HITACHI
Inspire the Next

Hitachi Advanced Data Binder SQL リファレンス

3000-6-504-P0

前書き

■ 著作権

■ 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

HITACHI，HA モニタ，HiRDB，JP1 は，株式会社 日立製作所の商標または登録商標です。

Access，Azure，Excel，Microsoft，Visual C++，Visual Studio，Windows，Windows Server は，マイクロソフト 企業グループの商標です。

Amazon Web Services，AWS，Powered by AWS ロゴ，Amazon Elastic Compute Cloud (Amazon EC2)，Amazon Simple Storage Service (Amazon S3)，Amazon Virtual Private Cloud (Amazon VPC) は，Amazon.com, Inc. またはその関連会社の商標です。

AMD は，Advanced Micro Devices, Inc.の商標です。

Docker および Docker ロゴは，Docker Inc. の米国およびその他の国における商標もしくは登録商標です。

Intel は，Intel Corporation またはその子会社の商標です。

Linux は，Linus Torvalds 氏の米国およびその他の国における登録商標です。

Oracle(R)，Java，MySQL 及び NetSuite は，Oracle，その子会社及び関連会社の米国及びその他の国における登録商標です。文中の社名，商品名等は各社の商標または登録商標である場合があります。

Red Hat, and Red Hat Enterprise Linux are registered trademarks of Red Hat, Inc. in the United States and other countries. Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.

Red Hat，および Red Hat Enterprise Linux は，米国およびその他の国における Red Hat, Inc.の登録商標です。Linux(R)は，米国およびその他の国における Linus Torvalds 氏の登録商標です。

RHEL is a trademark or a registered trademark of Red Hat, Inc. in the United States and other countries.

RHEL は，米国およびその他の国における Red Hat, Inc.の商標または登録商標です。

UNIX は，The Open Group の登録商標です。

その他記載の会社名，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。

1. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

2. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

3. This product includes software written by Tim Hudson (tjh@cryptsoft.com).

4. 本製品には OpenSSL Toolkit ソフトウェアを OpenSSL License および Original SSLeay License に従い使用しています。OpenSSL License および Original SSLeay License は以下の通りです。

LICENSE ISSUES

==========

The OpenSSL toolkit stays under a double license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit.
See below for the actual license texts.

OpenSSL License

/*

==========

* Copyright (c) 1998-2019 The OpenSSL Project. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in
* the documentation and/or other materials provided with the
* distribution.
*
* 3. All advertising materials mentioning features or use of this
* software must display the following acknowledgment:
* "This product includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
*

```
* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
* endorse or promote products derived from this software without
* prior written permission. For written permission, please contact
* openssl-core@openssl.org.
*
* 5. Products derived from this software may not be called "OpenSSL"
* nor may "OpenSSL" appear in their names without prior written
* permission of the OpenSSL Project.
*
* 6. Redistributions of any form whatsoever must retain the following
* acknowledgment:
* "This product includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit (http://www.openssl.org/)"
*
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
*
================================================================
=========
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/
```

Original SSLeay License

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"

* The word 'cryptographic' can be left out if the rouines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the apps directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/

■Double precision SIMD-oriented Fast Mersenne Twister (dSFMT)

Copyright (c) 2007, 2008, 2009 Mutsuo Saito, Makoto Matsumoto
and Hiroshima University.
Copyright (c) 2011, 2002 Mutsuo Saito, Makoto Matsumoto, Hiroshima
University and The University of Tokyo.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are

■ マイクロソフト製品のスクリーンショットの使用について

マイクロソフトの許可を得て使用しています。

■ マイクロソフト製品の表記について

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

表記			製品名
Azure			Microsoft Azure
Windows	Windows 10	Windows 10 x86	Windows(R) 10 Pro 日本語版(32 ビット版)
			Windows(R) 10 Enterprise 日本語版(32 ビット版)
		Windows 10 x64	Windows(R) 10 Pro 日本語版(64 ビット版)
			Windows(R) 10 Enterprise 日本語版(64 ビット版)

表記		製品名
	Windows 11	Windows(R) 11 Pro 日本語版
		Windows(R) 11 Enterprise 日本語版
	Windows Server 2012	Windows Server(R) 2012 Standard 日本語版
		Windows Server(R) 2012 Datacenter 日本語版
	Windows Server 2012 R2	Windows Server(R) 2012 R2 Standard 日本語版
		Windows Server(R) 2012 R2 Datacenter 日本語版
	Windows Server 2016	Windows Server(R) 2016 Standard 日本語版
		Windows Server(R) 2016 Datacenter 日本語版
	Windows Server 2019	Windows Server(R) 2019 Standard 日本語版
		Windows Server(R) 2019 Datacenter 日本語版
	Windows Server 2022	Windows Server(R) 2022 Standard 日本語版
		Windows Server(R) 2022 Datacenter 日本語版

■ 発行

2024 年 1 月

変更内容

変更内容(3000-6-504-P0)　Hitachi Advanced Data Binder　05-09

追加・変更内容		変更個所
ALTER TABLE 文の機能拡張	データ長が 256 バイト以上のVARCHAR 型の列のデータ長を，最大 32,000 バイトまで長くできるようにしました。	3.1.1(1)(e)
CREATE INDEX 文の機能拡張	1 つの表に定義できるインデクスの数を 64 個に拡張しました。	3.5.1(4)(a)

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルは，Hitachi Advanced Data Binder でデータベース操作に使用する SQL の文法について説明しています。

なお，このマニュアル中，および製品が出力する情報中（メッセージ，コマンドの出力結果など）では，Hitachi Advanced Data Binder を HADB と表記することがあります。

■ 対象製品

- P-8462-C611 Hitachi Advanced Data Binder 05-09　（適用 OS：Red Hat Enterprise Linux Server 7（64-bit x86_64），Red Hat Enterprise Linux Server 8（64-bit x86_64））
- P-8862-C811 Hitachi Advanced Data Binder 05-09　（適用 OS：Red Hat Enterprise Linux Server 9（64-bit x86_64））
- P-9W62-C311 Hitachi Advanced Data Binder Client 05-09　（適用 OS：Red Hat Enterprise Linux Server 7（64-bit x86_64），Red Hat Enterprise Linux Server 8（64-bit x86_64），Red Hat Enterprise Linux Server 9（64-bit x86_64））
- P-2462-C114 Hitachi Advanced Data Binder Client 05-09　（適用 OS：Windows 10，Windows 11，Windows Server 2012，Windows Server 2012 R2，Windows Server 2016，Windows Server 2019，Windows Server 2022）

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。

■ 対象読者

このマニュアルは，次に示す方々を対象にしています。

- AP 開発者
- HADB のシステム設計・構築者およびシステム管理者

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説明しています。

- SQL の基本的な知識
- Java 言語のプログラミングの基本的な知識，および JDBC の基本的な知識（Java 言語の AP を作成する場合）
- C 言語または C++言語のプログラミングの基本的な知識（C 言語または C++言語の AP を作成する場合）
- ODBC の基本的な知識（ODBC 対応の AP を作成する場合）

■ 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　SELECT 文の例題集

述語，集合関数，GROUP BY 句，HAVING 句などを使ったSELECT 文の書き方について例題形式で説明しています。SELECT 文の基本的な書き方を理解したい場合にこの章を読んでください。

第 2 章　SQL の一覧

HADB がサポートしている SQL の一覧と，SQL 構文の指定形式の読み方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，指定形式，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，指定形式，および規則について説明しています。

第 5 章　制御系 SQL

制御系 SQL の機能，指定形式，および規則について説明しています。

第 6 章　基本項目

SQL の記述規則や，データ型および定数などの SQL の基本項目について説明しています。

第 7 章　構成要素

問合せ式，問合せ指定，述語，値式，集合関数などの SQL の構成要素について説明しています。

第 8 章　スカラ関数

スカラ関数の機能，指定形式，および規則について説明しています。

付録 A　SQL 逆引きリファレンス

SQL の構文の記載個所について説明しています。実行したいことから，使用する SQL の構文を調べられる逆引きリファレンスです。

付録 B　関数一覧

HADB がサポートしている関数の一覧と各関数の記載個所について説明しています。

■ 関連マニュアル

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。

- 『Hitachi Advanced Data Binder システム構築・運用ガイド』（3000-6-501）
- 『Hitachi Advanced Data Binder AP 開発ガイド』（3000-6-502）
- 『Hitachi Advanced Data Binder コマンドリファレンス』（3000-6-503）
- 『Hitachi Advanced Data Binder メッセージ』（3000-6-505）
- 『Hitachi Code Converter（UNIX 編）』（3020-7-358）

- 『高信頼化システム監視機能 HA モニタ Linux(R)(x86)編』（3000-9-201）
- 『高信頼化システム監視機能 HA モニタ パブリッククラウド編』（3000-9-204）
- 『JP1 Version 11 JP1/Base 運用ガイド』（3021-3-A01）
- 『JP1 Version 11 JP1/Automatic Job Management System 3 設計ガイド（業務設計編）』（3021-3-B14）
- 『JP1 Version 11 JP1/Audit Management - Manager 構築・運用ガイド』（3021-3-A17）
- 『JP1 Version 12 JP1/Base 運用ガイド』（3021-3-D65）
- 『JP1 Version 12 JP1/Automatic Job Management System 3 設計ガイド（業務設計編）』（3021-3-D23）

なお，Hitachi Advanced Data Binder のマニュアルを本文中で参照させる場合は，『Hitachi Advanced Data Binder』を『HADB』と表記します。

（例）『HADB　システム構築・運用ガイド』

また，HA モニタのマニュアルを本文中で参照させる場合は，次のように表記します。

- 『高信頼化システム監視機能 HA モニタ Linux(R)(x86)編』を『HA モニタ Linux(R)(x86)編』と表記します。
  （例）『HA モニタ Linux(R)(x86)編』
- 『高信頼化システム監視機能 HA モニタ パブリッククラウド編』を『HA モニタ パブリッククラウド編』と表記します。
  （例）『HA モニタ パブリッククラウド編』

JP1/Base のマニュアルを本文中で参照させる場合は，『JP1 Version 11 JP1/Base 運用ガイド』または『JP1 Version 12 JP1/Base 運用ガイド』を『JP1/Base　運用ガイド』と表記します。

（例）『JP1/Base　運用ガイド』

JP1/AJS3 のマニュアルを本文中で参照させる場合は，『JP1 Version 11 JP1/Automatic Job Management System 3 設計ガイド（業務設計編）』または『JP1 Version 12 JP1/Automatic Job Management System 3 設計ガイド（業務設計編）』を『JP1/AJS3　設計ガイド（業務設計編）』と表記します。

（例）『JP1/AJS3　設計ガイド（業務設計編）』

JP1/Audit のマニュアルを本文中で参照させる場合は，『JP1 Version 11 JP1/Audit Management - Manager 構築・運用ガイド』を『JP1/Audit　構築・運用ガイド』と表記します。

