返す条件を指定します。

(c) SQL 手続き文

繰り返し実行する SQL 手続き文を指定します。

(d) END 〔WHILE〕 〔終了ラベル〕

WHILE 文の終了を指定します。終了ラベルには，文ラベルを指定します。

WHILE は指定してもしなくても同じです。

(4) 共通規則

1. 終了ラベルを指定する場合は，開始ラベルと同じ名称の文ラベルを指定してください。
2. 文ラベルの有効範囲は，WHILE 文の開始から終了までの間です。その WHILE 文中に含まれるほかの文の文ラベル，及びループ変数名と同じ名称の文ラベルは指定できません。ただし，SQL 手続き文中にハンドラ宣言がある場合，そのハンドラ宣言中は除きます。同一名称の文ラベルの指定可否の例を次に示します。

```
AAA: WHILE X < 100 DO
  BEGIN .....................................1
```

```
    DECLARE CN1 CONDITION FOR SQLCODE VALUE -800;
    DECLARE EXIT HANDLER FOR CN1
    AAA: BEGIN  ......................................2
        :
    END AAA;
    AAA: BEGIN  ......................................3
      DECLARE CN2 CONDITION FOR SQLCODE VALUE -800;
      DECLARE EXIT HANDLER FOR CN2
        :
    END AAA;
    SET X=X+1;
  END
END WHILE AAA
```

〔説明〕

2 は，1 と同一名称ですが，ハンドラ宣言中のため指定できます。

3 は，1 と同一名称であり，ハンドラ宣言中でないため指定できません。

3. 探索条件を評価し，結果が真の場合は SQL 手続き文を実行します。探索条件が偽，又は不定になるまで，繰り返し SQL 手続き文を実行します。
4. 指定した SQL 手続き文は，指定した順序で実行されます。SQL 手続き文の実行でエラーが発生した場合，それ以降の SQL 手続き文は実行されません。WHILE 文の実行も終了します。
5. 探索条件中に副問合せを指定できません。

(5) 留意事項

1. WHILE 文は，SQL ルーチン中に指定できます。

7.7 FOR 文（各行に対する文の繰返し）

7.7.1 FOR 文の形式と規則

(1) 機能

表の各行に対して SQL 文の実行を繰り返します。

(2) 形式

```
 〔開始ラベル：〕
FOR ループ変数名 AS
 〔カーソル名 CURSOR 〔WITH HOLD〕 FOR〕
   カーソル指定　形式 1
   〈排他オプション〉
     〔 〔 {WITH  {SHARE | EXCLUSIVE} LOCK
         | WITHOUT LOCK 〔 {WAIT | NOWAIT} 〕 } 〕
     〔 {WITH ROLLBACK | NO WAIT} 〕 〕
         〔FOR {UPDATE 〔OF列名〔, 列名〕…〕 〔NOWAIT〕 | READ ONLY} 〕
         〔UNTIL DISCONNECT〕
DO
  SQL手続き文;〔SQL手続き文;〕…
END FOR 〔終了ラベル〕
```

(3) オペランド

(a) 〔開始ラベル〕

FOR 文の文ラベルを指定します。

(b) ループ変数名

暗黙的に宣言する SQL 変数の修飾子を指定します。

FOR 文実行時，カーソル指定の導出列名を変数名とする SQL 変数が暗黙的に宣言されます。暗黙的に宣言された SQL 変数はループ変数名での修飾ができます。

ループ変数名はラベル名の規則に従います。したがって，ラベル名に関する制限はループ変数名に対しても適用されます。

(c) カーソル名

カーソルの名称を指定します。

カーソル名を省略した場合は，HiRDB が固有のカーソル名を生成します。

カーソル名については，「名前の指定」を参照してください。

(d) 〔WITH HOLD〕

ホールダブルカーソルを使用する場合に指定します。

WITH HOLD は，UNTIL DISCONNECT を指定した場合の機能と同じなので，説明は UNTIL DISCONNECT を参照してください。また，この指定が UNTIL DISCONNECT と重複しているかどうかによる機能差はありません。

ホールダブルカーソルに関する制限事項については，「DECLARE CURSOR　形式 1（カーソル宣言）」を参照してください。

(e) カーソル指定　形式 1

問合せの内容を表すカーソルを指定します。

カーソル指定については「カーソル指定　形式 1」を参照してください。

FOR 文で指定するカーソル指定の規則

- 導出列に名前のない列は指定できません。名前のない列を導出列として指定する場合は，AS 句を指定し，導出列に対して別名を付与してください。
- 導出列名が重複する導出列を指定できません。
- 導出列に〔表指定.〕ROW を指定できません。
- 導出列に，FROM 句に平坦化の指定がない添字なし繰返し列を指定できません。

カーソル指定の導出列が次に示す項目から導き出された列で，AS 列名を省略した場合，その列は名前のない列となります。導出列がスカラ副問合せの場合，導出列名はスカラ副問合せの選択式の導出列名に依存します。

- スカラ演算（ウィンドウ関数を含む）
- 関数呼出し
- 集合関数
- 定数
- USER 値関数
- CURRENT_DATE 値関数
- CURRENT_TIME 値関数
- CURRENT_TIMESTAMP 値関数
- コンポネント指定
- GET_JAVA_STORED_ROUTINE_SOURCE 指定
- WRITE 指定
- SQL 変数
- SQL パラメタ

(f)　排他オプション

問合せをする場合の排他制御モードと，ほかのユーザが資源を占有していた場合の処置を指定します。

排他オプションについては，「排他オプション」を参照してください。

(g)　〔FOR {UPDATE 〔OF 列名〔，列名〕…〕〔NOWAIT〕 | READ ONLY}〕

FOR UPDATE 〔OF 列名〔，列名〕…〕を FOR UPDATE 句といいます。

FOR UPDATE 句，及び FOR READ ONLY については，「DECLARE CURSOR　形式 1（カーソル宣言）」を参照してください。

(h)　〔UNTIL DISCONNECT〕

ホールダブルカーソルを使用する場合に指定します。

この指定をしたときの機能は，WITH HOLD を指定した場合と全く同じです。また，この指定が WITH HOLD と重複しているかどうかによっての機能差はありません。

また，ホールダブルカーソルに関する制限事項については，「DECLARE CURSOR　形式 1（カーソル宣言）」を参照してください。

(i)　SQL 手続き文

繰り返し実行する SQL 手続き文を指定します。

FOR 文の中で指定する SQL 手続き文の規則

- FOR 文のカーソルを指定した OPEN 文，FETCH 文，及び CLOSE 文は指定できません。
- ループ変数名を指定した LEAVE 文は指定できません。
- FOR 文のカーソルがホールダブルカーソルではない場合，COMMIT 文，ROLLBACK 文，及び PURGE TABLE 文は指定できません。また，CALL 文でこれらを指定した手続きが呼び出された場合は実行時エラーとなります。
- 指定した SQL 手続き文は指定した順序で実行されます。SQL 手続き文の実行でエラーが発生した場合，それ以降の SQL 手続き文は実行されません。FOR 文の実行も終了します。

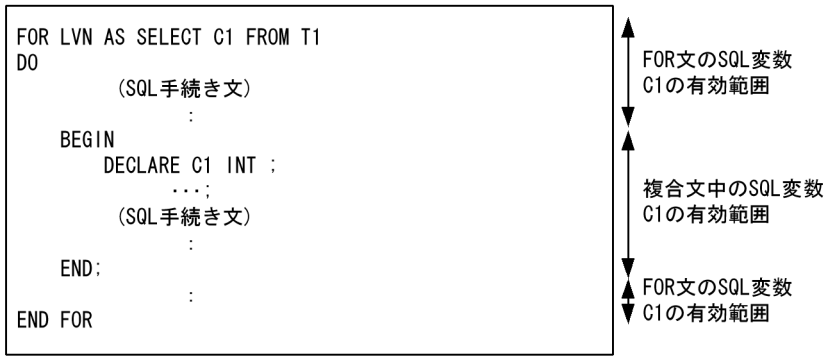
(j)　END FOR 〔終了ラベル〕

FOR 文の終了を指定します。終了ラベルには，文ラベルを指定します。

(4)　共通規則

1. 終了ラベルを指定する場合は，開始ラベルと同じ名称の文ラベルを指定してください。
2. 文ラベル，及びループ変数名の有効範囲は，FOR 文の開始から終了までの間です。その FOR 文中に含まれるほかの文の文ラベル，ループ変数名と同じ名称の文ラベル，及びループ変数名は指定できません。ただし，SQL 手続き文中にハンドラ宣言がある場合，そのハンドラ宣言中は除きます。

3. FOR 文中で暗黙的に宣言した SQL 変数，及び明示的に宣言したカーソルは，その FOR 文の最初に割り当てられ，最後に解放されます。
4. FOR 文で暗黙的に宣言した SQL 変数，及び明示的に宣言したカーソルの有効範囲は，その SQL 変数及びカーソルが宣言されている FOR 文の中になります。
5. FOR 文の中の SQL 手続き文に複合文を指定した場合，及び複合文の中の SQL 手続き文に FOR 文を指定した場合，FOR 文で暗黙的に宣言した SQL 変数と，複合文の中で宣言した SQL 変数が同じ名称のときは，内側のルーチン制御 SQL の中では内側のルーチン制御 SQL で宣言した SQL 変数が有効となります。内側のルーチン制御 SQL が終了したら，外側のルーチン制御 SQL で宣言した SQL 変数が有効となります。例を次に示します。



6. FOR 文の中の SQL 手続き文に FOR 文を指定した場合，外側の FOR 文で暗黙的に宣言した SQL 変数と内側の FOR 文で暗黙的に宣言した SQL 変数が同じ名称のときは，内側の FOR 文の中では内側の FOR 文で宣言した SQL 変数が有効となります。内側の FOR 文が終了したら，外側の FOR 文で宣言した SQL 変数が有効となります。
7. FOR 文の中の SQL 手続き文に複合文を指定した場合，及び複合文の中の SQL 手続き文に FOR 文を指定した場合，FOR 文で宣言したカーソルと，複合文の中で宣言したカーソルが同じ名称のときは，内側のルーチン制御 SQL の中では内側で宣言したカーソルが有効となります。内側のルーチン制御 SQL が終了したら，外側で宣言したカーソルが有効となります。
8. FOR 文の中の SQL 手続き文に FOR 文を指定した場合，外側の FOR 文で宣言したカーソルと内側の FOR 文で宣言したカーソルが同じ名称のときは，内側の FOR 文の中では内側の FOR 文で宣言したカーソルが有効となります。内側の FOR 文が終了したら，外側の FOR 文で宣言したカーソルが有効となります。
9. SQL 手続きの実行中にエラーが発生した場合に，トランザクションが無効になるのは，そのエラーが暗黙的ロールバックありのときだけです。トリガ動作の SQL 手続き文の実行中にエラーが発生した場合は，必ず暗黙的ロールバックとなります。
10. FOR 文，及び複合文のネスト数は，最大 255 です。

(5) 留意事項

1. FOR 文は SQL 手続き中，及びトリガ中に指定できます。

(6) 使用例

表 T1 のデータのうち，列 C1(INT 型)の値が 100 以下のデータは表 T2 に代入し，それ以外は表 T3 に代入する手続き(PROC1)を定義します。

```
CREATE PROCEDURE PROC1 ()
FLBL :
FOR LVN AS SELECT C1,C2,C3 FROM T1 DO
 IF C1 <= 100 THEN
  INSERT INTO T2 VALUES(LVN.C1,LVN.C2,LVN.C3) ;
 ELSE
  INSERT INTO T3 VALUES(LVN.C1,LVN.C2,LVN.C3) ;
 END IF ;
END FOR FLBL
```

上記使用例で示した FOR 文と等価な SQL は以下の SQL で実現できます。

```
CREATE PROCEDURE PROC1 ()
LVN :
BEGIN
  DECLARE C1,C2,C3 INT ;
  DECLARE FCN CURSOR FOR SELECT C1,C2,C3 FROM T1 ;
  DECLARE AT_END CHAR(1) DEFAULT 'N' ;
  OPEN FCN ;
  FLBL :
  WHILE AT_END != 'Y' DO
    FETCH FCN INTO C1,C2,C3 ;
    IF SQLCODE = 100 THEN
      SET AT_END = 'Y' ;
    ELSE
      IF C1 <= 100 THEN
        INSERT INTO T2 VALUES(LVN.C1,LVN.C2,LVN.C3) ;
      ELSE
        INSERT INTO T3 VALUES(LVN.C1,LVN.C2,LVN.C3) ;
      END IF ;
    END IF ;
  END WHILE FLBL ;
  CLOSE FCN ;
END LVN
```

7.8 WRITE LINE 文（ファイルへの文字列出力）

7.8.1 WRITE LINE 文の形式と規則

(1) 機能

指定した値式の文字列をファイルに出力します。

(2) 形式

```
WRITE LINE 値式
```

(3) オペランド

(a) 値式

ファイルに出力する値式を指定します。

値式についての規則を次に示します。

1. 値式に指定できるものを次に示します。
   - 定数
   - USER 値関数
   - SQL 変数，又は SQL パラメタ
   - 連結演算
   - スカラ関数
   - 関数呼出し
   - CASE 式
   - CAST 指定
   - SQLERRM_OF_LAST_CONDITION
2. 値式のデータ型は，文字データ型（CHAR 及び VARCHAR），各国文字データ型（NCHAR 及び NVARCHAR），又は混在文字データ型（MCHAR 及び MVARCHAR）にしてください。
3. 値式の結果のデータ型が文字データ型となる場合，ファイルに出力する文字の文字集合は値式で指定した文字集合となります。
4. 値式の出力先は，クライアント環境定義 PDWRTLNPATH で指定した出力先となります。また，値式を出力するファイルの最大サイズは，クライアント環境定義 PDWRTLNFILSZ で指定します。PDWRTLNPATH，及び PDWRTLNFILSZ については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

