HITACHI
Inspire the Next

OpenTP1 Version 7

高速トランザクション処理基盤

TP1/EE/Extended Data Cache SQL プログラミング

手引・文法書

3000-3-F56

マニュアルの購入方法

このマニュアル，および関連するマニュアルをご購入の際は，巻末の「ソフトウェアマニュアルのサービス ご案内」をご参照ください。

■対象製品

P-9V64-9411 uCosminexus TP1/EE/Extended Data Cache 01-01（適用 OS：Red Hat Enterprise Linux 5.1（IPF））
こ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。
この製品は，ISO9001 および TickIT の認証を受けた品質マネジメントシステムで開発されました。

■輸出時の注意

本製品を輸出される場合には，外国為替および外国貿易法ならびに米国の輸出管理関連法規などの規制をご確認の上，必要な手続きをお取りください。
なお，ご不明な場合は，弊社担当営業にお問い合わせください。

■商標類

Linux は，Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。
Red Hat は，米国およびその他の国で Red Hat, Inc. の登録商標若しくは商標です。

■発行

2008 年 11 月（第 1 版）3000-3-F56

■著作権

はじめに

このマニュアルは，プログラムプロダクト　P-9V64-9411　uCosminexus TP1/EE/Extended Data Cache（以降，XDB と表記します）に使用する，SQL の文法について説明したものです。

■対象読者

高速データ処理基盤である XDB を使ったシステムで表を設計・作成する方，および UAP を作成・実行する方を対象としています。

なお，このマニュアルは，次に示す知識があることを前提にして説明されています。

- SQL の基礎的な知識
- COBOL 言語のプログラミングの基礎的な知識
- Linux(R) の基礎的な知識

■マニュアルの構成

このマニュアルは，次に示す章と付録から構成されています。

第 1 章　基本項目

SQL を使用する上での基本項目について説明しています。

第 2 章　構成要素

SQL の構成要素について説明しています。

第 3 章　定義系 SQL

定義系 SQL の機能，形式，および規則について説明しています。

第 4 章　操作系 SQL

操作系 SQL の機能，形式および規則について説明しています。

第 5 章　埋め込み言語文法

埋め込み言語の機能，形式および規則について説明しています。

第 6 章　SQL の記述テクニック

性能が向上する SQL の書き方や，検索時に使用するインデクスを変更する SQL の書き方について説明しています。

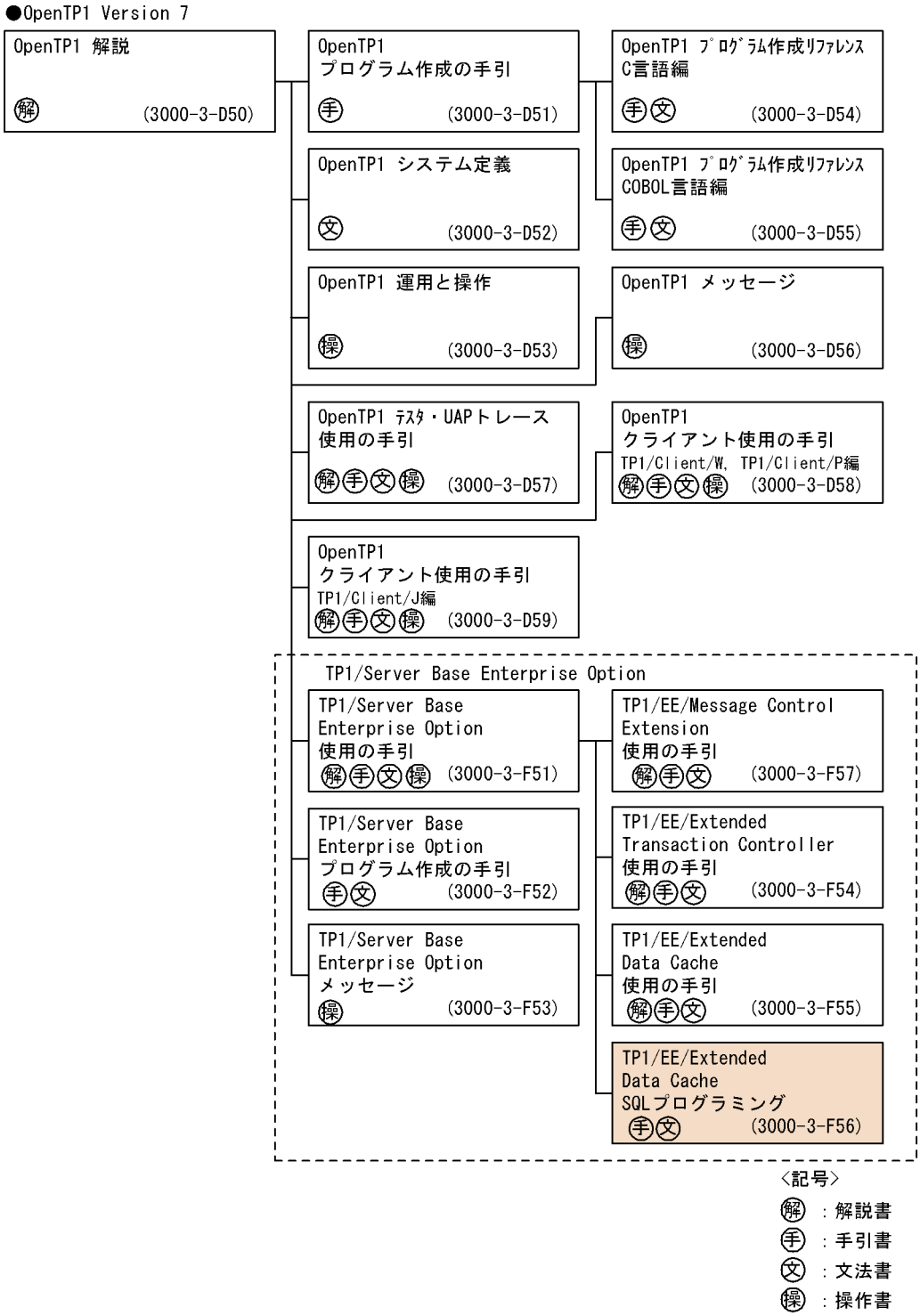
第 7 章　SQL プリプロセサ（eexdbcbl）

SQL プリプロセサ（eexdbcbl）について説明しています。

付録 A　サンプル UAP

サンプル UAP について説明しています。

■関連マニュアル


●OpenTP1 Version 7
OpenTP1 解説
㊎ (3000-3-D50)
OpenTP1
プログラム作成の手引
㊍ (3000-3-D51)
OpenTP1 システム定義
㊏ (3000-3-D52)
OpenTP1 運用と操作
㊐ (3000-3-D53)
OpenTP1 ﾃｽﾀ・UAPトレース
使用の手引
㊎㊍㊏㊐ (3000-3-D57)
OpenTP1
クライアント使用の手引
TP1/Client/J編
㊎㊍㊏㊐ (3000-3-D59)
OpenTP1 ﾌﾟﾛｸﾞﾗﾑ作成ﾘﾌｧﾚﾝｽ
C言語編
㊍㊏ (3000-3-D54)
OpenTP1 ﾌﾟﾛｸﾞﾗﾑ作成ﾘﾌｧﾚﾝｽ
COBOL言語編
㊍㊏ (3000-3-D55)
OpenTP1 メッセージ
㊐ (3000-3-D56)
OpenTP1
クライアント使用の手引
TP1/Client/W, TP1/Client/P編
㊎㊍㊏㊐ (3000-3-D58)
TP1/Server Base Enterprise Option
TP1/Server Base
Enterprise Option
使用の手引
㊎㊍㊏㊐ (3000-3-F51)
TP1/Server Base
Enterprise Option
プログラム作成の手引
㊍㊏ (3000-3-F52)
TP1/Server Base
Enterprise Option
メッセージ
㊐ (3000-3-F53)
TP1/EE/Message Control
Extension
使用の手引
㊎㊍㊏ (3000-3-F57)
TP1/EE/Extended
Transaction Controller
使用の手引
㊎㊍㊏ (3000-3-F54)
TP1/EE/Extended
Data Cache
使用の手引
㊎㊍㊏ (3000-3-F55)
TP1/EE/Extended
Data Cache
SQLプログラミング
㊍㊏ (3000-3-F56)
〈記号〉
㊎ : 解説書
㊍ : 手引書
㊏ : 文法書
㊐ : 操作書

●関連製品

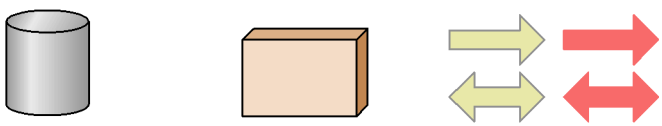
COBOL2002 使用の手引 手引編 ㊎㊍ (3000-3-D42)	COBOL2002 使用の手引 操作編 ㊐ (3000-3-D43)	
HiRDB Version 8 システム導入・設計ガイド （UNIX(R)用） ㊎㊍㊐ (3000-6-352)	HiRDB Version 8 システム定義 （UNIX(R)用） ㊏ (3000-6-353)	HiRDB Version 8 コマンドリファレンス （UNIX(R)用） ㊏ (3000-6-355)
HiRDB Version 8 UAP開発ガイド ㊎㊍ (3020-6-356)	HiRDB Version 8 SQLリファレンス ㊏ (3020-6-357)	HiRDB Version 8 メッセージ ㊐ (3020-6-358)
HiRDB Version 8 XDM/RD E2接続機能 ㊎㊍㊏ (3020-6-365)		
高信頼化システム監視機能 HAモニタ AIX(R)編 ㊎㊍㊐ (3000-9-130)	高信頼化システム監視機能 HAモニタ Linux(R)編 ㊎㊍㊐ (3000-9-132)	高信頼化システム監視機能 HAモニタ メッセージ ㊐ (3000-9-134)

〈記号〉
㊎ ：解説書
㊍ ：手引書
㊏ ：文法書
㊐ ：操作書

■図中で使用する記号

このマニュアルの図中で使用する記号を，次のように定義します。

●ファイル，メモリ　●プログラム　●データの流れ



■文法の記号

(1) 文法記述記号

SQL の指定形式を説明する記号です。

文法記述記号	意味
〔　　〕	この記号で囲まれている項目は省略できることを意味しています。 （例）〔WHERE 探索条件〕 　この場合，WHERE 句は省略できます。
…	この記号で示す直前の項目を繰り返し指定できることを意味しています。 （例）列名〔，列名〕… 　この場合，列名を複数指定できます。
｜	この記号で区切られた項目は選択できることを意味しています。 （例）{ASC ｜ DESC} 　この場合，ASC または DESC のどちらかを指定できます。
{　　}	この記号で囲まれている複数の項目のうち，一つだけ選択できることを意味しています。 （例）{列指定｜値指定｜集合関数｜行 ID} 　この場合，列指定，値指定，集合関数，行 ID のうち，どれか一つを指定できます。
_（下線）	この記号で示す項目は，該当する項目を省略した場合に仮定される値を意味しています。 （例）xdb_memory_fixed = Y ｜ N 　この場合，xdb_memory_fixed オペランドを省略すると，Y が仮定されます。
：：＝	：：＝の左にあるものを右にあるもので定義することを意味しています。 （例）表名：：＝〔スキーマ名 .〕表識別子

(2) 属性表示記号

SQL プリプロセサ（eexdbcbl）のオプションに指定できる値の範囲を説明する記号です。

属性表示記号	意味
～	この記号のあとに指定値の属性を示します。
〈　〉	指定値の構文要素を示します。

(3) 構文要素記号

SQL プリプロセサ（eexdbcbl）のオプションに指定できる文字を説明する記号です。

構文要素記号	意味
〈パス名〉	英字，数字，/，および　.（ピリオド） ただし，パス名は使用する OS に依存します。

■計算式の記号

このマニュアルで使用する計算式の記号の意味を次に示します。

記号	意味
↑↑	計算結果の値を小数点以下で切り上げます。 （例）↑ 34 ÷ 3 ↑ 　この場合，計算結果は 12 となります。

記号	意味
↓↓	計算結果の値を小数点以下で切り捨てます。 （例）↓34 ÷ 3 ↓ 　　この場合，計算結果は 11 となります。
MIN	括弧内の項目のうち，最も小さい値を選びます。 （例）MIN（3，8 ＋ 2，2 × 2） 　　この場合，計算結果は 3 となります。

■このマニュアルでの表記

このマニュアルでは，製品名称を省略して表記しています。製品名称と，このマニュアルでの表記を次に示します。

製品名称	このマニュアルでの表記
Itanium(R) Processor Family	IPF
uCosminexus TP1/EE/Extended Data Cache	XDB
uCosminexus TP1/EE/Extended Transaction Controller	XTC
uCosminexus TP1/EE/Message Control Extension	MCP
uCosminexus TP1/Server Base	TP1/Server Base
uCosminexus TP1/Server Base Enterprise Option	TP1/EE

■略語一覧

このマニュアルで使用する英略語の一覧を次に示します。

英略語	英字での表記
AP	Application Programming
API	Application Programming Interface
CR	Carriage Return
CSV	Comma Separated Value
DB	Database
DBMS	Database Management System
EOF	End of File
ID	Identifier
LRU	Least Recently Used
NL	New Line
OS	Operating System
RPC	Remote Procedure Call
SPP	Service Providing Program
SUP	Service Using Program

英略語	英字での表記
TAR	Tape Archival and Retrieval format
UAP	User Application Program

■常用漢字以外の漢字の使用について

このマニュアルでは，常用漢字を使用することを基本としていますが，次に示す用語については，常用漢字以外の漢字を使用しています。
個所（かしょ）

■ KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

■ XDB のリレーショナルデータベース言語の出典

このマニュアルで記述する XDB のリレーショナル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。

XDB のリレーショナルデータベース

（1）JIS：X3005-1997　データベース言語 SQL
（2）IS：ISO9075-1992　Information processing systems-Database Language SQL

注
JIS：日本工業規格（Japanese Industrial Standard）
IS：国際規格（International Standard）

目次

図目次

表目次

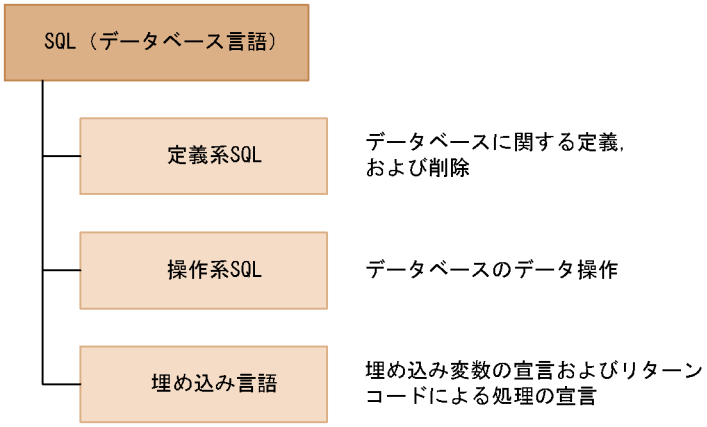
1 基本項目

この章では，SQL を使用する上での基本項目について説明します。

1.1　SQL の一覧

XDB でのデータベースの定義，操作および AP の制御は，SQL と呼ばれるデータベース言語の機能を用いて行います。XDB での SQL の機能体系を次の図に示します。

図 1-1　XDB での SQL の機能体系



機能体系ごとの SQL の分類および機能を次の表に示します。

表 1-1　機能体系ごとの SQL の分類および機能

項番	分類	SQL	機能
1	定義系 SQL	CREATE INDEX（インデクスの定義）	表の列にインデクスを定義します。
		CREATE TABLE（表の定義）	表を定義します。
2	操作系 SQL	CLOSE（カーソルのクローズ）	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。
3		DECLARE CURSOR（カーソルの宣言）	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。
4		DELETE（行の削除）	指定した探索条件を満たす行を表から削除します。
5		FETCH（行の取り出し）	取り出す行を示すカーソルの位置を次の行に進めます。
6		INSERT（行の挿入）	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。
7		OPEN（カーソルのオープン）	カーソルを開きます。 検索結果の先頭の行の直前にカーソルを位置づけて，検索結果を取り出せる状態にします。

項番	分類	SQL	機能
8		SELECT（表の検索）	表のデータを検索します。
9		1 行 SELECT（表の 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。
10		UPDATE（行の更新）	指定した探索条件を満たす行の指定した列の値を更新します。
11	埋め込み言語	BEGIN DECLARE SECTION（埋め込み SQL の開始宣言）	埋め込み SQL 宣言節の始まりを示します。 埋め込み SQL 宣言節には，SQL 中で使用する埋め込み変数および標識変数を指定します。
12		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。
13		END DECLARE SECTION（埋め込み SQL の終了宣言）	埋め込み SQL 宣言節の終わりを示します。
14		END-EXEC（SQL 終了子）	SQL の終わりを示します。
15		EXEC SQL（SQL 先頭子）	SQL の始まりを示します。
16		SQLCODE 変数	SQL の実行後，XDB から返されるリターンコードを受け取ります。
17		WHENEVER（埋め込み例外宣言）	SQL の実行後，XDB 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。

（1） 定義系 SQL の実行可否

定義系 SQL の SQL 文種別と実行可能なインタフェースを次の表に示します。

表 1-2　SQL 文種別と実行可能なインタフェース（定義系 SQL）

項番	SQL 文種別	インタフェース		
		XDB サービス定義	eexdbsql コマンド	UAP での使用
1	CREATE INDEX	○	×	×
2	CREATE TABLE	○	×	×

（凡例）

○：実行できます。

×：実行できません。

(2) 操作系 SQL の実行可否

操作系 SQL の SQL 文種別と実行可能なインタフェースを次の表に示します。

表 1-3 SQL 文種別と実行可能なインタフェース（操作系 SQL）

項番	SQL 文種別	インタフェース		
		XDB サービス定義	eexdbsql コマンド	UAP での使用
1	CLOSE	×	×	○
2	DECLARE CURSOR	×	×	○
3	DELETE	×	○	○
4	FETCH	×	×	○
5	INSERT	×	○	○
6	OPEN	×	×	○
7	SELECT	×	○	×
8	1 行 SELECT	×	×	○
9	UPDATE	×	○	○

（凡例）
○：実行できます。
×：実行できません。

(3) 埋め込み言語の実行可否

埋め込み言語は埋め込み型の UAP で使用できます。

1.2　SQL の記述形式

ここでは，SQL の記述形式について説明します。

1.2.1　オペランドの指定順序

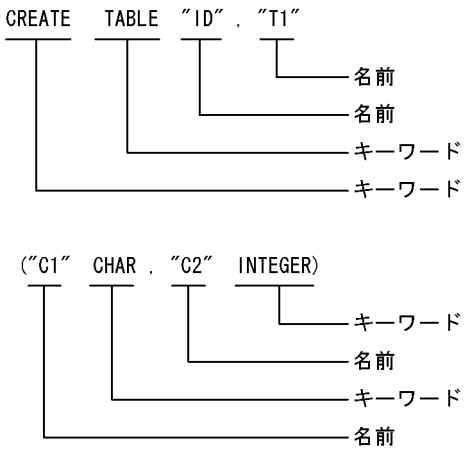
オペランドは，各 SQL の形式で記述している順序に従って指定します。

1.2.2　キーワードの指定

SELECT，UPDATE のように，SQL の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。ユーザは決められた位置以外にキーワードを指定できません。

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の図に示します。予約語については，「1.10　予約語」を参照してください。

図 1-2　キーワードの例



1.2.3　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および制限に従って指定します。詳細については，「1.4(2) 定数の表記法」を参照してください。

1.2.4　分離符号

分離符号は，SQL 中の語と語を分離するために使用します。

(1) 形式

分離符号：：＝空白

空白

空白には，次の文字を指定できます。

- 半角空白
- 全角空白
- CR（Carriage Return）：復帰
- NL（New Line）：改行
- TAB

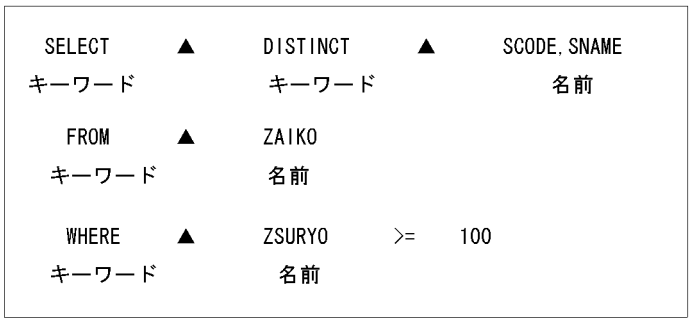
(2) 分離符号を挿入する位置

分離符号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

分離符号の挿入例を次の図に示します。

図 1-3 分離符号の挿入例



（凡例）
▲：分離符合

(3) 分離符号を挿入してはいけない位置

分離符号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
- 二重引用符（"）で囲まれていない名前の中
- 名前を囲む初めの二重引用符（"）の後ろ

- 名前を囲む終わりの二重引用符（"）の前
- 数定数の中
- 演算子の中（2 文字から成る比較演算子の中）

分離符号を挿入してはいけない位置の例を次の図に示します。

図 1-4　分離符号を挿入してはいけない位置の例

（凡例）
▲：分離符合

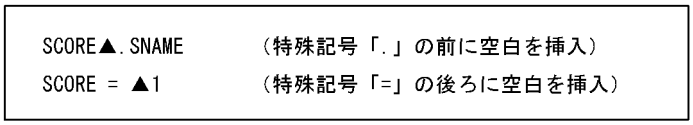
（4）分離符号を挿入してもよい位置

分離符号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「(3) 分離符号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
「,」「.」「-」「+」「*」「'」「"」「(」「)」「<」「>」「=」「^」「!」「TAB」「NL」「CR」「半角空白」「全角空白」