（例）『JP1/Audit　構築・運用ガイド』

■ このマニュアルで使用する製品名・機能名

このマニュアルでは，製品名を次のように表記しています。

表記		製品名
HADB	HADB サーバ	Hitachi Advanced Data Binder
	HADB クライアント	Hitachi Advanced Data Binder Client
Linux	Linux	Linux
	RHEL 7	Red Hat Enterprise Linux Server 7（64-bit x86_64）
	RHEL 8	Red Hat Enterprise Linux Server 8（64-bit x86_64）
	RHEL 9	Red Hat Enterprise Linux Server 9（64-bit x86_64）
HDLM		Hitachi Dynamic Link Manager Software
Hitachi Code Converter		Hitachi Code Converter - Runtime for C/COBOL（64）
JP1/AJS3		JP1/Automatic Job Management System 3
JP1/Audit		JP1/Audit Management - Manager

■ このマニュアルで使用する英略語

このマニュアルで使用する英略語を次に示します。

英略語	英字での表記
AD	Active Directory
Amazon S3	Amazon Simple Storage Service
AP	Application Program
APD	Application Parameter Descriptor
API	Application Programming Interface
ARD	Application Row Descriptor
AWS	Amazon Web Services
BI	Business Intelligence
BLOB	Binary Large Object
BNF	Backus-Naur Form
BOM	Byte Order Mark
CLI	Call Level Interface

英略語	英字での表記
CLOB	Character Large Object
CPU	Central Processing Unit
CSV	Character-Separated Values
DB	Database
DBMS	Database Management System
DMMP	Device Mapper Multipath
DNS	Domain Name System
DRBD	Distributed Replicated Block Device
EBS	Amazon Elastic Block Store
EC2	Amazon Elastic Compute Cloud
EFS	Amazon Elastic File System
ELF	Executable and Linking Format
ER	Entity Relationship
HBA	Host Bus Adapter
HDD	Hard Disk Drive
ID	Identification number
IEF	Integrity Enhancement Facility
IP	Internet Protocol
IPD	Implementation Parameter Descriptor
IRD	Implementation Row Descriptor
JAR	Java Archive File
JDBC	Java Database Connectivity
JDK	Java Developer's Kit
JNDI	Java Naming and Directory Interface
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JTA	Java Transaction API
LDAP	Lightweight Directory Access Protocol
LOB	Large Object

英略語	英字での表記
LRU	Least Recently Used
LV	Logical Volume
LVM	Logical Volume Manager
LWP	Light Weight Process
MSDN	Microsoft Developer Network
NFS	Network File System
NIC	Network Interface Card
NTP	Network Time Protocol
ODBC	Open Database Connectivity
OS	Operating System
OSS	Open Source Software
PAM	Pluggable Authentication Module
PP	Program Product
PV	Physical Volume
PVC	Persistent Volume Claim
RAID	Redundant Array of Independent Disks
RDBMS	Relational Database Management System
SELinux	Security-Enhanced Linux
SSD	Solid State Drive
SSSD	System Security Services Daemon
TLB	Translation Lookaside Buffer
URL	Uniform Resource Locator
VG	Volume Group
VPC	Amazon Virtual Private Cloud
WWN	World Wide Name
XFS	Extents File System

■ このマニュアルで使用する記号

サーバ定義などのオペランド，およびコマンドの説明で使用している記号を次に示します。

なお，これらの記号は説明のために使用している記号のため，オペランドまたはコマンド中に記述しないでください。

記号	意味	例
〔 〕	この記号で囲まれている項目は省略できます。	adbsql 〔-V〕 この例の場合，adbsql と指定してもよいし，adbsql -V と指定してもよいことを意味しています。
{ }	この記号で囲まれている複数の項目のうちから，1 つを選択できます。	adbcancel {--ALL | -u *コネクション ID*} この例の場合，--ALL または-u *コネクション ID* のどちらかを指定できることを意味しています。
…	この記号の直前の項目を繰り返し指定できます。	adbbuff -n *DB エリア名*〔, *DB エリア名*〕… この例の場合，*DB エリア名*を繰り返し指定できることを意味しています。
{ { } }	この記号で囲まれた複数の項目を 1 つの単位として，繰り返し指定できます。	{ {adbinitdbarea -n *データ用 DB エリア名*} } この例の場合，「adbinitdbarea -n *データ用 DB エリア名*」を繰り返し指定できることを意味しています。
__ （下線）	この記号で示す項目は，省略時の解釈値です。	adb_import_errmsg_lv = {0 | 1} この例の場合，オペランドの指定を省略したとき，0 が仮定されることを意味しています。
～	この記号のあとに，指定値の属性を説明しています。	adb_sys_max_users = *最大同時接続数* ～〈整数〉((1～1,024))《10》 この例の場合，1～1,024 の整数が指定できます。オペランドの指定を省略した場合は，10 が仮定されます。
〈 〉	指定値の種別を説明しています。	
(( ))	指定値の範囲を説明しています。	
《 》	省略値を説明しています。	

■ このマニュアルで使用する構文要素記号

構文要素記号	意味
〈パス名〉	パス名には次に示す文字が使用できます。 • OS が Linux の場合 英字，数字，#，-，/，@，_ • OS が Windows の場合 英字，数字，#，-，/，@，_，¥，： ただし，OS によって使用できる文字が異なります。
〈OS パス名〉	OS パス名には，OS でパス名として使用できるすべての文字が使用できます。使用できる文字の詳細については，OS のマニュアルを参照してください。
〈文字列〉	任意の文字列を指定できます。

構文要素記号	意味
〈単位付き整数〉	数字（0～9）の末尾に，MB（メガバイト），GB（ギガバイト），またはTB（テラバイト）のどれかの単位を付けた形式で指定します。数字と単位の間に空白を入れることはできません。 • 指定例 1024MB 512GB 32TB • エラーになる指定例 512 GB

注　すべて半角文字を使用してください。

■ このマニュアルで使用する計算式の記号

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	内容
↑　↑	計算結果の値を小数点以下で切り上げることを意味しています。 （例）↑34÷3↑の計算結果は 12 になります。
↓　↓	計算結果の値を小数点以下で切り捨てることを意味しています。 （例）↓34÷3↓の計算結果は 11 になります。
MAX	計算結果のうち，最も大きい値が有効になることを意味しています。 （例）MAX（3×6，4 ＋ 7）の計算結果は 18 になります。
MIN	計算結果のうち，最も小さい値が有効になることを意味しています。 （例）MIN（3×6，4 ＋ 7）の計算結果は 11 になります。

■ パス名の表記について

- サーバディレクトリ（インストール時）のパスは，$INSTDIR と表記します。
- サーバディレクトリ（運用時）のパスは，$ADBDIR と表記します。
- DB ディレクトリのパスは，$DBDIR と表記します。
- クライアントディレクトリのパスは，%ADBCLTDIR%（HADB クライアントが Windows 版の場合）または$ADBCLTDIR（HADB クライアントが Linux 版の場合）と表記します。
- HADB ODBC ドライバトレースファイルの格納フォルダのパスは，%ADBODBTRCPATH%と表記します。

■ ¥の表記について

本文中で使用されている¥は，Linux 版の場合は半角のバックスラッシュを意味しています。

■ このマニュアルで使用する KB（キロバイト）などの単位表記

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト），1PB（ペタバイト），1EB（エクサバイト）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，$1{,}024^5$ バイト，$1{,}024^6$ バイトです。

■ HADB のデータベース言語の出典

このマニュアルで記述する HADB 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 SELECT 文の例題集

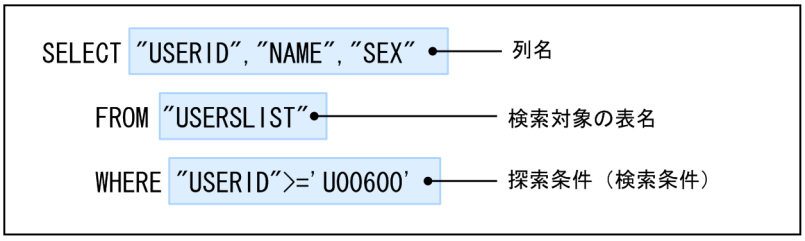
この章では，SELECT 文の基本的な書き方を例題を使って説明します。

1.1 節では，SELECT 文の基本的な書き方を説明しています。1.2 節以降では，SELECT 文の書き方を例題形式で説明しています。

1.1　SELECT 文の基本的な書き方および規則

ここでは，SELECT 文の基本的な書き方と基本的な規則について説明します。

1.1.1　SELECT 文の基本的な書き方



列名：

検索結果を取り出す列（表示する列）を指定します。複数の列名を指定できます。

検索対象の表名：

FROM 句には，検索対象表を指定します。複数の表名を指定できます。

探索条件（検索条件）：

WHERE 句には，検索データを絞り込むための探索条件を指定します。WHERE 句に複数の探索条件を指定する場合は，AND またはOR で探索条件をつなぎます。

```
（例）WHERE ”USERID”>=’U00600’ AND ”SEX”=’M’
```

次に示すSELECT 文を実行した場合，検索結果は次のようになります。

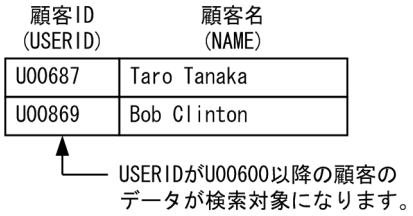
```
SELECT ”USERID”,”NAME”
    FROM ”USERSLIST”
        WHERE ”USERID”>=’U00600’
```

- **検索対象表（顧客表：USERSLIST）の構成**

顧客ID (USERID)	顧客名 (NAME)	性別 (SEX)
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

注　（　）内は，列名を示しています。

- **検索結果**



メモ

探索条件，FROM 句，WHERE 句の文法規則の詳細については，次に示す該当個所を参照してください。

- FROM 句：「7.5.1　FROM 句の指定形式および規則」
- WHERE 句：「7.6.1　WHERE 句の指定形式」
- 探索条件：「7.19.1　探索条件の指定形式および規則」

1.1.2　SELECT 文を書く際の基本的な規則

SELECT 文を書く際の基本的な規則を次に示します。

- SELECT 文中に指定する表名および列名は，二重引用符（”）で囲んで指定することを推奨します。二重引用符で囲んで指定すると，SQL の予約語と同じ名称を指定できるため，将来，表名や列名と同じ名称の予約語が追加されたときに，SQL を書き換える必要がなくなります。

  また，二重引用符で囲まない場合，英小文字の指定は英大文字として扱われます。例えば，name と指定した場合，NAME として扱われます。

- CHAR 型およびVARCHAR 型の文字データは，アポストロフィ（’）で囲んでください。

```
（例）WHERE ”NAME”='Taro Tanaka'
```

- DATE 型の日付データは，次のように記述してください。

```
（例1）WHERE ”PUR-DATE”>=DATE'2011-09-06'
（例2）WHERE ”PUR-DATE”>=DATE'2011/09/06'
```