5. 値式の文字列の最後に，HiRDB サーバ側の環境の改行コードが付加されます。改行コードは，HiRDB サーバの OS によって異なります。また，値式の値がクライアント環境定義 PDWRTLNCOMSZ で指定したサイズを超えた場合，最後に付加する HiRDB サーバ側の改行コードは，PDWRTLNCOMSZ で指定したサイズに収まるように，値式の文字列の後方を削除して，改行コードを付加します。PDWRTLNCOMSZ については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

   付加される HiRDB サーバ側の改行コードを次の表に示します。

   表 7-4　付加される HiRDB サーバ側の改行コード

HiRDB サーバの OS	付加される改行コード
AIX	NL(X'0a')
Linux	NL(X'0a')
Windows	CR(X'0d')＋ NL(X'0a')

6. 値式の値がナル値の場合は，ファイルに出力しません。
7. 値式中には，副問合せを指定できません。

(4)　共通規則

1. WRITE LINE 文は，SQL 手続き，及びトリガに指定できます。SQL 関数中には指定できません。
2. クライアント環境定義 PDWRTLNFILSZ を指定しなかった場合，WRITE LINE 文は実行できません。

(5)　使用例

日付データ型の SQL パラメタ「fromdate」を文字列に変換して，ファイルに出力します。

```
CREATE PROCEDURE proc_1(IN fromdate DATE)
  BEGIN
    WRITE LINE 'fromdate='||char(fromdate);
  END;
```

7.9 SIGNAL 文（エラーの通知）

7.9.1 SIGNAL 文の形式と規則

(1) 機能

エラーを発生させて，通知します。このとき，この時点までに診断領域へ設定していた情報をクリアします。

(2) 形式

```
SIGNAL 信号値

信号値：：＝ {SQLSTATE値｜条件名}
SQLSTATE値：：＝SQLSTATE 〔VALUE〕 文字列定数
```

(3) オペランド

(a) 信号値：：＝ {SQLSTATE 値｜条件名}

UAP に返す値を指定します。

(b) SQLSTATE 値：：＝ SQLSTATE 〔VALUE〕 文字列定数

SQLSTATE 値として有効な値（A～Z の大文字，及び 0～9 の数字の組み合わせ）を 5 文字で指定してください。ただし，次の HiRDB での SQLSTATE の規則に従って値を指定してください。

- SQLSTATE クラス（最初の 2 文字）に'00'，'01'，'02'，及び'R2'は指定できません。これらの値はエラーを表すクラスではありません。
- 最初の 1 文字が'0'～'5'，'A'～'I'，及び'R'で始まる SQLSTATE クラスは指定できません。これらの値は HiRDB で予約されています。

SQLSTATE クラスがこれらの規則に従っていない場合，定義時にエラーとなります。SQLSTATE 値については，「SQLSTATE 変数」を参照してください。

(c) 条件名

条件宣言で宣言した条件名を指定してください。

条件名を指定した場合，SQLSTATE にはエラーを示す'R0000'（異常終了で暗黙的ロールバックなし）が設定されます。

条件名に対応した SQLCODE 値を定義している場合，エラーとなります。

(4) 共通規則

1. SIGNAL 文を実行した場合は，SQLCODE に-1400 が設定されます。
2. SIGNAL 文を実行しても，暗黙的ロールバックにはなりません。ただし，トリガの中で実行した場合は，WITHOUT ROLLBACK を指定して定義した表を除いて，暗黙的ロールバックになります。WITHOUT ROLLBACK を指定して定義した表については，SIGNAL 文をトリガの中で実行しても，行更新（追加，削除を含む）の処理完了後はロールバックしません。
3. SIGNAL 文を実行すると，SIGNAL 文を実行する前に設定されていた診断領域の条件情報項目の情報はクリアされます。

   診断領域の文情報項目の NUMBER には 1 が，MORE には'N'が設定されます。

   1 番目（条件番号：1）の条件情報項目の ERROR_SQL には，'SIGNAL'が設定されます。また，信号値に条件名を指定した場合は，CONDITION_IDENTIFIER に条件名が，SQLSTATE 値を指定した場合は CONDITION_IDENTIFIER に'SQLSTATE:xxxxx'（xxxxx は，指定した SQLSTATE 値）が設定されます。

(5) 留意事項

1. SIGNAL 文は，SQL 手続き，及びトリガ中に指定できます。
2. SIGNAL 文を実行すると，実行前に設定されていた診断領域の診断情報はクリアされます。それ以前の診断情報の履歴をクリアしない場合は，RESIGNAL 文を使用してください。

(6) 使用例

1. 在庫表（ZAIKO）に対し，指定した商品コードの数量を更新する SQL 手続き（ZAIKO_KOUSIN1）を定義します。入力パラメタに指定した数量が負の値の場合，SIGNAL 文でエラーを発生させ，「数量として無効な値です。」というメッセージを出力パラメタに設定します。

```
CREATE PROCEDURE ZAIKO_KOUSIN1(IN USCODE INT, IN USURYO INT,
                               OUT MSG NVARCHAR(50))
BEGIN
  DECLARE illegal_value CONDITION ;
  DECLARE EXIT HANDLER FOR illegal_value
    SET MSG=N'数量として無効な値です。';
  DECLARE EXIT HANDLER FOR NOT FOUND
    SET MSG=N'指定された商品コードは登録されていません。';
  IF USURYO<0 THEN
    SIGNAL illegal_value;
  ELSE
    UPDATE ZAIKO SET ZSURYO=USURYO WHERE ZSCODE=USCODE;
    SET MSG=N'更新が完了しました。';
  END IF;
END
```

2. 在庫表（ZAIKO）の行を削除する前に，SIGNAL 文でエラーを発生させ，在庫表からの行の削除を抑止するトリガ（SIGNALTRIG）を定義します。

```
CREATE TRIGGER SIGNALTRIG
  BEFORE DELETE ON ZAIKO
  SIGNAL SQLSTATE '99001'
```

7.10　RESIGNAL 文（エラーの再通知）

7.10.1　RESIGNAL 文の形式と規則

(1)　機能

エラーを発生させ，通知します。このとき，診断領域に診断情報を追加します。

(2)　形式

```
RESIGNAL 〔信号値〕

信号値：：＝ {SQLSTATE値｜条件名}
SQLSTATE値：：＝SQLSTATE 〔VALUE〕 文字列定数
```

(3)　オペランド

(a)　信号値：：＝ {SQLSTATE 値｜条件名}

UAP に返す値を指定します。

(b)　SQLSTATE 値：：＝ SQLSTATE 〔VALUE〕 文字列定数

SQLSTATE 値として有効な値（A～Z の大文字，及び 0～9 の数字の組み合わせ）を 5 文字で指定してください。ただし，次の HiRDB での SQLSTATE の規則に従って値を指定してください。

- SQLSTATE クラス（最初の 2 文字）に'00'，'01'，'02'，及び'R2'は指定できません。これらの値はエラーを表すクラスではありません。
- 最初の 1 文字が'0'～'5'，'A'～'I'，及び'R'で始まる SQLSTATE クラスは指定できません。これらの値は HiRDB で予約されています。

SQLSTATE クラスがこれらの規則に従っていない場合，定義時にエラーとなります。SQLSTATE 値については，「SQLSTATE 変数」を参照してください。

(c)　条件名

条件宣言で宣言した条件名を指定してください。

条件名を指定した場合，SQLSTATE にはエラーを示す'R0000'（異常終了で暗黙的ロールバックなし）が設定されます。

条件名に対応した SQLCODE 値を定義している場合，エラーとなります。

(4) 共通規則

1. RESIGNAL 文を実行するまでに，ハンドラを呼び出した SQL 手続き文がない場合は，実行時にエラーとなります。
2. 信号値を指定していない場合，SQLSTATE にはエラーを示す'R0000'（異常終了で暗黙的ロールバックなし）が設定されます。
3. RESIGNAL 文を実行した場合は，SQLCODE に設定される値は次のようになります。
   - 信号値を指定していない場合

     変更されません。ハンドラを呼び出した SQL 手続き文を実行したときに設定された値が保持されます。
   - 信号値を指定した場合

     -1400 が設定されます。
4. RESIGNAL 文を実行しても，暗黙的ロールバックにはなりません。ただし，トリガの中で実行した場合は，WITHOUT ROLLBACK を指定して定義した表を除いて，暗黙的ロールバックになります。WITHOUT ROLLBACK を指定して定義した表については，RESIGNAL 文をトリガの中で実行しても，行更新（追加，削除を含む）の処理完了後はロールバックしません。
5. RESIGNAL 文を実行した場合，診断領域に設定される情報は次のようになります。
   - 信号値を指定していない場合

     診断情報は更新されません。
   - 信号値を指定した場合

     診断領域の文情報項目の NUMBER には i+1（RESIGNAL 文実行前の値を i とする）が，MORE には'N'が設定されます。診断領域の条件情報項目の i 番目（条件番号 i）に設定されていた情報は，i+1 番目（条件番号 i+1）に配置されます。ここで，条件情報項目数の最大値を超えた場合は，文情報項目の MORE に'Y'が設定されます。1 番目（条件番号:1）の条件情報項目の ERROR_SQL には，'RESIGNAL'が設定されます。また，信号値に条件名を指定した場合，CONDITION_IDENTIFIER に条件名，又は SQLSTATE 値を指定したときは，CONDITION_IDENTIFIER に'SQLSTATE:xxxxx'（xxxxx は指定した SQLSTATE 値）が設定されます。

(5) 留意事項

1. RESIGNAL 文は，SQL 手続き，及びトリガ中に指定できます。

(6) 使用例

在庫表（ZAIKO）に対し，指定した商品コードの数量を更新する SQL 手続き（ZAIKO_KOUSIN2）を定義します。指定した数量が 0 未満の値の場合，一致する商品がなかった場合，又は更新に失敗した場合に，RESIGNAL 文でエラーを発生させ，それぞれの SQLSTATE 値を設定します。

```
CREATE PROCEDURE ZAIKO_KOUSIN2(IN USCODE INT,IN USURYO INT)
BEGIN
  DECLARE illegal_value CONDITION ;
  DECLARE EXIT HANDLER FOR illegal_value
    RESIGNAL SQLSTATE VALUE '66001';
  DECLARE EXIT HANDLER FOR NOT FOUND
    SIGNAL SQLSTATE VALUE '66002';
  DECLARE EXIT HANDLER FOR SQLERROR
    RESIGNAL SQLSTATE VALUE '66003';
  IF USURYO<0 THEN
    SIGNAL illegal_value;
  ELSE
    UPDATE ZAIKO SET ZSURYO=USURYO WHERE ZSCODE=USCODE;
  END IF;
END
```

付録

付録 A　予約語一覧

付録 A.1　SQL の予約語

SQL には，ISO で「ISO 9075－1992 Database Language SQL」として規格された予約語（以降 SQL92 と呼びます）と，JIS で「JIS X 3005-1990 データベース言語 SQL」として規格された予約語があります。HiRDB で使用する予約語は，JIS 規格を基本にしています。

予約語は，SQL 文で使用するキーワードとして登録されています。したがって，予約語を表や列の名称として定義できません。なお，予約語を SQL 文中に使用する必要がある場合，引用符（"）で囲んでください。予約語を引用符（"）で囲むと，一般の文字列と同じように SQL 文で使用できます。

SQL の予約語を次の表に示します。

- SQL の予約語（W）
- SQL の予約語（X）
- SQL の予約語（Y）
- SQL の予約語（Z）

表中の凡例を次に示します。

○：予約語です。

－：予約語ではありません。

●：DatabaseMetaData インタフェースの getSQLKeywords メソッドで返却されるキーワードです。

×：DatabaseMetaData インタフェースの getSQLKeywords メソッドで返却されるキーワードではありません。

SQL92：ISO SQL 1992

SQL99：ISO SQL 1999

UNIFY：UNIFY2000

XDM/RD：XDM/RD E2

HiRDB（V6）：HiRDB Version 6

HiRDB（V7）：HiRDB Version 7

HiRDB（V8）：HiRDB Version 8

HiRDB（V9）：HiRDB Version 9

HiRDB（V10）：HiRDB Version 10

JDBC：Type4 JDBC ドライバの DatabaseMetaData インタフェースの getSQLKeywords メソッド

表 A-1　SQL の予約語（A）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
ABS	－	－	－	○	○	○	○	○	○	●
ABSOLUTE	○	○	－	－	○	○	○	○	○	×
ACCESS	－	－	○	－	○	○	○	○	○	●
ACTION	○	○	－	○	○	○	○	○	○	×
ADD	○	○	○	○	○	○	○	○	○	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
ADMIN	−	○	−	−	−	−	−	−	−	×
AFTER	−	○	−	−	○	○	○	○	○	●
AGGREGATE	−	○	−	−	−	−	−	−	−	×
ALIAS	−	○	−	−	○	○	○	○	○	●
ALL	○	○	○	○	○	○	○	○	○	×
ALLOCATE	○	○	○	○	○	○	○	○	○	×
ALTER	○	○	○	○	○	○	○	○	○	×
AMOUNT	−	−	○	−	○	○	○	○	○	●
AND	○	○	○	○	○	○	○	○	○	×
ANDNOT	−	−	−	○	○	○	○	○	○	●
ANSI	−	−	○	−	○	○	○	○	○	●
ANY	○	○	○	○	○	○	○	○	○	×
ARE	○	○	−	−	○	○	○	○	○	×
ARRAY	−	○	−	○	○	○	○	○	○	●
AS	○	○	○	○	○	○	○	○	○	×
ASC	○	○	○	○	○	○	○	○	○	×
ASSERTION	○	○	−	−	○	○	○	○	○	×
ASSIGN	−	−	−	○	○	○	○	○	○	●
ASYNC	−	−	−	−	○	○	○	○	○	●
AT	○	○	○	−	○	○	○	○	○	×
AUTHORIZATION	○	○	○	○	○	○	○	○	○	×
AUTO	−	−	○	−	○	○	○	○	○	●
AVG	○	−	○	○	○	○	○	○	○	×