分離符号を挿入してもよい位置の例を次の図に示します。

図 1-5　分離符号を挿入してもよい位置の例



（凡例）
▲：分離符合

1.2.5　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。

なお，SQL で使用できる文字は，JISX0201 または JISX0208 になります。また，SQL はシフト JIS で記述してください。

表 1-4　SQL で使用できる文字

項番	種別	SQL で使用できる文字
1	文字列定数	半角文字コード（X'00' を除く）および全角文字コードのすべての文字

項番	種別	SQL で使用できる文字
2	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 下線（_） カタカナ • 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（.） ハイフンまたは負符号（-） 正符号（+） アスタリスク（*） アポストロフィ（'） 二重引用符（"） 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（!） コロン（:） TAB NL CR

1.2.6 SQL の最大長

一つの SQL 文は，300000 バイトまで記述できます。

1.2.7 名前の指定

名前の指定には，二重引用符（"）で囲んで指定する方法と，二重引用符（"）で囲まないで指定する方法があります。

名前を指定する場合，二重引用符（"）で囲んで指定することを推奨します。なお，名前を二重引用符（"）で囲んだ場合，半角英小文字および半角英大文字は区別して扱われます。

参考

名前には，予約語と同じ名前を指定できませんが，二重引用符（"）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が追加される場合があります。名前をあらかじめ二重引用符（"）で囲んで指定しておくと，追加された予約語と既存の名前が重複するという問題を回避できます。

ただし，次に示す名前を UAP で使用する場合は，SQL の予約語と同じでも二重引用符（"）で囲まないで指定してください。

- カーソル名
- 埋め込み変数名，標識変数名，ホスト識別子

また，名前は識別子として指定します。

識別子には，二重引用符（"）で囲んで指定した区切り識別子と，二重引用符（"）で囲まないで指定した通常識別子があります。

（1） 形式

```
識別子：：＝｛通常識別子｜区切り識別子｝
  区切り識別子：：＝”区切り識別子本体”
```

（2） 名前を指定するときの規則

- 通常識別子および区切り識別子本体に指定できる文字の種類と長さは，名前に使用できる文字と長さ（バイト数）の制限に従います。詳細については，「(3) 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。
- 名前の先頭の文字は，半角英大文字，半角英小文字もしくは半角カタカナ大文字（ア～ン，ヲ）または全角文字にしてください。
- 全角空白を含むことはできません。

(a) 名前を通常識別子として指定する場合の規則

- 半角英小文字は，半角英大文字として扱います。
- 次の文字を含む場合は，通常識別子として指定できません。区切り識別子として指定してください。
  - 半角空白
  - 半角ハイフン（-）
- SQL の予約語と同じ名前は，通常識別子として指定できません。区切り識別子として指定してください。

(b) 名前を区切り識別子として指定する場合の規則

- 半角英小文字および半角英大文字を区別して扱います。
- 名前の最後の文字に半角空白は指定できません。

（3） 名前に使用できる文字と制限

名前に使用できる文字と長さの制限を次の表に示します。

表 1-5 名前に使用できる文字と長さの制限

項番	名前の種類	長さの制限（バイト）	半角文字の使用の可否						全角文字の使用の可否
			英大文字，数字	英小文字	カタカナ	下線（_）	空白	ハイフン（-）	
1	インデクス識別子	100	○	○	○	○	○	○	○
2	相関名	100	○	○	○	○	○	○	○
3	認可識別子	30	○	○	×	×	×	×	×
4	表識別子	100	○	○	○	○	○	○	○
5	列名	100	○	○	○	○	○	○	○
6	カーソル名	30	○	○	○	○	×	○	○
7	埋め込み変数名※1	30	○	○	○	○	×	○	○
8	標識変数名※1	30	○	○	○	○	×	○	○
9	DB エリア名	30	○※2	○	×	○	×	○	×

（凡例）

○：使用できます。

×：使用できません。

注※ 1

ホスト言語によって制限されます。

COBOL2002 の場合，埋め込み変数名および標識変数名には，全角文字とほかの 1 バイト文字を混在させて使用できます。また，埋め込み変数と標識変数の長さは，1 バイト文字，全角文字に関係なく，31 文字までです。

注※ 2

「¥」「#」「@」は使用できません。

（4） 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は，予約語と重複する名前を二重引用符（"）で囲んでください。

なお，名前を二重引用符（"）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。

1.2.8 名前の修飾

ピリオド（.）によってスキーマ名，表識別子などを連結することを名前の修飾といいます。名前の修飾は，スキーマ名を明示したり，名前を一意にしたりするときに使用します。

(1) スキーマ名，表名，インデクス名

(a) スキーマ名

スキーマを所有する認可識別子を指定します。

指定できる認可識別子を次に示します。

- XDBUSER
- MASTER

形式

```
スキーマ名：：＝認可識別子
```

(b) 表名

スキーマ名で修飾された表識別子を表名といいます。

形式

```
表名：：＝〔スキーマ名.〕表識別子
```

(c) インデクス名

スキーマ名で修飾されたインデクス識別子をインデクス名といいます。

形式

```
インデクス名：：＝〔スキーマ名.〕インデクス識別子
```

(d) 指定内容と規則

指定項目ごとの指定内容と規則を次の表に示します。

表 1-6 指定項目ごとの指定内容と規則

項番	指定項目	指定内容	規則
1	スキーマ名	ディクショナリ表を検索する場合は，"MASTER" を指定します。それ以外の場合は，"XDBUSER" を指定します。	スキーマ名を省略すると，スキーマ名には，"XDBUSER" が仮定されます。
2	表識別子	表の名前を指定します。	
3	インデクス識別子	インデクスの名前を指定します。	

(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，指定する列または * がどの表に対応するかを一意にするための修飾子です。表指定には表名または相関名を指定します。

形式

表指定：：＝ {表名｜相関名}

表指定での修飾例を次に示します。

(例)

複数の表（EMP，DEPT）に存在する同じ名前の列（DNO）を参照するために，表名で修飾する場合

```
SELECT "ENO","ENAME","EMP","DNO","DNAME" FROM "EMP","DEPT"
  WHERE "EMP"."DNO"="DEPT"."DNO"
```

(3) 列指定

表指定で修飾された列名を列指定といいます。

形式

列指定：：＝〔表指定.〕列名

列名を表指定で修飾する場合，指定する表名の有効範囲の中でなければ，列名をその表名で修飾できません。表名の有効範囲については，「(4) 範囲変数」を参照してください。

列名は，その列名を指定する位置に有効範囲を持つ表または導出される表に存在しなければなりません。i 番目の選択式によって導出される列名が，i 番目に導出される表の列名になります。

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。各形式中に「列指定」と記述

している個所は，列名が修飾できることを示し，「列名」と記述している個所は，修飾できないことを示しています。

なお，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）をする場合に，検索対象の複数の表に同じ列名があるときは，列名を表指定で修飾しなければなりません。修飾しないと，指定した列がどの表の列なのかわかりません。

(4) 範囲変数

列指定の修飾子となる可能性のある識別子を範囲変数といいます。範囲変数は，名称と有効範囲を持ちます。相関名が指定された場合，相関名の指定された表名は有効範囲がなくなります。

(a) 範囲変数となる識別子の種類

範囲変数となる識別子の種類を次の表に示します。

表 1-7 範囲変数となる識別子の種類

項番	範囲変数となる識別子の種類		扱い
1	相関名		○
2	表名	相関名の指定がない場合	○
		相関名の指定がある場合	×
3	UPDATE の更新対象表または DELETE の削除対象表の表名		○

（凡例）
○：範囲変数となります。
×：範囲変数となりません。

(b) 範囲変数の名称の例

範囲変数の名称の例を次の表に示します。

表 1-8 範囲変数の名称の例

項番	SQL 例	範囲変数の名称
1	SELECT T1.C1,X.C1,Y.C1 FROM	―
2	T1	A.T1
3	,A.T1 X	X
4	,A.T2 Y	Y

（凡例）
―：該当しません。
A：スキーマ名
T1，T2：表識別子
X，Y：相関名

C1：列名

(c) 範囲変数の有効範囲の例

範囲変数の有効範囲の例を次の表に示します。

表 1-9 範囲変数の有効範囲の例

項番	SQL 例	範囲変数		
		A.T1	X	A.T2
1	SELECT X.C1,T2.C2	×	○	○
2	FROM A.T1 X,	×	○	○
3	A.T2	×	○	○
4	WHERE X.C1=100	×	○	○

（凡例）

○：有効範囲となります。

×：有効範囲となりません。

A：スキーマ名

T1，T2：表識別子

X：相関名

C1，C2：列名

1.3　データ型

ここでは，XDB で使用できるデータ型について説明します。

1.3.1　データ型の一覧

XDB で使用できるデータ型を次の表に示します。

表 1-10　XDB で使用できるデータ型

項番	分類	データ型	データ形式
1	数データ	INTEGER	整数（4 バイト）
2		SMALLINT ※	整数（2 バイト）
3		DECIMAL	固定小数点数
4	文字データ	CHARACTER	固定長文字列
5		VARCHAR ※	可変長文字列
6	時刻印データ	TIMESTAMP	年，月，日，時，分，秒の六つの属性を持つ時刻印
7	行 ID データ	ROWID	行 ID（12 バイト）

注※

ディクショナリ表だけで使用できます。

(1) 数データ

(a) INTEGER 型（整数）

- 値の範囲が -2147483648 ～ 2147483647 の整数を扱うデータ型です。INT または INTEGER と記述します。
- データは，4 バイトの 2 進形式です。
- 定数は，100，200 のように表現します。定数表現については，「1.4　定数」を参照してください。

(b) SMALLINT 型（整数）

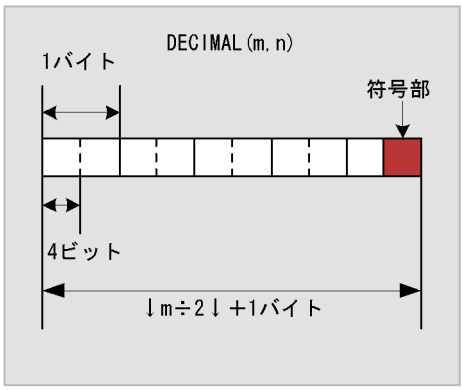
- 列のデータ型としては指定できません。
- 値の範囲が -32768 ～ 32767 の整数を扱うデータ型です。
- データは，2 バイトの 2 進形式です。
- 定数は，100，200 のように表現します。定数表現については，「1.4　定数」を参照してください。

(c) DECIMAL 型（固定小数点数）

- 固定小数点数を扱うデータ型です。{DEC ｜ DECIMAL}〔(m〔,n〕)〕と記述します。

  - 精度（全体のけた数）を m，位取り（小数秒のけた数）を n で指定します。
  - m，n は正の整数で，1 ≦ m ≦ 29，0 ≦ n ≦ 29，n ≦ m です。
  - m を省略すると，29 が仮定されます。
  - n を省略すると，0 が仮定されます。
- データは，↓ m ÷ 2 ↓＋ 1 バイトのパック 10 進形式です。
- データは，次の図に示すように 4 ビットで 1 けたの数値を表し，末尾の 4 ビットに符号を持ちます。

図 1-6　DECIMAL 型のデータ形式



- 符号部の値が 0xC の場合，正の値として扱い，符号部の値が 0xD の場合，負の値として扱います。ただし，符号部の値が前述以外の場合は，不正なデータと見なされます。また，数の値が 0 の場合，符号部の値は，0xC でなければなりません。
- 精度が偶数の場合，データの先頭 4 ビットは，0x0 でなければなりません。
- 定数は，123.4，12.345 のように表現します。定数表現については，「1.4　定数」を参照してください。

(2) 文字データ

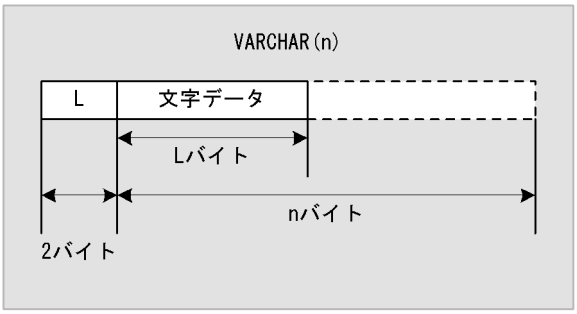
(a) CHARACTER 型（固定長文字列）

- 固定長文字列を扱うデータ型です。{CHAR ｜ CHARACTER}〔(n)〕と記述します。
  - 文字列の長さ（バイト数）を n で指定します。
  - n は，1 ～ 32000 の整数でなければなりません。
  - n を省略すると，1 が仮定されます。
- 入力するデータの長さ（バイト数）は，n と等しくなければなりません。
- 定数は，'char'，'Ｃｈａｒ'，'charＣｈａｒ' のように表現します。定数表現については，「1.4　定数」を参照してください。
- 半角文字，全角文字の両方とも扱うことができます。
- 文字データの比較を行う場合，文字コードの大小が比較するデータの大小となります。

(b) VARCHAR 型（可変長文字列）

- 列のデータ型としては指定できません。
- 可変長文字列を扱うデータ型です。
  - 文字列の最大の長さ（バイト数）を n で指定します。
  - n は省略できません。
- データは，次の図に示すように文字列の長さを 2 バイトで表します。

図 1-7　VARCHAR 型のデータ形式



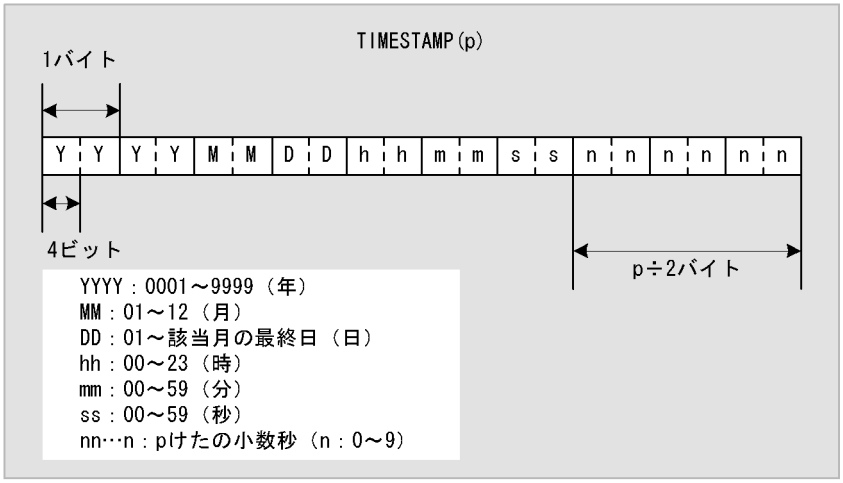
- 半角文字，全角文字の両方とも扱うことができます。また，文字列の長さが 0 バイトのデータを扱うこともできます。
- 文字データの比較を行う場合，文字コードの大小が比較するデータの大小となります。

(3) 時刻印データ

(a) TIMESTAMP 型（時刻印）

- 年，月，日，時，分，秒の六つの属性を持つ時刻印のデータ型で，TIMESTAMP〔(p)〕と記述します。
  - 秒数の小数点以下のけたを p で指定します。
  - p は，0，2，4 または 6 でなくてはなりません。
  - p を省略すると，0 が仮定されます。
- 入力するデータの長さ（バイト数）は，7 ＋ p ÷ 2 と等しくなくてはなりません。

図 1-8 TIMESTAMP 型のデータ形式



- 定数は，TIMESTAMP'2008-07-08 11:03:58.12'，TIMESTAMP'2008-07-08 11:03:58.1234'，TIMESTAMP'2008-07-08 11:03:58.123456' のように表現します。定数表現については，「1.4 定数」を参照してください。

(4) 行 ID データ

(a) ROWID 型

- 列のデータ型としては指定できません。
- 表に格納された行を識別する値を扱うデータ型です。
- このデータ型を定数として表現することはできません。必ず，特殊レジスタの ROWID を指定して取得した値を使用してください。特殊レジスタの ROWID を指定して取得した値以外を指定した場合の動作は保証されません。特殊レジスタの ROWID については，「1.5 特殊レジスタ」を参照してください。

1.3.2 比較，代入できるデータ型

(1) 比較

WHERE 句に指定した探索条件中で比較できるデータ型の組み合わせを次の表に示します。WHERE 句については，「2.5 WHERE 句」を参照してください。

ここでの比較は，ユニークインデクスの重複キーチェック，および ORDER BY 句にも適用されます。

表 1-11 比較できるデータ型の組み合わせ

項番	データ型	数データ	文字データ	時刻印データ	行 ID データ
1	数データ	○	×	×	×
2	文字データ	×	○	×	×

項番	データ型	数データ	文字データ	時刻印データ	行 ID データ
3	時刻印データ	×	×	○	×
4	行 ID データ	×	×	×	○

（凡例）
○：比較できます。
×：比較できません。

(a) 数データの比較

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されます。範囲の広さの順を次に示します。

DECIMAL ＞ INTEGER ＞ SMALLINT

(b) 文字データの比較

- 比較する文字データの長さが異なる場合は，短い方のデータの末尾に半角空白を補い，長さをそろえてから比較されます。
- VARCHAR 型同士の比較の場合でも，半角空白を補って長さをそろえてから比較されます。
- 異なる値に空白を補って比較した結果，同じ値として扱われるデータがある場合，次のように動作するため注意が必要です。
  - ユニークインデクスを定義している場合，重複キーとして扱われます。
  - ORDER BY 句を指定した検索を行う場合，同じ値として扱われます。

(c) 時刻印データの比較

小数秒のけた数が異なる場合は，精度の高い方にそろえて，拡張した小数秒部分に 0 を補って比較されます。

(d) 行 ID データの比較

行 ID データは，WHERE 句中の比較述語（=）にだけ指定できます。

(2) 格納代入

INSERT の挿入値または UPDATE の更新値として，指定できるデータ型の組み合わせを次の表に示します。ただし，埋め込み変数，および標識変数を使用して格納代入を行う場合には，代入先と代入元のデータ型をそろえてください。埋め込み変数，および標識変数については，「1.6　変数（埋め込み変数および標識変数）」を参照してください。

表 1-12　格納代入できるデータ型の組み合わせ

項番	代入元のデータ型	代入先のデータ型			
		数データ	文字データ	時刻印データ	行 ID データ
1	数データ	○	×	×	×

項番	代入元のデータ型	代入先のデータ型			
		数データ	文字データ	時刻印データ	行 ID データ
2	文字データ	×	○	×	×
3	時刻印データ	×	×	○	×
4	行 ID データ	×	×	×	×

（凡例）

○：格納代入できます。

×：格納代入できません。

(a) 数データの格納代入

- 代入元が代入先で扱える値の範囲を超える場合，代入できません。
- 代入先が INTEGER 型で代入元が DECIMAL 型の場合，端数（小数）部分は切り捨てられます。
- 代入元，代入先ともに DECIMAL 型のデータの場合，代入先の位取りより下位のけた部分は切り捨てられます。代入先の位取りより代入元の位取りが小さい場合，小数部分に 0 を補って格納されます。

(b) 文字データの格納代入

- 代入元のデータ長が代入先のデータ長よりも長い場合，代入できません。
- 代入元のデータ長が代入先のデータ長より短い場合は，最後に半角空白を補って格納されます。

(c) 時刻印データの格納代入

- 代入元の秒数の小数秒のけた数より代入先の秒数の小数秒のけた数が少ない場合，代入できない部分は切り捨てられます。
- 代入元の秒数の小数秒のけた数より代入先の秒数の小数秒のけた数が多い場合，0 を補って格納されます。

(d) 行 ID データの格納代入

INSERT の挿入値または UPDATE の更新値として，行 ID データは使用できません。

(3) 検索代入

検索結果を受け取る場合，代入先と代入元のデータ型をそろえてください。

1.3.3 文字データ使用上の注意事項

文字データを使用する場合，文字データの定義長によっては制限のある項目があります。文字データ使用上の制限を次の表に示します。

表 1-13 文字データ使用上の制限

項番	定義長	項目	
		集合関数	ソート
1	1 ～ 4036	○	○
2	4037 以上	×	×

（凡例）

○：使用できます。

×：使用できません。

1.3.4 DECIMAL 型使用上の注意事項

DECIMAL 型では，右端のバイトの右 4 ビットで演算符号が表現されます。ビット列と符号の関係を次の表に示します。

表 1-14 ビット列と符号の関係

項番	ビット列の値	ビット列の値の 16 進表示	符号との関係
1	1100	C	正の符号と見なされます。
2	1101	D	負の符号と見なされます。
3	上記以外	－	符号とは見なされません。不正なデータと見なされます。

（凡例）

－：該当しません。

注

数の値が 0 の場合，符号部の値は，0xC でなければなりません。

1.3.5 SQL のデータ型と COBOL 言語のデータ記述の対応

SQL のデータ型と COBOL 言語のデータ記述の対応を次の表に示します。なお，表中のデータ記述は，次のように表記することもできます。

PICTURE：PIC

COMPUTATIONAL：COMP

COMPUTATIONAL-n：COMP-n

9(n)：99…9

X(n)：XX…X

データの受け渡しには，データ型およびデータ長（精度および位取りも含む）が一致する埋め込み変数を使用してください。

表 1-15　SQL のデータ型と COBOL 言語のデータ記述の対応

項番	SQL のデータ型	COBOL 言語のデータ記述	項目の記述	備考
1	SMALLINT	L1 基本項目名 PICTURE S9(4) COMPUTATIONAL.	基本項目または独立項目	－
2	INTEGER	L1 基本項目名 PICTURE S9(9) COMPUTATIONAL.	基本項目または独立項目	－
3	DECIMAL〔(p〔,s〕)〕	L1 基本項目名 PICTURE S9(p-s)[V9(s)] COMPUTATIONAL-3.	基本項目または独立項目	1 ≦ p ≦ 29 ※5，0 ≦ s ≦ p，p=s の場合，SV9(s) とします。s=0 の場合，[V9(s)] を省略します。
4	CHARACTER〔(n)〕	L1 基本項目名 PICTURE X(n). ※1	基本項目または独立項目	1 ≦ n ≦ 32000
5	VARCHAR(n)	L2 集団項目名 . L3 基本項目名 1 PICTURE S9(4) COMPUTATIONAL. L3 基本項目名 2 PICTURE X(n). ※1	二つの基本項目から構成される集団項目 基本項目名 1：文字列長 基本項目名 2：文字列	1 ≦ n ≦ 32000
6	TIMESTAMP〔(p)〕	L1 基本項目名 PICTURE X(n). ※2 ※3	基本項目または独立項目	• p=0 の場合：n=19 • p=2 の場合：n=22 • p=4 の場合：n=24 • p=6 の場合：n=26
7	ROW	この表中のデータ項目と集団項目の組み合わせ※4	複数の基本項目から構成される集団項目	－
		L1 基本項目名 PICTURE X(n). ※1	基本項目または独立項目	n は行長