  この章で説明しているSELECT 文の例は，例 1 の形式を使用しています。

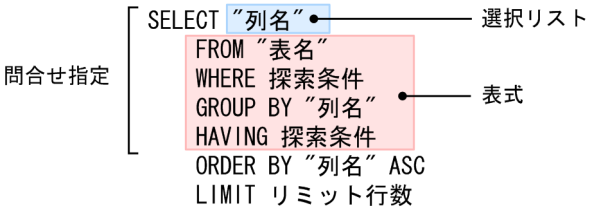
- INTEGER 型などの数データは，アポストロフィ（’）で囲みません。

```
（例）WHERE ”PUR-NUM”=10
```

1.1.3　SELECT 文の構文と構成要素の関係（ご参考）

このマニュアルでは，SELECT 文の構文を構成要素に分解して説明しています。SELECT 文の構文と構成要素の関係を次の図に示します。

図 1-1　SELECT 文の構文と構成要素の関係



各構成要素について説明します。

問合せ指定：

検索結果を取り出す列，検索対象表，および探索条件を指定する部分を**問合せ指定**といいます。

選択リスト：

検索結果として取り出す項目の指定を**選択リスト**といいます。通常は列名を指定しますが，集合関数などを指定することもできます。

表式：

FROM 句，WHERE 句，GROUP BY 句，およびHAVING 句を総称して**表式**といいます。

ORDER BY 句：

検索結果を昇順または降順に並べたいときに指定します。指定例については，「1.3　検索結果を昇順に並べる（ORDER BY 句）」を参照してください。

LIMIT 句：

検索結果の行数の上限を指定したいときに指定します。指定例については，「1.4　検索結果行数の上限を指定する（LIMIT 句）」を参照してください。

> **メモ**
>
> 問合せ指定，選択リスト，表式の文法規則の詳細については，次に示す該当個所を参照してください。
>
> - 問合せ指定：「7.2.1　問合せ指定の指定形式および規則」
> - 選択リスト：「7.2.1　問合せ指定の指定形式および規則」の「(2)　指定形式の説明」の「(c)　選択リスト」
> - 表式：「7.4.1　表式の指定形式および規則」

1.1.4　1.2 節以降を読むに当たっての注意

- 1.2 節以降では，SELECT 文の書き方を例題形式で説明していますが，例題が複数ある場合，基本の例題から始まり，順次応用の例題になっています。
- 例題の検索結果の行の並び順は，見やすさを考慮して実際の検索結果の行の並び順とは異なっています。

1.2　表の全行を検索する

1.2.1　例（全顧客の顧客情報を検索する）

顧客表（USERSLIST）のすべての行を検索し，結果を表示します。顧客表は，顧客 ID（USERID），名前（NAME），性別（SEX）の列で構成されています。

検索対象表

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT ”USERID”,”NAME”,”SEX”
    FROM ”USERSLIST”
```

検索結果

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

メモ

表のすべての列を検索結果とする場合は，列名の代わりにアスタリスク（＊）を指定できます。指定例を次に示します。

指定例

```
SELECT * FROM ”USERSLIST”
```

検索結果

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

1.3　検索結果を昇順に並べる（ORDER BY 句）

検索結果を昇順または降順に並べたいときは，ORDER BY 句を使います。ORDER BY 句の指定形式を次に示します。

指定形式

```
SELECT ”列名” FROM ”表名”
    WHERE 探索条件
    ORDER BY ”列名” ASC
```

ORDER BY ”*列名*” ASC：

*列名*には，並べ替えの対象とする列を指定します。また，検索結果を昇順に並べる場合はASC を，降順に並べる場合はDESC を指定します。

> **メモ**
>
> ORDER BY 句には列名以外の指定もできます。ORDER BY 句の文法規則の詳細については，「7.25 ソート指定リスト」を参照してください。

1.3.1　例 1（顧客 ID 順に検索結果を並べる）

顧客表（USERSLIST）の全データを顧客 ID（USERID）順に並べます。顧客表は，顧客 ID（USERID），名前（NAME），性別（SEX）の列で構成されています。

検索対象表

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT ”USERID”,”NAME”,”SEX”
    FROM ”USERSLIST”
    ORDER BY ”USERID” ASC
```

検索結果

USERID	NAME	SEX
U00212	Maria Gomez	F
U00358	Nancy White	F
U00555	Mike Johnson	M
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

検索結果が顧客ID順に並びます。

メモ

ORDER BY 句には，並べ替えの対象とする列の名称を指定します。この例の場合は，顧客 ID 順に並べるため，ORDER BY 句にUSERID を指定しています。

1.3.2 例 2（購入日順，顧客 ID 順に検索結果を並べる）

販売履歴表（SALESLIST）の全データを購入日（PUR-DATE）順に並べます。購入日が同じ場合は，顧客 ID（USERID）順に並べます。販売履歴表は，顧客 ID（USERID），商品コード（PUR-CODE），販売個数（PUR-NUM），購入日（PUR-DATE）の列で構成されています。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT "USERID","PUR-CODE","PUR-NUM","PUR-DATE"
    FROM "SALESLIST"
    ORDER BY "PUR-DATE" ASC,"USERID" ASC
```

検索結果

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00212	P003	10	2011-09-03
U00358	P001	9	2011-09-03
U00358	P001	1	2011-09-04
U00358	P002	3	2011-09-04
U00212	P001	6	2011-09-05
U00212	P002	12	2011-09-05
U00358	P003	2	2011-09-05
U00555	P002	5	2011-09-06
U00687	P002	8	2011-09-06
U00358	P002	6	2011-09-07
U00687	P003	5	2011-09-07

購入日順に並びます。

購入日が同じ場合は，顧客ID順に並びます。

メモ

ORDER BY 句には複数の列を指定できます。並び替えの優先順位は，最初に指定した列がいちばん高くなります。この例の場合，まずは購入日（PUR-DATE）順に並び替え，購入日が同じ場合は顧客 ID（USERID）順に並び替えています。

1.4　検索結果行数の上限を指定する（LIMIT 句）

検索結果行数の上限を指定したいときは，LIMIT 句を使います。LIMIT 句の指定形式を次に示します。

指定形式

```
SELECT ”列名” FROM ”表名”
    WHERE 探索条件
    LIMIT リミット行数
```

LIMIT *リミット行数*：

リミット行数には，検索結果行数の上限値を指定します。

> **メモ**
>
> LIMIT 句には，リミット行数のほかにオフセット行数を指定することもできますが，ここではオフセット行数の説明は省略します。LIMIT 句の文法規則の詳細については，「7.9.1　LIMIT 句の指定形式および規則」を参照してください。

1.4.1　例（検索結果行数の上限を指定する）

販売履歴表（SALESLIST）を検索し，販売個数（PUR-NUM）の上位３位までを表示します。

検索対象表

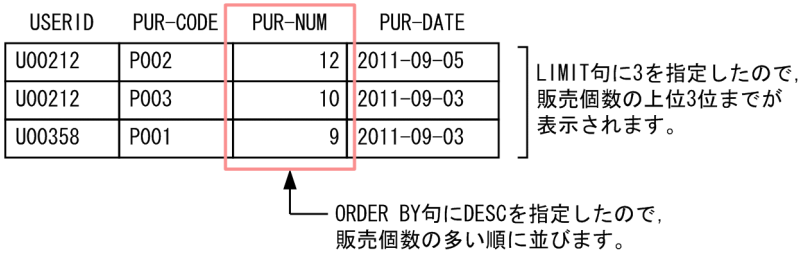
■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT ”USERID”,”PUR-CODE”,”PUR-NUM”,”PUR-DATE”
    FROM ”SALESLIST”
    ORDER BY ”PUR-NUM” DESC
    LIMIT 3
```

検索結果


USERID
PUR-CODE
PUR-NUM
PUR-DATE
U00212
P002
12
2011-09-05
U00212
P003
10
2011-09-03
U00358
P001
9
2011-09-03
LIMIT句に3を指定したので,
販売個数の上位3位までが
表示されます。
ORDER BY句にDESCを指定したので,
販売個数の多い順に並びます。

1.5　探索条件を指定して検索する

WHERE 句に探索条件（検索条件）を指定して検索対象の行を絞り込みます。WHERE 句の指定形式を次に示します。

指定形式

- 探索条件を 1 つだけ指定する場合

```
SELECT ”列名” FROM ”表名”
    WHERE 探索条件
```

- 2 つ以上の探索条件を指定する場合

```
SELECT ”列名” FROM ”表名”
    WHERE 探索条件1 AND 探索条件2 …
```

または

```
SELECT ”列名” FROM ”表名”
    WHERE 探索条件1 OR 探索条件2 …
```

WHERE 句に複数の探索条件を指定する場合，AND またはOR で探索条件をつなぎます。AND とOR を混在して指定できます。

WHERE *探索条件 1* AND *探索条件 2*：

探索条件 1 と*探索条件 2* の両方を満たす行が検索対象になります。

WHERE *探索条件 1* OR *探索条件 2*：

探索条件 1 と*探索条件 2* のどちらかを満たす行が検索対象になります。

> **メモ**
>
> WHERE 句，探索条件の文法規則の詳細については，次に示す該当個所を参照してください。
>
> - WHERE 句：「7.6.1　WHERE 句の指定形式」
> - 探索条件：「7.19.1　探索条件の指定形式および規則」

1.5.1　例 1（購入日を条件にして検索する）

販売履歴表（SALESLIST）から，2011/9/6 以降に商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-06'
```

検索結果

USERID	PUR-CODE	PUR-DATE
U00555	P002	2011-09-06
U00687	P002	2011-09-06
U00358	P002	2011-09-07
U00687	P003	2011-09-07

2011/9/6以降のデータが検索対象になります。

メモ

- WHERE 句に探索条件を指定する際，次の表に示す比較演算子が使えます。比較演算子の種類と意味を次の表に示します。

表 1-1　比較演算子の種類と意味

項番	比較演算子	意味
1	=	等しい
2	<>，!=，または^=	等しくない
3	<	より小さい
4	<=	以下
5	>	より大きい
6	>=	以上

- 条件式に指定する値がCHAR 型およびVARCHAR 型の文字列の場合，値をアポストロフィ（’）で囲んでください。

  （例）WHERE ”NAME”=’Taro Tanaka’
- 条件式に指定する値がDATE 型の日付の場合，次のように指定してください。

  （例）WHERE ”PUR-DATE”>=DATE’2011-09-06’

1.5.2 例 2（購入日と商品コードを条件にして検索する）

販売履歴表（SALESLIST）から，2011/9/6 以降に商品コード P002 の商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

検索対象表

■SALESLIST

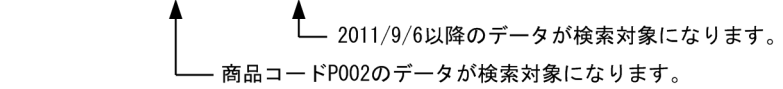
USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT ”USERID”,”PUR-CODE”,”PUR-DATE”
    FROM ”SALESLIST”
        WHERE ”PUR-DATE”>=DATE’2011-09-06’
        AND ”PUR-CODE”=’P002’
```