表 A–2　SQL の予約語（B）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
BASE	−	−	○	−	○	○	○	○	○	●
BEFORE	−	○	−	−	○	○	○	○	○	●
BEGIN	○	○	○	○	○	○	○	○	○	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
BETWEEN	○	－	○	○	○	○	○	○	○	×
BINARY	－	○	○	○	○	○	○	○	○	●
BIT	○	○	－	－	○	○	○	○	○	×
BIT_AND_TEST	－	－	－	－	○	○	○	○	○	●
BIT_LENGTH	○	－	－	－	○	○	○	○	○	×
BLOB	－	○	－	○	○	○	○	○	○	●
BOOLEAN	－	○	－	○	○	○	○	○	○	●
BOTH	○	○	－	○	○	○	○	○	○	×
BREADTH	－	○	－	－	○	○	○	○	○	●
BTREE	－	－	○	－	○	○	○	○	○	●
BUFFER	－	－	○	－	○	○	○	○	○	●
BY	○	○	○	○	○	○	○	○	○	×
BYTE	－	－	○	－	○	○	○	○	○	●

表 A-3　SQL の予約語（C）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
CALL	－	○	－	○	○	○	○	○	○	●
CASCADE	○	○	－	－	○	○	○	○	○	×
CASCADED	○	○	－	－	－	－	－	－	－	×
CASE	○	○	－	○	○	○	○	○	○	×
CAST	○	○	－	○	○	○	○	○	○	×
CATALOG	○	○	－	－	○	○	○	○	○	×
CHANGE	－	－	－	○	○	○	○	○	○	●
CHAR	○	○	○	○	○	○	○	○	○	×
CHARACTER	○	○	○	○	○	○	○	○	○	×
CHAR_LENGTH	○	－	－	－	○	○	○	○	○	×
CHARACTER_LENGTH	○	－	－	－	○	○	○	○	○	×
CHECK	○	○	○	○	○	○	○	○	○	×
CLASS	－	○	－	－	－	－	－	－	－	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
CLOB	−	○	−	○	−	−	−	−	−	×
CLOSE	○	○	○	○	○	○	○	○	○	×
CLUSTER	−	−	−	○	○	○	○	○	○	●
COALESCE	○	−	−	○	○	○	○	○	○	×
COLLATE	○	○	−	−	○	○	○	○	○	×
COLLATION	○	○	−	−	○	○	○	○	○	×
COLUMN	○	○	○	○	○	○	○	○	○	×
COLUMNS	−	−	○	−	○	○	○	○	○	●
COMMENT	−	−	−	○	○	○	○	○	○	●
COMMIT	○	○	○	○	○	○	○	○	○	×
COMPLETION	−	○	−	−	○	○	○	○	○	●
COMPRESSED	−	−	−	○	−	−	−	○	○	×
CONDITION	−	○	−	−	−	○	○	○	○	●
CONFIGURATION	−	−	○	−	○	○	○	○	○	●
CONNECT	○	○	○	○	○	○	○	○	○	×
CONNECTION	○	○	−	−	○	○	○	○	○	×
CONST	−	−	○	−	○	○	○	○	○	●
CONSTRAINT	○	○	−	○	○	○	○	○	○	×
CONSTRAINTS	○	○	−	−	○	○	○	○	○	×
CONSTRUCTOR	−	○	−	−	○	○	○	○	○	●
CONTIGUOUS	−	−	○	−	○	○	○	○	○	●
CONTINUE	○	○	○	−	○	○	○	○	○	×
CONVERT	○	−	−	−	○	○	○	○	○	×
CORR	−	−	−	○	−	−	−	−	−	×
CORRESPONDING	○	○	−	−	○	○	○	○	○	×
COUNT	○	−	○	○	○	○	○	○	○	×
COUNT_FLOAT	−	−	−	−	−	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
COVAR_POP	−	−	−	○	−	−	−	−	−	×
COVAR_SAMP	−	−	−	○	−	−	−	−	−	×
CREATE	○	○	○	○	○	○	○	○	○	×
CROSS	○	○	−	○	○	○	○	○	○	×
CUBE	−	○	−	○	−	−	−	−	−	×
CUME_DIST	−	−	−	○	−	−	−	−	−	×
CURAID	−	−	○	−	○	○	○	○	○	●
CURRENT	○	○	○	○	○	○	○	○	○	×
CURRENT_DATE	○	○	−	○	○	○	○	○	○	×
CURRENT_DEFAULT_TRANSFORM_GROUP	−	○	−	−	−	−	−	−	−	×
CURRENT_PATH	−	○	−	−	−	−	−	−	−	×
CURRENT_ROLE	−	○	−	−	−	−	−	−	−	×
CURRENT_ROLL	−	○	−	−	−	−	−	−	−	×
CURRENT_TIME	○	○	−	○	○	○	○	○	○	×
CURRENT_TIMESTAMP	○	○	−	○	○	○	○	○	○	×
CURRENT_TRANSFORM_GROUP_FOR_TYPE	−	○	−	−	−	−	−	−	−	×
CURRENT_USER	○	○	−	○	○	○	○	○	○	×
CURSOR	○	○	○	○	○	○	○	○	○	×
CYCLE	−	○	−	−	○	○	○	○	○	●

表 A-4　SQL の予約語（D）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
DATA	−	○	○	○	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
DATABASE	−	−	○	−	○	○	○	○	○	●
DATE	○	○	○	○	○	○	○	○	○	×
DAY	○	○	−	○	○	○	○	○	○	×
DAYS	−	−	−	○	○	○	○	○	○	●
DBA	−	−	○	○	○	○	○	○	○	●
DEALLOCATE	○	○	○	−	○	○	○	○	○	×
DEC	○	○	○	○	○	○	○	○	○	×
DECIMAL	○	○	○	○	○	○	○	○	○	×
DECLARE	○	○	○	○	○	○	○	○	○	×
DEFAULT	○	○	○	○	○	○	○	○	○	×
DEFER	−	−	○	−	○	○	○	○	○	●
DEFERRABLE	○	○	−	−	○	○	○	○	○	×
DEFERRED	○	○	○	−	○	○	○	○	○	×
DELETE	○	○	○	○	○	○	○	○	○	×
DEMOTING	−	−	○	−	○	○	○	○	○	●
DENSE_RANK	−	−	−	○	−	−	−	−	−	×
DEPTH	−	○	−	−	○	○	○	○	○	●
DEREF	−	○	−	−	−	−	−	−	−	×
DESC	○	○	○	○	○	○	○	○	○	×
DESCRIBE	○	○	○	○	○	○	○	○	○	×
DESCRIPTION	−	−	○	−	−	−	−	−	−	×
DESCRIPTOR	○	○	○	○	○	○	○	○	○	×
DESTROY	−	○	−	−	−	−	−	−	−	×
DESTRUCTOR	−	○	−	−	−	−	−	−	−	×
DETERMINISTIC	−	○	−	−	−	−	−	−	−	×
DEVICE	−	−	○	−	○	○	○	○	○	●
DIAGNOSTICS	○	○	−	○	○	○	○	○	○	×
DICTIONARY	−	○	−	−	○	○	○	○	○	●
DIGITS	−	−	−	○	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
DIRECT	−	−	○	−	○	○	○	○	○	●
DISCONNECT	○	○	○	○	○	○	○	○	○	×
DISPLAY	−	−	○	−	−	−	−	−	−	×
DISTINCT	○	○	○	○	○	○	○	○	○	×
DO	−	○	−	○	○	○	○	○	○	●
DOMAIN	○	○	−	−	−	−	−	−	−	×
DOUBLE	○	○	○	○	○	○	○	○	○	×
DOUBLE_PRECISION	−	−	○	−	○	○	○	○	○	●
DROP	○	○	○	○	○	○	○	○	○	×
DYNAMIC	−	○	−	−	−	−	−	−	−	×

表 A-5　SQL の予約語（E）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
EACH	−	○	−	○	○	○	○	○	○	●
EDIT	−	−	○	−	○	○	○	○	○	●
ELSE	○	○	−	○	○	○	○	○	○	×
ELSEIF	−	○	−	○	○	○	○	○	○	●
ENCRYPT	−	−	−	−	−	○	○	○	○	●
END	○	○	○	○	○	○	○	○	○	×
END-EXEC	○	○	−	−	−	−	−	−	−	×
EQUALS	−	○	−	−	○	○	○	○	○	●
ESCAPE	○	○	○	○	○	○	○	○	○	×
ESTIMATED	−	−	○	−	○	○	○	○	○	●
EVERY	−	○	−	○	−	−	−	−	−	×
EXCEPT	○	○	−	○	○	○	○	○	○	×
EXCEPTION	○	○	−	○	○	○	○	○	○	×
EXCLUSIVE	−	−	−	○	○	○	○	○	○	●
EXEC	○	○	○	−	○	○	○	○	○	×
EXECUTE	○	○	○	○	○	○	○	○	○	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
EXISTS	○	−	○	○	○	○	○	○	○	×
EXIT	−	○	−	−	−	○	○	○	○	●
EXTERN	−	−	○	−	○	○	○	○	○	●
EXTERNAL	○	○	−	−	○	○	○	○	○	×
EXTRACT	○	−	−	−	○	○	○	○	○	×

表 A-6　SQL の予約語（F）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
FALSE	○	○	−	○	○	○	○	○	○	×
FETCH	○	○	○	○	○	○	○	○	○	×
FILE	−	−	○	−	○	○	○	○	○	●
FILTER	−	−	−	○	−	−	−	−	−	×
FIRST	○	○	−	−	○	○	○	○	○	×
FIX	−	−	−	○	○	○	○	○	○	●
FIXED	−	−	○	−	○	○	○	○	○	●
FLAT	−	−	−	○	○	○	○	○	○	●
FLOAT	○	○	○	○	○	○	○	○	○	×
FOR	○	○	○	○	○	○	○	○	○	×
FORCE	−	−	○	○	○	○	○	○	○	●
FOREIGN	○	○	−	○	○	○	○	○	○	×
FOUND	○	○	○	−	○	○	○	○	○	×
FREE	−	○	−	−	−	○	○	○	○	●
FROM	○	○	○	○	○	○	○	○	○	×
FULL	○	○	−	○	○	○	○	○	○	×
FUNCTION	−	○	−	○	○	○	○	○	○	●

表 A-7　SQL の予約語（G）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
GENERAL	−	○	−	−	○	○	○	○	○	●
GET	○	○	○	○	○	○	○	○	○	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
GET_JAVA_STORED_ROUTINE_SOURCE	−	−	−	−	○	○	○	○	○	●
GLOBAL	○	○	−	−	○	○	○	○	○	×
GO	○	○	○	−	○	○	○	○	○	×
GOTO	○	○	○	−	○	○	○	○	○	×
GRANT	○	○	○	○	○	○	○	○	○	×
GROUP	○	○	○	○	○	○	○	○	○	×
GROUPING	−	○	−	○	−	−	−	−	−	×

表 A-8　SQL の予約語（H）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
HANDLER	−	○	−	−	−	○	○	○	○	●
HASH	−	−	○	−	○	○	○	○	○	●
HAVING	○	○	○	○	○	○	○	○	○	×
HELP	−	−	○	−	○	○	○	○	○	●
HEX	−	−	−	○	○	○	○	○	○	●
HOST	−	○	−	−	−	−	−	−	−	×
HOUR	○	○	−	○	○	○	○	○	○	×
HOURS	−	−	−	○	○	○	○	○	○	●
HUGE	−	−	○	−	○	○	○	○	○	●

表 A-9　SQL の予約語（I）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
IDENTIFIED	−	−	−	○	○	○	○	○	○	●
IDENTITY	○	○	−	−	○	○	○	○	○	×
IF	−	○	−	○	○	○	○	○	○	●
IGNORE	−	○	−	−	○	○	○	○	○	●
IMMEDIATE	○	○	○	○	○	○	○	○	○	×
IN	○	○	○	○	○	○	○	○	○	×
INDEX	−	−	○	○	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
INDICATOR	○	○	○	○	○	○	○	○	○	×
INITIALIZE	−	○	−	−	−	−	−	−	−	×
INITIALLY	○	○	−	−	○	○	○	○	○	×
INNER	○	○	−	○	○	○	○	○	○	×
INOUT	−	○	−	○	○	○	○	○	○	●
INPUT	○	○	○	−	○	○	○	○	○	×
INSENSITIVE	○	−	−	−	○	○	○	○	○	×
INSERT	○	○	○	○	○	○	○	○	○	×
INT	○	○	○	○	○	○	○	○	○	×
INTEGER	○	○	○	○	○	○	○	○	○	×
INTERSECT	○	○	−	○	○	○	○	○	○	×
INTERVAL	○	○	−	○	○	○	○	○	○	×
INTO	○	○	○	○	○	○	○	○	○	×
IS	○	○	○	○	○	○	○	○	○	×
ISOLATION	○	○	−	○	○	○	○	○	○	×
IS_USER_CONTAINED_IN_HDS_GROUP	−	−	−	−	○	○	○	○	○	●
ITERATE	−	○	−	○	−	−	−	−	−	×

表 A-10　SQL の予約語（J）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
JOIN	○	○	−	○	○	○	○	○	○	×

表 A-11　SQL の予約語（K）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
KEY	○	○	○	○	○	○	○	○	○	×