項番	SQL のデータ型	COBOL 言語のデータ記述	項目の記述	備考
8	ROWID	L1 基本項目名 PICTURE X(12).	基本項目または 独立項目	－
9	標識変数	L1 基本項目名 PICTURE S9(4) COMPUTATIONAL.	基本項目または 独立項目	－

（凡例）

－：該当しません。

L1：レベル番号 01 ～ 49，または 77

n：長さ（バイト）

p：精度（全体のけた数）

s：位取り（小数秒のけた数）

注※ 1

X の代わりに 9 を使用して定義できます。9 を使用して定義した場合，数字以外の文字を含む文字列の代入や，検索結果として受け取ったときの動作は COBOL 言語のコンパイラの実装に依存します。

注※ 2

X の代わりに 9 を使用して定義した場合，プリプロセス時にエラーとなりませんが，使用しないでください。

注※ 3

定数表現（TIMESTAMP'YYYY-MM-DD hh:mm:ss〔.nn…n〕'）の YYYY-MM-DD hh:mm:ss〔.nn…n〕の部分を格納します。

注※ 4

行単位（ROW 指定）の検索，または更新をする場合，ROW に対する埋め込み変数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。

注※ 5

使用できる精度の上限は，COBOL 言語のコンパイラの仕様に依存します。

1.4 定数

ここでは，定数について説明します。

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や時刻印を表す一般定数があります。

(1) 形式

```
定数：：＝ {数定数｜一般定数}
数定数：：＝ {真数定数}
真数定数：：＝ {整数定数｜10進数定数}
一般定数：：＝ {文字列定数｜時刻印定数}
```

(2) 定数の表記法

定数の表記法を次の表に示します。

表 1-16 定数の表記法

項番	定数	表記法		XDB で解釈するデータ型
1	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，＋または－で表します。	INTEGER
2	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：表記したけた数
3	文字列定数	'文字列' ＜例＞ 'HITACHI' '88' '''95.7.30'	文字列は半角文字，または全角文字の列で表します。 文字列にアポストロフィ(')を記述する場合，1 個のアポストロフィを表すために 2 個続けて記述してください。	CHAR(n) n：表記した文字列長

項番	定数	表記法		XDB で解釈するデータ型
4	時刻印定数	TIMESTAMP'YYYY-MM-DD hh:mm:ss.nn…n' ＜例＞ TIMESTAMP'2008-07-08 11:03:58' TIMESTAMP'2008-07-08 11:03:58.12' TIMESTAMP'2008-07-08 11:03:58.1234' TIMESTAMP'2008-07-08 11:03:58.123456'	年（YYYY），月（MM），日（DD）をハイフン（-）で結び，空白を含めてから，さらに時（hh），分（mm），秒（ss）をコロン（:）で結び「'YYYY-MM-DD hh:mm:ss'」として表現します。 年（YYYY），月（MM），日（DD）が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）が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。	TIMESTAMP〔(p)〕

数定数の使用上の制限を次の表に示します。

表 1-17　数定数の使用上の制限

項番	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）
1	整数定数※	-2147483648 ～ 2147483647	10 けた
2	10 進数定数	-（10^{30} − 1）～ -10^{-29}，0，および，10^{-29} ～（10^{30} − 1）	29 けた

注※

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，XDB 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。

1.5 特殊レジスタ

SQL 中で参照できる特別な値のことを特殊レジスタといいます。特殊レジスタは，SQL 中で値指定として指定できます。

ここでは，特殊レジスタである CURRENT_TIMESTAMP 値関数および行 ID について説明します。

1.5.1 CURRENT_TIMESTAMP 値関数

(1) 機能

CURRENT_TIMESTAMP 値関数は，現在の時刻印を表します。

(2) 形式

```
CURRENT_TIMESTAMP値関数：：＝　CURRENT_TIMESTAMP〔(p)〕
```

(3) 共通規則

- p は，0，2，4 または 6 でなくてはなりません。
- 小数秒精度 p（p=0，2，4 または 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のデータを返します。
- 小数秒精度 p を指定しない場合，p=0 が仮定されます。
- 結果のデータ型は，TIMESTAMP 型となります。
- 一つの SQL 文中で CURRENT_TIMESTAMP 値関数を複数回指定した場合，それらはすべて同じ値となります。

1.5.2 行 ID

(1) 機能

行 ID は，表に格納された行を識別する値を表します。

探索条件に行 ID を指定すると，ROWID 型のデータから行の格納位置を特定して，目的の行にアクセスできます。

(2) 形式

```
行ID：：＝　ROWID
```

(3) 共通規則

- 行 ID は，選択式および WHERE 句にだけ指定できます。

- WHERE 句に行 ID を指定する場合，比較述語（=）にだけ指定できます。
- 結果のデータ型は，ROWID 型となります。
- 選択式中に行 ID を指定した場合，その問合せ指定に対する次の項目と同時に指定できません。
  - 集合関数
  - WHERE 句に対する行 ID の指定
- SQL 文中に行 ID を指定した場合，FROM 句に二つ以上の表を指定できません。
- WHERE 句に行 ID を指定した場合，ORDER BY 句および LIMIT 句を指定できません。
- ソートキーになる項目に行 ID は指定できません。
- 行 ID が不変であることが保証できるのは，同一トランザクション中だけです。
- トランザクションをわたって行 ID を使用する場合は，再度行 ID を取得してください。行の削除・追加によって，行 ID が変更されている可能性があります。

1.6 変数（埋め込み変数および標識変数）

ここでは，SQL に値を渡す変数（埋め込み変数および標識変数）について説明します。

1.6.1 埋め込み変数および標識変数

(1) 機能

UAP の SQL 中に埋め込み変数および標識変数を指定できます。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。

(2) 形式

```
：埋め込み変数〔：標識変数〕
```

(3) 規則

1. SQL の文字列中に埋め込み変数および標識変数を指定した場合，そのデータ型およびデータ長は，埋め込み変数および標識変数を指定した個所によって，システムが仮定します。
   埋め込み変数および標識変数を指定した個所，ならびに仮定されるデータ型およびデータ長を次の表に示します。

表 1-18 埋め込み変数および標識変数を指定した個所，ならびに仮定されるデータ型およびデータ長

項番	埋め込み変数および標識変数を指定した個所	仮定されるデータ型およびデータ長
1	述語中に単独で指定した場合	比較相手となる値式の結果のデータ型およびデータ長
2	挿入値，更新値に単独で指定した場合	格納代入相手の列のデータ型およびデータ長
3	そのほかの個所に指定した場合	各項目の説明を参照してください。

2. SQL の中に指定する埋め込み変数の数は，3000 以下にしてください。
3. 埋め込み変数および標識変数に値を渡す場合，システムが仮定したデータ型およびデータ長の値を指定してください。

(4) 用途および指定個所

埋め込み変数および標識変数の機能，ならびにその用途および指定個所を次の表に示します。

表 1-19　埋め込み変数および標識変数の機能，ならびにその用途および指定個所

項番	機能	用途		指定個所
		埋め込み変数	標識変数	
1	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。※	ナル値も含む列の値を受け取るために，埋め込み変数とともに使用します。埋め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。	1 行 SELECT 文，FETCH 文の INTO 句
2	定数値の変更	ナル値以外の値を受け取ります。※	ナル値を指定するために，埋め込み変数とともに使用します。SQL に渡す埋め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。
3	埋め込み変数の変更	カーソル宣言中で指定した SELECT 文中の埋め込み変数の代わりに，ほかの埋め込み変数を指定します。	－	OPEN 文の USING 句

（凡例）

－：該当しません。

注※

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，「1.6.3　標識変数の値の設定」を参照してください。

（5）埋め込み変数および標識変数のデータ型と COBOL 言語のデータ記述の関係

埋め込み変数および標識変数を UAP 中に記述するときのデータ型とデータ記述の関係については，「1.3.5　SQL のデータ型と COBOL 言語のデータ記述の対応」を参照してください。

（6）埋め込み変数および標識変数の修飾指定（COBOL 言語）

埋め込み変数およ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，埋め込み変数また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となります。

1.6.2 埋め込み変数および標式変数を指定できる個所

埋め込み変数および標識変数を指定できる個所を次の表に示します。

表 1-20 埋め込み変数および標識変数を指定できる個所

項番	SQL 文	指定できる個所	埋め込み変数	標識変数
1	DECLARE CURSOR	探索条件中で定数が指定できる個所※	○	○
2		LIMIT 句のリミット行数	○	×
3	1 行 SELECT	探索条件中で定数が指定できる個所※	○	○
4		INTO 句	○	○
5		LIMIT 句のリミット行数	○	×
6	INSERT	挿入値	○	○
7	UPDATE	更新値	○	○
8		探索条件中で定数が指定できる個所※	○	○
9	DELETE	探索条件中で定数が指定できる個所※	○	○
10	OPEN	USING 句	○	×
11	FETCH	INTO 句	○	○

（凡例）

○：指定できます。

×：指定できません。

注※

ただし，次の個所には指定できません。

- 比較述語の両辺
- BETWEEN 述語の左側

1.6.3 標識変数の値の設定

（1）データを受け取る場合（FETCH 文または 1 行 SELECT 文の INTO 句）

FETCH 文または 1 行 SELECT 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。埋め込み変数にナル値が返された場合，埋め込み変数の値は保証されません。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 1-21　FETCH 文または 1 行 SELECT 文で返される標識変数の値

項番	標識変数の値	対応する埋め込み変数が受け取る値
1	負の値	ナル値です。
2	0	ナル値ではない値です。

（2）データを渡す場合（FETCH 文および 1 行 SELECT 文の INTO 句以外）

FETCH 文および 1 行 SELECT 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め込み変数の値を使用してください。

表 1-22　SQL 実行前に設定する標識変数の値

項番	標識変数の値	対応する埋め込み変数が SQL に渡す値
1	-1	ナル値です。埋め込み変数が持っている値は無視されます。
2	0	ナル値ではない値です。
3	上記以外の値	SQL 実行時にエラーとなります。

1.7 ナル値

ここでは，ナル値の扱いについて説明します。

ナル値とは，値がないこと，また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，または設定されていなければ，ナル値が設定されます。

ディクショナリ表にナル値を持つ列が存在します。

(1) 検索結果の列の値の受け取り

ナル値の識別には標識変数を使用します。詳細については，「1.6.3　標識変数の値の設定」を参照してください。

(2) 比較

述語に指定した値式や列の値がナル値の場合，その条件で検索した行に対する述語の評価結果は，真でも偽でもなく不定になります。

(3) ソート

昇順の場合はナル値を最後に出力します。降順の場合はナル値を最初に出力します。

(4) 重複排除

ナル値同士は，重複するものとして扱います。

(5) 集合関数

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。

1.8 SQL のエラーの判定と対処

UAP で SQL 文を実行する場合，SQL 文が正常に実行されたかを判定する必要があります。

ここでは，UAP を作成するときに考慮する必要がある基本的な内容として，SQL 文が正常に実行されたかを判定する方法と，エラーを検知した場合の対処方法について説明します。

なお，SQL エラーに伴うロールバック処理については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」の「SQL エラーに伴うロールバック」を参照してください。

1.8.1 エラーの判定と対処

(1) リターンコードの参照

SQL 実行時に XDB でリターンコード（SQLCODE）が設定されます。ただし，DECLARE CURSOR のような宣言文の場合は，リターンコードが設定されません。リターンコードを参照する場合の変数名称を次に示します。

- SQLCODE を参照する場合の変数名称：SQLCODE

SQLCODE 変数の設定値を参照することで，SQL 文の実行状態が判定できます。

SQL 文の実行状態と変数に設定される値の関係を次の表に示します。

表 1-23 SQL 文の実行状態と変数に設定される値の関係

項番	SQL 文の実行状態		SQLCODE 変数の値	SQLWARN0 の値	SQLWARN6 の値
1	正常終了	警告なし	0	'△'	'△'
2		警告あり※	0	'W'	'△'
3	データなし		100	'△'	'△'
4	エラー終了	中断状態なし	0 より小さい	'△' または 'W' ※	'△'
5		中断状態発生	0 より小さい	'W'	'W'

（凡例）

△：空白を示します。

注※

警告情報は SQLWARN1 ～ F に設定されます。警告情報が設定された場合は，SQLWARN0 に 'W' が設定されます。

SQLWARN0 に 'W' が設定されている場合，SQLWARN1 ～ F の領域を確認します。

なお，SQLWARN0 ～ F の内容については，「1.9　SQL 連絡領域」を参照してください。

(2) リターンコードの判定例

リターンコードの判定例を次に示します。

(a) SQLCODE=100 の場合

検索する行がなくなったと判定します。

特に次に示す内容を判定するときに有効です。

- FETCH 文で取り出す行がなくなった
- 1 行 SELECT 文で該当する行がなかった
- DELETE 文，または UPDATE 文で削除または更新対象の行がなかった

(b) SQLCODE<0 の場合

SQL エラーが発生したと判定します。

SQL エラーが発生した場合，中断状態になる場合とならない場合があります。SQLWARN6='W' の場合，暗黙的にデータベースへの更新が無効にされて中断状態になったと判定します。

(c) 上記の (a) および (b) 以外の場合

正常終了したと判定します。

正常終了には，警告情報がある場合とない場合があります。SQLWARN0='W' の場合，警告付き正常終了と判定します。

(3) エラー検出時の対処

エラーを検知した場合，次に示す順番で対処します。

1. リターンコードを出力，または表示します。
2. リターンコードだけではエラーの内容が判別しにくい場合，各コードの付加情報を表示，または出力します。また，必要に応じて，エラーになった SQL 文，またはエラーになった SQL 文を識別するための情報を表示します。
   リターンコードの付加情報と参照先を次の表に示します。

表 1-24　リターンコードの付加情報と参照先

項番	付加情報	参照先
1	SQLCODE に対応するメッセージ	SQL 連絡領域中の SQLERRML 領域，および SQLERRMC 領域の内容

3. トランザクションを取り消します（ロールバック，または UAP を異常終了させま

す)。中断状態を発生させた UAP からはロールバック以外は受け付けられません。
4. UAP の終了，またはトランザクションの開始（別のトランザクションの新規実行，または同じトランザクションの再実行）をします。
なお，同じトランザクションを再実行する場合，実行前にエラーの対策をしてください。エラーの原因が取り除かれていない状態でトランザクションを再実行すると，無限ループになるおそれがあります。また，トランザクションを再実行しても同じエラーが発生する場合は，UAP の終了を検討する必要があります。

1.8.2 エラーの自動判定と対処

WHENEVER 文を使用すると，エラーが発生したかどうかを自動的に判定できます。WHENEVER 文では，次に示す内容について判定できます。

- エラーが発生したかどうか
- 検索する行がなくなったかどうか
- 正常終了時の警告情報があるかどうか

なお，WHENEVER 文の詳細については，「5.7　WHENEVER（埋め込み例外宣言）」を参照してください。

(1) エラーが発生したかどうかの判定（SQLCODE<0）

エラーが発生したかどうかを判定するために，SQLERROR を指定した WHENEVER 文を使用してください。

エラーが発生したときに必要な処理を実装してください。

エラーの内容を参照する処理を実装すると，リターンコードと関連する情報を参照できます。

(2) 検索する行がないかどうかの判定（SQLCODE=100）

検索する行がないかどうかを判定するために，NOT FOUND を指定した WHENEVER 文を使用してください。

検索する行がなくなったときに必要な処理を実装してください。

(3) 正常終了時に警告情報があるかどうかの判定（SQLWARN0='W'）

UAP の正常終了時に警告情報があるかどうかを判定するために，SQLWARNING を指定した WHENEVER 文を使用してください。

正常終了時に警告情報があったときに必要な処理を実装してください。

1.9 SQL 連絡領域

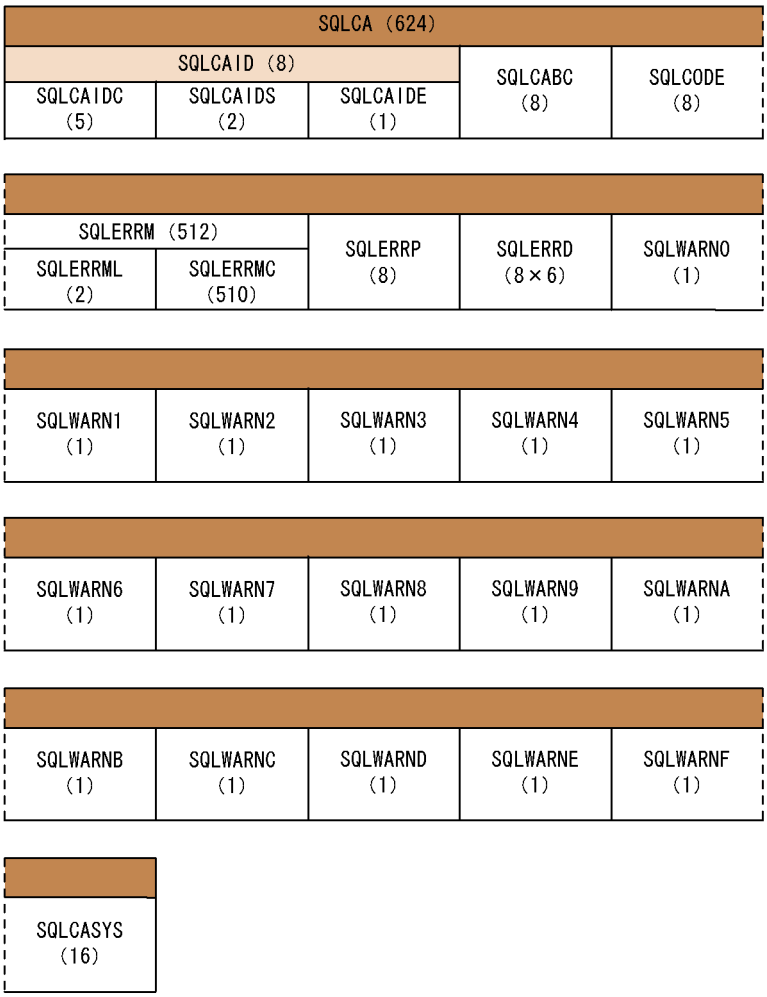
SQL を実行すると，XDB は SQL が正常に実行されたかどうかを示すリターンコードと関連する情報を UAP に返します。これらの情報を受け取るための領域を SQL 連絡領域といいます。ここでは，SQL 連絡領域の構成と内容および SQL 連絡領域の展開について説明します。

なお，SQL 連絡領域の使用方法については，「1.8　SQL のエラーの判定と対処」を参照してください。

1.9.1 SQL 連絡領域の構成

SQL 連絡領域の構成を次の図に示します。

図 1-9　SQL 連絡領域の構成



注
（ ）内は領域の長さ（単位：バイト）を示します。

1.9.2　SQL 連絡領域の内容

SQL 連絡領域の内容を次の表に示します。

表 1-25　SQL 連絡領域の内容

レベル番号	連絡領域名	データ型	長さ（バイト）	内容
1	SQLCA	—	624	SQL 連絡領域全体の名前を表します。

レベル番号	連絡領域名	データ型	長さ（バイト）	内容
2	SQLCAID	—	8	SQLCAIDC 領域，SQLCAIDS 領域および SQLCAIDE 領域の内容を値として持ちます。
3	SQLCAIDC	文字	5	SQL 連絡領域である文字列（'SQLCA'）が設定されます。
3	SQLCAIDS	文字	2	予備
3	SQLCAIDE	文字	1	予備
2	SQLCABC	整数	8	SQL 連絡領域の長さ（624）が設定されています。
2	SQLCODE	整数	8	SQL 実行後に XDB から返されるリターンコードを受け取る領域です。リターンコードには次に示す意味があります。 0：正常に終了しました。 正：正常に終了しましたが，メッセージ情報があります。 負：正常に終了していません。
2	SQLERRM	—	512	SQLERRML 領域，SQLERRMC 領域の内容を値として持ちます。
3	SQLERRML	整数	2	SQLERRMC 領域に返されるメッセージの長さを示します。
3	SQLERRMC	文字	510	SQLCODE 領域に返されるリターンコードに対応するメッセージが設定されます。メッセージの文字コードはシフト JIS です。
2	SQLERRP	文字	8	予備
2	SQLERRD	整数	8 × 6	XDB の内部状態を示す領域です。 SQLERRD(1)：未使用 SQLERRD(2)：未使用 SQLERRD(3)：次のどれかの値が返されます。 • SELECT 文で取り出した行数 • UPDATE 文で更新した行数 • DELETE 文で削除した行数 • INSERT 文で挿入した行数 • FETCH 文で取り出した行数 SQLERRD(4)：未使用 SQLERRD(5)：未使用 SQLERRD(6)：未使用
2	SQLWARN0	文字	1	SQLWARN1 ～ SQLWARN F の領域のどれかに警告フラグ（'W'）が設定された場合に 'W' が設定されます。それ以外の場合は，空白が設定されます。