検索結果

USERID	PUR-CODE	PUR-DATE
U00555	P002	2011-09-06
U00687	P002	2011-09-06
U00358	P002	2011-09-07



メモ

WHERE 句に次に示す 2 つの探索条件をAND でつないで指定しています。

- 2011/9/6 以降に商品を購入
- 商品コード P002 の商品を購入

1.5.3 例 3（購入日と 2 つの商品コードを条件にして検索する）

販売履歴表（SALESLIST）から，2011/9/4 以降に商品コード P001 または P003 の商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-04'
        AND ("PUR-CODE"='P001' OR "PUR-CODE"='P003')
```

検索結果

USERID	PUR-CODE	PUR-DATE
U00358	P001	2011-09-04
U00358	P003	2011-09-05
U00212	P001	2011-09-05
U00687	P003	2011-09-07

2011/9/4以降のデータが検索対象になります。

商品コードがP001またはP003のデータが検索対象になります。

メモ

AND とOR を両方指定した場合，AND が最初に評価されます。評価の優先順位を変更するには，指定例のように（　）を指定してください。

1.6　検索範囲を指定して検索する（BETWEEN 述語）

検索範囲を指定する場合は，BETWEEN 述語を使います。BETWEEN 述語の指定形式を次に示します。

指定形式

```
SELECT ”列名” FROM ”表名”
    WHERE ”列名” BETWEEN 値1 AND 値2
```

列名：

検索範囲を絞り込む列を指定します。

BETWEEN *値 1* AND *値 2*：

値 1 には，検索範囲の下限値を指定します。*値 2* には，検索範囲の上限値を指定します。

```
（例）WHERE C1 BETWEEN 10 AND 20
```

この例の場合，列C1 の値が10～20（10 と20 も含まれます）の行が検索範囲になります。

> **メモ**
>
> BETWEEN 述語の文法規則の詳細については，「7.20.1　BETWEEN 述語」を参照してください。

1.6.1　例 1（期間中に商品を購入した顧客を検索する）

販売履歴表（SALESLIST）から，2011/9/4～2011/9/5 に商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
       WHERE "PUR-DATE" BETWEEN DATE'2011-09-04' AND DATE'2011-09-05'
```

検索結果

USERID	PUR-CODE	PUR-DATE
U00358	P001	2011-09-04
U00358	P002	2011-09-04
U00212	P002	2011-09-05
U00212	P001	2011-09-05
U00358	P003	2011-09-05

メモ

AND 条件の指定をBETWEEN 述語で書き換えることができます。例えば，次に示すAND 条件を指定したSELECT 文と，上記の指定例に示すBETWEEN 述語を指定したSELECT 文は，同じ検索結果になります。

AND 条件については，「1.5　探索条件を指定して検索する」を参照してください。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
       WHERE "PUR-DATE">=DATE'2011-09-04'
       AND "PUR-DATE"<=DATE'2011-09-05'
```

1.6.2　例 2（期間外に商品を購入した顧客を検索する）

販売履歴表（SALESLIST）から，2011/9/4～2011/9/5 を除いた日に商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
       WHERE "PUR-DATE" NOT BETWEEN DATE'2011-09-04' AND DATE'2011-09-05'
```

検索結果

USERID	PUR-CODE	PUR-DATE
U00212	P002	2011-09-03
U00212	P003	2011-09-03
U00358	P001	2011-09-03
U00555	P002	2011-09-06
U00687	P002	2011-09-06
U00687	P003	2011-09-07
U00358	P002	2011-09-07

メモ

NOT を指定すると，直後の条件式を満たさない値が検索対象になります。指定例のように「NOT BETWEEN DATE'2011-09-04' AND DATE'2011-09-05'」と指定した場合，2011/9/4～2011/9/5 を除いた条件で検索されます。

1.7　複数の条件のどれかに一致するデータを検索する（IN 述語）

複数の条件（値）を指定し，そのうちのどれかに一致するデータを検索する場合は，IN 述語を使います。IN 述語の指定形式を次に示します。

指定形式

```
SELECT ”列名” FROM ”表名”
    WHERE ”列名” IN（値1,値2,…）
```

列名：

検索対象を絞り込む列を指定します。

IN（*値 1*，*値 2*，…）：

検索対象とする値を指定します。ここで指定した値のどれかに一致する行が検索対象になります。

> **メモ**
>
> IN 述語の文法規則の詳細については，「7.20.3　IN 述語」を参照してください。

1.7.1　例 1（商品コード P001 または P003 の商品を購入した顧客を検索する）

販売履歴表（SALESLIST）から，商品コード P001 または P003 の商品を 2011/9/5 以降に購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT ”USERID”,”PUR-CODE”,”PUR-DATE”
    FROM ”SALESLIST”
```

```
        WHERE "PUR-CODE" IN ('P001','P003')
        AND "PUR-DATE">=DATE'2011-09-05'
```

検索結果

USERID	PUR-CODE	PUR-DATE
U00212	P001	2011-09-05
U00358	P003	2011-09-05
U00687	P003	2011-09-07

メモ

OR 条件の指定をIN 述語で書き換えることができます。例えば，次に示すOR 条件を指定したSELECT 文と，上記の指定例に示すIN 述語を指定したSELECT 文は，同じ検索結果になります。

OR 条件については，「1.5　探索条件を指定して検索する」を参照してください。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
       WHERE ("PUR-CODE"='P001' OR "PUR-CODE"='P003')
       AND "PUR-DATE">=DATE'2011-09-05'
```

1.7.2　例 2（特定の顧客を除いて，商品を購入した顧客を検索する）

販売履歴表（SALESLIST）から，顧客 ID（USERID），商品コード（PUR-CODE），販売個数（PUR-NUM）を検索します。ただし，顧客 ID（USERID）が U00212 および U00358 の顧客は，検索対象外とします。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT "USERID","PUR-CODE","PUR-NUM"
    FROM "SALESLIST"
       WHERE "USERID" NOT IN ('U00212','U00358')
```

検索結果

USERID	PUR-CODE	PUR-NUM
U00555	P002	5
U00687	P002	8
U00687	P003	5

メモ

NOT を指定すると，直後の条件式を満たさない値が検索対象になります。指定例のように「NOT IN ('U00212','U00358')」と指定した場合，U00212 および U00358 以外の USERID が検索対象になります。

1.8　特定の文字列が含まれているデータを検索する（LIKE 述語）

特定の文字列が含まれているデータを検索するには，LIKE 述語を使います。LIKE 述語の指定形式を次に示します。

指定形式

```
SELECT ”列名” FROM ”表名”
    WHERE ”列名” LIKE ’パターン文字列’
```

列名：

検索対象を絞り込む列を指定します。

> **メモ**
>
> 列名以外の指定もできます。LIKE 述語の文法規則の詳細については，「7.20.4　LIKE 述語」を参照してください。

LIKE ’*パターン文字列*’：

検索対象とするパターン文字列を指定します。主なパターン文字列を次に示します。

- %

  任意の 0 文字以上の文字列を意味しています。’ACT%’と指定した場合，ACT，ACTOR，ACTION などの文字列が検索対象になります。

- _（下線）

  任意の 1 文字を意味しています。’_I_’と指定した場合，BIT，HIT，KIT などの文字列が検索対象になります。

> **メモ**
>
> - パターン文字列の文法規則の詳細については，「7.20.4　LIKE 述語」を参照してください。
> - LIKE 述語にはESCAPE を指定することもできます。詳細については，「7.20.4　LIKE 述語」を参照してください。

1.8.1　例 1（名前が M で始まる顧客を検索する）

顧客表（USERSLIST）から，名前が M で始まる顧客の，顧客 ID（USERID），名前（NAME），性別（SEX）を検索します。

検索対象表

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT "USERID","NAME","SEX"
    FROM "USERSLIST"
        WHERE "NAME" LIKE 'M%'
```

検索結果

USERID	NAME	SEX
U00212	Maria Gomez	F
U00555	Mike Johnson	M

1.8.2 例 2（名前が M で始まらない顧客を検索する）

顧客表（USERSLIST）から，名前が M で始まらない女性の顧客の，顧客 ID（USERID），名前（NAME），性別（SEX）を検索します。

検索対象表

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT "USERID","NAME","SEX"
    FROM "USERSLIST"
        WHERE "NAME" NOT LIKE 'M%'
        AND "SEX"='F'
```

検索結果

USERID	NAME	SEX
U00358	Nancy White	F

メモ

NOT を指定すると，直後の条件式を満たさない値が検索対象になります。指定例のように「NOT LIKE 'M%'」と指定した場合，M で始まらない文字列が検索対象になります。

1.9　複数の表を指定して検索する（表の結合）

複数の表に検索対象のデータが分散している場合，共通の情報を持つ列を対応づけて検索を行います。これを**表の結合**といいます。販売履歴表（SALESLIST）と顧客表（USERSLIST）を例にして，表の結合について説明します。

(例)

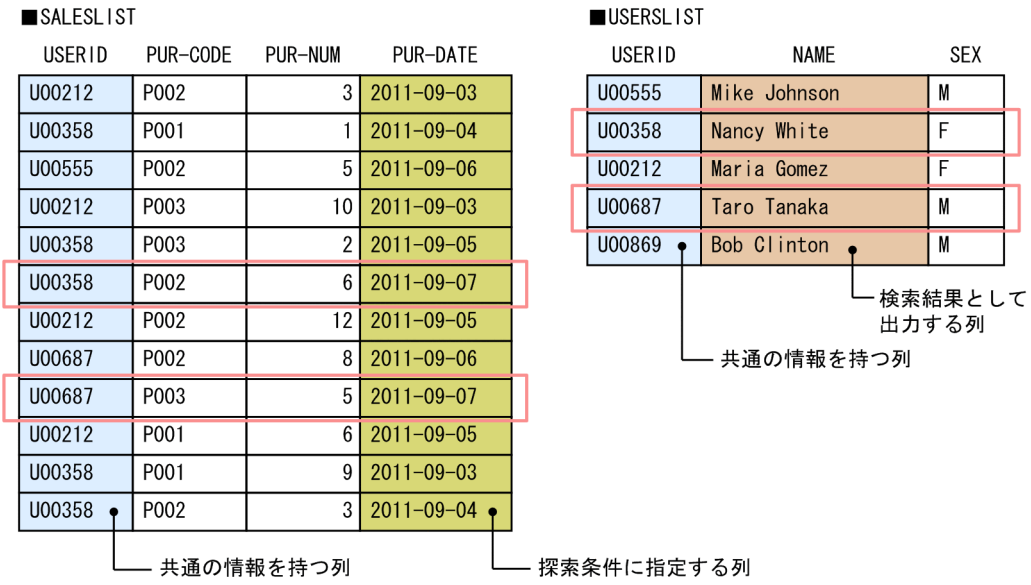
次に示す販売履歴表（SALESLIST）と顧客表（USERSLIST）から，2011/9/7 に商品を購入した顧客の名前（NAME）を検索するとします。

SELECT 文の指定