表 A-12　SQL の予約語（L）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
LABEL	−	−	−	○	−	−	−	−	−	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
LANGUAGE	○	○	○	○	○	○	○	○	○	×
LARGE	−	○	−	○	○	○	○	○	○	●
LAST	○	○	−	−	○	○	○	○	○	×
LATERAL	−	○	−	−	−	−	−	−	−	×
LEADING	○	○	○	○	○	○	○	○	○	×
LEAVE	−	○	−	○	○	○	○	○	○	●
LEFT	○	○	−	−	○	○	○	○	○	×
LENGTH	−	−	○	○	○	○	○	○	○	●
LESS	−	○	−	−	○	○	○	○	○	●
LEVEL	○	○	○	○	○	○	○	○	○	×
LIKE	○	○	○	○	○	○	○	○	○	×
LIMIT	−	○	−	−	○	○	○	○	○	●
LINES	−	−	○	−	○	○	○	○	○	●
LINK	−	−	○	−	○	○	○	○	○	●
LIST	−	−	−	○	○	○	○	○	○	●
LOCAL	○	○	−	−	○	○	○	○	○	×
LOCALTIME	−	○	−	−	−	−	−	−	−	×
LOCALTIMESTAMP	−	○	−	−	−	−	−	−	−	×
LOCATOR	−	○	−	−	−	○	○	○	○	●
LOCK	−	−	−	○	○	○	○	○	○	●
LOCKS	−	−	○	−	○	○	○	○	○	●
LOGID	−	−	○	−	○	○	○	○	○	●
LOGNAME	−	−	○	−	○	○	○	○	○	●
LONG	−	−	○	○	○	○	○	○	○	●
LOOP	−	○	−	○	○	○	○	○	○	●
LOWER	○	−	−	○	○	○	○	○	○	×

表 A–13　SQL の予約語（M）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
MAP	－	○	－	－	－	－	－	－	－	×
MATCH	○	○	－	－	○	○	○	○	○	×
MAX	○	－	○	○	○	○	○	○	○	×
MAXUSAGES	－	－	－	－	○	○	○	○	○	●
MCHAR	－	－	－	○	○	○	○	○	○	●
MICROSECOND	－	－	－	○	－	－	－	－	－	×
MICROSECONDS	－	－	－	○	－	－	－	－	－	×
MIN	○	－	○	○	○	○	○	○	○	×
MINUTE	○	○	－	○	○	○	○	○	○	×
MINUTES	－	－	－	○	○	○	○	○	○	●
MOD	－	－	－	○	○	○	○	○	○	●
MODE	－	－	○	○	○	○	○	○	○	●
MODIFIES	－	○	－	－	－	－	－	－	－	×
MODIFY	－	○	－	－	○	○	○	○	○	●
MODULE	○	○	○	○	○	○	○	○	○	×
MONTH	○	○	－	○	○	○	○	○	○	×
MONTHS	－	－	－	○	○	○	○	○	○	●
MOVE	－	－	○	－	○	○	○	○	○	●
MVARCHAR	－	－	－	○	○	○	○	○	○	●

表 A–14　SQL の予約語（N）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
NAMES	○	○	－	－	○	○	○	○	○	×
NATIONAL	○	○	－	○	○	○	○	○	○	×
NATURAL	○	○	－	－	○	○	○	○	○	×
NCHAR	○	○	－	○	○	○	○	○	○	×
NCLOB	－	○	－	－	－	－	－	－	－	×
NESTING	－	○	－	－	－	－	－	－	－	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
NEW	−	○	−	○	○	○	○	○	○	●
NEXT	○	○	−	−	○	○	○	○	○	×
NO	○	○	−	○	○	○	○	○	○	×
NONE	−	○	−	−	○	○	○	○	○	●
NONLOCAL	−	−	−	○	−	−	−	−	−	×
NOT	○	○	○	○	○	○	○	○	○	×
NOWAIT	−	−	−	○	○	○	○	○	○	●
NULL	○	○	○	○	○	○	○	○	○	×
NULLABLE	−	−	○	−	○	○	○	○	○	●
NULLIF	○	−	−	○	○	○	○	○	○	×
NUMERIC	○	○	○	○	○	○	○	○	○	×
NVARCHAR	−	−	−	○	○	○	○	○	○	●

表 A–15　SQL の予約語（O）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
OBJECT	−	○	−	−	○	○	○	○	○	●
OCTET_LENGTH	○	−	−	−	○	○	○	○	○	×
OF	○	○	○	○	○	○	○	○	○	×
OFF	−	○	−	−	○	○	○	○	○	●
OFFSET	−	−	○	−	○	○	○	○	○	●
OID	−	−	−	−	○	○	○	○	○	●
OLD	−	○	−	○	○	○	○	○	○	●
ON	○	○	○	○	○	○	○	○	○	×
ONLY	○	○	○	○	○	○	○	○	○	×
OPEN	○	○	○	○	○	○	○	○	○	×
OPERATION	−	○	−	−	○	○	○	○	○	●
OPERATORS	−	−	−	−	○	○	○	○	○	●
OPTION	○	○	○	○	○	○	○	○	○	×
OPTIMIZE	−	−	−	○	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
OR	○	○	○	○	○	○	○	○	○	×
ORDER	○	○	○	○	○	○	○	○	○	×
ORDINALITY	−	○	−	−	−	−	−	−	−	×
OTHERS	−	−	−	−	○	○	○	○	○	●
OUT	−	○	−	○	○	○	○	○	○	●
OUTER	○	○	−	○	○	○	○	○	○	×
OUTPUT	○	○	○	−	○	○	○	○	○	×
OVER	−	−（2001 年以降○に変更）	−	○	−	○	○	○	○	●
OVERFLOW	−	−	○	−	○	○	○	○	○	●
OVERLAPS	○	−	−	−	−	−	−	−	−	×
OVERWRITE	−	−	○	−	−	−	−	−	−	×
OWN	−	−	−	○	○	○	○	○	○	●
OWNER	−	−	○	−	−	−	−	−	−	×

表 A–16　SQL の予約語（P）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
PAD	○	○	−	−	○	○	○	○	○	×
PAGE	−	−	−	−	○	○	○	○	○	●
PARAMETER	−	○	−	−	−	−	−	−	−	×
PARAMETERS	−	○	−	−	○	○	○	○	○	●
PARTIAL	○	○	−	−	○	○	○	○	○	×
PARTITION	−	−	−	○	−	−	−	−	−	×
PARTITIONED	−	−	−	−	○	○	○	○	○	●
PATH	−	○	○	−	−	−	−	−	−	×
PCTFREE	−	−	−	○	○	○	○	○	○	●
PENDANT	−	−	−	−	○	○	○	○	○	●
PERCENT_RANK	−	−	−	○	−	−	−	−	−	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
PERCENTILE_CONT	−	−	−	○	−	−	−	−	−	×
PERCENTILE_DISC	−	−	−	○	−	−	−	−	−	×
PIC	−	−	○	−	○	○	○	○	○	●
PICTURE	−	−	○	○	○	○	○	○	○	●
POSITION	○	−	−	−	○	○	○	○	○	×
POSTFIX	−	○	−	−	−	−	−	−	−	×
PREALLOCATED	−	−	○	−	○	○	○	○	○	●
PRECISION	○	○	○	○	○	○	○	○	○	×
PREFERRED	−	−	○	−	○	○	○	○	○	●
PREFIX	−	○	−	−	−	−	−	−	−	×
PREORDER	−	○	−	−	○	○	○	○	○	●
PREPARE	○	○	○	○	○	○	○	○	○	×
PRESERVE	○	○	−	−	○	○	○	○	○	×
PRIMARY	○	○	○	○	○	○	○	○	○	×
PRIMLEGES	−	−	○	−	−	−	−	−	−	×
PRIOR	○	○	−	−	○	○	○	○	○	×
PRIVATE	−	−	○	○	○	○	○	○	○	●
PRIVILEGES	○	○	−	○	○	○	○	○	○	×
PROCEDURE	○	○	○	○	○	○	○	○	○	×
PROGRAM	−	−	−	○	○	○	○	○	○	●
PROTECTED	−	−	−	○	○	○	○	○	○	●
PUBLIC	○	○	○	○	○	○	○	○	○	×
PURGE	−	−	−	○	○	○	○	○	○	●

表 A-17　SQL の予約語（R）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
RANDOM	−	−	−	−	○	○	○	○	○	●
RANGE	−	−	−	○	−	−	−	−	−	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
RANK	−	−	−	○	−	−	−	−	−	×
RD	−	−	−	−	○	○	○	○	○	●
RDAREA	−	−	−	○	○	○	○	○	○	●
RDNODE	−	−	−	○	−	−	−	−	−	×
READ	○	○	○	○	○	○	○	○	○	×
READS	−	○	−	−	−	−	−	−	−	×
REAL	○	○	○	○	○	○	○	○	○	×
RECOMPILE	−	−	−	−	○	○	○	○	○	●
RECOVERABLE	−	−	○	−	○	○	○	○	○	●
RECOVERY	−	−	−	−	○	○	○	○	○	●
RECURSIVE	−	○	−	○	○	○	○	○	○	●
REDO	−	○	−	−	−	−	−	−	−	×
REF	−	○	−	−	○	○	○	○	○	●
REFERENCES	○	○	○	○	○	○	○	○	○	×
REFERENCING	−	○	−	○	○	○	○	○	○	●
REGLIKE	−	−	○	−	○	○	○	○	○	●
REGR_AVGX	−	−	−	○	−	−	−	−	−	×
REGR_AVGY	−	−	−	○	−	−	−	−	−	×
REGR_COUNT	−	−	−	○	−	−	−	−	−	×
REGR_INTERCEPT	−	−	−	○	−	−	−	−	−	×
REGR_R2	−	−	−	○	−	−	−	−	−	×
REGR_SLOPE	−	−	−	○	−	−	−	−	−	×
REGR_SXX	−	−	−	○	−	−	−	−	−	×
REGR_SXY	−	−	−	○	−	−	−	−	−	×
REGR_SYY	−	−	−	○	−	−	−	−	−	×
RELATIVE	○	○	−	−	○	○	○	○	○	×
RELEASE	−	−	−	○	○	○	○	○	○	●
RELEASING	−	−	○	−	○	○	○	○	○	●
RENAME	−	−	○	−	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
REPEAT	−	○	−	○	−	−	−	−	−	×
RESIGNAL	−	○	−	−	○	○	○	○	○	●
RESTART	−	−	○	−	○	○	○	○	○	●
RESTRICT	○	○	−	−	○	○	○	○	○	×
RESULT	−	○	−	−	−	−	−	−	−	×
RETURN	−	○	−	○	○	○	○	○	○	●
RETURNS	−	○	−	○	○	○	○	○	○	●
REVOKE	○	○	○	○	○	○	○	○	○	×
RIGHT	○	○	−	−	○	○	○	○	○	×
ROLE	−	○	−	−	○	○	○	○	○	●
ROLLBACK	○	○	○	○	○	○	○	○	○	×
ROLLUP	−	○	−	○	−	−	−	−	−	×
ROOT	−	−	○	−	○	○	○	○	○	●
ROUTINE	−	○	−	○	○	○	○	○	○	●
ROW	−	○	−	○	○	○	○	○	○	●
ROW_NUMBER	−	−	−	○	−	−	−	−	−	×
ROWID	−	−	○	○	○	○	○	○	○	●
ROWS	○	○	−	○	○	○	○	○	○	×

表 A-18　SQL の予約語（S）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
SAVEPOINT	−	○	−	−	○	○	○	○	○	●
SCALE	−	−	○	−	○	○	○	○	○	●
SCAN	−	−	○	−	○	○	○	○	○	●
SCATTERED	−	−	○	−	−	−	−	−	−	×
SCHEMA	○	○	○	○	○	○	○	○	○	×
SCHEMAS	−	−	○	−	○	○	○	○	○	●
SCOPE	−	○	−	−	○	○	○	○	○	●
SCROLL	○	○	−	−	○	○	○	○	○	×
SD	−	−	−	−	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
SEARCH	−	○	−	−	○	○	○	○	○	●
SECOND	○	○	−	○	○	○	○	○	○	×
SECONDS	−	−	−	○	○	○	○	○	○	●
SECTION	○	○	○	−	○	○	○	○	○	×
SEGMENT	−	−	○	−	○	○	○	○	○	●
SELECT	○	○	○	○	○	○	○	○	○	×
SENSITIVE	−	−	−	−	○	○	○	○	○	●
SEPARATE	−	−	○	−	○	○	○	○	○	●
SEPARATOR	−	−	○	−	○	○	○	○	○	●
SEQUENCE	−	○	−	−	○	○	○	○	○	●
SESSION	○	○	−	−	○	○	○	○	○	×
SESSION_USER	○	○	−	−	○	○	○	○	○	×
SET	○	○	○	○	○	○	○	○	○	×
SETS	−	○	−	−	−	−	−	−	−	×
SFLIKE	−	−	−	−	○	○	○	○	○	●
SHARE	−	−	−	○	○	○	○	○	○	●
SHLIKE	−	−	○	−	−	−	−	−	−	×
SHORT	−	−	○	−	○	○	○	○	○	●
SIGN	−	−	○	−	−	−	−	−	−	×
SIGNAL	−	○	−	○	○	○	○	○	○	●
SIMILAR	−	−	−	−	○	○	○	○	○	●
SIZE	○	○	○	−	○	○	○	○	○	×
SLOCK	−	−	○	−	○	○	○	○	○	●
SMALLFLT	−	−	−	○	○	○	○	○	○	●
SMALLINT	○	○	○	○	○	○	○	○	○	×
SOME	○	○	○	○	○	○	○	○	○	×
SPACE	○	○	−	−	○	○	○	○	○	×
SPECIFIC	−	○	−	○	−	−	−	−	−	×
SPECIFICTYPE	−	○	−	−	−	−	−	−	−	×
SPLIT	−	−	○	−	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
SQL	○	○	○	−	○	○	○	○	○	×
SQL_STANDARD	−	−	○	−	○	○	○	○	○	●
SQLCODE	○	−	−	○	○	○	○	○	○	×
SQLCODE_OF_LAST_CONDITION	−	−	−	−	−	−	○	○	○	●
SQLCODE_TYPE	−	−	○	−	○	○	○	○	○	●
SQLCOUNT	−	−	−	○	○	○	○	○	○	●
SQLDA	−	−	−	○	○	○	○	○	○	●
SQLERRM	−	−	−	○	○	○	○	○	○	●
SQLERRM_OF_LAST_CONDITION	−	−	−	−	−	−	○	○	○	●
SQLERRMC	−	−	−	○	○	○	○	○	○	●
SQLERRML	−	−	−	○	○	○	○	○	○	●
SQLERROR	○	−	○	−	○	○	○	○	○	×
SQLEXCEPTION	−	○	−	−	○	○	○	○	○	●
SQLNAME	−	−	−	○	○	○	○	○	○	●
SQLSTATE	○	○	−	○	○	○	○	○	○	×
SQLWARN	−	−	−	○	○	○	○	○	○	●
SQLWARNING	−	○	○	−	○	○	○	○	○	●
START	−	○	○	−	○	○	○	○	○	●
STATE	−	○	−	−	−	−	−	−	−	×
STATEMENT	−	○	−	−	−	−	−	−	−	×
STATIC	−	○	○	−	○	○	○	○	○	●
STDDEV_POP	−	−	−	○	−	−	−	−	−	×
STOP	−	−	○	−	○	○	○	○	○	●
STOPPING	−	−	−	○	○	○	○	○	○	●
STRUCTURE	−	○	−	−	○	○	○	○	○	●
SUBSTR	−	−	−	○	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
SUBSTRING	○	−	−	−	○	○	○	○	○	×
SUM	○	−	○	○	○	○	○	○	○	×
SUPPRESS	−	−	−	−	○	○	○	○	○	●
SYNONYM	−	−	○	−	○	○	○	○	○	●
SYSTEM_USER	○	○	−	−	○	○	○	○	○	×