レベル番号	連絡領域名	データ型	長さ（バイト）	内容
2	SQLWARN1	文字	1	予備
2	SQLWARN2	文字	1	予備
2	SQLWARN3	文字	1	検索結果の列の数と，その検索結果を受け取る埋め込み変数の数が一致しない場合，'W' が設定されます。それ以外の場合は，空白が設定されます。
2	SQLWARN4	文字	1	予備
2	SQLWARN5	文字	1	予備
2	SQLWARN6	文字	1	暗黙的にデータベースへの更新が無効にされて中断状態になった場合に，'W' が設定されます。それ以外の場合は，空白が設定されます。
2	SQLWARN7	文字	1	予備
2	SQLWARN8	文字	1	予備
2	SQLWARN9	文字	1	予備
2	SQLWARNA	文字	1	予備
2	SQLWARNB	文字	1	予備
2	SQLWARNC	文字	1	予備
2	SQLWARND	文字	1	予備
2	SQLWARNE	文字	1	予備
2	SQLWARNF	文字	1	予備
2	SQLCASYS	文字	16	予備

（凡例）

－：該当しません。

1.9.3 COBOL 言語での展開形

SQL 連絡領域の展開形を次に示します。

```
01 SQLCA IS EXTERNAL SYNCHRONIZED.
   02 SQLCAID  PIC X(8).
   02 FILLER   REDEFINES SQLCAID.
      03 SQLCAIDC  PIC X(5).
      03 SQLCAIDS  PIC X(2).
      03 SQLCAIDE  PIC X.
   02 SQLCABC PIC S9(18) COMP.
   02 SQLCODE PIC S9(18) COMP.
   02 SQLERRM.
      03 SQLERRML  PIC S9(4) COMP.
```

```
   03 SQLERRMC  PIC X(510).
02 SQLERRP  PIC X(8).
02 SQLERRD  PIC S9(18) COMP OCCURS 6 TIMES.
02 SQLWARN.
   03 SQLWARN0 PIC X.
   03 SQLWARN1 PIC X.
   03 SQLWARN2 PIC X.
   03 SQLWARN3 PIC X.
   03 SQLWARN4 PIC X.
   03 SQLWARN5 PIC X.
   03 SQLWARN6 PIC X.
   03 SQLWARN7 PIC X.
02 SQLEXT.
   03 SQLWARN8 PIC X.
   03 SQLWARN9 PIC X.
   03 SQLWARNA PIC X.
   03 SQLWARNB PIC X.
   03 SQLWARNC PIC X.
   03 SQLWARND PIC X.
   03 SQLWARNE PIC X.
   03 SQLWARNF PIC X.
02 SQLCASYS1 PIC X(16).
```

1.10 予約語

ここでは，予約語について説明します。

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，二重引用符（"）で囲んでください。予約語を二重引用符（"）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。

XDB の予約語を次の表に示します。

表 1-26 予約語

頭文字	予約語
A	ABS, ABSOLUTE, ACCESS, ACTION, ADD, AFTER, ALIAS, ALL, ALLOCATE, ALTER, AND, ANDNOT, ANY, ARE, ARRAY, AS, ASC, ASSERTION, ASSIGN, AT, AUTHORIZATION, AVG
B	BEFORE, BEGIN, BETWEEN, BINARY, BIT, BIT_AND_TEST, BIT_LENGTH, BLOB, BOOLEAN, BOTH, BREADTH, BY
C	CALL, CASCADE, CASE, CAST, CATALOG, CHANGE, CHAR, CHAR_LENGTH, CHARACTER, CHARACTER_LENGTH, CHECK, CLOB, CLOSE, CLUSTER, COALESCE, COLLATE, COLLATION, COLUMN, COMMENT, COMMIT, CONDITION, CONNECT, CONNECTION, CONSTRAINT, CONSTRAINTS, CONSTRUCTOR, CONTINUE, CONVERT, CORRESPONDING, COUNT, COUNT_FLOAT, CREATE, CROSS, CUBE, CURRENT, CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_USER, CURSOR, CYCLE
D	DATA, DATE, DAY, DAYS, DBA, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DEFERRABLE, DEFERRED, DELETE, DEPTH, DESC, DESCRIBE, DESCRIPTOR, DIAGNOSTICS, DIGITS, DISCONNECT, DISTINCT, DO, DOUBLE, DROP
E	EACH, ELSE, ELSEIF, END, ESCAPE, EXCEPT, EXCEPTION, EXCLUSIVE, EXEC, EXECUTE, EXISTS, EXIT, EXTERNAL, EXTRACT
F	FALSE, FETCH, FIRST, FIX, FLAT, FLOAT, FOR, FOREIGN, FOUND, FREE, FROM, FULL, FUNCTION
G	GET, GLOBAL, GO, GOTO, GRANT, GROUP, GROUPING
H	HANDLER, HASH, HAVING, HEX, HOUR, HOURS
I	IDENTIFIED, IDENTITY, IF, IMMEDIATE, IN, INDEX, INDICATOR, INITIALLY, INNER, INOUT, INPUT, INSENSITIVE, INSERT, INT, INTEGER, INTERSECT, INTERVAL, INTO, IS, ISOLATION, ITERATE
J	JOIN
K	KEY
L	LANGUAGE, LARGE, LAST, LEADING, LEAVE, LEFT, LENGTH, LEVEL, LIKE, LIMIT, LIST, LOCAL, LOCATOR, LOCK, LONG, LOOP, LOWER

頭文字	予約語
M	MATCH, MAX, MCHAR, MICROSECOND, MICROSECONDS, MIN, MINUTE, MINUTES, MOD, MODE, MODULE, MONTH, MONTHS, MVARCHAR
N	NAMES, NATIONAL, NATURAL, NCHAR, NCLOB, NEW, NEXT, NO, NONE, NOT, NOWAIT, NULL, NULLIF, NUMERIC, NVARCHAR
O	OBJECT, OCTET_LENGTH, OF, OFFSET, OLD, ON, ONLY, OPEN, OPTIMIZE, OPTION, OR, ORDER, OUT, OUTER, OUTPUT
P	PAD, PAGE, PARTIAL, PARTITIONED, PCTFREE, POSITION, PRECISION, PREPARE, PRESERVE, PRIMARY, PRIOR, PRIVATE, PRIVILEGES, PROCEDURE, PROGRAM, PROTECTED, PUBLIC, PURGE
Q	－
R	READ, REAL, RECOVERY, RECURSIVE, REFERENCES, REFERENCING, RELATIVE, RELEASE, REPEAT, RESIGNAL, RESTRICT, RETURN, RETURNS, REVOKE, RIGHT, ROLLBACK, ROLLUP, ROUTINE, ROW, ROWID, ROWS
S	SAVEPOINT, SCHEMA, SCROLL, SEARCH, SECOND, SECONDS, SECTION, SELECT, SENSITIVE, SEQUENCE, SESSION, SESSION_USER, SET, SHARE, SIGNAL, SIMILAR, SIZE, SMALLFLT, SMALLINT, SOME, SPACE, SQL, SQLCODE, SQLCODE_OF_LAST_CONDITION, SQLCOUNT, SQLERRM_OF_LAST_CONDITION, SQLERROR, SQLEXCEPTION, SQLSTATE, SQLWARNING, STRUCTURE, SUBSTR, SUBSTRING, SUM, SUPPRESS, SYSTEM_USER
T	TABLE, TEMPORARY, TEST, THEN, TIME, TIMESTAMP, TIMESTAMP_FORMAT, TIMEZONE_HOUR, TIMEZONE_MINUTE, TO, TRAILING, TRANSACTION, TRANSLATE, TRANSLATION, TRIGGER, TRIM, TRUE, TYPE
U	UNDER, UNION, UNIQUE, UNKNOWN, UNTIL, UPDATE, UPPER, USAGE, USER, USING
V	VALUE, VALUES, VARCHAR, VARCHAR_FORMAT, VARYING, VIEW
W	WAIT, WHEN, WHENEVER, WHERE, WHILE, WITH, WITHOUT, WORK, WRITE
X	XLIKE
Y	YEAR, YEARS
Z	－

（凡例）

－：該当しません。

2 構成要素

この章では，SQL の構成要素について説明します。

2.1 カーソル指定

(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，カーソル宣言および SELECT 文中に指定します。

(2) 形式

カーソル指定：：＝問合せ式 〔ORDER BY句〕 〔LIMIT句〕
ORDER BY**句：：＝**ORDER BY **列指定 〔 {**ASC｜DESC**} 〕 〔，列指定 〔 {**ASC｜DESC**} 〕 〕…**
LIMIT**句：：＝**LIMIT **リミット行数**
リミット行数：：＝ {整数定数｜：埋め込み変数}

(a) 問合せ式

検索条件（表式）と検索した結果を出力する項目（選択式）を指定すると，検索結果を格納した仮想的な表が生成されます。この表を導出表といいます。導出表は，選択式で指定した列で構成されます。

なお，XDB では，問合せ式は問合せ指定と等しくなります。問合せ指定については，「2.2　問合せ指定」を参照してください。

(b) ORDER BY 句

問合せ式での検索結果を，昇順また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されません。

ORDER BY 句についての規則を次に示します。

1. ソートキーに指定する列の数は，16 以下にしてください。
2. 同じ列は 2 回以上指定できません。
3. 問合せ式で導出される表に重複する列名があるときは，その列名をソートキーに指定できません。
4. ソートキー中に指定した導出列名は，単一の列指定だけから導出されている場合，その列指定によって置き換えられます。
5. ソートキーに列指定を指定した場合，列指定が参照する列を参照します。

列指定

ソート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。

- ソートキーに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）以外のものを指定する場合は，選択式中に集合関数を指定できません。
- 次のデータ型の列は指定できません。

・定義長が 4037 バイト以上の CHAR 型
・ROWID 型

{ASC ｜ DESC}

ASC
検索結果を昇順に並べ替える場合に指定します。

DESC
検索結果を降順に並べ替える場合に指定します。

(c) LIMIT 句

問合せ式の検索結果のうち，取得する行数を指定します。

LIMIT **リミット行数**
問合せ式の検索結果から取得する行数を指定します。

LIMIT 句についての規則を次に示します。

1. リミット行数のデータ型には整数（INTEGER 型）を指定してください。指定の範囲は 0 ～ 2147483647 です。
2. リミット行数にナル値は指定できません。
3. リミット行数に埋め込み変数を指定した場合，埋め込み変数のデータ型には，INTEGER 型が仮定されます。
4. リミット行数の指定がある場合，検索結果行数は次のようになります。
   MIN**（問合せ式の検索結果行数，リミット行数）**
5. 問合せ式の検索結果がリミット行数を超える場合，次のときには検索結果が一意に定まりません。
   - リミット行数で指定した最終行と同じソートキーの値を持つ行が複数存在する
   - ソートキーの指定がない

   検索結果が一意に定まらない場合の例を次に示します。

   (例)

   次の SQL を使用して表 ZAIKO を検索した場合，ソートキーとなる列 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。

```
SELECT * FROM "ZAIKO" ORDER BY "TANKA" LIMIT 3
```

図 2-1　LIMIT 句の指定によって，検索結果が不定となる例

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

●検索結果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

●検索結果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

(3) 注意事項

ORDER BY 句を指定すると，作業表を作成することがあります。

(4) 指定例

(例 1)

SELECT 文中に ORDER BY 句を指定した例を次に示します。

```
SELECT "SNAME" FROM "ZAIKO" ORDER BY "SNAME" ASC
```

注

下線部分が ORDER BY 句の部分です。

(例 2)

SELECT 文中に LIMIT 句を指定した例を次に示します。

```
SELECT "SCODE","ZSURYO" FROM "ZAIKO"
WHERE "ZSURYO">20 ORDER BY "ZSURYO","SCODE"
LIMIT 10
```

注

下線部分が LIMIT 句の部分です。

2.2 問合せ指定

(1) 機能

検索条件（表式）と検索した結果を出力する項目（選択式）を指定すると，検索結果を格納した仮想的な表が生成されます。この表を導出表といいます。導出表は，選択式で指定した列で構成されます。

(2) 形式

```
問合せ指定：：＝
  SELECT 選択リスト 表式
選択リスト：：＝｛*｜選択式〔，選択式〕…｝
選択式：：＝｛値式 〔列名〕｜表指定．*｜〔表指定．〕ROW｝
```

(a) 選択リスト

選択リスト：：＝ ｛*｜選択式〔，選択式〕…｝

*****

表のすべての列を出力する場合に指定します。
* 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とします。

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。

(b) 選択式

選択式：：＝｛値式 〔列名〕｜表指定．*｜〔表指定．〕ROW｝

値式 〔列名〕

i 番目の選択式の値式に対して列名を指定した場合，問合せ指定によって導出される表の i 番目の列の列名は指定した列名となります。
i 番目の選択式の値式に対して列名を指定しない場合は，次のようになります。

- 列指定から導出された値式の場合，その列指定で指定した列名が，問合せ指定によって導出される表の i 番目の列の列名となります。
- 列指定以外から導出された値式の場合，問合せ指定によって導出される表の i 番目の列の列名は設定されません（名前のない列となります）。

表指定．*

指定した表のすべての列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。

〔表指定．〕ROW

ROW は行全体を意味します。ROW 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。

- ROW は，ディクショナリ以外の表に対してだけ指定できます。
- 取り出されるデータに指定する埋め込み変数は，各列のデータ型に関係なく，ROW に対応する埋め込み変数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。
- 選択式中に ROW を指定する場合は，その問合せ指定に対する集合関数と同時に ROW を指定できません。
- 埋め込み変数中の時刻印データのデータ長は，秒数の小数点以下のけたを p とした場合，(7 ＋ (p ÷ 2)) バイトです。また，形式は X'YYYYMMDDhhmmss〔nn…n〕' となります。

(c) 表式

検索の対象となる表を指定します。また，表を検索するときの条件（探索条件）が指定できます。表式については，「2.3　表式」を参照してください。

(3) 共通規則

問合せ指定によって導出される列の数は，3000 以下にしてください。

(4) 指定例

問合せ指定の例を次に示します。

```
SELECT "GNO","GNAME","KIKAKU","TANKA","SURYO" FROM "ZAIKO"
ORDER BY "GNO"
```

注

下線部分が問合せ指定の部分です。

2.3　表式

(1) 機能

検索の対象となる一つまたは複数の表を指定します。

また，表を検索または結合するときの条件（探索条件）が指定できます。

表式は問合せ指定中に指定します。

(2) 形式

表式：：＝FROM**句**
　　〔WHERE**句**〕

(a) FROM 句

FROM 句については，「2.4　FROM 句」を参照してください。

(b) WHERE 句

WHERE 句については，「2.5　WHERE 句」を参照してください。

(3) 構文規則

表式の結果は，i 番目の列の記述子が FROM 句で指定した表の i 番目の列の記述子と等しい導出表になります。

(4) 一般規則

WHERE 句を省略した場合，表式の結果は FROM 句の結果になります。

それ以外の場合は，指定した各句の結果が直後に指定した句に適用されて，表式の結果は，最後に指定した句の結果になります。

(5) 指定例

SELECT 文中で表式を指定した例を次に示します。

```
SELECT "GNO" FROM "ZAIKO"
WHERE "GENKA" >= 25000
```

注

　下線部分が表式の部分です。

2.4 FROM 句

(1) 機能

一つの名前付きの表から導出される表を指定します。

(2) 形式

```
FROM句：：＝FROM 表参照リスト
  表参照リスト：：＝表参照〔，表参照〕…
```

(3) 構文規則

1. 表参照リストに，表名を複数含む問合せ（複数の表にわたる問合せ）を結合といいます。表参照については，「2.6　表参照」を参照してください。
2. 表参照リストに指定する表参照中の表名の数は，8 以下にしてください。
3. 表参照リストに一つだけ表参照を指定した場合，表参照リストの結果の記述子は，その表参照の記述子と同じになります。
4. 表参照リストに複数の表参照を指定した場合，表参照リストの結果の列の記述子は，表参照によって識別される表の列の記述子と同じになります。この表の列の記述子の順序は，表参照リスト中の表参照の順序と，その表内で定義した列の順序と同じになります。
5. FROM 句の結果の記述子は，表参照リストの結果の記述子と同じになります。

(4) 一般規則

1. 表参照リストに一つだけ表参照を指定した場合，表参照リストの結果はその表参照の結果になります。
2. FROM 句の結果は，表参照リストの結果になります。

2.5 WHERE 句

(1) 機能

先行する FROM 句の結果に，探索条件を適用することで導出される表を指定します。

探索条件を省略すると，表から導き出されるすべての行が検索されます。

(2) 形式

WHERE**句**：：＝WHERE **探索条件**

探索条件については，「2.7　探索条件」を参照してください。

(3) 構文規則

ありません。

(4) 一般規則

先行する FROM 句の結果を T とした場合，探索条件は T のそれぞれの行に適用されます。WHERE 句の結果は，探索条件の結果が真となる T の行の集合から成る一つの表です。

2.6 表参照

(1) 機能

検索する表を指定します。表参照は表式中に指定します。

(2) 形式

表参照：：＝表名 〔相関名〕

(a) 表名

検索する表の表名を指定します。

ディクショナリ表を検索する場合は，スキーマ名に "MASTER" を指定します。

表名については，「1.2.8　名前の修飾」を参照してください。

(b) 相関名

同じ表を結合する場合，表同士を区別するために付ける名称を指定します。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。また，ほかの範囲変数の表識別子と同じ名称も指定できません。

2.7 探索条件

(1) 機能

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。

探索条件は WHERE 句に指定します。

(2) 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：：＝｛BETWEEN述語｜比較述語｝
```

述語については，「2.8 述語」を参照してください。

(3) 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

(4) 論理演算の結果

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，ナル値を含む述語の結果は不定になります。

図 2-2 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

(5) 規則

SQL の探索条件の中に指定する論理演算の数は，255 以下にしてください。

2.8 述語

2.8.1 述語

(1) 機能

真，偽，不定の論理値を与えるために評価ができる条件を指定します。

(2) 形式

```
述語：：＝｛BETWEEN述語｜比較述語｝
```

2.8.2 BETWEEN 述語

(1) 形式

```
BETWEEN述語：：＝値式1〔NOT〕 BETWEEN 値式2 AND 値式3
値式1：：＝値式
値式2：：＝値式
値式3：：＝値式
```

(2) 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。

値式 2 **≦値式** 1 **≦値式** 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。

(3) 規則

(a) 値式 1

埋め込み変数，標識変数だけの値式は指定できません。

(b) 共通

1. 各値式中のデータ型は，それぞれ変換できるデータ型にしてください。
2. 値式の結果がナル値の場合，その比較結果は不定となります。

2.8.3 比較述語

(1) 形式

```
比較述語：：＝比較演算項 比較演算子 比較演算項
比較演算子：：＝ {= | <> | != | ^= | < | <= | > | >=}
比較演算項：：＝値式
```

(2) 述語が真となる場合

左右の比較演算項が，比較条件を満たす場合に真となります。

なお，比較演算項のどれかがナル値の場合は不定となります。

(3) 規則

1. 比較述語を「比較演算項 X　比較演算子　比較演算項 Y」として，比較演算子の意味を次の表に示します。

表 2-1　比較演算子の意味

項番	比較演算子の指定	意味
1	比較演算項 X = 比較演算項 Y	比較演算項 X と比較演算項 Y が等しい。
2	比較演算項 X <> 比較演算項 Y 比較演算項 X != 比較演算項 Y 比較演算項 X ^= 比較演算項 Y	比較演算項 X と比較演算項 Y が等しくない。
3	比較演算項 X < 比較演算項 Y	比較演算項 X が比較演算項 Y より小さい。
4	比較演算項 X <= 比較演算項 Y	比較演算項 X が比較演算項 Y 以下である。
5	比較演算項 X > 比較演算項 Y	比較演算項 X が比較演算項 Y より大きい。
6	比較演算項 X >= 比較演算項 Y	比較演算項 X が比較演算項 Y 以上である。

2. 左右の比較演算項のデータ型は比較できるデータ型にしてください。
3. 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。範囲の広さの順を次に示します。
   DECIMAL ＞ INTEGER ＞ SMALLINT
4. 比較演算項の結果がナル値の場合，その比較結果は不定となります。
5. 比較できるデータについては，「1.3.2　比較，代入できるデータ型」を参照してください。
6. 比較演算子の両側に，埋め込み変数，標識変数だけの比較演算項は指定できません。
7. 値式に ROWID 型のデータを指定した場合は，比較演算子の「=」だけ指定できます。また，特殊レジスタの行 ID と比較できる値式は，変数（埋め込み変数，標識変数）だけです。

2.9 値式

(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：：＝値式一次子
値式一次子：：＝｛列指定｜値指定｜集合関数｜行ID｝

(3) 規則

値式の結果のデータ型は，値式一次子の結果と同じデータ型となります。

2.10 値指定

(1) 機能

SQL 中に一つ以上の値，もしくは埋め込み変数および標識変数を指定する場合に，値指定を指定できます。

(2) 形式