```
SELECT "NAME"
    FROM "SALESLIST","USERSLIST"
        WHERE "PUR-DATE"=DATE'2011-09-07'
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
```

説明

探索条件に指定した購入日（PUR-DATE）の情報は，販売履歴表（SALESLIST）にあります。一方，検索結果として出力する名前（NAME）の情報は，顧客表（USERSLIST）にあります。このような場合に，販売履歴表（SALESLIST）と顧客表（USERSLIST）を，共通の情報を持つ顧客 ID（USERID）で結合します。



検索結果

Nancy White
Taro Tanaka

1.9.1　例 1（販売履歴表と顧客表から商品を購入した顧客を検索する その 1）

販売履歴表（SALESLIST）と顧客表（USERSLIST）から，2011/9/6 以降に商品を購入した顧客の，顧客 ID（USERID），名前（NAME），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT "SALESLIST"."USERID","NAME","PUR-CODE","PUR-DATE"
    FROM "SALESLIST","USERSLIST"
        WHERE "PUR-DATE">=DATE'2011-09-06'
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
```

検索結果

USERID	NAME	PUR-CODE	PUR-DATE
U00555	Mike Johnson	P002	2011-09-06
U00687	Taro Tanaka	P002	2011-09-06
U00358	Nancy White	P002	2011-09-07
U00687	Taro Tanaka	P003	2011-09-07

メモ

- 両方の表に同じ列名がある場合，指定した列名がどちらの表の列かを識別するために「"*表名*"."*列名*"」の形式で指定します。この例の場合は，USERID 列が該当します。"SALESLIST"."USERID"または"USERSLIST"."USERID"と指定します。
- FROM 句には，検索対象とするすべての表を指定します。
- 顧客 ID 列（USERID）をキーにして表を結合するため，「AND "SALESLIST"."USERID"="USERSLIST"."USERID"」の条件式を指定します。

1.9.2 例 2（販売履歴表と顧客表から商品を購入した顧客を検索する その 2）

販売履歴表（SALESLIST）と顧客表（USERSLIST）から，次に示す条件に当てはまる顧客の，顧客 ID（USERID），名前（NAME），性別（SEX），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

- 2011/9/6 以降に商品を購入した男性の顧客

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT "SALESLIST"."USERID","NAME","SEX","PUR-CODE","PUR-DATE"
    FROM "SALESLIST","USERSLIST"
        WHERE "PUR-DATE">=DATE'2011-09-06'
```

```
        AND "SEX"='M'
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
```

検索結果

USERID	NAME	SEX	PUR-CODE	PUR-DATE
U00555	Mike Johnson	M	P002	2011-09-06
U00687	Taro Tanaka	M	P002	2011-09-06
U00687	Taro Tanaka	M	P003	2011-09-07

1.9.3 例 3（販売履歴表と顧客表から商品を購入した顧客を検索する その 3）

販売履歴表（SALESLIST）と顧客表（USERSLIST）から，次に示すどちらかの条件に当てはまる顧客の，顧客 ID（USERID），名前（NAME），性別（SEX），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

- 2011/9/6 以降に商品を購入した男性の顧客
- 2011/9/5 以降に商品を購入した女性の顧客

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT "SALESLIST"."USERID","NAME","SEX","PUR-CODE","PUR-DATE"
    FROM "SALESLIST","USERSLIST"
        WHERE (("PUR-DATE">=DATE'2011-09-06' AND "SEX"='M')
```

```
        OR ("PUR-DATE">=DATE'2011-09-05' AND "SEX"='F'))
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
```

検索結果

USERID	NAME	SEX	PUR-CODE	PUR-DATE
U00212	Maria Gomez	F	P001	2011-09-05
U00212	Maria Gomez	F	P002	2011-09-05
U00358	Nancy White	F	P003	2011-09-05
U00555	Mike Johnson	M	P002	2011-09-06
U00687	Taro Tanaka	M	P002	2011-09-06
U00358	Nancy White	F	P002	2011-09-07
U00687	Taro Tanaka	M	P003	2011-09-07

1.10　検索結果の重複を排除する（SELECT DISTINCT）

検索結果の重複を排除したいときは，SELECT DISTINCT を使います。SELECT DISTINCT の指定形式を次に示します。

指定形式

```
SELECT DISTINCT ”列名” FROM ”表名”
    WHERE 探索条件
```

DISTINCT：

検索結果の重複を排除する場合に指定します。

> **メモ**
>
> SELECT DISTINCT の文法規則の詳細については，「7.2.1　問合せ指定の指定形式および規則」を参照してください。

1.10.1　例（商品を購入した顧客を検索する）

販売履歴表（SALESLIST）と顧客表（USERSLIST）から，2011/9/5 に商品を購入した顧客の，顧客 ID（USERID），名前（NAME）を検索します。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT DISTINCT "SALESLIST"."USERID","NAME"
    FROM "SALESLIST","USERSLIST"
        WHERE "PUR-DATE"=DATE'2011-09-05'
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
```

検索結果

USERID	NAME
U00212	Maria Gomez
U00358	Nancy White

メモ

SELECT DISTINCT を指定しない場合の検索結果は次のようになります。

指定例

```
SELECT "SALESLIST"."USERID","NAME"
    FROM "SALESLIST","USERSLIST"
        WHERE "PUR-DATE"=DATE'2011-09-05'
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
```

検索結果

USERID	NAME
U00212	Maria Gomez
U00212	Maria Gomez
U00358	Nancy White

1.11　検索対象データの件数を求める（COUNT(*)）

検索対象データの件数を求めたいときは，集合関数COUNT(*)を使います。

> **メモ**
>
> COUNT(*)の文法規則の詳細については，「7.23.3　COUNT」を参照してください。

1.11.1　例 1（顧客の総数を求める）

顧客表（USERSLIST）から，顧客総数を求めます。

検索対象表

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT COUNT(*)
    FROM "USERSLIST"
```

検索結果

COUNT(*)
5

1.11.2　例 2（商品を購入した人数を求める）

販売履歴表（SALESLIST）から，2011/9/5 以降に商品コード P003 の商品を購入した延べ人数を求めます。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT COUNT(*)
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-05'
        AND "PUR-CODE"='P003'
```

検索結果

COUNT(*)
2

1.12　検索対象データの最大値，最小値，平均値，合計値を求める（集合関数）

検索対象データの最大値，最小値，平均値，合計値を求めたいときは，MAX，MIN，AVG，SUM の集合関数を使います。

> **メモ**
>
> 集合関数の文法規則の詳細については，「7.23　集合関数」を参照してください。

1.12.1　例 1（商品の販売個数の最大値，最小値，平均値を求める）

販売履歴表（SALESLIST）から，商品コード P002 の販売個数（PUR-NUM）の最大値，最小値，平均値を求めます。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT MAX("PUR-NUM"),MIN("PUR-NUM"),AVG("PUR-NUM")
    FROM "SALESLIST"
        WHERE "PUR-CODE"='P002'
```

検索結果

MAX(PUR-NUM)	MIN(PUR-NUM)	AVG(PUR-NUM)
12	3	6
最大値	最小値	平均値

1.12.2　例 2（商品の販売個数の合計値を求める）

販売履歴表（SALESLIST）から，商品コード P002 の 2011/9/6 の販売個数（PUR-NUM）の合計値を求めます。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT SUM("PUR-NUM")
    FROM "SALESLIST"
        WHERE "PUR-CODE"='P002'
        AND "PUR-DATE"=DATE'2011-09-06'
```

検索結果

SUM(PUR-NUM)
13

合計値

1.13 グループごとに検索データを集計する（GROUP BY 句，HAVING 句）

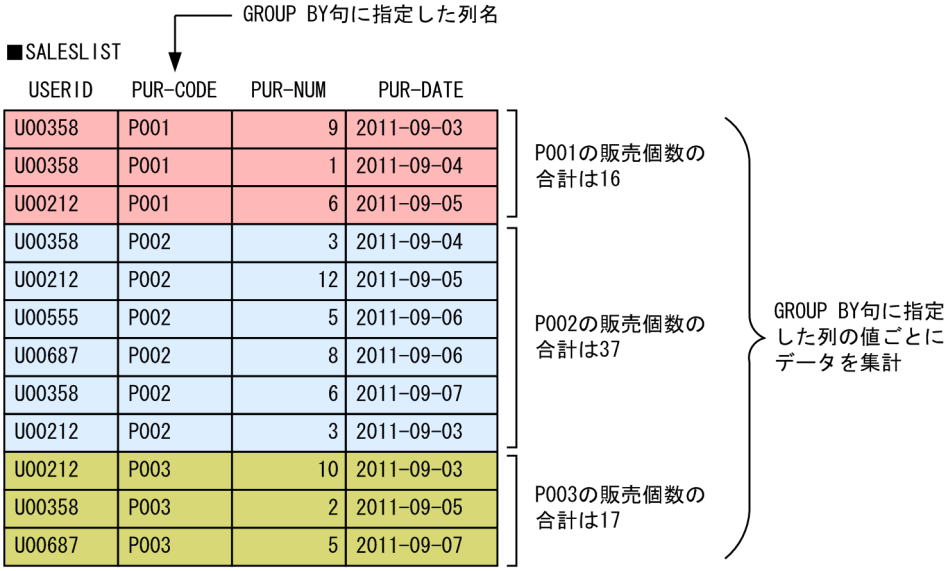
グループごとに検索データを集計したい場合は，GROUP BY 句を使います。販売履歴表（SALESLIST）を例にして，GROUP BY 句について説明します。

(例)

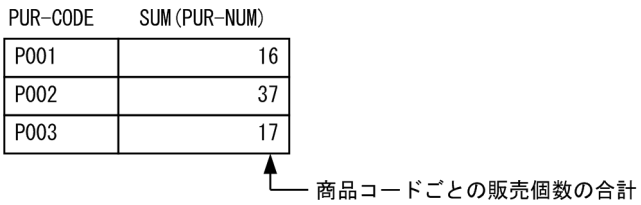
次に示す販売履歴表（SALESLIST）から，商品コード（PUR-CODE）ごとの販売個数の合計を求めます。

SELECT 文の指定

```
SELECT "PUR-CODE",SUM("PUR-NUM")
    FROM "SALESLIST"
    GROUP BY "PUR-CODE"
```



検索結果



GROUP BY 句，HAVING 句の指定形式を次に示します。

指定形式

```
SELECT "列名" FROM "表名"
    WHERE 探索条件
    GROUP BY "列名"
    HAVING 探索条件
```

GROUP BY "*列名*"：

検索データを集計する列を指定します。例えば，商品コード（PUR-CODE）ごとに検索データを集計する場合は，次のように指定します。

```
（例）GROUP BY ”PUR-CODE”
```

HAVING *探索条件*：

GROUP BY 句によってグループごとに集計された検索データを，指定した探索条件で絞り込むことができます。指定例については，「1.13.4　例４（商品コードごとに販売個数を求める（HAVING 句を指定して検索対象を絞り込む））」を参照してください。

メモ

GROUP BY 句には列名以外の指定もできます。GROUP BY 句，HAVING 句の文法規則の詳細については，次に示す該当個所を参照してください。

- GROUP BY 句：「7.7.1　GROUP BY 句の指定形式および規則」
- HAVING 句：「7.8.1　HAVING 句の指定形式および規則」

1.13.1　例１（顧客ごとの商品購入回数を求める）

販売履歴表（SALESLIST）から，顧客ごとの商品購入回数の一覧を求めます。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P001	6	2011-09-05
U00212	P002	3	2011-09-03
U00212	P002	12	2011-09-05
U00212	P003	10	2011-09-03
U00358	P001	1	2011-09-04
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04
U00358	P002	6	2011-09-07
U00358	P003	2	2011-09-05
U00555	P002	5	2011-09-06
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07

指定例