表 A-19　SQL の予約語（T）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
TABLE	○	○	○	○	○	○	○	○	○	×
TABLES	−	−	○	−	−	−	−	−	−	×
TEMPORARY	○	○	−	−	○	○	○	○	○	×
TERMINATE	−	○	−	−	−	−	−	−	−	×
TEST	−	−	−	−	○	○	○	○	○	●
TEXT	−	−	○	−	○	○	○	○	○	●
THAN	−	○	−	−	−	−	−	−	−	×
THEN	○	○	−	○	○	○	○	○	○	×
THERE	−	−	−	−	○	○	○	○	○	●
TIME	○	○	○	○	○	○	○	○	○	×
TIMESTAMP	○	○	−	○	○	○	○	○	○	×
TIMESTAMP_FORMAT	−	−	−	−	−	○	○	○	○	●
TIMEZONE_HOUR	○	○	−	−	○	○	○	○	○	×
TIMEZONE_MINUTE	○	○	−	−	○	○	○	○	○	×
TO	○	○	○	○	○	○	○	○	○	×
TRAILING	○	○	−	○	○	○	○	○	○	×
TRANSACTION	○	○	○	−	○	○	○	○	○	×
TRANSLATE	○	−	−	−	○	○	○	○	○	×
TRANSLATION	○	○	−	−	○	○	○	○	○	×
TREAT	−	○	−	−	○	○	○	○	○	●

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
TRIGGER	−	○	−	○	○	○	○	○	○	●
TRIM	○	−	−	○	○	○	○	○	○	×
TRUE	○	○	−	○	○	○	○	○	○	×
TRUNCATE※	−	−	−	−	−	−	−	○	○	×
TYPE	−	○	○	○	○	○	○	○	○	●

注※

ISO SQL 2008 での予約語です。

表 A-20　SQL の予約語（U）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
UAMT	−	−	○	−	○	○	○	○	○	●
UBINBUF	−	−	○	−	○	○	○	○	○	●
UCHAR	−	−	○	−	○	○	○	○	○	●
UDATE	−	−	○	−	○	○	○	○	○	●
UHAMT	−	−	○	−	−	−	−	−	−	×
UHANT	−	−	−	−	○	○	○	○	○	●
UHDATE	−	−	○	−	○	○	○	○	○	●
UNBOUNDED	−	−	−	○	−	−	−	−	−	×
UNDER	−	○	−	○	○	○	○	○	○	●
UNDO	−	○	−	−	−	−	−	−	−	×
UNIFY_2000	−	−	○	−	○	○	○	○	○	●
UNION	○	○	○	○	○	○	○	○	○	×
UNIONALL	−	−	−	−	○	○	○	○	○	●
UNIQUE	○	○	○	○	○	○	○	○	○	×
UNKNOWN	○	○	−	○	○	○	○	○	○	×
UNLIMITED	−	−	○	−	○	○	○	○	○	●
UNLOCK	−	−	○	−	○	○	○	○	○	●
UNTIL	−	○	−	○	○	○	○	○	○	●
UNNEST	−	○	−	−	−	−	−	−	−	×
UPDATE	○	○	○	○	○	○	○	○	○	×

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
UPPER	○	−	−	○	○	○	○	○	○	×
USAGE	○	○	○	○	○	○	○	○	○	×
USE	−	−	○	−	○	○	○	○	○	●
USER	○	○	○	○	○	○	○	○	○	×
USER_GROUP	−	−	−	○	−	−	−	−	−	×
USER_LEVEL	−	−	−	○	−	−	−	−	−	×
USING	○	○	○	○	○	○	○	○	○	×
UTIME	−	−	○	−	○	○	○	○	○	●
UTXTBUF	−	−	○	−	○	○	○	○	○	●

表 A–21　SQL の予約語（V）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
VALUE	○	○	○	○	○	○	○	○	○	×
VALUES	○	○	○	○	○	○	○	○	○	×
VAR_POP	−	−	−	○	−	−	−	−	−	×
VAR_SAMP	−	−	−	○	−	−	−	−	−	×
VARCHAR	○	○	−	○	○	○	○	○	○	×
VARCHAR_FORMAT	−	−	−	−	−	○	○	○	○	●
VARIABLE	−	○	−	−	○	○	○	○	○	●
VARYING	○	○	−	○	○	○	○	○	○	×
VIEW	○	○	○	○	○	○	○	○	○	×
VIRTUAL	−	−	−	−	○	○	○	○	○	●
VISIBLE	−	−	−	−	○	○	○	○	○	●
VOLATILE	−	−	○	−	○	○	○	○	○	●
VOLUME	−	−	○	−	○	○	○	○	○	●
VOLUMES	−	−	○	−	○	○	○	○	○	●

表 A-22　SQL の予約語（W）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
WAIT	－	－	－	○	○	○	○	○	○	●
WHEN	○	○	－	○	○	○	○	○	○	×
WHENEVER	○	○	○	－	○	○	○	○	○	×
WHERE	○	○	○	○	○	○	○	○	○	×
WHILE	－	○	－	○	○	○	○	○	○	●
WINDOW	－	－	－	○	－	－	－	－	－	×
WITH	○	○	○	○	○	○	○	○	○	×
WITHIN	－	－	－	○	－	－	－	－	－	×
WITHOUT	－	○	－	○	○	○	○	○	○	●
WORK	○	○	○	○	○	○	○	○	○	×
WRITE	○	○	○	－	○	○	○	○	○	×

表 A-23　SQL の予約語（X）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
XLIKE	－	－	－	－	○	○	○	○	○	●
XLOCK	－	－	○	－	○	○	○	○	○	●
XML	－	－	－	－	－	－	○	○	○	●
XMLAGG	－	－	－	－	－	－	○	○	○	●
XMLEXISTS	－	－	－	－	－	－	○	○	○	●
XMLPARSE	－	－	－	－	－	－	○	○	○	●
XMLQUERY	－	－	－	－	－	－	○	○	○	●
XMLSERIALIZE	－	－	－	－	－	－	○	○	○	●

表 A-24　SQL の予約語（Y）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
YEAR	○	○	－	○	○	○	○	○	○	×
YEARS	－	－	－	○	○	○	○	○	○	●

表 A-25　SQL の予約語（Z）

予約語	SQL92	SQL99	UNIFY	XDM/RD	HiRDB (V6)	HiRDB (V7)	HiRDB (V8)	HiRDB (V9)	HiRDB (V10)	JDBC
ZONE	－	○	－	－	－	－	－	－	－	×

付録 A.2　HiRDB の予約語

HiRDB が使用する予約語は，認可識別子や表識別子などの名前として定義できません。

HiRDB の予約語を次の表に示します。

表 A-26　HiRDB の予約語

予約語	定義できない名称
ALL	認可識別子，表識別子，RD エリア名
HiRDB※1	認可識別子
MASTER※1※2	認可識別子
PUBLIC	認可識別子
SQL 及び sql で始まる名称	埋込み変数，標準変数名，ホスト識別子

注※1

Type4 JDBC DatabaseMetaData インタフェースの getSQLKeywords メソッドが返却するキーワードの対象です。

注※2

データディクショナリ表，及びプラグインが提供する次のリソースの認可識別子で使用します。

- ユーザ定義型
- インデクス型
- 関数

付録 A.3　SQL 予約語削除機能で削除できる予約語

SQL 予約語削除機能で削除できる予約語，及び削除した場合に使用できなくなる機能を次の表に示します。

- 削除できる予約語（A）
- 削除できる予約語（B）
- 削除できる予約語（C）

- 削除できる予約語（D）
- 削除できる予約語（E）
- 削除できる予約語（F）
- 削除できる予約語（G）
- 削除できる予約語（H）
- 削除できる予約語（I）
- 削除できる予約語（L）
- 削除できる予約語（M）
- 削除できる予約語（N）
- 削除できる予約語（O）
- 削除できる予約語（P）
- 削除できる予約語（R）
- 削除できる予約語（S）
- 削除できる予約語（T）
- 削除できる予約語（U）
- 削除できる予約語（V）
- 削除できる予約語（W）
- 削除できる予約語（X）
- 削除できる予約語（Y）

表中の凡例を次に示します。

－：予約語を削除しても使用できなくなる機能はありません。

表 A-27　削除できる予約語（A）

予約語	使用できなくなる機能
ABS	・スカラ関数 ABS
ALLOCATE	・抽象データ型の LOB 属性 ・プラグイン
AMOUNT	－
ANDNOT	・リスト間の差集合
ANSI	－
ARRAY	・繰返し列
ASSERTION	－

予約語	使用できなくなる機能
ASYNC	－
AUTO	－

表 A-28　削除できる予約語（B）

予約語	使用できなくなる機能
BASE	－
BEGIN	・ルーチン制御 SQL の複合文 ・ユーザ定義関数
BINARY	・データ型 BINARY，BINARY LARGE OBJECT
BIT_AND_TEST	・スカラ関数 BIT_ADN_TEST
BLOB	・データ型 BLOB
BOOLEAN	・データ型 BOOLEAN
BOTH	・スカラ関数 TRIM の TRIM 指定 BOTH
BREADTH	－
BTREE	－
BUFFER	－
BYTE	－

表 A-29　削除できる予約語（C）

予約語	使用できなくなる機能
CALL	・ストアドプロシジャ ・コマンド又はユティリティの実行
CASE	・CASE 式
CAST	・CAST 指定
COALESCE	・CASE 略式
COLUMNS	－
COMPLETION	－
CONDITION	・ルーチン制御 SQL 複合文の条件宣言
CONFIGURATION	－
CONST	－
CONSTRAINT	・参照制約 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY，DROP CONNECTION SECURITY）の IP アドレスによる接続制限

予約語	使用できなくなる機能
CONTIGUOUS	−
CORRESPONDING	−
COUNT_FLOAT	・集合関数 COUNT_FLOAT
CROSS	・結合表 CROSS JOIN
CURAID	−
CURRENT_DATE	・CURRENT_DATE 値関数 ・DEFAULT 句
CURRENT_ROLE	−
CURRENT_TIME	・CURRENT_TIME 値関数 ・DEFAULT 句
CURRENT_TIMESTAMP	・CURRENT_TIMESTAMP 値関数 ・DEFAULT 句
CURRENT_USER	・USER 値関数
CYCLE	CREATE SEQUENCE 文の順序数生成子循環オプション

表 A-30　削除できる予約語（D）

予約語	使用できなくなる機能
DATA	・抽象データ型の削除
DATABASE	−
DATE	・データ型 DATE ・スカラ関数 DATE ・CURRENT_DATE 値関数 ・DEFAULT 句
DAY	・データ型 INTERVAL YEAR TO DAY ・スカラ関数 DAY ・DAY によるラベル付き間隔の指定 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の連続認証失敗回数制限での連続認証失敗アカウントロック状態継続期間の DAY による設定 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の過去に使用したパスワードを新規パスワードとして再利用することを禁止する DAY による設定
DAYS	・スカラ関数 DAYS ・DAYS によるラベル付き間隔の指定 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の連続認証失敗回数制限での連続認証失敗アカウントロック状態継続期間の DAYS による設定 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の過去に使用したパスワードを新規パスワードとして再利用することを禁止する DAYS による設定
DEFER	−

予約語	使用できなくなる機能
DEMOTING	－
DEPTH	－
DEVICE	－
DIAGNOSTICS	・埋込み言語文法 GET DIAGNOSTICS
DICTIONARY	－
DIGITS	・スカラ関数 DIGITS
DIRECT	－
DO	・ルーチン制御 SQL FOR 文，WHILE 文
DOUBLE_PRECISION	－

表 A–31　削除できる予約語（E）

予約語	使用できなくなる機能
EACH	・トリガ
EDIT	－
ELSE	・ルーチン制御 SQL IF 文
ELSEIF	・ルーチン制御 SQL IF 文
ENCRYPT	－
END	・CASE 式 ・ルーチン制御 SQL 複合文，FOR 文，IF 文，WHILE 文
EQUALS	－
ESTIMATED	－
EXCEPTION	・埋込み言語文法 GET DIAGNOSTICS
EXIT	・ルーチン制御 SQL 複合文のハンドラ宣言
EXTERN	－
EXTRACT	－