```
値指定：：＝｛定数｜一般値指定｝
一般値指定：：＝｛：埋め込み変数〔：標識変数〕｜CURRENT_TIMESTAMP値関数｝
```

(3) 共通規則

値指定の結果のデータ型は，定数，または一般値指定の結果と同じデータ型となります。一般値指定に埋め込み変数を指定した場合，結果のデータ型は，埋め込み変数を指定した個所によって仮定されるデータ型となります。仮定されるデータ型については，「1.6.1 埋め込み変数および標識変数」を参照してください。

2.11 集合関数

2.11.1 集合関数の概要

(1) 機能

集合関数は，複数行から算出される値を求めることができます。

集合関数の機能を次の表に示します。

表 2-2 集合関数の機能

項番	集合関数	内容
1	COUNT	集合関数の入力行数を求めます。

集合関数の特徴を次の表に示します。

表 2-3 集合関数の特徴

項番	項目	COUNT
1	ナル値の扱い	無視※
2	適用対象が 0 件またはナル値だけの集合の場合の関数値	0

注※

COUNT(*) 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。

(2) 形式

集合関数：：＝ {COUNT(*)｜**一般集合関数**}
一般集合関数：：＝ {ALL**集合関数**｜DISTINCT**集合関数**}
ALL**集合関数：：＝**COUNT（〔ALL〕**列指定**）
DISTINCT**集合関数：：＝**COUNT（DISTINCT **列指定**）

(3) 規則

1. 集合関数は，その集合関数が含まれる問合せ指定の選択式中に指定してください。
2. WHERE 句または 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を集合関数の入力とします。
3. 一般集合関数を指定した場合，集合関数の入力行数に等しい行数から構成され，引数に指定する列の値がナル値である行を除いた表を，一般集合関数の入力とします。
4. DISTINCT を指定した場合，引数に指定する列の値が重複する行を除いた結果を，一般集合関数の入力とします。

5. 一つの SQL 文中に DISTINCT 集合関数を複数指定する場合，それらの引数に指定する列は，すべて同じにしてください。
6. 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。

(4) 注意事項

DISTINCT 集合関数を指定すると，作業表が作成されることがあります。

2.11.2 COUNT(*)

(1) 機能

集合関数の入力行数を求めます。

(2) 規則

1. 結果のデータ型は INTEGER となります。
2. 入力行数が 0（ゼロ）の場合，結果は 0 となります。

2.11.3 COUNT

(1) 機能

集合関数の入力行数を求めます。

(2) 規則

1. 対象となる行数が 0（ゼロ）の場合，結果は 0 となります。
2. 引数に指定できる列のデータ型および結果のデータ型を次の表に示します。

表 2-4 列のデータ型と集合関数 COUNT の結果のデータ型の関係

項番	列のデータ型	結果のデータ型
1	INTEGER	INTEGER
2	SMALLINT	INTEGER
3	DECIMAL(m,n)	INTEGER
4	CHAR(n) ※	INTEGER
5	VARCHAR(n) ※	INTEGER
6	TIMESTAMP(p)	INTEGER

（凡例）
m，n：正の整数
p：{0 ｜ 2 ｜ 4 ｜ 6}

注※

n は 4036 以下にしてください。

3 定義系 SQL

定義系 SQL は，表またはインデクスを定義するために使用する SQL です。
この章では，定義系 SQL の機能，形式，および規則について説明します。

3.1 CREATE INDEX（インデクスの定義）

(1) 機能

表にインデクスを定義します。

CREATE INDEX 文は，定義系 SQL 文記述ファイルに記述します。定義系 SQL 文記述ファイルについては，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。

(2) 形式

```
インデクス定義：：＝
CREATE 〔UNIQUE〕 INDEX インデクス名
        ON 表名 （列名〔 {ASC｜DESC}〕〔，列名 〔 {ASC｜DESC}〕〕…）
          インデクス用DBエリア指定
         〔インデクスオプション〕…

インデクス用DBエリア指定：：＝IN DBエリア名
インデクスオプション：：＝｛インデクス未使用領域指定｜セグメント再利用指定｝
インデクス未使用領域指定：：＝PCTFREE＝未使用領域の比率
        未使用領域の比率：：＝符号なし整数定数
セグメント再利用指定：：＝SEGMENT REUSE ｛再利用契機セグメント数
                                  ｜（再利用契機セグメント数〔，再利用契機セグメント数増分値〕）
                                    ｜（，再利用契機セグメント数増分値）
                                      ｜NOUSE｝
        再利用契機セグメント数：：＝符号なし整数定数
        再利用契機セグメント数増分値：：＝符号なし整数定数
```

(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また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。このようなインデクスをユニークインデクスと呼びます。

UNIQUE を指定している場合に，インデクスの作成または更新時に重複キー値を検出すると，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。

(b) インデクス名

定義するインデクスのインデクス名を指定します。インデクス名はシステム内で重複できません。インデクス名については，「1.2.8　名前の修飾」を参照してください。

(c) 表名

インデクスを定義する表の表名を指定します。表名については，「1.2.8　名前の修飾」を参照してください。

(d) （列名 〔{ASC ｜ DESC}〕〔，列名 〔{ASC ｜ DESC}〕〕…）

インデクスを定義する列の列名およびインデクスの並び順を指定します。

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 16 個指定できます。
列名を複数個指定する場合は，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) インデクス用 DB エリア指定

インデクス用DBエリア指定：：＝IN DBエリア名

この指定は省略できません。必ず指定してください。

インデクスを格納するインデクス用 DB エリア名を指定します。表用 DB エリアは指定できません。

(f) インデクスオプション

インデクスオプションには，同一のオプションを繰り返して指定できません。

(g) インデクス未使用領域指定

インデクス未使用領域指定：：＝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。

インデクス未使用領域指定は次の場合に適用されます。そのほかの場合は，PCTFREE=0 が仮定されます。

- データをインポートするとき
- INSERT 文や UPDATE 文によって，インデクス内の最大キー値を管理するページにデータを追加するとき

未使用領域の比率

未使用領域の比率は，0 ～ 99（単位：%）を指定します。この指定を省略すると，30% が仮定されます。
インデクスを定義したあと，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。
未使用領域の指定については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」の「ページ内の未使用領域の設定」を参照してください。

(h) セグメント再利用指定

セグメント再利用指定：：＝SEGMENT REUSE
{再利用契機セグメント数

｜（再利用契機セグメント数〔，再利用契機セグメント数増分値〕）
｜（，再利用契機セグメント数増分値）
｜NOUSE}

インデクスはページを再利用する契機がないため，通常はこの指定を行わないでください。

セグメントの再利用機能を使用する場合に，セグメントの再利用を実行する契機（再利用契機セグメント数または再利用契機セグメント数増分値）を指定します。

セグメントの再利用とは，使用中のインデクスが管理するセグメントで，最後に確保したセグメント内の空きページを使い切った場合，新規セグメントを確保しないで，そのインデクスが管理するセグメントから空きページを検索して，割り当てることをいいます。

この指定を省略した場合は，NOUSE を指定したものとして扱います。

再利用契機セグメント数

セグメントの再利用を実行する契機となるセグメント数を指定します。
再利用契機セグメント数には，0 ～ 16777216 の符号なし整数を指定できます。
再利用契機セグメント数を省略した場合は，0 が仮定されます。

1 以上を指定した場合：

確保したセグメント数が，指定した再利用契機セグメント数に達した場合に，セグメントの再利用を実行します。ただし，確保したセグメント数が再利用契機セグメント数に達する前に，新規セグメントが確保できなくなると，その時点でセグメントの再利用を実行します。
セグメントの再利用を実行したあと，次回，空きページを検索する場合は，前回，割り当てたページを起点とします。

0 を指定した場合：

新規セグメントが確保できなくなったときにセグメントの再利用を行います。
セグメントの再利用を実行したあと，次回，空きページを検索する場合は，前回，割り当てたページを起点とします。

再利用契機セグメント数増分値

再利用契機セグメント数の増分値を指定します。
再利用契機セグメント数増分値には，0 ～ 16777216 の符号なし整数を指定できます。
再利用契機セグメント数増分値を省略した場合は，0 が仮定されます。

1 以上を指定した場合：

セグメントの再利用の実行時，そのインデクスが管理するセグメントに空きページがなくなり，新規セグメントを確保した場合，新たな再利用契機セグメント数を決定するための増分値を指定します。再利用契機セグメント数に増分値を加算した値を新たな再利用契機セグメント数とします。ただし，確保したセグメント数が再利用契機セグメント数に達する前に，新規セグメントが確保できなくなると，その時点でセグメントの再利用を実行します。

0 を指定した場合：

再利用契機セグメント数を増分しません。
確保したセグメント数が再利用契機セグメント数を超えたあとは，新規セグメントが確保できなくなった時点で，セグメントの再利用を実行します。

NOUSE

セグメントの再利用機能を使用しない場合に指定します。
新規セグメントが確保できなくなったときに，セグメントの再利用を実行します。
セグメントの再利用を実行したあと，次回，空きページを検索する場合は，前回，割り当てたページを起点とします。

（3） 規則

1. XDBUSER が所有するインデクスを定義できます。
2. XDBUSER が所有する表にインデクスを定義できます。
3. インデクスは，一つの表に最大 16 個定義できます。
4. インデクスはシステム内に最大 2048 個定義できます。ただし，一つの DB エリアに格納できるインデクスは 400 個です。
5. インデクスは，次のデータ型の列に対して定義できません。
   - 精度が 20 けた以上の DECIMAL 型
6. 単一列インデクスを定義する場合，インデクスを定義する列の定義長が次に示す値以下である必要があります。

MIN｛（a÷2）－100，4036｝　**（単位：バイト）**

a：インデクス用 DB エリアのページサイズ

7. 単一列インデクスに対して DESC を指定した場合は，ASC を指定したものとして扱います。
8. 次のインデクスは複数個定義できません。
   - 列構成が同じであり，かつすべての列の昇順，降順の指定が同じであるインデクス
   - 列構成が同じであり，かつすべての列の昇順，降順の指定が逆であるインデクス
9. 複数列インデクスは，次の表に示すデータ型の列に対して定義できます。複数列インデクスを構成する列の長さはデータ型によって異なります。各列の長さの合計長が次の値以下であれば，複数列インデクスを定義できます。

MIN｛（a÷2）－100，4036｝　**（単位：バイト）**

a：インデクス用 DB エリアのページサイズ

表 3-1 複数列インデクスを構成する列の長さ

項番	データ型	複数列インデクスを構成する列の長さ※1		
		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	
			構成列が固定長だけの場合	構成列に可変長を含む場合※3
1	INTEGER	5	5	6
2	SMALLINT※2	3	3	4
3	DECIMAL(m,n)	↓ m ÷ 2 ↓ + 2	↓ m ÷ 2 ↓ + 2	↓ m ÷ 2 ↓ + 3
4	CHAR(n)	n + 1	n + 1	n + 2
5	VARCHAR(n)※2	n + 1	－	n + 2
6	TIMESTAMP(p)	8 + p ÷ 2	8 + p ÷ 2	9 + p ÷ 2
7	ROWID※2	13	13	14

（凡例）

m，n：正の整数

p：{0 ｜ 2 ｜ 4 ｜ 6}

－：該当しません。

注※ 1

「各列の長さの合計が 255 バイト以下の場合」の列長を基に計算した結果，合計が 256 バイト以上になったときには，「各列の長さの合計が 256 バイト以上の場合」を基に列の長さを計算し直してください。

注※ 2

これらのデータ型は作業表だけで使用します。そのため，CREATE TABLE 文でこれらのデータ型を指定することはできません。

注※ 3

作業表だけに該当する説明です。

(4) 留意事項

1. 複数列インデクスを定義する場合，各列の指定順位は，キー値作成の優先順位になります。
2. 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。

(5) 使用例

(例 1)

飲食店表（INSHOKU）に次に示す条件でインデクスを定義します。

- 飲食店表の店舗コード（SHOP_CODE）列に，重複キーがない（UNIQUE），昇順のインデクス（IDX1）を定義します。
- 行の追加が頻繁に発生すると予想されるため，インデクスページ内の未使用領域の比率を 50% にします。
- インデクスは DBIDX001 という名称のインデクス用 DB エリアに格納します。

```
CREATE UNIQUE INDEX "IDX1" ON "INSHOKU"("SHOP_CODE" ASC)
IN DBIDX001 PCTFREE = 50
```

（例 2）

飲食店表（INSHOKU）に次に示す条件で複数列インデクスを定義します。

- 飲食店表の店舗コード（SHOP_CODE）列と地域コード（AREA_CODE）列を組にして，店舗コードを昇順に，地域コードを降順にした複数列インデクス（IDX3）を定義します。
- インデクスは DBIDX001 という名称のインデクス用 DB エリアに格納します。

```
CREATE INDEX "IDX3" ON "INSHOKU"
    ("SHOP_CODE" ASC,"AREA_CODE" DESC)
IN DBIDX001
```

3.2 CREATE TABLE（表の定義）

(1) 機能

表を定義します。

CREATE TABLE 文は，定義系 SQL 文記述ファイルに記述します。定義系 SQL 文記述ファイルについては，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。

(2) 形式

```
表定義：：＝CREATE FIX TABLE 表名（表要素〔，表要素〕…）
          表用DBエリア指定
          〔表オプション〕…

表要素：：＝列定義
列定義：：＝列名 データ型
表用DBエリア指定：：＝IN DBエリア名
表オプション：：＝｛表未使用領域指定｜セグメント再利用指定｝
表未使用領域指定：：＝PCTFREE＝未使用領域の比率
        未使用領域の比率：：＝符号なし整数定数
セグメント再利用指定：：＝SEGMENT REUSE
            {再利用契機セグメント数
            |（再利用契機セグメント数〔，再利用契機セグメント数増分値〕）
              |（，再利用契機セグメント数増分値）
                |NOUSE}
        再利用契機セグメント数：：＝符号なし整数定数
        再利用契機セグメント数増分値：：＝符号なし整数定数
```

(a) FIX

行の長さが固定となる表を定義します。

(b) 表名

定義する表の表名を指定します。表名はシステム内で重複できません。表名については，「1.2.8　名前の修飾」を参照してください。

(c) 表要素

表要素：：＝列定義

列定義

表を構成する列自体に関する定義（列名，データ型の指定など）を行います。

(d) 列定義

列定義：：＝列名 データ型

列名

表を構成する列の名前を指定します。同一表内では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。

(e) 表用 DB エリア指定

表用DBエリア指定：：＝IN DBエリア名

この指定は省略できません。必ず指定してください。

表を格納する表用 DB エリア名を指定します。インデクス用 DB エリアは指定できません。

(f) 表オプション

表オプションには，同一のオプションを繰り返して指定できません。

(g) 表未使用領域指定

表未使用領域指定：：＝PCTFREE＝未使用領域の比率

ユーザが定義する表の列のデータ型に VARCHAR を指定できないため，通常はこの指定を行わないでください。

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。

表未使用領域指定は，データのインポート時に適用されます。INSERT 文や UPDATE 文では PCTFREE=0 として扱われます。

未使用領域の比率
未使用領域の比率は，0 ～ 99（単位：%）を指定します。この指定を省略すると，0% が仮定されます。
VARCHAR 型を含む表に対して，行長の長くなる更新が頻繁に発生する場合は，同一ページ内に行を格納できなくなり，行の配置に乱れが生じるため，未使用領域の比率を高くしてください。

(h) セグメント再利用指定

セグメント再利用指定：：＝SEGMENT REUSE
{再利用契機セグメント数
｜（再利用契機セグメント数〔，再利用契機セグメント数増分値〕）
｜（，再利用契機セグメント数増分値）
｜NOUSE}

セグメントの再利用機能を使用して，セグメントの再利用を実行する契機（再利用契機セグメント数または再利用契機セグメント数増分値）を指定します。

セグメントの再利用とは，使用中の表が管理するセグメントで，最後に確保したセグメント内の空きページを使い切った場合，新規セグメントを確保しないで，その表が管理

するセグメントから空きページを検索して，割り当てることをいいます。

この指定を省略した場合は，NOUSE を指定したものとして扱います。

セグメントの再利用については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」の「セグメントの再利用の設定」を参照してください。

再利用契機セグメント数

セグメントの再利用を実行する契機となるセグメント数を指定します。
再利用契機セグメント数には，0 ～ 16777216 の符号なし整数を指定できます。
再利用契機セグメント数を省略した場合，0 が仮定されます。

1 以上を指定した場合：

確保したセグメント数が，指定した再利用契機セグメント数に達した場合に，セグメントの再利用を実行します。ただし，確保したセグメント数が再利用契機セグメント数に達する前に，新規セグメントが確保できなくなると，その時点でセグメントの再利用を実行します。
セグメントの再利用を実行したあと，次回，空きページを検索する場合は，前回，割り当てたページを起点とします。

0 を指定した場合：

新規セグメントが確保できなくなったときにセグメントの再利用を行います。
セグメントの再利用を実行したあと，次回，空きページを検索する場合は，前回，割り当てたページを起点とします。

再利用契機セグメント数増分値

再利用契機セグメント数の増分値を指定します。
再利用契機セグメント数増分値には，0 ～ 16777216 の符号なし整数を指定できます。
再利用契機セグメント数増分値を省略した場合，0 が仮定されます。

1 以上を指定した場合：

セグメントの再利用の実行時，その表が管理するセグメントに空きページがなくなり，新規セグメントを確保した場合，再利用契機セグメント数に再利用契機セグメント数増分値を加算した値を新たな再利用契機セグメント数とします。ただし，確保したセグメント数が再利用契機セグメント数に達する前に，新たなセグメントが確保できなくなると，その時点でセグメントの再利用を実行します。

0 を指定した場合：

再利用契機セグメント数を増分しません。
確保したセグメント数が再利用契機セグメント数を超えたあとは，新規セグメントが確保できなくなった時点で，セグメントの再利用を実行します。

NOUSE

セグメントの再利用機能を使用しない場合に指定します。
新規セグメントが確保できなくなったときに，セグメントの再利用を実行します。

セグメントの再利用を実行したあと，次回，空きページを検索する場合は，前回，割り当てたページを起点とします。

(3) 規則

1. XDBUSER が所有する表を定義できます。
2. 表は最大 1024 個定義できます。ただし，一つの DB エリアに格納できる表は 200 個です。
3. 列は一つの表に最大 3000 個定義できます。
   列の長さ（データ長）の合計（BL）は，次の式を満たす必要があります。各列のデータ型ごとの長さ（データ長）を次の表に示します。

$$BL = \uparrow \left(14 + \sum_{i=1}^{n} Ci \right) \div 2 \uparrow \times 2 \leqq \text{ページサイズ} - 40$$

作業表の場合は，次の計算式となります。

$$BL = \uparrow \left(16 + 2 \times n + \sum_{i=1}^{n} Ci \right) \div 2 \uparrow \times 2 \leqq \text{ページサイズ} - 40$$

n：表に定義する列の総数（個）
Ci：各列のデータ長（バイト）

表 3-2　データ型ごとのデータ長

項番	分類	データ型	データ長（バイト）
1	数データ	INTEGER	4
2		SMALLINT ※1	2
3		DECIMAL(m,n) ※2	↓ m ÷ 2 ↓ + 1
4	文字データ	CHAR(n)	n
5		VARCHAR(n) ※1	7
6	時刻印データ	TIMESTAMP〔(p)〕	7 + p ÷ 2
7	行 ID データ	ROWID ※1	12

（凡例）
m，n：正の整数
p：{0 ｜ 2 ｜ 4 ｜ 6}

注※ 1

これらのデータ型は作業表だけで使用します。そのため，CREATE TABLE 文でこれらのデータ型を指定することはできません。

注※ 2

全体のけた数が m けたで，小数秒のけた数が n けたの固定小数点数です。m を省略した場合は 29 が仮定されます。

(4) 使用例

(例 1)

飲食店表（INSHOKU）を次に示す条件で定義します。

- 飲食店表は DBDATA001 という名称の表用 DB エリアに格納します。

```
CREATE FIX TABLE "INSHOKU"
    ("SID" INT,"TEL_NO" CHAR(10),"SHOP_NAME" CHAR(120),
     "POST_CODE" CHAR(7), "ADDRESS" CHAR(300)) IN DBDATA001
```

(例 2)

飲食店表（INSHOKU）を次に示す条件で定義します。

- セグメント再利用機能を使用します。再利用契機セグメント数を 20，再利用契機セグメント数増分値を 10 とします。
- 飲食店表は DBDATA001 という名称の表用 DB エリアに格納します。

```
CREATE FIX TABLE "INSHOKU"
    ("SID" INT,"TEL_NO" CHAR(10),"SHOP_NAME" CHAR(120),
     "POST_CODE" CHAR(7), "ADDRESS" CHAR(300))
    IN DBDATA001
    SEGMENT REUSE(20,10)
```

4 操作系 SQL

操作系 SQL は，表のデータを操作（検索，追加，削除および更新）するときに使用します。
この章では，操作系 SQL の機能，形式および規則について説明します。

4.1 CLOSE（カーソルのクローズ）

(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) 形式

CLOSE**文：：**＝CLOSE **カーソル名**

(a) 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。

(3) 共通規則

1. 開いた状態のカーソルを指定してください。
2. トランザクションが終了した場合，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，暗黙的にデータベースへの更新が無効にされて中断状態になった場合にも，カーソルはすべて閉じられます。

(4) 注意事項

カーソル名は，埋め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す。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。したがって，SQL 中にカーソル名を指定している場合は，そのカーソル名をプリプロセスする UAP ソース中で宣言する必要があります。

(5) 使用例

カーソル（CR1）を閉じます。

```
CLOSE CR1
```

4.2 DECLARE CURSOR（カーソルの宣言）

(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。

(2) 形式

カーソル宣言：：=DECLARE **カーソル名** CURSOR FOR **カーソル指定**

(a) カーソル名

カーソルの名称を指定します。

(3) 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 中のカーソル指定で指定した埋め込み変数の値は，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 文を実行したときの値が有効です。値を変更する場合は，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 1023 個のカーソルを宣言できます。

(4) 注意事項

1. カーソル名は，埋め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す。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。したがって，SQL 中にカーソル名を指定している場合は，そのカーソル名をプリプロセスする UAP ソース中で宣言する必要があります。
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 を実行しても，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) 使用例

(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
```

(例 2)

在庫表（ZAIKO）から，単価（TANKA）が 5000 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
```

（例 3）

カーソル（CR1）を使用して在庫表（ZAIKO）の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
OPEN CR1
FETCH CR1 INTO ＜各列を取り出す変数名＞
DELETE FROM ZAIKO
    WHERE SNAME='セーター'
CLOSE CR1
```