```
SELECT ”USERID”,COUNT(*)
    FROM ”SALESLIST”
        GROUP BY ”USERID”
```

検索結果

USERID	COUNT(*)
U00212	4
U00358	5
U00555	1
U00687	2

メモ

GROUP BY 句に指定した列をSELECT とFROM 句の間に指定してください。指定されていない場合，SQL エラーになります。上記の例の場合，USERID 列を指定しています。

エラーになる SQL の例を次に示します。

エラーになる SQL の例

```
SELECT "USERID","PUR-CODE",COUNT(*)
    FROM "SALESLIST"
        GROUP BY "USERID"
```

正しい SQL の例

```
SELECT "USERID","PUR-CODE",COUNT(*)
    FROM "SALESLIST"
        GROUP BY "USERID","PUR-CODE"
```

上記の SQL は，顧客（USERID）と商品コード（PUR-CODE）別に商品購入回数を求める SQL です。検索結果を次に示します。

検索結果

USERID	PUR-CODE	COUNT(*)
U00212	P001	1
U00212	P002	2
U00212	P003	1
U00358	P001	2
U00358	P002	2
U00358	P003	1
U00555	P002	1
U00687	P002	1
U00687	P003	1

1.13.2 例 2（商品コードごとに販売回数を求める）

販売履歴表（SALESLIST）から，2011/9/5 以降の商品コード（PUR-CODE）ごとの販売回数を求めます。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00358	P001	9	2011-09-03
U00212	P002	3	2011-09-03
U00212	P003	10	2011-09-03
U00358	P001	1	2011-09-04
U00358	P002	3	2011-09-04
U00212	P001	6	2011-09-05
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00555	P002	5	2011-09-06
U00358	P002	6	2011-09-07
U00358	P003	2	2011-09-05
U00687	P003	5	2011-09-07

指定例

```
SELECT "PUR-CODE",COUNT(*)
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-05'
        GROUP BY "PUR-CODE"
```

検索結果

PUR-CODE	COUNT(PUR-CODE)
P001	1
P002	4
P003	2

1.13.3 例 3（商品コードごとに販売個数の合計値と平均値を求める）

販売履歴表（SALESLIST）から，2011/9/3 以降の商品コード（PUR-CODE）ごとの販売個数の合計値，平均値を求めます。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00358	P001	9	2011-09-03
U00358	P001	1	2011-09-04
U00212	P001	6	2011-09-05
U00358	P002	3	2011-09-04
U00212	P002	12	2011-09-05
U00555	P002	5	2011-09-06
U00687	P002	8	2011-09-06
U00358	P002	6	2011-09-07
U00212	P002	3	2011-09-03
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00687	P003	5	2011-09-07

指定例

```
SELECT "PUR-CODE",SUM("PUR-NUM"),AVG("PUR-NUM")
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-03'
        GROUP BY "PUR-CODE"
```

検索結果

PUR-CODE	SUM(PUR-NUM)	AVG(PUR-NUM)
P001	16	5
P002	37	6
P003	17	5
	合計値	平均値

1.13.4 例 4（商品コードごとに販売個数を求める（HAVING 句を指定して検索対象を絞り込む））

販売履歴表（SALESLIST）から，2011/9/3 以降の商品コード（PUR-CODE）ごとの販売個数の合計値，平均値を求めます。

その際，販売個数の合計値が 20 個以下の商品コードだけを検索対象にします。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00358	P001	9	2011-09-03
U00358	P001	1	2011-09-04
U00212	P001	6	2011-09-05
U00358	P002	3	2011-09-04
U00212	P002	12	2011-09-05
U00555	P002	5	2011-09-06
U00687	P002	8	2011-09-06
U00358	P002	6	2011-09-07
U00212	P002	3	2011-09-03
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00687	P003	5	2011-09-07

指定例

```
SELECT "PUR-CODE",SUM("PUR-NUM"),AVG("PUR-NUM")
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-03'
        GROUP BY "PUR-CODE"
        HAVING SUM("PUR-NUM")<=20
```

検索結果

PUR-CODE	SUM(PUR-NUM)	AVG(PUR-NUM)
P001	16	5
P003	17	5

合計値　　平均値

販売個数の合計値が20個以下の商品コードだけが検索対象になります。

1.13.5　例５（販売履歴表と顧客表からデータを集計する）

販売履歴表（SALESLIST）と顧客表（USERSLIST）から，2011/9/4 以降の商品コード P002 の販売個数（PUR-NUM）の合計値を顧客ごとに求めます。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

■USERSLIST

USERID	NAME	SEX
U00555	Mike Johnson	M
U00358	Nancy White	F
U00212	Maria Gomez	F
U00687	Taro Tanaka	M
U00869	Bob Clinton	M

指定例

```
SELECT "NAME",SUM("PUR-NUM")
    FROM "SALESLIST","USERSLIST"
        WHERE "PUR-DATE">=DATE'2011-09-04'
        AND "PUR-CODE"='P002'
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
        GROUP BY "NAME"
```

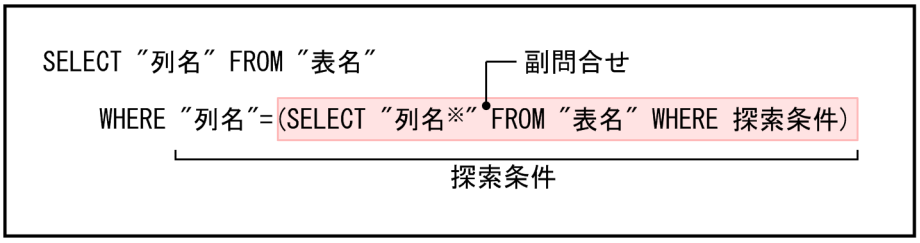
検索結果

NAME	SUM(PUR-NUM)
Maria Gomez	12
Nancy White	9
Mike Johnson	5
Taro Tanaka	8

1.14　探索条件に SELECT 文を指定して検索する（副問合せ）

探索条件にSELECT 文を指定して，表を検索できます。これによって，SELECT 文で求めた検索結果を条件にして，さらにSELECT 文で検索ができます。探索条件中に指定したSELECT 文を**副問合せ**といいます。副問合せの指定形式の例を次の図に示します。

図 1-2　副問合せの指定形式の例



注※　列名のほかに集合関数などを指定することもできます。

[説明]

副問合せに指定したSELECT 文の検索結果を条件にして，表を検索できます。

メモ

副問合せの文法規則の詳細については，「7.3.1　副問合せの指定形式および規則」を参照してください。

1.14.1　例（商品の購入数がいちばん多い顧客を求める）

販売履歴表（SALESLIST）から，商品コードP001 の 1 回当たりの商品購入数がいちばん多い顧客の，顧客 ID（USERID）と購入数（PUR-NUM）を求めます。

検索対象表

■SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2011-09-03
U00358	P001	1	2011-09-04
U00555	P002	5	2011-09-06
U00212	P003	10	2011-09-03
U00358	P003	2	2011-09-05
U00358	P002	6	2011-09-07
U00212	P002	12	2011-09-05
U00687	P002	8	2011-09-06
U00687	P003	5	2011-09-07
U00212	P001	6	2011-09-05
U00358	P001	9	2011-09-03
U00358	P002	3	2011-09-04

指定例

```
SELECT "USERID","PUR-NUM"
    FROM "SALESLIST"
        WHERE "PUR-NUM"=(SELECT MAX("PUR-NUM") FROM "SALESLIST"
                                WHERE "PUR-CODE"='P001')
```

検索結果

USERID	PUR-NUM
U00358	9

ヒント

下線部分の副問合せの指定で，販売履歴表（SALESLIST）から，商品コードP001 の 1 回当たりの最多購入数（9 個）を求めています。

次に，「購入数（PUR-NUM）＝最多購入数（9 個）」である顧客 ID（USERID）と購入数（PUR-NUM）を求めています。

1.15　よくあるSQL文の間違いと対処方法

ここでは，SQL文を実行したときによくある間違いと対処方法について説明します。

なお，ここでは，よくある間違いの代表的な対処方法だけを説明しています。そのため，ここで説明している方法では対処できないことがあります。その場合は，出力されたメッセージの対処方法に従ってください。

1.15.1　KFAA30104-Eメッセージが表示された場合

次に示すような誤りがないかを確認してください。

- 文字列をアポストロフィ（'）で囲んでいない

  条件式に指定する値がCHAR型またはVARCHAR型の文字列の場合，文字列をアポストロフィ（’）で囲む必要があります。

  （例）

```
SELECT ”USERID” FROM ”USERSLIST” WHERE ”NAME”=Taro Tanaka

KFAA30104-E There is an unnecessary part ”Tanaka” in the SQL statement.
```

  上記の例の場合，下線部分の指定に誤りがあります。正しい指定は，「’Taro Tanaka’」です。

- 列名または表名を二重引用符（"）で正しく囲んでいない

  （例）

```
SELECT ”USERID,”PUR-CODE”,”PUR-DATE” FROM ”SALESLIST”
    WHERE ”PUR-DATE”>=DATE’2011-09-06’

KFAA30104-E There is an unnecessary part ”,(0x2c)” in the SQL statement.
```

  上記の例の場合，下線部分の指定に誤りがあります。USERIDの終わりに二重引用符（”）の指定がありません。

- WHEREのつづりに誤りがある

  （例）

```
SELECT ”USERID”,”PUR-CODE”,”PUR-DATE” FROM ”SALESLIST”
    WHRER ”PUR-DATE”>=DATE’2011-09-06’

KFAA30104-E There is an unnecessary part ””PUR-DATE”>=DATE’2011-09-06’”
in the SQL statement.
```

  上記の例の場合，下線部分の指定に誤りがあります。正しい指定は，「WHERE」です。

1.15.2 KFAA30105-E メッセージが表示された場合

次に示すような誤りがないかを確認してください。

(例)

```
SELECT "USERID","PUR-CODE","PUR-DATE" FROM "SALESLIST"
    WHERE "PUR-DATE"=>DATE'2011-09-06'

KFAA30105-E Token ">"(non-reserved word), which is after token "=", is invalid.
```

下線部分の指定に誤りがあります。正しい指定は「>=」です。

KFAA30105-E メッセージのToken の後ろの部分「">"」に構文規則を満たしていない文字列が表示されます。

1.15.3 KFAA30119-E メッセージが表示された場合

SELECT の直後に指定する列とGROUP BY 句に指定する列が異なっていないかを確認してください。

(例 1)

```
SELECT "USERID",COUNT(*) FROM "SALESLIST"
    GROUP BY "PUR-CODE"