表 A–32　削除できる予約語（F）

予約語	使用できなくなる機能
FALSE	・論理述語 IS FALSE
FIXED	－
FORCE	・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）のパスワードを構成する文字に使用する文字種の設定
FREE	・操作系 SQL FREE LOCATOR 文

予約語	使用できなくなる機能
FULL	－
FUNCTION	・ユーザ定義関数 ・プラグイン ・オブジェクト名による監査証跡絞込み

表 A-33　削除できる予約語（G）

予約語	使用できなくなる機能
GENERAL	－
GET	・埋込み言語文法 GET DIAGNOSTICS
GET_JAVA_STORED_ROUTINE_SOURCE	・スカラ関数 GET_JAVA_STORED_ROUTINE_SOURCE

表 A-34　削除できる予約語（H）

予約語	使用できなくなる機能
HANDLER	・ルーチン制御 SQL ハンドラ宣言
HELP	－
HEX	・スカラ関数 HEX
HOUR	・データ型 INTERVAL HOUR TO SECOND ・スカラ関数 HOUR ・HOUR によるラベル付き間隔の指定 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の連続認証失敗回数制限での連続認証失敗アカウントロック状態継続期間の HOUR による設定
HOURS	・HOURS によるラベル付き間隔の指定 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の連続認証失敗回数制限での連続認証失敗アカウントロック状態継続期間の HOURS による設定
HUGE	－

表 A-35　削除できる予約語（I）

予約語	使用できなくなる機能
IF	・ルーチン制御 SQL IF 文
IGNORE	－
INNER	・結合表 INNER JOIN ・暗号化機能
INOUT	・ストアドプロシジャ
INTERSECT	－
INTERVAL	・データ型 INTERVAL HOUR TO SECOND，INTERVAL YEAR TO DAY

予約語	使用できなくなる機能
	・順序数生成子定義（CREATE SEQUENCE 文）のログ出力間隔オプション ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の指定された認可識別子に対してパスワードの有効期間を設定
ISOLATION	・SQL コンパイルオプション ISOLATION
IS_USER_CONTAINED_IN_HDS_GROUP	・スカラ関数 IS_USER_CONTAINED_IN_HDS_GROUP

表 A-36　削除できる予約語（L）

予約語	使用できなくなる機能
LARGE	・データ型 LARGE DECIMAL，BINARY LARGE OBJECT
LEADING	・スカラ関数 TRIM の TRIM 指定 LEADING
LEAVE	・ルーチン制御 SQL LEAVE 文
LENGTH	・スカラ関数 LENGTH ・SQL コンパイルオプション SUBSTR LENGTH ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）のパスワード文字制限強化でのパスワードの最小許容バイト数の設定
LESS	－
LEVEL	・SQL コンパイルオプション OPTIMIZE LEVEL，ADD OPTIMIZE LEVEL
LIMIT	・LIMIT 句
LINES	－
LINK	－
LOCATOR	－
LOCKS	－
LOGID	－
LOGNAME	－
LOOP	－
LOWER	・スカラ関数 LOWER ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）のパスワードを構成する文字に使用する文字種の LOWER による設定

表 A-37　削除できる予約語（M）

予約語	使用できなくなる機能
MAXUSAGES	－
MINUTE	・スカラ関数 MINUTE ・MINUTE によるラベル付き間隔の指定

予約語	使用できなくなる機能
	・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の連続認証失敗回数制限での連続認証失敗アカウントロック状態継続期間の MINUTE による設定
MINUTES	・MINUTES によるラベル付き間隔の指定 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の連続認証失敗回数制限での連続認証失敗アカウントロック状態継続期間の MINUTES による設定
MOD	・スカラ関数 MOD
MONTH	・スカラ関数 MONTH ・MONTH によるラベル付き間隔の指定
MONTHS	・MONTHS によるラベル付き間隔の指定
MOVE	－

表 A-38　削除できる予約語（N）

予約語	使用できなくなる機能
NATURAL	－
NEW	・トリガ
NOWAIT	・排他オプション NO WAIT
NULLIF	・CASE 略式

表 A-39　削除できる予約語（O）

予約語	使用できなくなる機能
OFF	－
OID	－
OLD	・トリガ
ONLY	・改竄防止表 ・読み込み専用のビュー表 ・FOR READ ONLY
OPERATION	－
OPERATORS	－
OPTIMIZE	・SQL コンパイルオプション OPTIMIZE LEVEL，ADD OPTIMIZE LEVEL
OTHERS	・OTHERS を使用した分割格納条件変更
OUT	・ストアドプロシジャ
OUTER	・結合表 OUTER JOIN
OVER	・ウィンドウ関数
OVERFLOW	－

予約語	使用できなくなる機能
OWN	－

表 A-40　削除できる予約語（P）

予約語	使用できなくなる機能
PARAMETERS	－
PENDANT	－
PIC	－
PICTURE	－
PREALLOCATED	－
PREFERRED	－
PREORDER	－
PRIVATE	・抽象データ型
PROTECTED	・抽象データ型
PURGE	・表の分割条件変更（定義系 SQL ALTER TABLE WITHOUT PURGE） ・キーワード PURGE を使用した PURGE TABLE 文

表 A-41　削除できる予約語（R）

予約語	使用できなくなる機能
RANDOM	－
RD	－
READ	・読み込み専用のビュー表 ・FOR READ ONLY
RECOMPILE	－
RECOVERABLE	－
RECURSIVE	－
REF	－
REFERENCING	・トリガ
REGLIKE	－
RELEASING	－
RESTART	－
RETURN	・ユーザ定義関数
RETURNS	・ユーザ定義関数
RIGHT	・結合表 RIGHT OUTER JOIN

予約語	使用できなくなる機能
ROLE	－
ROOT	－
ROUTINE	・ルーチンのリコンパイル
ROW	・行単位の排他 LOCK ROW ・トリガ ・行単位（ROW 指定）インタフェース ・列追加定義（ALTER TABLE 文）で ON ROW EXISTS 指定の列を追加
ROWS	・一時表定義での一時表のデータが存在する期間

表 A-42　削除できる予約語（S）

予約語	使用できなくなる機能
SAVEPOINT	－
SCALE	－
SCAN	－
SCHEMAS	－
SCOPE	－
SD	－
SEARCH	－
SECOND	・データ INTERVAL HOUR TO SECOND ・スカラ関数 SECOND ・SECOND によるラベル付き間隔の指定
SECONDS	・SECONDS によるラベル付き間隔の指定
SENSITIVE	－
SEPARATE	－
SEPARATOR	－
SEQUENCE	・CREATE SEQUENCE 文 ・DROP SEQUENCE 文
SESSION_USER	－
SFLIKE	－
SHORT	－
SIGNAL	・ルーチン制御 SQL 文 SIGNAL 文
SIMILAR	・SIMILAR 述語 ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）のパスワード文字制限強化でのパスワードを構成する文字種に関する設定

予約語	使用できなくなる機能
SLOCK	－
SQL_STANDARD	－
SQLCODE_OF_LAST_CONDITION	・値指定　SQLCODE_OF_LAST_CONDITION
SQLCODE_TYPE	－
SQLDA	－
SQLERRM	－
SQLERRM_OF_LAST_CONDITION	・値指定　SQLERRM_OF_LAST_CONDITION
SQLERRMC	－
SQLERRML	－
SQLEXCEPTION	－
SQLWARN	－
STATIC	－
STOP	－
STOPPING	－
SUBSTR	・スカラ関数 SUBSTR ・SQL コンパイルオプション SUBSTR LENGTH
SYSTEM_USER	－

表 A-43　削除できる予約語（T）

予約語	使用できなくなる機能
TEST	・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）のパスワード文字制限定義での不適合パスワード確認
TEXT	－
THEN	・CASE 式 ・ルーチン制御 SQL IF 文
THERE	－
TIME	・データ型 TIME ・スカラ関数 TIME ・CURRENT_TIME 値関数 ・DEFAULT 句 規定値
TIMESTAMP	・データ型 TIMESTAMP ・スカラ関数 TIMESTAMP

予約語	使用できなくなる機能
	・CURRENT_TIMESTAMP 値関数 ・DEFAULT 句 規定値
TIMESTAMP_FORMAT	・スカラ関数 TIMESTAMP_FORMAT
TRAILING	・スカラ関数 TRIM の TRIM 指定 TRAILING
TRANSACTION	－
TREAT	－
TRIGGER	・トリガ ・オブジェクト名による監査証跡絞込み
TRIM	・スカラ関数 TRIM
TRUE	・論理述語 IS TRUE
TRUNCATE	・キーワード TRUNCATE を使用した PURGE TABLE 文
TYPE	・抽象データ型 ・プラグイン ・オブジェクト名による監査証跡絞込み

表 A-44　削除できる予約語（U）

予約語	使用できなくなる機能
UAMT	－
UBINBUF	－
UCHAR	－
UDATE	－
UHANT	－
UHDATE	－
UNDER	・抽象データ型
UNIFY_2000	－
UNIONALL	－
UNKNOWN	・論理述語 IS UNKNOWN
UNLIMITED	・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）の連続認証失敗アカウントロック状態を継続する期間での継続する期間を無期限にする機能
UNLOCK	－
UNTIL	・UNTIL DISCONNECT
UPPER	・スカラ関数 UPPER ・CONNECT 関連セキュリティ機能（CREATE CONNECTION SECURITY）のパスワードを構成する文字に使用する文字種の UPPER による設定

予約語	使用できなくなる機能
USAGE	・順序数生成子定義（CREATE SEQUENCE 文）の FOR PUBLIC USAGE
USE	－
UTIME	－
UTXTBUF	－

表 A-45　削除できる予約語（V）

予約語	使用できなくなる機能
VALUE	・スカラ関数 VALUE ・NEXT VALUE 式
VARCHAR_FORMAT	・スカラ関数 VARCHAR_FORMAT
VARIABLE	－
VARYING	・データ型 CHARACTER VARYING，NATIONAL CHARACTER VARYING
VIRTUAL	－
VISIBLE	－
VOLATILE	－
VOLUME	－
VOLUMES	－

表 A-46　削除できる予約語（W）

予約語	使用できなくなる機能
WHEN	・CASE 式 ・トリガ
WHILE	・改竄防止表 ・ルーチン制御 SQL WHILE 文

表 A-47　削除できる予約語（X）

予約語	使用できなくなる機能
XLOCK	－
XML	・XML 型 ・XML コンストラクタ関数
XMLAGG	・XMLAGG 集合関数
XMLEXISTS	・XMLEXISTS 述語
XMLPARSE	・XMLPARSE 関数
XMLQUERY	・XMLQUERY 関数

予約語	使用できなくなる機能
XMLSERIALIZE	・XMLSERIALIZE 関数

表 A-48　削除できる予約語（Y）

予約語	使用できなくなる機能
YEAR	・データ型 INTERVAL YEAR TO DAY ・スカラ関数 YEAR ・YEAR によるラベル付き間隔の指定
YEARS	・YEARS によるラベル付き間隔の指定

付録 B　SQL 一覧

各 SQL の種類と使用できる SQL の一覧を次の表に示します。「OLTP 下」とは，OLTP 下の X/Open に従った UAP で，該当する SQL が使用できるかどうかを表しています。

表 B-1　SQL 一覧（定義系 SQL）

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
ALTER INDEX （インデクス定義変更）	インデクスの名称を変更します。	○	○	×
ALTER PROCEDURE （手続きの SQL オブジェクトの再作成）	手続きの SQL オブジェクトを再作成します。	○	○	×
ALTER ROUTINE （関数，手続き，及びトリガの SQL オブジェクトの再作成）	関数，手続き，及びトリガの SQL オブジェクトを再作成します。	○	○	×
ALTER TABLE （表定義変更）	• 実表に新しい列を追加します。 • データ型を変更します。 • 可変長データ型の列の最大長を大きくします。 • データの格納されていない実表の列を削除します。 • データの格納されていない実表のクラスタキーの一意性制約を変更します。 • 表や列の名称を変更します。	○	○	×
ALTER TRIGGER （トリガの SQL オブジェクトの再作成）	トリガの SQL オブジェクトを再作成します。	○	○	×
COMMENT （注釈付加）	表，及び列に注釈を付けます。	○	○	×
CREATE AUDIT （監査対象イベントの定義）	監査証跡として記録する監査イベント及びその対象を定義します。	○	○	×
CREATE CONNECTION SECURITY （CONNECT 関連セキュリティ機能の定義）	CONNECT 関連セキュリティ機能に関するセキュリティ項目を定義します。	○	○	×
CREATE ［PUBLIC］ FUNCTION （関数定義，パブリック関数定義）	関数，パブリック関数を定義します。	○	○	×

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
CREATE INDEX （インデクス定義）	実表の列にインデクス（昇順，降順）を定義します。	○	○	×
CREATE ［PUBLIC］ PROCEDURE （手続き定義，パブリック手続き定義）	手続き，パブリック手続きを定義します。	○	○	×
CREATE SCHEMA （スキーマ定義）	スキーマを定義します。	○	○	×
CREATE SEQUENCE （順序数生成子定義）	順序数生成子を定義します。	○	○	×
CREATE TABLE （表定義）	実表を定義します。	○	○	×
CREATE TRIGGER （トリガ定義）	トリガを定義します。	○	○	×
CREATE TYPE （型定義）	抽象データ型を定義します。	○	○	×
CREATE ［PUBLIC］ VIEW （ビュー定義，パブリックビュー定義）	ビュー表，パブリックビュー表を定義します。	○	○	×
DROP AUDIT （監査対象イベントの削除）	CREATE AUDIT で定義した監査対象イベントと内容が一致する定義を，監査対象から削除します。	○	○	×
DROP CONNECTION SECURITY （CONNECT 関連セキュリティ機能の削除）	CONNECT 関連セキュリティ機能に関するセキュリティ項目を削除します。	○	○	×
DROP DATA TYPE （ユーザ定義型削除）	ユーザ定義型を削除します。	○	○	×
DROP［PUBLIC］FUNCTION （関数削除，パブリック関数削除）	関数，パブリック関数を削除します。	○	○	×
DROP INDEX （インデクス削除）	インデクスを削除します。	○	○	×
DROP［PUBLIC］PROCEDURE （手続き削除，パブリック手続き削除）	手続き，パブリック手続きを削除します。	○	○	×
DROP SCHEMA	スキーマを削除します。	○	○	×