4.3 DELETE（行の削除）

(1) 機能

指定した探索条件を満たす行を表から削除します。

(2) 形式

DELETE**文：：＝**DELETE FROM **削除対象表** 〔WHERE **探索条件**〕
削除対象表：：＝表名

(a) 削除対象表

削除したい行を含む表の表名を指定してください。

表名については，「1.2.8　名前の修飾」を参照してください。

スキーマ名に，"MASTER" は指定できません。

スキーマ名を省略した場合については，「1.2.8　名前の修飾」を参照してください。

(b) WHERE 句

WHERE **探索条件**

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。

探索条件

削除する行を選択する条件を指定してください。
探索条件中に埋め込み変数を指定できます。
探索条件については，「2.7　探索条件」を参照してください。

(3) 共通規則

削除の対象になる行がない場合は，SQLCODE に 100 が設定されます。

(4) 使用例

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'
```

4.4 FETCH（行の取り出し）

(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ます。

(2) 形式

```
FETCH文：：=FETCH カーソル名 INTO ：埋め込み変数〔：標識変数〕
                                  〔，：埋め込み変数〔：標識変数〕〕…
```

(a) カーソル名

検索結果を取り出すカーソルのカーソル名を指定します。

(b) ：埋め込み変数

ナル値以外の列の値を読み込むための埋め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，埋め込み変数と標識変数の両方を指定します。

(c) ：標識変数

標識変数を指定します。標識変数には，埋め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ます。

(3) 共通規則

1. FETCH 文で指定するカーソルは，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 句で指定した埋め込み変数の個数は同じにしてください。列の個数と，埋め込み変数の個数が異なる場合は，SQL 連絡領域の SQLWARN3 領域に 'W'（警告フラグ）が設定されます。
3. INTO 句で指定する埋め込み変数のデータ型およびデータ長は，対応する検索項目のデータ型およびデータ長と同じにしてください。
4. 一つのカーソルに複数の FETCH 文を記述する場合，FETCH 文ごとに異なる埋め込み変数〔：標識変数〕を指定することはできません。
5. 取り出す行がない場合，次の個所にリターンコード 100 が設定されます。
   - SQL 連絡領域の SQLCODE 領域
   - SQLCODE 変数

(4) 注意事項

1. 検索結果の列の値がナル値の場合は，対応する埋め込み変数の値は保証されません。
2. カーソル名は，埋め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。したがって，SQL 中にカーソル名を指定している場合は，そのカーソル名をプリプロセスする UAP ソース中で宣言する必要があります。

(5) 使用例

カーソル（CR1）による在庫表（ZAIKO）の検索結果を，商品コード列（SCODE），商品名列（SNAME），色列（COL），単価列（TANKA），在庫量列（ZSURYO）に対応する埋め込み変数および標識変数に読み込みます。

＜埋め込み変数を指定した場合＞

```
FETCH CR1
    INTO :XSCODE,:XSNAME,:XCOL,
         :XTANKA,:XZSURYO
```

4.5 INSERT（行の挿入）

(1) 機能

表に，行を列単位，または行単位で挿入します。直接，値を指定して一つの行を挿入できます。

(2) 形式

```
INSERT文：：＝
   INSERT INTO 挿入対象表 〔｛（列名リスト）｜（ROW）｝〕
      VALUES（挿入値〔，挿入値〕…）
挿入対象表：：＝表名
列名リスト：：＝列名〔，列名〕…
挿入値：：＝値指定
```

(a) 挿入対象表

行を挿入する表の表名を指定してください。

スキーマ名に，"MASTER" は指定できません。

（列名リスト）

データを挿入する列の名称を指定してください。
指定されていない列にはナル値が挿入されます。
列名を一つも指定しない（列名リストを省略した）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，すべての列を指定したことになります。

（ROW）

行単位でデータを挿入する場合に指定します。ROW は行全体を意味し，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。ROW を指定する場合の規則を次に示します。

- 挿入値には，埋め込み変数を指定してください。
- 指定する埋め込み変数は，各列のデータ型に関係なく，ROW に対応する埋め込み変数（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。データ長の計算方法については，「3.2(3) 規則」を参照してください。

VALUES（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てください。

(b) 挿入値

挿入値：：＝値指定

値指定

値指定については，「2.10　値指定」を参照してください。

(3) 共通規則

1. 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。データ型は，データを挿入する列のデータ型または変換できるデータ型にしてください。
2. 挿入値に埋め込み変数を指定する場合，仮定されるデータ型およびデータ長は挿入対象となる列のデータ型およびデータ長になります。
3. DECIMAL 型のデータを INTEGER 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。
   また，DECIMAL 型のデータを DECIMAL 型の列に挿入する場合，列の位取りより下位のけた部分が切り捨てられます。
4. 表の定義時に指定した長さ以上の文字データは，挿入できません。
5. 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。
6. 固定長文字列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，余りの部分に半角空白が設定されます。
7. TIMESTAMP 型のデータを TIMESTAMP 型の列に挿入する場合，列の小数秒のけた数より下位のけた部分が切り捨てられます。挿入するデータの小数秒のけた数より列の小数秒のけた数が多い場合，0 を補って格納されます。

(4) 使用例

在庫表（ZAIKO）の商品コード（SCODE），商品名（SNAME），色（COL）の各列に，612S，SLACKS，WHITE のデータを挿入します。

```
INSERT INTO "ZAIKO"("SCODE","SNAME","COL")
VALUES('612S','SLACKS','WHITE')
```

4.6 OPEN（カーソルのオープン）

(1) 機能

カーソルを開きます。DECLARE CURSOR で宣言したカーソルを，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。

(2) 形式

```
OPEN文：：＝OPEN カーソル名 〔USING ：埋め込み変数
                              〔，：埋め込み変数〕…〕
```

(a) カーソル名

開くカーソルのカーソル名を指定します。

(b) USING ：埋め込み変数

DECLARE CURSOR 中のカーソル指定で指定した埋め込み変数を別の埋め込み変数に変更する場合に，変更後の埋め込み変数を指定します。

DECLARE CURSOR 中の SELECT 文で指定した埋め込み変数の値は，カーソルを閉じるまで有効になります。したがって，カーソルのオープン中に埋め込み変数の値を更新しても，オープン中のカーソルの問合せ指定に影響を与えません。埋め込み変数の値を変更する場合は，一度カーソルを閉じてから，再度カーソルを開いてください。

USING 句で指定する埋め込み変数の数は，カーソル宣言で指定した埋め込み変数の数と一致しなければなりません。

USING 句で指定する埋め込み変数は，指定した順序に，カーソル宣言中の SELECT 文で指定した埋め込み変数と置き換えられます。

置き換えられる埋め込み変数のデータ型およびデータ長は完全に一致しなければなりません。

(3) 注意事項

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. 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。
3. USING 句で埋め込み変数を指定する場合は，埋め込み変数に値を設定してから OPEN 文を実行してください。
4. カーソル名は，埋め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す。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。

5. カーソルを開くときに使用する埋め込み変数と標識変数を次の表に示します。

表 4-1 カーソルを開くときに使用する埋め込み変数と標識変数

項番	DECLARE CURSOR のカーソル指定		OPEN 文の USING 句	カーソルを開くときに使用する埋め込み変数，標識変数
	埋め込み変数	標識変数	埋め込み変数	
1	指定あり	指定あり	指定なし	DECLARE CURSOR で指定した埋め込み変数，標識変数を使用します。
2			指定あり	OPEN 文で指定した埋め込み変数を使用します。標識変数は使用しません。
3		指定なし	指定なし	DECLARE CURSOR で指定した埋め込み変数を使用します。標識変数は使用しません。
4			指定あり	OPEN 文で指定した埋め込み変数を使用します。標識変数は使用しません。
5	指定なし	－	－	埋め込み変数，標識変数は使用しません。

（凡例）

－：該当しません。

（4） 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
```

4.7 SELECT（表の検索）

(1) 機能

一つの表からデータを検索します。

(2) 形式

SELECT**文：：＝カーソル指定**

(3) 共通規則

「2.1　カーソル指定」を参照してください。

(4) 使用例

「2.1(4) 指定例」を参照してください。

4.8 1行 SELECT（表の1行検索）

(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，問合せ指定とオペランドは同じですが，問合せ指定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を指定する INTO 句を含んでいます。

(2) 形式

```
1行SELECT文：：＝SELECT 選択リスト
   INTO ：埋め込み変数〔：標識変数〕
   〔，：埋め込み変数〔：標識変数〕〕…
   表式
```

選択リストについては「2.2　問合せ指定」を，表式については「2.3　表式」をそれぞれ参照してください。

(a) INTO 句

SELECT 文を単独で UAP に直接記述する場合は，INTO 句を必ず指定してください。ただし，カーソル宣言中の SELECT 文には指定できません。

(b) ：埋め込み変数

一つの行の列の値を読み込む埋め込み変数を指定します。

(c) ：標識変数

埋め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。

(3) 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 句で指定した埋め込み変数の個数は同じにしてください。列の個数と，埋め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 'W'（警告フラグ）が設定されます。また，列の個数より埋め込み変数の個数が多い場合，列に対応していない埋め込み変数の値は保証されません。

- INTO 句で指定する埋め込み変数のデータ型およびデータ長は，対応する列のデータ型およびデータ長にしてください。
- 検索結果の列の値がナル値の場合，対応する埋め込み変数の値は保証されません。
- 検索結果の列の値がナル値の場合は，埋め込み変数に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，次の個所にリターンコード 100 が設定されます。
   - SQL 連絡領域の SQLCODE 領域
   - SQLCODE 変数
3. 「2.1　カーソル指定」の説明に従って，ORDER BY 句および LIMIT 句を指定できます。

(4) 使用例

使用例については，「4.2　DECLARE CURSOR（カーソルの宣言）」を参照してください。

4.9 UPDATE（行の更新）

(1) 機能

指定した探索条件を満たす表内の行の，指定した列の値を更新します。

(2) 形式

```
UPDATE文：：＝
  UPDATE 更新対象表
            SET SET句〔，SET句〕…
             〔WHERE 探索条件〕
更新対象表：：＝表名
SET句：：＝ {更新対象列＝更新値｜ROW＝行更新値}
更新対象列：：＝列名
更新値：：＝値指定
行更新値：：＝：埋め込み変数〔：標識変数〕
```

(a) 更新対象表

更新する表の表名を指定します。

スキーマ名に "MASTER" は指定できません。

(b) 更新対象列

更新対象列：：＝列名

列名

更新する列の名称を指定します。

(c) 更新値

更新値：：＝値指定

値指定

値指定については，「2.10　値指定」を参照してください。

(d) ROW

行単位でデータを更新する場合に指定します。ROW は行全体を意味し，これを指定すると，行全体を一つのデータと見なされます。ROW を指定する場合の規則を次に示します。

- 更新するデータに指定する埋め込み変数は，各列のデータ型に関係なく，ROW に対応する埋め込み変数（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。データ長の計算方法については，「3.2(3) 規則」を参照してください。

(3) 共通規則

1. 更新値は，更新対象列と変換または比較できるデータ型にしてください。
2. 更新値に埋め込み変数を指定する場合，仮定されるデータ型およびデータ長は更新対象列のデータ型およびデータ長になります。
3. DECIMAL 型のデータで INTEGER 型の列を更新する場合，端数（小数）部分は切り捨てられます。
   また，DECIMAL 型のデータで DECIMAL 型の列を更新する場合，列の位取りより下位のけた部分が切り捨てられます。
4. 更新対象列の値として，表の定義時に指定した長さ以上の文字データは入力できません。
5. 数データを更新する場合，範囲外の数値を入力することはできません。
6. 固定長文字列の列を更新するデータが列の長さより短い場合は，左詰めにされて，余りの部分に半角空白が格納されます。
7. TIMESTAMP 型のデータで TIMESTAMP 型の列を更新する場合，列の小数秒のけた数より下位のけた部分が切り捨てられます。更新するデータの小数秒のけた数より列の小数秒のけた数が多い場合，0 を補って格納されます。
8. SET 句は，最大 3000 個指定できます。
9. 更新の対象となる行がない場合は，SQLCODE に 100 が設定されます。
10. 同じ列名を更新対象列に重複して指定できません。また ROW を指定する場合は，SET 句を二つ以上指定できません。

(4) 使用例

在庫表（ZAIKO）の商品コード列（SCODE）が 302S の商品の在庫量列（ZSURYO）を 100 に変更します。

```
UPDATE "ZAIKO"
SET "ZSURYO" = 100
WHERE "SCODE" = '302S'
```

5 埋め込み言語文法

埋め込み言語は，埋め込み変数の宣言およびリターンコードによる処理の宣言をする SQL です。埋め込み型の UAP を作成する場合に操作系 SQL と一緒に使用します。
この章では，埋め込み言語の機能，形式および規則について説明します。

5.1 BEGIN DECLARE SECTION（埋め込みSQLの開始宣言）

(1) 機能

埋め込みSQL宣言節の始まりを示します。埋め込みSQL宣言節には，SQL中で使用する埋め込み変数および標識変数を指定します。

(2) 形式

```
BEGIN DECLARE SECTION
```

(3) 共通規則

1. 埋め込みSQL宣言節の終わりには，END DECLARE SECTION（埋め込みSQL終了宣言）を指定してください。
2. SQL中で使用する埋め込み変数および標識変数は，埋め込みSQL宣言節で宣言します。
3. 埋め込み型のUAPには，0個以上の埋め込みSQL宣言節を指定できます。
4. 埋め込みSQL宣言節には，変数の宣言だけを指定できます。ただし，変数の宣言を含まない埋め込みSQL宣言節も指定できます。

(4) 使用例

SQL中で使用する埋め込み変数を宣言します。

```
    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.
```

5.2 COPY（登録集原文の引き込み）

(1) 機能

登録集原文をソースプログラム中に引き込みます。

(2) 形式

```
COPY 原文名
```

(a) 原文名

登録集原文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。

(3) 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 終了子のあとには，行末まで何も記述しないでください。
2. 登録集原文が登録されているディレクトリは，次に示す順位で検索されます。また，ディレクトリ内のファイルの検索優先順位も次のようになっています。

   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 の場合は EEXDBCBLLIB）
   - カレントディレクトリ

   ファイルの検索順位
   - 環境変数 EEXDBCBLFIX 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

3. COPY 文で引き込んだ登録集原文中に COPY 文を指定することはできません。
4. 登録集原文は固定形式正書法で記述してください。

(4) 注意事項

COBOL 言語の COPY 文および INCLUDE 文で，埋め込み変数，または SQL の含まれた登録集原文を引き込まないでください。

(5) 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。

```
EXEC SQL
  COPY SAMPLE
END-EXEC.
```

5.3 END DECLARE SECTION（埋め込み SQL の終了宣言）

（1） 機能

埋め込み 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. 埋め込み SQL 宣言節の始まりには，BEGIN DECLARE SECTION（埋め込み SQL 開始宣言）を指定してください。
2. SQL 中で使用する埋め込み変数および標識変数は，埋め込み SQL 宣言節で宣言します。

（4） 使用例

使用例については「5.1　BEGIN DECLARE SECTION（埋め込み SQL の開始宣言）」を参照してください。

5.4　END-EXEC（SQL 終了子）

（1） 機能

SQL の終わりを示します。

（2） 形式

```
END-EXEC
```

（3） 共通規則

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。

（4） 使用例

使用例については，「5.5　EXEC SQL（SQL 先頭子）」を参照してください。

5.5 EXEC SQL（SQL 先頭子）

(1) 機能

SQL の始まりを示します。

(2) 形式

```
EXEC SQL
```

(3) 共通規則

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。

(4) 使用例

OPEN 文を記述します。

```
EXEC SQL
  OPEN CR1
END-EXEC.
```

5.6　SQLCODE 変数

SQL を実行すると XDB 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，COBOL 言語では S9(18) COMPUTATIONAL で，SQLCA の基本項目として宣言しています。

SQLCODE 変数に設定された内容を参照する場合は，変数名称 SQLCODE を指定してください。

5.7 WHENEVER（埋め込み例外宣言）

（1） 機能

SQL の実行後，XDB 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
       ｜PERFORM〔：〕ホスト識別子}
```

(a) {SQLERROR ｜ SQLWARNING ｜ NOT FOUND}

SQLERROR

ユーザの誤りや XDB 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 領域および 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。

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，または SQL 連絡領域の SQLCODE 領域および 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。

NOT FOUND

表の検索で，検索する行がなくなったとき（SQL 連絡領域の SQLCODE 領域と，SQLCODE 変数に 100 が返されたとき）の処理を指示する場合に指定します。

(b) {CONTINUE ｜ {GO TO ｜ GOTO}〔：〕ホスト識別子｜ PERFORM〔：〕ホスト識別子}

CONTINUE

UAP の実行を続行させる場合に指定します。

{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。

- 節名，または段落名

PERFORM〔：〕ホスト識別子

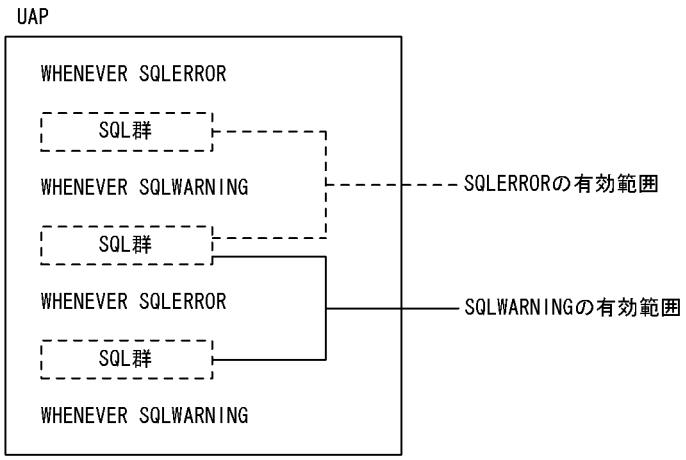
指定した手続きを実行させる場合，次に示すホスト識別子によって実行させる手続きを指定します。

- 節名，または段落名

(3) 共通規則

1. 埋め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. 埋め込み例外宣言は，同じ UAP 中に複数個指定できます。
3. SQLERROR に該当する状態で，かつ，SQL 連絡領域の SQLWARN6 が 'W' である場合，SQL は実行できません。
4. 手続き実行後の制御は，特異状態（SQLERROR など）が発生した SQL の次の命令へ戻ります。
5. 埋め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。つまり，ある一つの埋め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め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。
   埋め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 5-1 埋め込み例外宣言で指定した処理の有効範囲

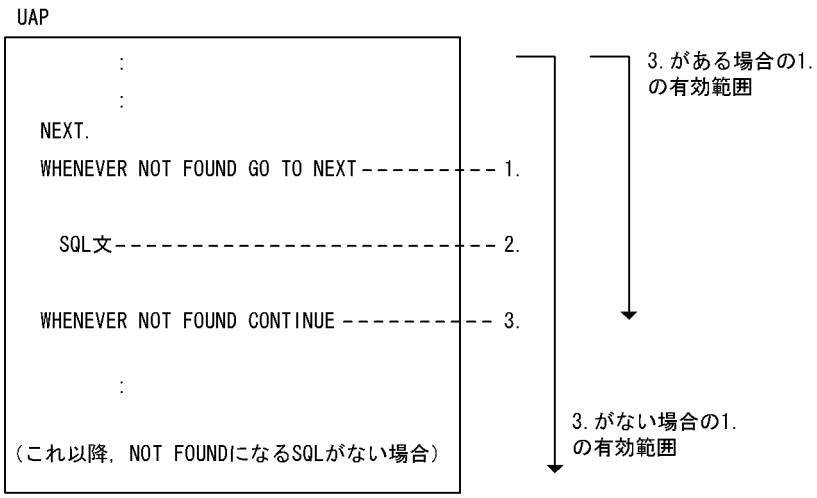


6. SQLERROR，SQLWARNING，NOT FOUND の有効範囲が重複した場合は，次に示す優先順位で SQL が処理されます。
   SQLERROR→NOT FOUND→SQLWARNING

(4) 注意事項

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 5-2 WHENEVER 文の記述例 1

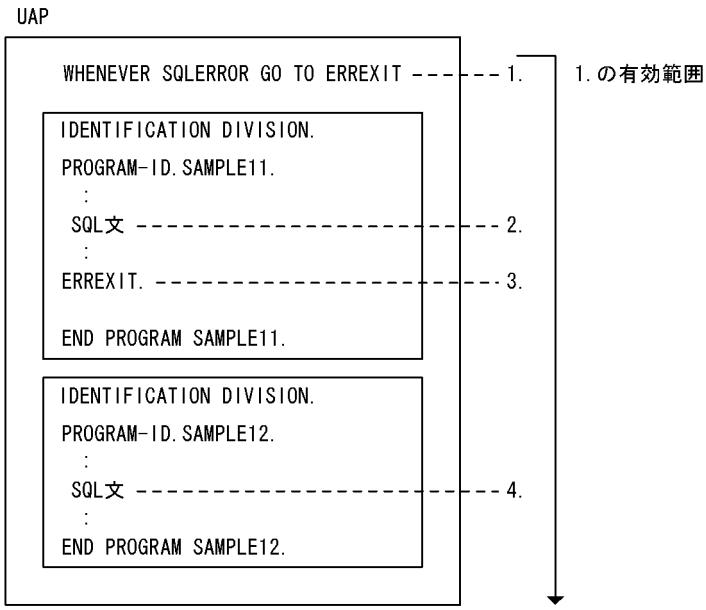


説明

3. 以降で NOT FOUND になる SQL 文がない場合は，NOT FOUND になったときの処理が不要になるため，3. の WHENEVER 文には CONTINUE を指定します。

2. 埋め込み例外宣言は，UAP 中に記述している複数の関数にわたって有効になります。
WHENEVER 文で指定した GOTO 文の分岐先は，SQL 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 文の記述例を図 5-3 および図 5-4 に示します。

図 5-3　WHENEVER 文の記述例 2

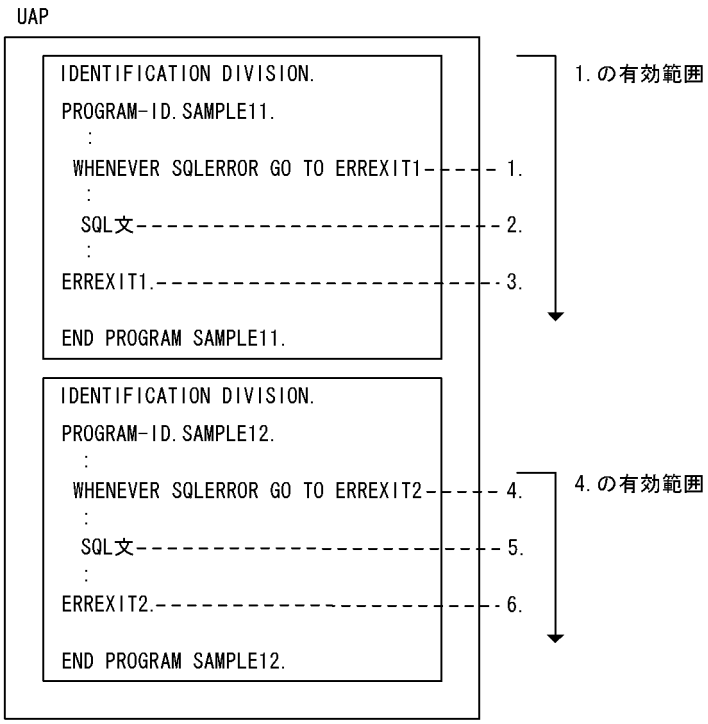


説明

2. で SQLERROR が発生すると，1. で指定した分岐先 3. は 2. と同じ内部プログラム内にあるため，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. と同じ内部プログラム内にないため，コンパイルエラーとなります。

図 5-4 WHENEVER 文の記述例 3



説明

2. で SQLERROR が発生したときの分岐先 3. と，5. で SQLERROR が発生したときの分岐先 6. は，それぞれ SQL 文で同じ内部プログラムのため，分岐できます。

(5) 使用例

埋め込み例外宣言の使用例を次に示します。

(a) コーディング

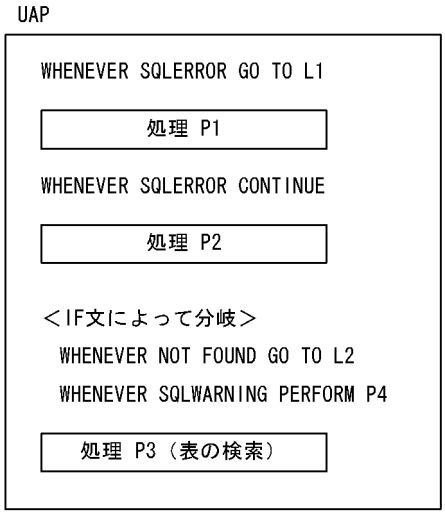
```
EXEC SQL
  WHENEVER SQLERROR GO TO L1
END-EXEC.
```

(b) エラー判定と動作の制御

- 処理 P1 中の SQL を実行したあと，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 を実行したあと，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。

WHENEVER 文の記述例を次の図に示します。

図 5-5　WHENEVER **文の記述例** 4



6 SQL の記述テクニック

この章では，性能が向上する SQL の書き方や，検索時に使用するインデクスを変更する SQL の書き方について説明します。

6.1 性能が向上する SQL の書き方

ここでは，性能が向上する SQL の書き方について説明します。

6.1.1 OR 条件中の共通部分を外側にくくり出す

OR 条件を指定した場合，検索時にインデクスが使用されません。インデクスを使用するには，SQL を次のように書き換えてください。

■共通項のくくり出し

OR 条件中の共通部分を OR 条件の外側にくくり出すと，検索時にインデクスが使用されます。

書き換え前の SQL 例