KFAA30119-E In a query using a GROUP BY clause or a set function specification,
the column "USERID" specified in a select expression, a HAVING clause
or an ORDER BY clause must be specified as an argument of the GROUP BY clause
or the set function. (query number = 1, 1)
```

上記の例の場合，下線部分の指定に誤りがあります。下線部分に指定する列名を同じにしてください。

(例 2)

```
SELECT "USERID","PUR-CODE",COUNT(*) FROM "SALESLIST"
    GROUP BY "USERID"

KFAA30119-E In a query using a GROUP BY clause or a set function specification,
the column "PUR-CODE" specified in a select expression, a HAVING clause
or an ORDER BY clause must be specified as an argument of the GROUP BY clause
or the set function. (query number = 1, 1)
```

上記の例の場合，下線部分の指定に誤りがあります。下線部分に指定する列数，列名を同じにしてください。

1.15.4 KFAA30202-E メッセージが表示された場合

指定した列名に誤りがないかを確認してください。

（例）

```
SELECT "USRID","PUR-CODE","PUR-DATE" FROM "SALESLIST"
    WHERE "PUR-DATE">=DATE'2011-09-06'

KFAA30202-E Column "USRID" is not found in any table. (query number = 1)
```

上記の例の場合，下線部分の指定に誤りがあります。正しい列名は"USERID"です。

1.15.5　KFAA30203-E メッセージが表示された場合

2 つの表に対する検索で，両方の表に同じ列名がある場合，指定した列名がどちらの表の列かを識別するために「"*表名*"."*列名*"」の形式で指定する必要があります。

（例）

```
SELECT "USERID","NAME","PUR-CODE","PUR-DATE"
    FROM "SALESLIST","USERSLIST" WHERE "PUR-DATE">=DATE'2011-09-06'
    AND "SALESLIST"."USERID"="USERSLIST"."USERID"

KFAA30203-E Column "USERID" cannot be determined in the SQL statement.
(query number = 1)
```

上記の例の場合，下線部分の指定に誤りがあります。「"*表名*"."USERID"」（"SALESLIST"."USERID"など）の形式で指定してください。

1.15.6　KFAA30204-E メッセージが表示された場合

指定した表名に誤りがないかを確認してください。

（例）

```
SELECT "USERID","PUR-CODE","PUR-DATE" FROM "SALELIST"
    WHERE "PUR-DATE">=DATE'2011-09-06'

KFAA30204-E The table or index "ADBUSER01"."SALELIST" is not found in the system.
```

上記の例の場合，下線部分の指定に誤りがあります。正しい表名は"SALESLIST"です。

1.15.7　KFAA30401-E メッセージが表示された場合

探索条件の指定などに誤りがないかを確認してください。

（例 1）

```
SELECT "USERID","PUR-CODE","PUR-DATE" FROM "SALESLIST"
    WHERE "USERID">=DATE'2011-09-06'

KFAA30401-E The data types of both operands specified in predicate
"COMPARISON" are not compatible. (query number = 1)
```

上記の例の場合，下線部分の指定に誤りがあります。USERID 列（顧客 ID）に対して 2011 年 9 月 6 日以降という比較できない条件を指定しています。正しい指定例を次に示します。

- "USERID">='U00500'
- "PUR-DATE">=DATE'2011-09-06'

（例 2）

```
SELECT "USERID","PUR-CODE","PUR-DATE" FROM "SALESLIST"
    WHERE "PUR-DATE">='2011-9-6'

KFAA30401-E The data types of both operands specified in predicate
"COMPARISON" are not compatible. (query number = 1)
```

上記の例の場合，下線部分の指定に誤りがあります。正しい指定は，「DATE'2011-09-06'」です。

1.16　目的別リファレンス

SELECT 文で使用する句，述語，関数と，例題の参照先の一覧を，検索の目的別に次の表に示します。

なお，表中の（例）の説明は，販売履歴表（SALESLIST）と顧客表（USERSLIST）に類似した表を使用していることを前提としています。

表 1-2　SELECT 文で使用する句，述語，関数と，例題の参照先の一覧

項番	検索の目的	使用する句，述語，または関数	例題の参照先
1	表の全データを見たい （例） • 全顧客の顧客情報を見たい • 商品の販売履歴情報をすべて見たい	－	「1.2　表の全行を検索する」
2	検索結果を昇順または降順に並べたい （例） • 顧客情報を顧客 ID 順に並べたい • 商品の販売履歴情報を日付順に並べたい	ORDER BY 句	「1.3　検索結果を昇順に並べる（ORDER BY 句）」
3	検索結果行数の上限を指定したい （例） • 顧客情報を何件か見たい • 商品の販売履歴情報を何件か見たい	LIMIT 句	「1.4　検索結果行数の上限を指定する（LIMIT 句）」
4	条件を指定して検索したい （例） • 昨日の商品販売履歴情報を求めたい • 特定の顧客の商品購入履歴を求めたい	WHERE 句	「1.5　探索条件を指定して検索する」
5	範囲を指定して検索したい （例） • 今週の商品販売履歴情報を求めたい	BETWEEN 述語	「1.6　検索範囲を指定して検索する（BETWEEN 述語）」
6	複数の条件のどれかに一致するデータを検索したい （例） • 商品コード P001 または P003 の商品を購入した顧客を求めたい	IN 述語	「1.7　複数の条件のどれかに一致するデータを検索する（IN 述語）」
7	特定の文字列が含まれているデータを検索したい （例） • 名字が「Johnson」の顧客の顧客情報を求めたい • 名前の頭文字が A の顧客の顧客情報を求めたい	LIKE 述語	「1.8　特定の文字列が含まれているデータを検索する（LIKE 述語）」
8	複数の表からデータを検索したい（表の結合） （例） • 昨日商品を購入した顧客の顧客情報を求めたい	WHERE 句	「1.9　複数の表を指定して検索する（表の結合）」,

項番	検索の目的	使用する句，述語，または関数	例題の参照先
			「1.10.1　例（商品を購入した顧客を検索する）」， 「1.13.5　例5（販売履歴表と顧客表からデータを集計する）」
9	検索結果の重複を排除したい （例） • 商品を購入した顧客の名前を求めたい • 売れた商品の商品コードを求めたい	SELECT DISTINCT	「1.10　検索結果の重複を排除する（SELECT DISTINCT）」
10	表の全データ件数を求めたい （例） • 顧客の総数を求めたい	集合関数COUNT(*)	「1.11.1　例1（顧客の総数を求める）」
11	検索対象行の件数を求めたい （例） • 商品を購入した人数を求めたい • 昨日の商品販売回数を求めたい • 特定の顧客の商品購入回数を求めたい	集合関数COUNT(*)	「1.11.2　例2（商品を購入した人数を求める）」
12	検索データの最大値を求めたい （例） • 商品の販売個数の最大値を求めたい	集合関数MAX	「1.12.1　例1（商品の販売個数の最大値，最小値，平均値を求める）」
13	検索データの最小値を求めたい （例） • 商品の販売個数の最小値を求めたい	集合関数MIN	
14	検索データの平均値を求めたい （例） • 商品の販売個数の平均値を求めたい	集合関数AVG	
15	検索データの合計値を求めたい （例） • 昨日の商品販売個数を求めたい • 特定の顧客の商品購入個数を求めたい	集合関数SUM	「1.12.2　例2（商品の販売個数の合計値を求める）」
16	グループごとにデータを集計したい （例） • 顧客ごとに，商品の購入回数，または商品の購入個数を求めたい • 商品コードごとに，商品の販売回数，または商品の販売個数を求めたい	GROUP BY 句 HAVING 句	「1.13　グループごとに検索データを集計する（GROUP BY 句，HAVING 句）」
17	SELECT 文で求めた検索結果を条件に指定して，検索をしたい （例）	副問合せ	「1.14　探索条件にSELECT 文を指定して検索する（副問合せ）」

項番	検索の目的	使用する句，述語，または関数	例題の参照先
	• 商品購入数がいちばん多い顧客の顧客情報を求めたい		

（凡例）－：該当しません。

2 SQL の一覧

この章では，HADB がサポートしている SQL の一覧と，SQL 構文の指定形式の読み方について説明します。

2.1 SQL の一覧

HADB がサポートしている SQL の一覧を次の表に示します。

表 2-1 HADB がサポートしている SQL の一覧

項番	分類	HADB がサポートしている SQL	説明
1	定義系 SQL	ALTER TABLE	実表の定義を変更します。
2		ALTER USER	HADB ユーザの情報を変更します。
3		ALTER VIEW	ビュー表を再作成します。
4		CREATE AUDIT	監査対象を定義します。
5		CREATE INDEX	実表の列にインデクスを定義します。
6		CREATE SCHEMA	スキーマを定義します。
7		CREATE TABLE	実表を定義します。
8		CREATE USER	HADB ユーザを作成します。
9		CREATE VIEW	ビュー表を定義します。
10		DROP AUDIT	監査対象の定義を削除します。
11		DROP INDEX	インデクスを削除します。
12		DROP SCHEMA	スキーマを削除します。
13		DROP TABLE	実表を削除します。
14		DROP USER	HADB ユーザを削除します。
15		DROP VIEW	ビュー表を削除します。
16		GRANT	HADB ユーザに権限を付与します。
17		REVOKE	HADB ユーザの権限を取り消します。
18	操作系 SQL	DELETE	行を削除します。
19		INSERT	表に行を挿入します。
20		PURGE CHUNK	チャンク内のすべての行を削除します。
21		SELECT	表のデータを検索します。
22		TRUNCATE TABLE	実表内のすべての行を削除します。
23		UPDATE	行の値を更新します。
24	制御系 SQL	COMMIT	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て，トランザクションを正常終了します。
25		ROLLBACK	トランザクションが更新したデータベースの内容を無効にして，トランザクションを取り消します。

注

上記の SQL を AP またはadbsql コマンドで実行することができます。ただし，制御系 SQL のCOMMIT およびROLLBACK については，AP で使用することはできません。

- JDBC ドライバを使用している場合は，Connection インタフェースのcommit メソッドまたはrollback メソッドを使用してください。これらのメソッドについては，マニュアル『HADB　AP 開発ガイド』を参照してください。
- ODBC ドライバを使用している場合は，ODBC 関数のSQLEndTran を使用してください。SQLEndTran については，マニュアル『HADB　AP 開発ガイド』を参照してください。
- CLI 関数を使用している場合は，a_rdb_SQLEndTran()を使用してください。a_rdb_SQLEndTran()については，マニュアル『HADB　AP 開発ガイド』を参照してください。

メモ

- SELECT 文を検索系 SQL と呼びます。
- INSERT 文，UPDATE 文，DELETE 文，PURGE CHUNK 文，およびTRUNCATE TABLE 文を更新系 SQL と呼びます。

2.2　SQL 構文の指定形式の読み方

SQL 構文の指定形式は，BNF 表記法を使って説明しています。LIKE 述語の説明を例にして，SQL 構文の指定形式の読み方を説明します。

LIKE 述語の指定形式の説明