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
（スキーマ削除）				
DROP SEQUENCE （順序数生成子削除）	順序数生成子を削除します。	○	○	×
DROP TABLE （表削除）	実表を削除します。さらに，その実表に対するインデクス，注釈，アクセス権限，ビュー表，及びトリガも削除します。	○	○	×
DROP TRIGGER （トリガ削除）	トリガを削除します。	○	○	×
DROP［PUBLIC］VIEW （ビュー表，パブリックビュー表の削除）	ビュー表，パブリックビュー表を削除します。	○	○	×
GRANT AUDIT （監査人のパスワード変更）	監査人のパスワードを変更します。	○	○	×
GRANT CONNECT （CONNECT 権限定義）	ユーザに CONNECT 権限を与えます。	○	○	×
GRANT DBA （DBA 権限定義）	ユーザに DBA 権限を与えます。	○	○	×
GRANT RDAREA （RD エリア利用権限定義）	ユーザに RD エリアの利用権限を与えます。	○	○	×
GRANT SCHEMA （スキーマ定義権限定義）	ユーザにスキーマ定義権限を与えます。	○	○	×
GRANT SCHEMA OPERATION （スキーマ操作権限定義）	ユーザにスキーマ操作権限を与えます。	○	○	×
GRANT アクセス権限 （アクセス権限定義）	ユーザにアクセス権限を与えます。	○	○	×
REVOKE CONNECT （CONNECT 権限削除）	ユーザに与えた CONNECT 権限を取り消します。	○	○	×
REVOKE DBA （DBA 権限削除）	ユーザに与えた DBA 権限を取り消します。	○	○	×
REVOKE RDAREA （RD エリア利用権限削除）	ユーザに与えた RD エリアの利用権限を取り消します。	○	○	×
REVOKE SCHEMA （スキーマ定義権限削除）	ユーザに与えたスキーマ定義権限を取り消します。	○	○	×
REVOKE SCHEMA OPERATION	ユーザに与えたスキーマ操作権限を取り消します。	○	○	×

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
（スキーマ操作権限削除）				
REVOKE アクセス権限 （アクセス権限削除）	ユーザに与えたアクセス権限を取り消します。	○	○	×

（凡例）

○：使用できます。

×：使用できません。

表 B-2　SQL 一覧（操作系 SQL）

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
ALLOCATE CURSOR 文 （カーソル割当て）	PREPARE 文で前処理した SELECT 文，又は手続きから返却された結果集合の組に対してカーソルを割り当てます。	○	○	○
ASSIGN LIST 文 （リスト作成）	実表からリストを作成します。	○	○	○
CALL 文※ （手続きの呼び出し）	手続きを呼び出します。	○	○	○
CLOSE 文 （カーソルクローズ）	カーソルを閉じます。	○	○	○
DEALLOCATE PREPARE 文 （前処理解除）	PREPARE 文で前処理された SQL 文の割り当てを解放します。	○	○	○
DECLARE CURSOR （カーソル宣言）	SELECT 文の検索結果を FETCH 文で 1 行ずつ取り出すために，カーソルを宣言します。	○	○	○
DELETE 文 （行削除）	指定した探索条件を満足する行，又はカーソルが指している行を削除します。	○	○	○
準備可能動的 DELETE 文：位置付け （前処理可能なカーソルを使用した行削除）	指定したカーソルが指している行を削除します。動的に実行する場合に使用します。	○	○	○
DESCRIBE 文 （検索情報，入出力情報の受け取り）	PREPARE 文で前処理した SQL の検索情報，出力情報，又は入力情報を SQL 記述領域に返します。	○	○	○

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
DESCRIBE CURSOR 文 （カーソルの検索情報の受け取り）	手続きから返却された結果集合を参照するカーソルの検索情報を，SQL 記述領域に返します。	○	○	○
DESCRIBE TYPE 文 （ユーザ定義型の定義情報の受け取り）	PREPARE 文で前処理した SQL の検索項目情報に直接又は間接的に含まれるユーザ定義型の定義情報（各属性のデータコード，データ長など）を SQL 記述領域に受け取ります。	○	○	○
DROP LIST 文 （リスト削除）	リストを削除します。	○	○	○
EXECUTE 文 （SQL の実行）	PREPARE 文で前処理した SQL を実行します。	○	○	○
EXECUTE IMMEDIATE 文 （SQL の前処理と実行）	文字列で与えられた SQL を，前処理して実行します。	○	○	○
FETCH 文 （データの取り出し）	取り出す行を示すカーソルの位置を次の行に進め，その行の列の値を INTO 句で指定した埋込み変数に読み込みます。	○	○	○
FREE LOCATOR 文 （位置付け子の無効化）	位置付け子を無効にします。	○	○	○
INSERT 文 （行挿入）	表に行を挿入します。直接，値を指定して一つの行を挿入できます。また，SELECT 文を使用して一つ，又は複数の行を挿入できます。	○	○	○
OPEN 文 （カーソルオープン）	カーソルを開きます。DECLARE CURSOR で宣言したカーソル，又は ALLOCATE CURSOR 文で割り当てたカーソルを，検索結果の先頭の行の直前に位置づけて，検索結果を取り出せる状態にします。	○	○	○
PREPARE 文 （SQL の前処理）	文字列で与えられた SQL を実行するための前処理をして，その SQL に名称（SQL 文識別子，又は拡張文名）を付けます。	○	○	○
PURGE TABLE 文 （全行削除）	実表中のすべての行を削除します。	○	○	×
1 行 SELECT 文 （1 行検索）	表のデータを検索します。表から 1 行だけデータを取り出す場合は，カーソルを使用しないでデータを取り出す 1 行 SELECT 文を指定します。	○	○	○

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
動的 SELECT 文 (動的検索)	表のデータを検索します。動的 SELECT 文は，PREPARE 文で前処理します。検索するときは，DECLARE CURSOR によってカーソルを宣言するか，ALLOCATE CURSOR 文によってカーソルを割り当ててから，そのカーソルを使用して検索結果を 1 行ずつ取り出します。	○	○	○
UPDATE 文 (データ更新)	表の指定した探索条件を満足する行，又はカーソルが指している行の指定した列の値を更新します。	○	○	○
準備可能動的 UPDATE 文：位置付け (前処理可能なカーソルを使用したデータ更新)	指定したカーソルが指している行の指定した列の値を更新します。動的に実行する場合に使用します。	○	○	○
代入文 (値の代入)	値を代入します。	○	○	○

(凡例)

○：使用できます。

×：使用できません。

注※

OLTP 下で手続きを呼び出す場合，その手続き中に PURGE TABLE 文，COMMIT 文，又は ROLLBACK 文があると，手続きは実行できません。

表 B-3　SQL 一覧（制御系 SQL）

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
CALL COMMAND 文 (コマンド又はユティリティの実行)	HiRDB のコマンド又はユティリティを実行します。	○	○	○
COMMIT 文 (トランザクションの正常終了)	現在のトランザクションを正常終了させて，同期点を設定し 1 コミットメント単位を生成します。そのトランザクションが更新したデータベースの内容を有効にします。	○	○	×
CONNECT 文 (HiRDB との接続)	HiRDB に認可識別子及びパスワードを連絡して，UAP が HiRDB を使用できる状態にします。	○	○	×

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
DISCONNECT 文 (HiRDB との切り離し)	現在のトランザクションを正常終了させて，同期点を設定し 1 コミットメント単位を生成します。その後，UAP を HiRDB から切り離します。	○	○	×
LOCK 文 (表の排他制御)	指定した表に排他制御をします。	○	○	○
ROLLBACK 文 (トランザクションの取り消し)	現在のトランザクションを取り消して，そのトランザクション内でのデータベースの更新を無効にします。	○	○	×
SET SESSION AUTHORIZATION 文 (実行ユーザの変更)	接続中のユーザを変更します。	○	○	○

(凡例)

○：使用できます。

×：使用できません。

表 B-4　SQL 一覧（埋込み言語）

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
BEGIN DECLARE SECTION (埋込み SQL 開始宣言)	埋込み変数宣節の始まりを示します。埋込み変数宣言節には，SQL 中で使用する埋込み変数，及び標識変数を指定します。	○	○	○
END DECLARE SECTION (埋込み SQL 終了宣言)	埋込み変数宣言節の終わりを示します。	○	○	○
ALLOCATE CONNECTION HANDLE (接続ハンドルの割り当て)	複数接続機能を使用した環境で，UAP が使用する接続ハンドルを割り当てます。	○	○	×
FREE CONNECTION HANDLE (接続ハンドルの解放)	ALLOCATE CONNECTION HANDLE で割り当てた接続ハンドルを解放します。	○	○	×
DECLARE CONNECTION HANDLE SET (使用する接続ハンドルの宣言)	複数接続機能を使用した環境で，UAP 中の SQL が使用する接続ハンドルを宣言します。	○	○	○※1
DECLARE CONNECTION HANDLE UNSET (使用する接続ハンドルの全解除)	この文以前に DECLARE CONNECTION HANDLE SET で指定	○	×	○※2

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
	した接続ハンドルの，使用の宣言をすべて解除します。			
GET CONNECTION HANDLE (接続ハンドル取得)	X/Open XA インタフェース環境下で複数接続機能を使用する場合，UAP が使用する接続ハンドルを割り当てます。	○	○	○
COPY (登録集原文の引き込み)	登録集原文をソースプログラム中に引き込みます。	○	○	○
GET DIAGNOSTICS (診断情報取得)	直前に実行した SQL 文が CREATE PROCEDURE，又は CALL 文の場合に，そのエラー情報を診断領域から取得します。	○	○	○
COMMAND EXECUTE (UAP からのコマンド実行)	UAP 中から，HiRDB のコマンド，及び OS のコマンドを実行します。	○	×	×
SQL 先頭子	SQL の始まりを示します。	○	○	○
SQL 終了子	SQL の終わりを示します。	○	○	○
WHENEVER (埋込み例外宣言)	SQL の実行後に HiRDB が SQL 連絡領域に設定したリターンコードによって，UAP の処理を宣言します。	○	○	○
SQLCODE 変数	SQL の実行後に HiRDB から返されるリターンコードを受け取ります。	○	○	○
SQLSTATE 変数	SQL の実行後に HiRDB から返されるリターンコードを受け取ります。	○	○	○
PDCNCTHDL 型変数の宣言	複数接続機能を使用した環境で，使用する接続ハンドル型の変数を宣言します。	○	○	○
INSTALL JAR (JAR ファイルの登録)	HiRDB サーバに JAR ファイルをインストールします。	○	×	×
REPLACE JAR (JAR ファイルの再登録)	HiRDB サーバに JAR ファイルを上書きインストールします。	○	×	×
REMOVE JAR (JAR ファイルの削除)	HiRDB サーバから JAR ファイルをアンインストールします。	○	×	×
INSTALL CLIB (外部 C ライブラリファイルの登録)	HiRDB サーバに外部 C ライブラリファイルをインストールします。	○	×	×
REPLACE CLIB (外部 C ライブラリファイルの再登録)	HiRDB サーバに外部 C ライブラリファイルを上書きインストールします。	○	×	×

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
REMOVE CLIB (外部 C ライブラリファイルの削除)	HiRDB サーバから外部 C ライブラリファイルをアンインストールします。	○	×	×
DECLARE AUDIT INFO SET (ユーザ任意接続情報の設定)	HiRDB サーバにアクセスするアプリケーションのアカウント情報などのユーザ任意接続情報を設定します。	○	○	○

(凡例)

○：使用できます。

×：使用できません。

注※1

GET CONNECTION HANDLE で接続ハンドルを取得した場合に使用できます。

注※2

使用できる SQL は C 言語だけです。

表 B-5　SQL 一覧（ルーチン制御 SQL）

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
複合文 (複数文実行)	複数の SQL 文をまとめて一つの SQL 文として実行します。	△	△	×
代入文 (値の代入)	SQL 変数，又は SQL パラメタに値を代入します。	△	△	×
IF 文 (条件分岐)	条件によって，SQL 文を実行します。	△	△	×
RETURN 文 (関数の戻り値の返却)	関数の戻り値を返却します。	△※1	△※1	×
WHILE 文 (繰り返し実行)	SQL 文の実行を繰り返します。	△	△	×
FOR 文 (各行に対する繰り返し実行)	表の各行に対して SQL 文の実行を繰り返します。	△※3	△※3	×
LEAVE 文 (実行の途中終了)	複合文，又は WHILE 文から抜けて，その文の実行を終了します。	△	△	×
WRITE LINE 文 (ファイルへの文字列出力)	指定した値式の文字列をファイルに出力します。	△	△	×

種類	機能	使用できる SQL		
		C 言語	COBOL 言語	OLTP 下
SIGNAL 文 （エラーの通知）	エラーを発生させて通知します。	△※2	△※2	×
RESIGNAL 文 （エラーの再通知）	エラーを発生させて再通知します。	△※2	△※2	×

（凡例）

△：直接 UAP で使用することはできませんが，CREATE PROCEDURE，CREATE FUNCTION，及び CREATE TRIGGER の中で，SQL 手続き，SQL 関数，及びトリガの動作を定義するために使用できます。