```
SELECT * FROM "T1"
  WHERE ("C1" = 1 AND "C2" = 'A') OR ("C1" = 1 AND "C3" = 10)
```

列 C1 にインデクスが定義されていても，インデクスが使用されません。

書き換え後の SQL 例

```
SELECT * FROM "T1" WHERE "C1" = 1 AND ("C2" = 'A' OR "C3" = 10)
```

列 C1 にインデクスが定義されている場合，インデクスが使用されます。

6.1.2 同じ SQL を記述する

同じ SQL を実行する回数が多いほど，SQL プール機能によって SQL の前処理が省略できる回数が多くなるため，性能が向上します。

SQL プール機能については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。

(1) XDB が行う SQL の比較判定処理

SQL を受け付けたときに，その SQL が以前実行された SQL と同じかどうかを XDB が比較します。比較した結果，SQL のテキストが一致した場合，同じ SQL と判定し，SQL プールに保存されている前処理結果を再利用します。

XDB は，SQL のテキストを比較して SQL が同じかどうかを判定しています。SQL の構文が同じで，テキストの長さ，空白の位置や数などがすべて一致した場合に同じ SQL と判定しています。

同じ SQL と判定されるケースと，異なる SQL と判定されるケースの例を示します。なお，例題中の△は一つの空白を意味しています。

（例 1）同じ SQL と判定されるケース

```
SELECT△*△FROM△"T1"
SELECT△*△FROM△"T1"
```

（例 2）異なる SQL と判定されるケース

```
SELECT△*△FROM△"T1"
SELECT△△*△FROM△"T1"
```

空白の数が異なるため，異なる SQL と判定されます。

（2）SQL のコーディングポイント

XDB が SQL を比較する際，SQL のテキストの一部は比較対象とされません。比較対象されない部分をうまく利用して，同じ SQL と判定されるように SQL をコーディングすることをお勧めします。SQL のテキストの比較対象外部分を次の表に示します。

表 6-1　SQL のテキストの比較対象外部分

項番	SQL 文の種類	比較対象外の部分
1	SQL 全般	埋め込み変数，標識変数
2	DECLARE CURSOR	DECLARE カーソル名 CURSOR FOR
3	1 行 SELECT 文	INTO ：埋め込み変数〔：標識変数〕〔,：埋め込み変数〔：標識変数〕〕…

同じ SQL と判定されるケースと，異なる SQL と判定されるケースの例を次に示します。

（a）カーソルの例

カーソル名が異なっていても同じ SQL と判定されます。例を次に示します。

（例）同じ SQL と判定されるケース

```
DECLARE△CR1△CURSOR△FOR△SELECT△*△FROM△"T1"
DECLARE△CR2△CURSOR△FOR△SELECT△*△FROM△"T1"
```

△は一つの空白を意味しています。

（b）定数と埋め込み変数の例

探索条件に定数を指定している場合，その定数が SQL によって異なると，異なる SQL と判定されます。この場合，探索条件に埋め込み変数を使用することで，同じ SQL と判定されます。例を次に示します。

（例 1）異なる SQL と判定されるケース

```
SELECT * FROM "T1" WHERE "C1"=1
SELECT * FROM "T1" WHERE "C1"=2
```

この場合，定数部分の SQL 文のテキストが異なるため，異なる SQL と判定され，SQL プール機能が適用されません。

（例 2）同じ SQL と判定されるケース

```
X1=1;
SELECT * FROM "T1" WHERE "C1"=:X1
X2=2;
SELECT * FROM "T1" WHERE "C1"=:X2
```

埋め込み変数は比較対象外のため，同じ SQL と判定され，SQL プール機能が適用されます。

6.2 検索時に使用するインデクスを変更する方法

検索時に使用するインデクスを変更することで，性能を向上できることがあります。

ここでは，インデクスの定義を変更しないで，検索時に使用するインデクスを変更する方法について説明します。

(1) 条件の指定順序を変更する

WHERE 句の述語の指定順序を変更すると，検索時に使用するインデクスを変更できます。

例えば，列 C1 と列 C2 に，それぞれインデクス T1IX1 とインデクス T1IX2 が定義されている場合，次の SQL ではインデクス T1IX1 を使用します。

- **インデクス** T1IX1 **を使用する** SQL **の例**

```
SELECT * FROM "T1" WHERE "C1">1 AND "C2">'B'
```

インデクス T1IX2 を使用するには，SQL を次のように書き換えます。

- **インデクス** T1IX2 **を使用する** SQL **の例**

```
SELECT * FROM "T1" WHERE "C2">'B' AND "C1">1
```

(2) 述語の種類を変更する

WHERE 句に指定する述語の種類を変更すると，検索時に使用するインデクスを変更できます。

例えば，列 C1 と列 C2 に，それぞれインデクス T1IX1 とインデクス T1IX2 が定義されている場合，次の SQL ではインデクス T1IX1 を使用します。

- **インデクス** T1IX1 **を使用する** SQL **例**

```
SELECT * FROM "T1" WHERE "C1"=1 AND "C2"='B'
```

インデクス T1IX2 を使用するには，SQL を次のように書き換えます。

- **インデクス** T1IX2 **を使用する** SQL **例**

```
SELECT * FROM "T1" WHERE "C1" BETWEEN 1 AND 1 AND "C2"='B'
```

7 SQL プリプロセサ（eexdbcbl）

この章では，SQL プリプロセサ（eexdbcbl）について説明します。

7.1 概要

ここでは，SQL プリプロセサ（eexdbcbl）の概要について説明します。

7.1.1 機能

SQL プリプロセサ（eexdbcbl）は，SQL を埋め込んだ UAP ソースを，言語コンパイラで変換できるポストソースに変換して，ファイルに出力するユティリティです。SQL プリプロセサ（eexdbcbl）を実行して，UAP ソースをポストソースに変換することをプリプロセスといいます。

SQL プリプロセサ（eexdbcbl）は，UAP ソースを入力し，SQL を XDB の呼び出し手続きとして展開します。

SQL プリプロセサ（eexdbcbl）を実行しても，次に示す内容についてはチェックされません。SQL 実行時にチェックされます。

- 厳密な SQL の文法
- XDB に問い合わせが必要な項目

例えば，SQL に記述された表や列が存在するか，それらに対応づけられた埋め込み変数にデータ型の互換性があるかどうかなどは，SQL プリプロセサ（eexdbcbl）の実行時にはチェックされません。

7.1.2 入力ファイルおよび出力ファイルの形式

入力ファイルおよび出力ファイルの形式について説明します。

（1） 入力ファイルの形式

固定形式正書法で記述された COBOL UAP ソースを入力ファイルとしてください。自由形式正書法で記述された COBOL UAP ソースはプリプロセスできません。

なお，COBOL UAP ソースの文字集合はシフト JIS である必要があります。

（2） 出力ファイルの形式

SQL プリプロセサ（eexdbcbl）を実行すると，COBOL 言語コンパイラの入力となるポストソースが出力されます（固定形式正書法で記述されたポストソースが出力されます）。

SQL プリプロセサ（eexdbcbl）が出力するポストソースの文字集合はシフト JIS です。

7.1.3 SQL プリプロセサを実行する前の確認項目

SQL プリプロセサ（eexdbcbl）を実行する前に確認が必要な項目について説明します。

（1）使用できるコンパイラの確認

SQL プリプロセサ（eexdbcbl）を使用するための前提条件を次に示します。

- COBOL コンパイラ
  COBOL2002 Net Server Suite

（2）最大値と最小値の確認

プリプロセスする前に次の表に示す最大値と最小値を確認してください。

表 7-1　最大値と最小値

項番	項目	最小値（個）	最大値（個）
1	1 プリプロセス単位に記述できる SQL 文の数※ 1	0	4095
2	1 プリプロセス単位に宣言できるカーソルの数	0	1023
3	1SQL 文に指定できる埋め込み変数の数	0	3000
4	1SQL 文に指定できる検索項目数	0	3000
5	1 トランザクション内で同時に実行できる SQL 文の数※ 2	0	4095

注

項番 1 ～ 4 の最大値に違反した場合は，SQL プリプロセサ（eexdbcbl）を実行したときにエラーとなります。
項番 5 の最大値に違反した場合は，UAP を実行したときにエラーとなります。

注※ 1

SQL 先頭子，SQL 終了子で囲まれた SQL 文を 1 個として数えます。

注※ 2

数える対象の SQL を次の表に示します。

表 7-2　数える対象の SQL

項番	分類	SQL	対象可否
1	操作系 SQL	CLOSE※	○
2		DECLARE CURSOR	×
3		DELETE	○
4		FETCH※	○
5		INSERT	○
6		OPEN※	○
7		SELECT	○
8		1 行 SELECT	○

項番	分類	SQL	対象可否
9		UPDATE	○
10	埋め込み言語	BEGIN DECLARE SECTION	×
11		COPY	×
12		END DECLARE SECTION	×
13		SQL 先頭子	×
14		SQL 終了子	×
15		SQLCODE 変数	×
16		WHENEVER	×

（凡例）

○：数える対象とします。

×：数える対象としません。

注※

同じカーソルの OPEN，FETCH，CLOSE については，1SQL として数えてください。それぞれを数える必要はありません。

7.1.4 SQL プリプロセサを実行する前の準備

SQL プリプロセサ（eexdbcbl）を実行する前に準備しておく内容について説明します。

(1) 環境変数の設定

SQL プリプロセサ（eexdbcbl）を実行する前に，必要に応じて，環境変数を設定します。設定が必要となる環境変数を次の表に示します。

表 7-3 環境変数

項番	環境変数名	内容
1	EEXDBCBLFIX	UAP ソースファイルの規定の識別子（.ecb，.cob，および .cbl）以外のファイル識別子を使用する場合に指定します。 ファイル識別子には，ピリオドで始まる 4 文字までの文字列を設定します。 この環境変数に設定したファイル識別子は，入力ファイルにだけ使用できます。 なお，.prp はプリプロセサの中間ファイルの識別子となるため，指定できません。
2	EEXDBCBLLIB	SQL の COPY 文で UAP ソースファイルに引き込まれる登録集原文を検索するディレクトリを指定します。 複数のディレクトリを指定する場合，ディレクトリ同士をコロン（:）で区切ります。 この環境変数を省略すると，カレントディレクトリだけが検索されます。

(a) 環境変数の指定例

■ bash（バッシュ）または sh（ボーンシェル）で環境設定をする場合

bash（バッシュ）または sh（ボーンシェル）で環境設定をする場合の指定例を次に示します。

```
$ EEXDBCBLFIX=".Cob"                                          …1.
$ EEXDBCBLLIB="$HOME/cobol/include:$HOME/cobol/source"        …2.
$ export EEXDBCBLFIX EEXDBCBLLIB                              …3.
```

1. UAP ソースファイルの識別子として，.Cob を有効にします。
   識別子に空白が含まれる場合は，識別子を二重引用符（"）で囲んでください。
2. 登録集原文を検索するディレクトリを指定します。
3. 環境変数をエクスポートします。

■ csh（C シェル）で環境設定をする場合

csh（C シェル）で環境設定をする場合の指定例を次に示します。

```
% setenv   EEXDBCBLFIX ".Cob"                                      …1.
% setenv   EEXDBCBLLIB "$HOME/cobol/include:$HOME/cobol/source"    …2.
```

1. UAP ソースファイルの識別子として，.Cob を有効にします。
   識別子に空白が含まれる場合は，識別子を二重引用符（"）で囲んでください。
2. 登録集原文を検索するディレクトリを指定します。

(b) 環境変数およびオプションの組み合わせ

SQL プリプロセサ（eexdbcbl）を起動するコマンドのオプション，および環境変数の指定の組み合わせを次の表に示します。オプションについては，「7.2(2) オプションおよび引数」を参照してください。

表 7-4　指定できるオプションおよび環境変数の組み合わせ

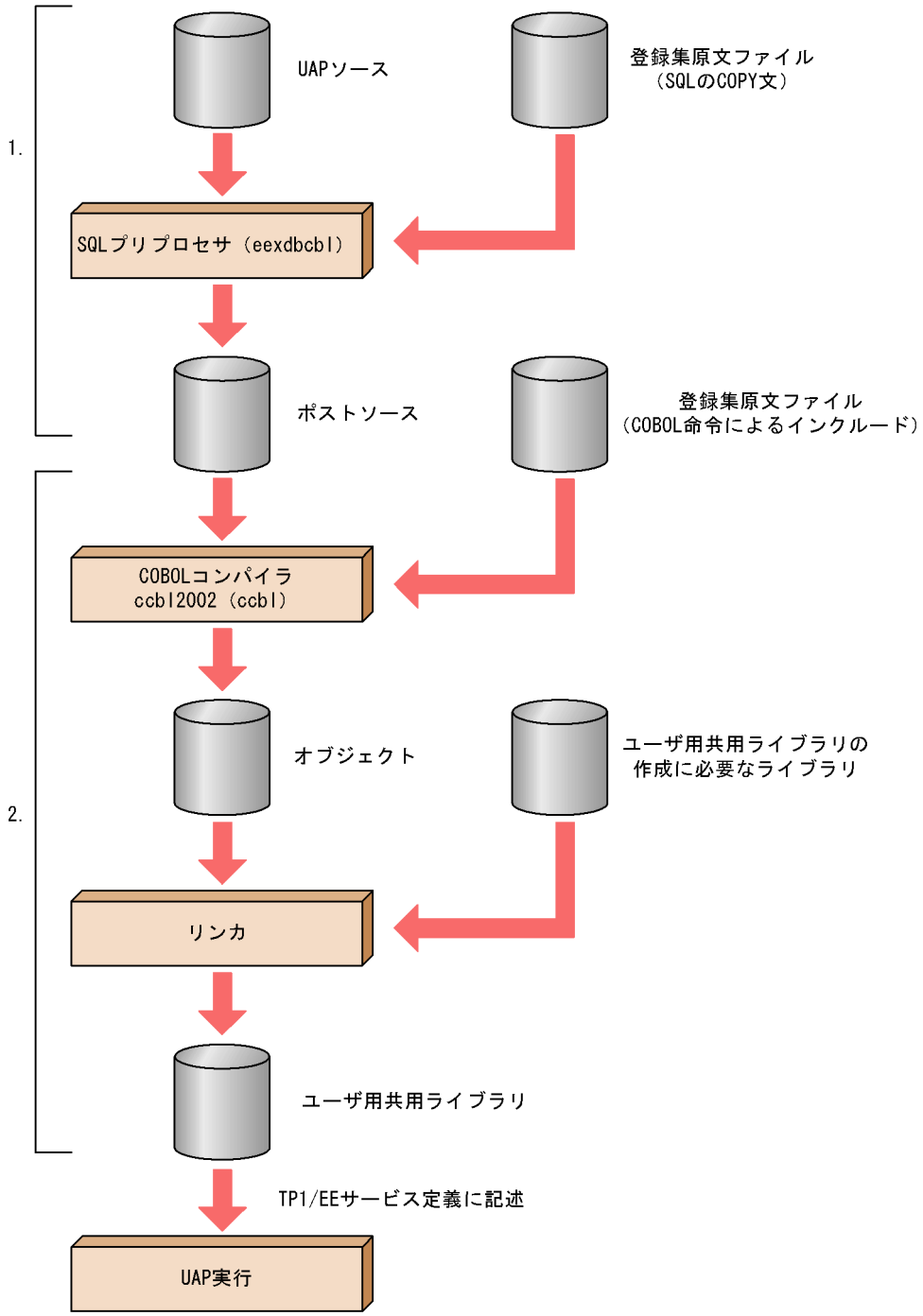
オプション		環境変数		オプション	
		EEXDBCBLFIX	EEXDBCBLLIB	-o	-Xc
環境変数	EEXDBCBLFIX	―	○	○	○
	EEXDBCBLLIB	○	―	○	○
オプション	-o	○	○	―	○
	-Xc	○	○	○	―

（凡例）
○：指定できます。
―：該当しません。

7.1.5 UAP 実行までの手順

COBOL 言語で作成した UAP を実行するまでの手順を次の図に示します。

図 7-1　COBOL 言語で作成した UAP を実行するまでの手順



1. SQL を埋め込んだ COBOL 言語の UAP ソースを，SQL プリプロセサ（eexdbcbl）でプリプロセスします。

2. 1. で得たポストソースを専用の言語コンパイラでコンパイル，およびリンケージをすることで，ロードモジュール（実行可能ファイル）が生成されます。
   コンパイル，リンケージの実行については使用する COBOL のコンパイラのマニュアルを参照してください。
   コンパイル時には，TP1/EE のインクルードディレクトリを，COBOL のコンパイラが登録集原文を検索するディレクトリに追加してください。また，コンパイル時には，次に示すコンパイラオプションを指定してください。
   ・DynamicLink,Call
   ・MainNotCBL
   ・MinusZero
   ・MultiThread

7.1.6 注意事項

SQL プリプロセサ（eexdbcbl）に関する注意事項を次に示します。

- SQL プリプロセサ（eexdbcbl）が中間ファイル中で使用するコメントは UAP で使用できません。UAP で使用できないコメントを次に示します。
  - PSTSRC-DCLDVLP-S
  - PSTSRC-DCLDVLP-E
  - PSTSRC-SQLPROCDVLP-S
  - PSTSRC-SQLPROCDVLP-E
  - PSTSRC-IDENTIFICATION
- 1 プリプロセス単位（UAP ソースファイル）中には，XDB を操作する SQL と，そのほかの DBMS を操作する SQL を混在できません。
- カーソルオープン中に制御を呼び出し元に戻す UAP を COBOL 言語で作成する場合，UAP を初期化属性プログラムとすることはできません。UAP ではデータ項目を明示的に初期化してください。

7.2　コマンドの形式

ここでは，SQL プリプロセサ（eexdbcbl）を起動するコマンドの形式について説明します。

（1）形式

```
eexdbcbl 〔-o 出力ファイル名〕
         〔-Xc〕
         入力ファイル名
```

（2）オプションおよび引数

● -o 出力ファイル名

～〈パス名〉((1 ～ 510 バイト))

SQL プリプロセサ（eexdbcbl）が出力するポストソースのファイル名を変更する場合に指定します。

このオプションを省略すると，入力ファイルのファイル識別子を .cbl に変更したものが出力ファイル名となります。

入力ファイルの識別子が .cbl の場合，必ずこのオプションを指定し，出力ファイル名の識別子を .cbl 以外に変更してください。ただし，.prp は SQL プリプロセサ（eexdbcbl）が出力する中間ファイル※の識別子となるため，指定できません。指定するとエラーとなります。

注※

SQL プリプロセサ（eexdbcbl）はプリプロセスする際，ポストソースと同じディレクトリ下に中間ファイルを出力します。この中間ファイルはプリプロセスの終了時に SQL プリプロセサ（eexdbcbl）によって削除されます。

ただし，中間ファイルの削除処理中に SQL プリプロセサ（eexdbcbl）が異常終了した場合，中間ファイルは削除されません。この場合，出力されたメッセージに従って中間ファイルを削除してください。

このオプションでは，ファイル名だけの指定，絶対パス指定，および相対パス指定ができます。ポストソースの出力先を次の表に示します。

表 7-5　ポストソースの出力先

項番	-o オプションの指定		ポストソースの出力先
1	あり	ファイル名だけ指定	カレントディレクトリ
2		絶対パス指定	指定ディレクトリ
3		相対パス指定	指定ディレクトリ
4	なし		入力ファイルと同じディレクトリ

● -Xc

SQL プリプロセサ（eexdbcbl）が生成する文字列の囲みを二重引用符（"）にする場合に指定します。
このオプションを省略すると，文字列の囲みはアポストロフィ（'）となります。

●入力ファイル名

～〈パス名〉((1 ～ 510 バイト))

UAP ソースファイル名を指定します。
ファイル識別子には，.ecb，.cob，または .cbl を指定します。
環境設定でファイル識別子を登録してある場合は，その識別子を使用できます。ただし，.prp は SQL プリプロセサ（eexdbcbl）が出力する中間ファイル※の識別子となるため，指定できません。

注※
-o オプションの注※の説明を参照してください。

入力ファイル名と出力ファイル名が同じ場合は，異なる識別子を指定してください。
入力ファイル名と出力ファイル名が同じ場合，エラーとなります。
また，パス指定の方法（絶対パス，相対パス，または入力ファイル名だけの指定）が異なっていても，格納先ディレクトリが同じになる場合，入力ファイルは出力ファイルに上書きされます。
入力ファイル名は，ファイル名だけの指定，絶対パス指定，および相対パス指定ができます。入力ファイルの指定方法と対象となるディレクトリを次の表に示します。

表 7-6　入力ファイルの指定方法と対象となるディレクトリ

項番	入力ファイルの指定方法	対象となるディレクトリ
1	入力ファイル名だけ指定	カレントディレクトリ
2	絶対パス指定	指定ディレクトリ
3	相対パス指定	指定ディレクトリ

（3）オプションの指定例

UAP ソースファイル名が sample で，出力するポストソースのファイル名を main にする場合の指定例を次に示します。

```
eexdbcbl -o main.cbl sample.ecb
```

UAP ソースファイル名が sample で，SQL プリプロセサ（eexdbcbl）が生成する文字列の囲みを二重引用符（"）にする場合の指定例を次に示します。

```
eexdbcbl -Xc sample.ecb
```

（4）リターンコード

SQL プリプロセサ（eexdbcbl）のリターンコードを次に示します。

0：正常終了

8：エラー発生
　プリプロセスを途中で終了します。ただし，プリプロセスを最後まで続行できる場合は，最後まで処理を続行します。

16：エラー発生
　プリプロセスを途中で終了します。

（5）エラーの出力

SQL プリプロセサ（eexdbcbl）は，SQL 文の文法に誤りがある場合，誤りのある SQL 文を無視して処理を続行します。オプションの指定に誤りがある場合は，処理を中断します。また，メモリ不足やファイル入出力エラーなどのエラーが発生し，それ以降の処理ができない場合は，処理の途中でも終了します。

SQL 文の文法に誤りがある場合，SQL プリプロセサ（eexdbcbl）はエラーメッセージを標準エラー出力へ出力します。エラーメッセージを参照すると，エラー内容，UAP ソースファイル名，エラーが発生した個所（SQL 文の行番号）などがわかります。

付録

付録 A　サンプル UAP

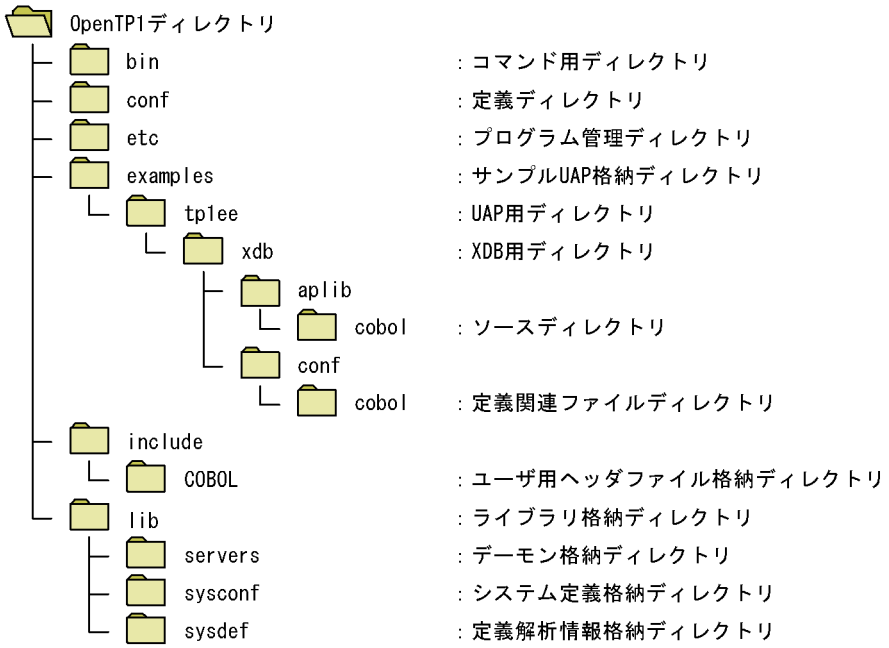
付録 A　サンプル UAP

ここでは，XDB が提供するサンプル UAP について説明します。

付録 A.1　サンプル UAP のディレクトリ構成

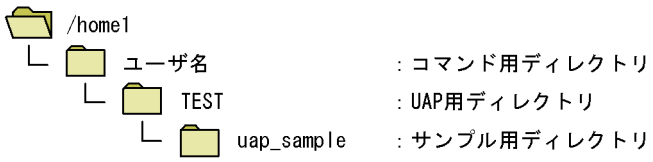
サンプル UAP が格納されているディレクトリの構成を次の図に示します。

図 A-1　サンプル UAP のディレクトリ構成



作業ディレクトリを作成したときの構成を次の図に示します。

図 A-2　サンプル UAP の作業ディレクトリの構成



説明

- コマンド用ディレクトリ
  コマンドを実行するために，ユーザが作成するディレクトリです。
- UAP 用ディレクトリ
  UAP 用のディレクトリです。

- サンプル用ディレクトリ
  サンプルを実行するためのディレクトリです。

付録 A.2　環境変数の設定

次に示す環境変数を .bashrc ファイルに指定する必要があります。

LD_LIBRARY_PATH
　共用ライブラリが格納されているパスを指定します。

DCCONFPATH
　OpenTP1 の定義ファイルを格納するディレクトリを絶対パス名で指定します。

DCDIR
　OpenTP1 ディレクトリを絶対パス名で指定します。

PATH
　プリプロセスを行うコマンドが格納されているパスを指定します。

環境変数の指定例を次に示します。

■ bash（バッシュ）または sh（ボーンシェル）で環境設定をする場合

```
export ID=$LOGNAME
export LD_LIBRARY_PATH=OpenTP1ホームディレクトリ/lib:/opt/HILNGcbl2k/lib:$DCDIR/
lib:$LD_LIBRARY_PATH
export PATH=OpenTP1ホームディレクトリ/bin:/opt/HILNGcbl2k/bin:$PATH
export DCCONFPATH=OpenTP1のシステム定義ファイルを格納するディレクトリ
export DCDIR=OpenTP1ホームディレクトリ
export VAR_PATH=COBOLコンパイル実行に必要なパスを指定します
export VAR_LDLIB=COBOLコンパイルおよびプリプロセスコマンド実行に必要なライブラリパスを指
定します
export VAR_CBLLIB=COBOL登録集原文が保管されているパスを指定します
```

■ csh（C シェル）で環境設定をする場合

```
setenv ID $LOGNAME
setenv LD_LIBRARY_PATH “ OpenTP1ホームディレクトリ/lib:/opt/HILNGcbl2k/lib:
$DCDIR/lib:$LD_LIBRARY_PATH ”
setenv PATH “ OpenTP1ホームディレクトリ/bin:/opt/HILNGcbl2k/bin:$PATH ”
setenv DCCONFPATH “ OpenTP1のシステム定義ファイルを格納するディレクトリ ”
setenv DCDIR “ OpenTP1ホームディレクトリ ”
setenv VAR_PATH “ COBOLコンパイル実行に必要なパスを指定します ”
setenv VAR_LDLIB “ COBOLコンパイルおよびプリプロセスコマンド実行に必要なライブラリパスを
指定します ”
setenv VAR_CBLLIB “ COBOL登録集原文が保管されているパスを指定します ”
```

付録 A.3　コンパイル

ソースファイルが格納されているディレクトリで，make コマンドを実行することでコンパイルが行われます。

ユーザがコンパイル用に用意したディレクトリでコンパイルすることを推奨します。

コンパイルを実行すると生成されるファイルを次の表に示します。

表 A-1　コンパイルを実行すると生成されるファイル

項番	ファイル名	説明
1	sup.o	SUP オブジェクトファイル
2	SUP	SUP ロードモジュールファイル
3	sampleuap.cbl	COBOL ソースファイル
4	libxdb_sampleuap.so	SPP 共用ライブラリファイル
5	sampleuap.o	SPP オブジェクトファイル

付録 A.4　サンプル UAP の実行手順

サンプル UAP の実行手順を次に示します。

1. OpenTP1 を開始します。
   ```
   dcstart
   ```
2. TP1/EE を開始します。
   ```
   eesvstart -u SMPLSV1
   ```
3. ユーザサーバを起動します。
   ```
   dcsvstart -u SMPLSV0
   ```
   ユーザサービス構成定義の dcsvstart 定義コマンドに，開始するユーザサーバ名を指定します。
   dcsvstart 定義コマンドを実行する前に，dcsetup コマンドでサービス定義の設定を行う必要があります。
   なお，実行結果は標準出力に出力されます。

付録 A.5　makefile およびコンパイル用シェルの内容

make コマンドを実行するには，makefile のほかに ccbl2002 コンパイル用シェル（prp_ccbl2002.sh）が必要です。

makefile および ccbl2002 コンパイル用シェル（prp_ccbl2002.sh）の内容を次に示します。

（1） makefile の内容

makefile の内容を次に示します。

```
#All Rights Reserved. Copyright (C) 2008, Hitachi, Ltd.
#Licensed Material of Hitachi, Ltd.
#Reproduction, use, modification or disclosure otherwise than
#permitted in the License Agreement is strictly prohibited.
#makefile
#makefile for making UAP(COBOL Language)
#

CBLDIR = /opt/HILNGcbl2k                    …[1]始まり
CPDIR = $(DCDIR)/aplib

IDIRS = -I$(DCDIR)/include
LDIRS = -L$(DCDIR)/lib -L$(CBLDIR)/lib      …[1]終わり

### -------- Normal Option --------
CBLFLAGS1 = -C2 -Mw
CBLFLAGS2 = -C2
CFLAGS = -c -ansi -O1 $(IDIRS)              …[2]
LFLAGS = $(LDIRS) -Wl,-Bdynamic -lbetran -lcbl2k -lcbl2kml -ltactk -lm -ldl

CC= /usr/bin/gcc
CCBL= $(CBLDIR)/bin/ccbl


OBJECT1= sup.o
all : SUP libxdb_sampleuap.so

#make load module
libxdb_sampleuap.so : sampleuap.o
gcc -o libxdb_sampleuap.so  sampleuap.o -shared -L /opt/HILNGcbl2k/lib -lcbl2k
-lcbl2kml -lpthread  -lm

sampleuap.o : sampleuap.ecb
        ./prp_ccbl2002.sh

SUP : $(OBJECT1)
        $(CC) -o $@ $(OBJECT1) $(LFLAGS)

#make object file
sup.o : sup.cbl                             …[3]始まり
        $(CCBL) $(CBLFLAGS1) sup.cbl        …[3]終わり

#cleaning load, object
clean :                                     …[4]始まり
        -rm SUP $(OBJECT1)
        -rm libxdb_sampleuap.so
        -rm sampleuap.cbl
        -rm sampleuap.o                     …[4]終わり
```

説明

[　] 内の番号は，次に示す項番と対応しています。

1. ライブラリ情報の定義
2. コンパイラオプション
3. ccbl2002 の実行
4. 生成されたファイルの削除

コンパイル時には事前に生成されたファイルを削除してから make コマンドを実行する必要があります。生成されるファイルについては，「表 A-1　コンパイルを実行すると生

成されるファイル」を参照してください。

（2） ccbl2002 コンパイル用シェル（prp_ccbl2002.sh）の内容

ccbl2002 コンパイル用シェル（prp_ccbl2002.sh）の内容を次に示します。

```
#!/bin/bash
#All Rights Reserved. Copyright (C) 2008, Hitachi, Ltd.
#Licensed Material of Hitachi, Ltd.
#Reproduction, use, modification or disclosure otherwise than
#permitted in the License Agreement is strictly prohibited.
#makefile
#prp_ccbl2002.sh for making UAP(COBOL Language)
#

export ID=$LOGNAME
#*******************************#
# プリプロセサ名（パスを含む）
#*******************************#
    EEXDBCBL=$DCDIR/bin/eexdbcbl
#*******************************#
#  UAPソース名（パスを含む）
#*******************************#
    UAPSOURCE=./sampleuap.ecb
#*******************************#
# COBOLコンパイラ名(パスを含む)
#*******************************#
    CCBL2002=/opt/HILNGcbl/bin/ccbl2002

#*******************************#
#     環境変数設定
#*******************************#
    export PATH=$VAR_PATH
    export LD_LIBRARY_PATH=$VAR_LDLIB
    export CBLLIB=$VAR_CBLLIB
    export LANG=$VAR_LANG
#*******************************#
#                               #
#     プリプロセス
#*******************************#
echo '****** START PRP ******'
    $EEXDBCBL $UAPSOURCE
echo '****** END PRP   ******'
#*******************************#
#                               #
#     コンパイル
#*******************************#
echo '****** START CCBL2002 ******'
    $CCBL2002 -PIC,Std -DynamicLink,Call -MainNotCBL ./sampleuap.cbl ld -shared
-o libxdb_sampleuap.so ./sampleuap.o -Bstatic -MultiThread -lpthread
echo '****** END CCBL2002   ******'
```

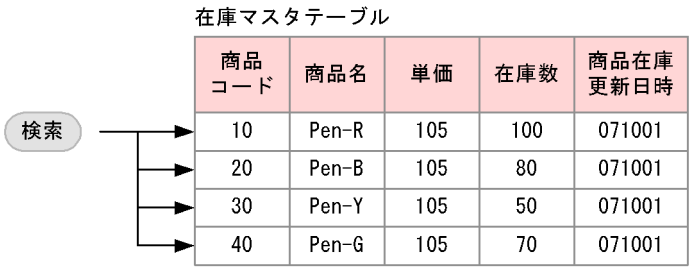
付録 A.6　サンプル UAP の処理概要

このサンプル UAP は，商品の入出荷テーブルを使用して，在庫データの参照や更新をすることで商品の在庫管理を行います。

処理の概要を次に示します。

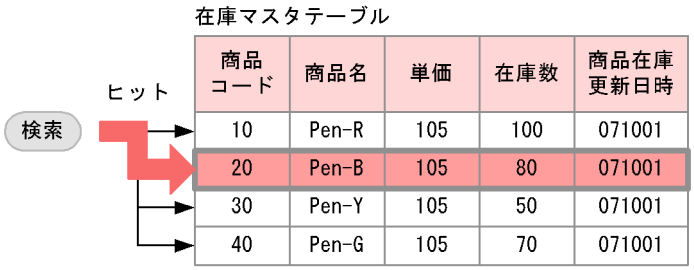
1. 入力種別が入荷の場合，在庫マスタテーブルの商品コード，商品名，および単価をキーに ROWID を検索します。

●商品コード，商品名，および単価をキーに検索します。

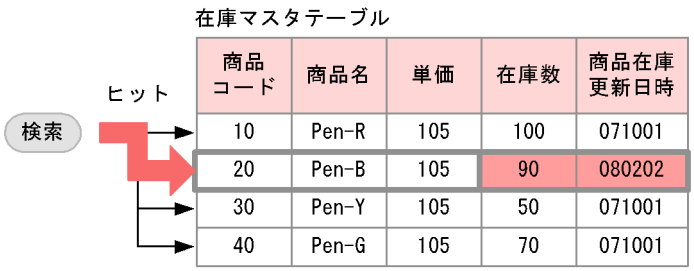


2. ヒットした場合，ROWID 指定で行を取り出し，在庫数と商品在庫更新日時を更新して ROWID 指定で行を更新します。

●検索がヒットします。



●ヒットしたROWIDで在庫数と商品在庫更新日時を更新します。



3. ヒットしない場合，行を追加します。

●ヒットしなかった場合，行を追加します。

4. 入力種別が出荷の場合，在庫マスタテーブルの商品コード，商品名，および単価をキーに ROWID を検索します。
5. ヒットした場合，ROWID 指定で行を取り出し，在庫数を更新したあと，ROWID 指定で行を更新します。
6. ヒットしない場合，エラーとします。
7. 在庫マスタテーブルの更新時に，商品在庫更新日時として CURRENT TIMESTAMP 値関数でタイムスタンプを更新します。

サンプル UAP が使用する表を次に示します。

● 在庫マスタテーブル

商品の在庫数を管理するマスタテーブルです。次に示す列から構成されています。

- 商品コード（ユニーク），商品名，単価，在庫数，商品在庫更新日時

● トランザクションテーブル

トランザクションが発生したときに使用するトランザクション用のテーブルです。次に示す列から構成されています。

- 店舗コード，商品コード，商品名，入荷／出荷，取引数

索引

数字

A

B

C

D

E

F

I

L

O

P

R

S

T

U

V

W

あ

い

う

え

か

き

く

け

こ

さ

し

す

せ

そ

た

つ

て

と

な

に

は

ひ

ふ

へ

み

も

ゆ

よ

り

れ

ろ

ソフトウェアマニュアルのサービス ご案内

１．マニュアル情報ホームページ

ソフトウェアマニュアルの情報をインターネットで公開しています。

URL　http://www.hitachi.co.jp/soft/manual/

ホームページのメニューは次のとおりです。

■マニュアル一覧	日立コンピュータ製品マニュアルを製品カテゴリ，マニュアル名称，資料番号のいずれかから検索できます。
■CD-ROMマニュアル	日立ソフトウェアマニュアルと製品群別CD-ROMマニュアルの仕様について記載しています。
■マニュアルのご購入	マニュアルご購入時のお申し込み方法を記載しています。
■オンラインマニュアル	一部製品のマニュアルをインターネットで公開しています。
■サポートサービス	ソフトウェアサポートサービスお客様向けページでのマニュアル公開サービスを記載しています。
■ご意見・お問い合わせ	マニュアルに関するご意見，ご要望をお寄せください。

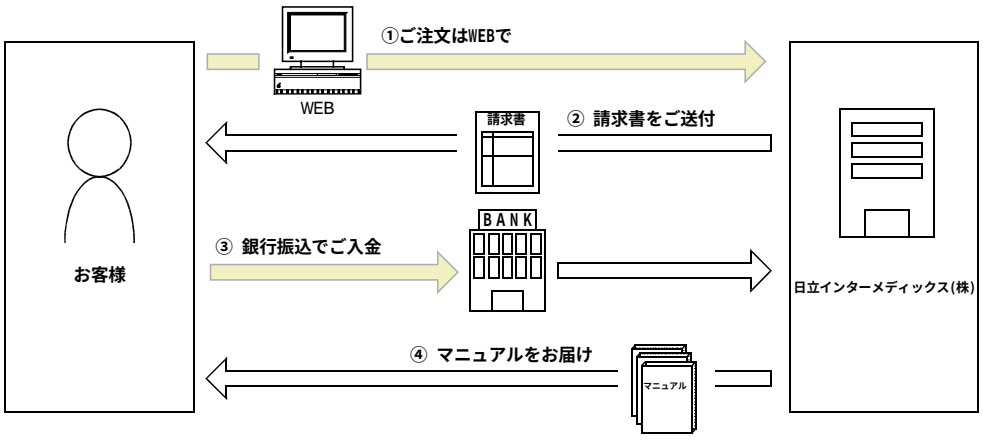
２．インターネットでのマニュアル公開

2 種類のマニュアル公開サービスを実施しています。

(1) マニュアル情報ホームページ「オンラインマニュアル」での公開
　製品をよりご理解いただくためのご参考として，一部製品のマニュアルを公開しています。

(2) ソフトウェアサポートサービスお客様向けページでのマニュアル公開
　ソフトウェアサポートサービスご契約のお客様向けにマニュアルを公開しています。公開しているマニュアルの一覧，本サービスの対象となる契約の種別などはマニュアル情報ホームページの「サポートサービス」をご参照ください。

３．マニュアルのご注文



① マニュアル情報ホームページの「マニュアルのご購入」にアクセスし，お申し込み方法をご確認のうえ WEB からご注文ください。ご注文先は日立インターメディックス(株)となります。

② ご注文いただいたマニュアルについて請求書をお送りします。

③ 請求書の金額を指定銀行へ振り込んでください。

④ 入金確認後 7 日以内にお届けします。在庫切れの場合は，納期を別途ご案内いたします。