×：使用できません。

注

ルーチン制御 SQL 以外の，手続き定義中で指定できる SQL 文は，CALL 文，CLOSE 文，DECLARE CURSOR，DELETE 文，FETCH 文，INSERT 文，OPEN 文，PURGE TABLE 文，1 行 SELECT 文，UPDATE 文，COMMIT 文，LOCK 文，及び ROLLBACK 文です。関数の場合は，ルーチン制御 SQL 以外の SQL は使用できません。

注※1

CREATE PROCEDURE，及び CREATE TRIGGER の中で，SQL 手続き，及びトリガの動作を定義する場合は使用できません。

注※2

CREATE FUNCTION の中で，SQL 関数を定義する場合は使用できません。

注※3

CREATE FUNCTION の中では使用できません。

付録 C　例題用データベース

マニュアル内の使用例として，使用している表の基本構成例を次の図に示します。

図 C-1　表の基本構成例

表名：在庫表（ZAIKO）

CHAR(4)	NCHAR(8)	NCHAR(1)	INTEGER	INTEGER
商品コード (SCODE)	商品名 (SNAME)	色 (COL)	単価 (TANKA)	在庫量 (ZSURYO)
101M	ブラウス	白	3500	85
101L	ブラウス	青	3500	62
201M	ポロシャツ	白	3640	29
202M	ポロシャツ	赤	3640	67
302S	スカート	白	5110	65
353M	スカート	緑	4760	56
353L	スカート	赤	4760	18
411M	セーター	青	8400	12
412M	セーター	赤	8400	22
591S	ソックス	白	250	280
591M	ソックス	青	250	90
591L	ソックス	赤	250	300

表名：受注表（JUTYU）

CHAR(6)	CHAR(5)	CHAR(4)	INTEGER	DATE	TIME
伝票番号 (DNO)	得意先コード (TCODE)	商品コード (SCODE)	受注量 (JSURYO)	受付日付 (JDATE)	受付時刻 (JTIME)
026551	TT002	101M	10	1995-06-14	09:23:11
026552	TT002	591M	25	1995-06-14	09:23:11
026553	TH001	353M	8	1995-06-14	10:10:55
026554	TK001	411M	6	1995-06-14	10:15:47
026555	TA001	591M	30	1995-06-14	10:15:47
026556	TT002	202M	10	1995-06-14	11:48:09
026557	TZ001	411M	5	1995-06-14	13:02:00
026558	TZ001	412M	4	1995-06-14	13:02:00
026559	TH001	591M	80	1995-06-14	14:04:16
026560	TT001	591L	10	1995-06-14	15:31:20

付録 D　バージョン，リビジョンによる SQL 構文の省略時解釈の変更点

付録 D.1　変更点

バージョン 09-50 より前のバージョンからバージョンアップを行う場合は，省略時解釈変更によるメリット及びデメリットを確認してください。確認の結果，旧バージョンとの互換性を重視する場合は，旧バージョンと同等の省略値解釈になる互換モードを適用してください。

(1)　バージョンによって省略時解釈が異なる SQL 構文

バージョンによって省略時解釈が異なる SQL 構文を次の表に示します。

表 D–1　バージョンによって省略時解釈が異なる SQL 構文

SQL		09-50 以降の省略時解釈	0904 互換モードを適用時の省略時解釈	09-50 より前の省略値解釈	省略値変更によるメリット	省略値変更によるデメリット
CREATE〔PUBLIC〕PROCEDURE（手続き定義，パブリック手続き定義）	SQL コンパイルオプションの「ISOLATION データ保証レベル〔FOR UPDATE { EXCLUSIVE | COMPATIBLE }〕」	FOR UPDATE { EXCLUSIVE | COMPATIBLE }を指定しない場合，FOR UPDATE EXCLUSIVE を仮定します。	→	FOR UPDATE EXCLUSIVE を指定しない場合，FOR UPDATE COMPATIBLE を仮定します。	更新を前提とした検索に対する排他モードのロックを自動設定	排他モード変更による SQL 文の見直し
CREATE TRIGGER（トリガ定義）	SQL コンパイルオプションの「ISOLATION データ保証レベル〔FOR UPDATE { EXCLUSIVE | COMPATIBLE }〕」	FOR UPDATE { EXCLUSIVE | COMPATIBLE }を指定しない場合，FOR UPDATE EXCLUSIVE を仮定します。	→	FOR UPDATE EXCLUSIVE を指定しない場合，FOR UPDATE COMPATIBLE を仮定します。	更新を前提とした検索に対する排他モードのロックを自動設定	排他モード変更による SQL 文の見直し
ALTER PROCEDURE（手続きの SQL オブジェクトの再作成）	SQL コンパイルオプションの「ISOLATION データ保証レベル〔FOR UPDATE { EXCLUSIVE |	FOR UPDATE { EXCLUSIVE | COMPATIBLE }を指定しない場合，FOR UPDATE	→	FOR UPDATE EXCLUSIVE を指定しない場合，FOR UPDATE COMPATIBLE を仮定します。	更新を前提とした検索に対する排他モードのロックを自動設定	排他モード変更による SQL 文の見直し

SQL		09-50 以降の省略時解釈	0904 互換モードを適用時の省略時解釈	09-50 より前の省略値解釈	省略値変更によるメリット	省略値変更によるデメリット
	COMPATIBLE }]」	EXCLUSIVE を仮定します。				
ALTER TRIGGER（トリガの SQL オブジェクトの再作成）	SQL コンパイルオプションの「ISOLATION データ保証レベル 〔FOR UPDATE { EXCLUSIVE | COMPATIBLE }]」	FOR UPDATE { EXCLUSIVE | COMPATIBLE }を指定しない場合，FOR UPDATE EXCLUSIVE を仮定します。	→	FOR UPDATE EXCLUSIVE を指定しない場合，FOR UPDATE COMPATIBLE を仮定します。	更新を前提とした検索に対する排他モードのロックを自動設定	排他モード変更による SQL 文の見直し
ALTER ROUTINE（関数，手続き，及びトリガの SQL オブジェクトの再作成）	SQL コンパイルオプションの「ISOLATION データ保証レベル 〔FOR UPDATE { EXCLUSIVE | COMPATIBLE }]」	FOR UPDATE { EXCLUSIVE | COMPATIBLE }を指定しない場合，FOR UPDATE EXCLUSIVE を仮定します。	→	FOR UPDATE EXCLUSIVE を指定しない場合，FOR UPDATE COMPATIBLE を仮定します。	更新を前提とした検索に対する排他モードのロックを自動設定	排他モード変更による SQL 文の見直し
CREATE TABLE（表定義）	表オプションの「PCTFREE」	「PCTFREE」を指定しない場合，PCTFREE=(30,10)を仮定します。ただし，クラスタキーのない FIX 表の場合で，PCTFREE を省略したときは，PCTFREE =（0，0）を仮定します。	「PCTFREE」を指定しない場合，PCTFREE=(30,10)を仮定します。ただし，クラスタキーのない FIX 表の場合で，PCTFREE を省略したときは，PCTFREE =（0，0）を仮定します。	「PCTFREE」を指定しない場合，PCTFREE=(30,10)を仮定します。	DB の格納効率向上	なし
	表オプションの「SEGMENT REUSE」	「SEGMENT REUSE」の指定有無に関係なく，空き領域の再利用機能を適用します。ただし，リバランス表，一時表の場合は，空き領域	「SEGMENT REUSE」の指定有無に関係なく，空き領域の再利用機能を適用します。ただし，リバランス表，一時表の場合は，空き領域	「SEGMENT REUSE」を指定しない場合，空き領域の再利用機能を適用しません。	DB の空き領域のサーチ方法の変更による SQL 安定稼働	なし

SQL		09-50 以降の省略時解釈	0904 互換モードを適用時の省略時解釈	09-50 より前の省略値解釈	省略値変更によるメリット	省略値変更によるデメリット
		の再利用機能を適用しません。	の再利用機能を適用しません。			
	表オプションの「SEGMENT REUSE」の「OPTION 再利用オプション値」	再利用オプション値として常に 3 を仮定し，再利用オプション機能を適用します。	再利用オプション値として常に 3 を仮定し，再利用オプション機能を適用します。	「OPTION 再利用オプション値」を指定しない場合，再利用オプション機能を適用しません。	DB の空き領域のサーチ方法の変更による SQL 安定稼働	なし
	列定義の「NO SPLIT」	定義長が 256 バイト以上の可変長文字列（VARCHAR, NVARCHAR, MVARCHAR）を定義する場合，ノースプリットオプションを常に適用します。	定義長が 256 バイト以上の可変長文字列（VARCHAR, NVARCHAR, MVARCHAR）を定義する場合，ノースプリットオプションを常に適用します。	「NO SPLIT」を指定しない場合，ノースプリットオプションを適用しません。	DB 容量削減	なし
ALTER TABLE（表定義変更）	列追加定義の「NO SPLIT」	定義長が 256 バイト以上の可変長文字列（VARCHAR, NVARCHAR, MVARCHAR）を定義する場合，ノースプリットオプションを常に適用します。	定義長が 256 バイト以上の可変長文字列（VARCHAR, NVARCHAR, MVARCHAR）を定義する場合，ノースプリットオプションを常に適用します。	「NO SPLIT」を指定しない場合，ノースプリットオプションを適用しません。	DB 容量削減	なし

（凡例）

→：バージョン 09-04 の省略時解釈と同じであることを示します。

(2) 指定不要になった SQL 構文

省略時解釈以外の指定をする必要がないため指定不要になった SQL 構文を，次の表に示します。

表 D–2 指定不要になった SQL 構文

SQL	
ALTER TABLE（表定義変更）	列追加定義の「NO SPLIT」
CREATE TABLE（表定義）	列定義の「NO SPLIT」

SQL	
	表オプションの「SEGMENT REUSE」の「OPTION 再利用オプション値」

付録 E　機能ごとの SQL 実行可否一覧

付録 E.1　リードレプリカ機能使用時の SQL 実行可否一覧

リードレプリカ機能使用時の SQL の実行可否を次の表に示します。なお，リードレプリカ機能の詳細は，マニュアル「HiRDB システム運用ガイド」の「リードレプリカ機能の運用」を参照してください。

表 E-1　リードレプリカ機能使用時の SQL 実行可否一覧（定義系 SQL）

定義系 SQL	リードレプリカ機能使用時	
	更新 DB での SQL の実行可否	参照 DB での SQL の実行可否
ALTER INDEX	○	×
ALTER PROCEDURE	○	×
ALTER ROUTINE	○	×
ALTER TABLE	○	×
ALTER TRIGGER	○	×
COMMENT	○	×
CREATE AUDIT	○	×
CREATE CONNECTION SECURITY	○	×
CREATE FUNCTION	○	×
CREATE INDEX	○	×
CREATE PROCEDURE	○	×
CREATE SCHEMA	○	×
CREATE SEQUENCE	○	×
CREATE TABLE	○	×
CREATE TRIGGER	○	×
CREATE TYPE	○	×
CREATE VIEW	○	×
DROP AUDIT	○	×
DROP CONNECTION SECURITY	○	×
DROP DATA TYPE	○	×
DROP FUNCTION	○	×
DROP INDEX	○	×
DROP PROCEDURE	○	×

定義系 SQL	リードレプリカ機能使用時	
	更新 DB での SQL の実行可否	参照 DB での SQL の実行可否
DROP SCHEMA	○	×
DROP SEQUENCE	○	×
DROP TABLE	○	×
DROP TRIGGER	○	×
DROP VIEW	○	×
GRANT AUDIT	○	×
GRANT CONNECT	○	×
GRANT DBA	○	×
GRANT RDAREA	○	×
GRANT SCHEMA	○	×
GRANT SCHEMA OPERATION	○	×
GRANT アクセス権限	○	×
REVOKE CONNECT	○	×
REVOKE DBA	○	×
REVOKE RDAREA	○	×
REVOKE SCHEMA	○	×
REVOKE SCHEMA OPERATION	○	×
REVOKE アクセス権限	○	×

（凡例）

○：実行できます。

×：実行できません。

表 E-2　リードレプリカ機能使用時の SQL 実行可否一覧（操作系 SQL）

操作系 SQL	リードレプリカ機能使用時	
	更新 DB での SQL の実行可否	参照 DB での SQL の実行可否
ALLOCATE CURSOR	○	○
ASSIGN LIST	○	○
CALL	○	○
CLOSE	○	○
DEALLOCATE PREPARE	○	○
DECLARE CURSOR	○	○

操作系 SQL	リードレプリカ機能使用時	
	更新 DB での SQL の実行可否	参照 DB での SQL の実行可否
DELETE	○	×
準備可能動的 DELETE	○	×
DESCRIBE	○	○
DESCRIBE TYPE	○	○
DESCRIBE CURSOR	○	○
DROP LIST	○	○
EXECUTE	○	○
EXECUTE IMMEDIATE	○	○
FETCH	○	○
FREE LOCATOR	○	○
INSERT	○	×
OPEN	○	○
PREPARE	○	○
PURGE TABLE	○	×
1 行 SELECT	○	○
動的 SELECT	○	○
UPDATE	○	×
準備可能動的 UPDATE	○	×
代入文	○	○

（凡例）

○：実行できます。

×：実行できません。

表 E-3　リードレプリカ機能使用時の SQL 実行可否一覧（制御系 SQL）

制御系 SQL	リードレプリカ機能使用時	
	更新 DB での SQL の実行可否	参照 DB での SQL の実行可否
CALL COMMAND	○	○
COMMIT	○	○
CONNECT	○	○
DISCONNECT	○	○
LOCK	○	○

制御系 SQL	リードレプリカ機能使用時	
	更新 DB での SQL の実行可否	参照 DB での SQL の実行可否
ROLLBACK	○	○
SET SESSION AUTHORIZATION	○	○

（凡例）

○：実行できます。

×：実行できません。

索引

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

あ

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

ふ

へ

ほ

ま

み

も

ゆ

よ

ら

り

る

れ

ろ