```
LIKE述語 ：：＝一致値　〔NOT〕 LIKE パターン文字列　〔ESCAPE エスケープ文字〕　...1

  一致値 ：：＝値式                                    ...2
  パターン文字列 ：：＝値式                            ...2
  エスケープ文字 ：：＝値式                            ...2
```

：：＝の左の項目は，右の項目で示す形式で記述することを意味しています。したがって，上記の 1～2 の意味は次のようになります。

1. LIKE 述語は，「*一致値* 〔NOT〕 LIKE *パターン文字列* 〔ESCAPE *エスケープ文字*〕」の形式で記述します。
2. *一致値*，*パターン文字列*，および*エスケープ文字*は，値式の形式で記述します。

よって，LIKE 述語は，次に示す形式で記述します。

```
値式　〔NOT〕 LIKE 値式 ESCAPE 値式
```

なお，値式を説明している個所（「7.21　値式」）があります。そこで値式の指定形式を確認できます。

3 定義系 SQL

この章では，定義系 SQL の機能，指定形式，および規則について説明します。

3.1　ALTER TABLE（表定義の変更）

ここでは，ALTER TABLE 文の指定形式および規則について説明します。

3.1.1　ALTER TABLE 文の指定形式および規則

ALTER TABLE 文で次に示すことができます。

- 実表に列を追加する
- 実表の列を削除する
- 実表の列名を変更する
- 実表の表名を変更する
- 実表の列のデータ型（VARCHAR 型のデータ長）を変更する
- マルチチャンク表のチャンク数の最大値を変更する
- 配列型の列の最大要素数を大きくする
- レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更する
- アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更する

なお，1 回のALTER TABLE 文で，上記の複数の操作を同時に実行することはできません。

(1)　指定形式および説明

(a)　実表に列を追加する場合

指定形式

```
ALTER TABLE文 : :＝ALTER TABLE 表名
                        ADD COLUMN 列定義

  列定義 : :＝列名 データ型 〔NOT NULL〕 〔BRANCH {YES | NO | AUTO} 〕 〔圧縮方式指定〕
```

●*表名*

列を追加する実表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2) 表名の指定形式」を参照してください。

なお，次の表は指定できません。

- ビュー表
- ディクショナリ表
- システム表

●ADD COLUMN *列定義*

列定義：：＝*列名 データ型* 〔NOT NULL〕 〔BRANCH {YES | NO | AUTO}〕 〔*圧縮方式指定*〕

追加する列の列定義を指定します。

列追加の仕様を次に示します。

- 追加できる列は 1 列だけです。また，実表の最後の列に追加されます。
- 追加した列にはナル値が格納されます。
- 次の実表には列を追加できません。
  - 操作対象の実表の列数が上限（4,000 列）に達している場合
  - 操作対象の実表が FIX 表であり，かつ行を格納するセグメントが割り当てられている状態の場合
  - 操作対象の実表が，CREATE TABLE 文を実行した際にBRANCH ALL を指定した実表であり，かつ行を格納するセグメントが割り当てられている状態の場合

  行を格納するセグメントが割り当てられている状態については，マニュアル『HADB　システム構築・運用ガイド』の『B-tree インデクスを定義する場合の注意点（B-tree インデクスの未完状態)』を参照してください。

メモ

ALTER TABLE 文でDEFAULT 句を指定することはできません。

列名：

追加する列の列名を指定します。

表中ですでに使用されている列名は指定できません。

また，HADB によって自動的に設定される導出列名と重複する可能性があるため，列名に EXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。

データ型：

追加する列のデータ型を指定します。指定できるデータ型を次の表に示します。

表 3-1　指定できるデータ型（ALTER TABLE 文の場合）

項番	データ型	指定形式
1	INTEGER	INT またはINTEGER
2	SMALLINT	SMALLINT
3	DECIMAL	DEC〔(*m*〔,*n*〕)〕またはDECIMAL〔(*m*〔,*n*〕)〕 *m*：精度（全体の桁数） *n*：位取り（小数部の桁数） *m* を省略すると38 が仮定され，*n* を省略すると0 が仮定されます。
4	NUMERIC※3	NUMERIC〔(*m*〔,*n*〕)〕 *m*：精度（全体の桁数） *n*：位取り（小数部の桁数）

項番	データ型	指定形式
		m を省略すると38 が仮定され，*n* を省略すると0 が仮定されます。
5	DOUBLE PRECISION	DOUBLE またはDOUBLE PRECISION
6	FLOAT※4	FLOAT
7	CHARACTER	CHAR(*n*)またはCHARACTER(*n*) *n*：文字列の長さ（バイト） CHAR またはCHARACTER と指定した場合は，文字列の長さに1 が仮定されます。
8	VARCHAR※1，※2	VARCHAR(*n*) *n*：文字列の最大長（バイト）
9	DATE	DATE
10	TIME	TIME(*p*)またはTIME *p*：小数秒精度（小数秒の桁数） *p* に指定できる値は，0，3，6，9，または12 です。TIME と指定した場合は，*p* に0 が仮定されます。
11	TIMESTAMP	TIMESTAMP(*p*)またはTIMESTAMP *p*：小数秒精度（小数秒の桁数） *p* に指定できる値は，0，3，6，9，または12 です。TIMESTAMP と指定した場合は，*p* に0 が仮定されます。
12	BINARY	BINARY(*n*) *n*：バイナリデータの長さ（バイト数） BINARY と指定した場合は，バイナリデータの長さに1 が仮定されます。
13	VARBINARY※1	VARBINARY(*n*) *n*：バイナリデータの最大長（バイト数）
14	ARRAY※5	*要素データ型* ARRAY[*最大要素数*] *要素データ型*： 配列要素のデータ型を指定します。要素データ型には，数データ，文字データ，日時データ，またはバイナリデータのどれかのデータ型を指定します。要素データ型の指定形式は，項番 1～13 で説明している各データ型の指定形式の規則に従います。 *最大要素数*： 配列要素の最大数を指定します。最大要素数には，2～30,000 の符号なし整数定数を指定してください。 ＜指定例＞ • 要素データ型がCHAR(5)で，最大要素数が 20 の場合 CHAR(5) ARRAY[20] • 要素データ型がINTEGER で，最大要素数が 5 の場合 INTEGER ARRAY[5]

注※1

FIX 表に列を追加する場合，VARCHAR 型およびVARBINARY 型は指定できません。

注※2

列の定義長が 32,000 バイトを超えるVARCHAR 型は指定できません。

注※3

NUMERIC 型が指定された場合，HADB はデータ型にDECIMAL 型が指定されたと見なします。

注※4

FLOAT 型が指定された場合，HADB はデータ型にDOUBLE PRECISION 型が指定されたと見なします。

注※5

配列型のデータ型です。配列型の列を追加する際の考慮点については，マニュアル『HADB　システム構築・運用ガイド』の『配列型の列の定義【カラムストア表】』を参照してください。

> **重要**
>
> - カラムストア表に配列型の列を追加できます。ローストア表には配列型の列を追加できません。
> - B-tree インデクスまたは一意性制約が定義されている表には，配列型の列を追加できません。

データ型の詳細については，「6.2　データ型」を参照してください。

NOT NULL：

追加する列に非ナル値制約（ナル値を許さない制約）を定義する場合に指定します。

注意事項と規則を次に示します。

- 行を格納するセグメントが割り当てられている状態の実表に対しては，NOT NULL を指定できません。行を格納するセグメントが割り当てられている状態については，マニュアル『HADB　システム構築・運用ガイド』の『B-tree インデクスを定義する場合の注意点（B-tree インデクスの未完状態)』を参照してください。
- FIX 表の場合，すべての列に対して非ナル値制約が設定されます。そのため，FIX 表に列を追加する場合，NOT NULL の指定を省略しても，指定されていると見なされます。

BRANCH {YES | NO | AUTO} ：

VARCHAR 型またはVARBINARY 型の列のデータの格納方法を指定します。

BRANCH にYES またはNO を指定した方がよいケースについては，マニュアル『HADB　システム構築・運用ガイド』の『可変長データ型の列データの分岐指定（BRANCH）【ローストア表】』を参照してください。

指定できる格納方法の種類は，CREATE TABLE 文と同じです。また，BRANCH の指定を省略した場合の動作も，CREATE TABLE 文と同じです。CREATE TABLE 文でのBRANCH の詳細については，「3.7.1 CREATE TABLE 文の指定形式および規則」の「(2)　指定形式の説明」の「(d)　列定義　【共通】」を参照してください。

なお，このオプションは，次の場合には指定できません。

- 列を追加する表の表定義時に，BRANCH ALL を指定している場合

- 追加する列のデータ型がVARCHAR 型またはVARBINARY 型以外の場合
- 列を追加する表がカラムストア表の場合

圧縮方式指定：

```
圧縮方式指定 : :=COMPRESSION TYPE  {AUTO | NONE | RUNLENGTH | DICTIONARY | DELTA | DELTA_R
UNLENGTH}
```

追加する列のデータを圧縮する方式（列データの圧縮方式）を指定します。

指定できる圧縮方式の種類は，CREATE TABLE 文と同じです。また，*圧縮方式指定*の指定を省略した場合の動作も，CREATE TABLE 文と同じです。CREATE TABLE 文での*圧縮方式指定*の詳細については，「3.7.1　CREATE TABLE 文の指定形式および規則」の「(2)　指定形式の説明」の「(d)　列定義　【共通】」を参照してください。

なお，このオプションは，列を追加する表がローストア表の場合は指定できません。

(b)　実表の列を削除する場合

指定形式

```
ALTER TABLE文 : :=ALTER TABLE 表名
                    DROP 〔COLUMN〕 列名 〔削除動作〕

 削除動作 : := {CASCADE | RESTRICT}
```

●*表名*

列を削除する実表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2) 表名の指定形式」を参照してください。

なお，次の表は指定できません。

- ビュー表
- ディクショナリ表
- システム表

●DROP 〔COLUMN〕 *列名* 〔*削除動作*〕

列名：

削除対象の列の列名を指定します。削除できる列は 1 列だけです。

削除動作：

```
削除動作 : := {CASCADE | RESTRICT}
```

次のどれかの条件を満たす場合に，列を削除するかどうかを指定します。

- 削除対象の列にインデクスが定義されている場合
- 列を削除する実表に依存するビュー表が定義されている場合
- 削除対象の列に表制約が定義されている場合

削除動作の各指定の説明を次の表に示します。

<table>
<tr><th>削除動作の指定</th><th>説明</th><th>列を削除した実表に依存するビュー表の扱い</th></tr>
<tr><td>削除動作の指定を省略した場合</td><td rowspan="2">次のどれかの条件を満たす場合でも，列を削除します。
• 削除対象の列にインデクスが定義されている場合
• 列を削除する実表に依存するビュー表が定義されている場合
• 削除対象の列に表制約が定義されている場合
このとき，次に示すものも削除されます。
• 削除対象の列に定義されているインデクス
• 削除対象の列に定義されている主キー，外部キー，および削除対象の列に定義されている主キーを参照する外部キー</td><td>列を削除した実表に依存するビュー表が無効化されます。自スキーマのビュー表だけではなく，ほかのスキーマの依存するビュー表も無効化されます。</td></tr>
<tr><td>CASCADE を指定した場合</td><td>列を削除した実表に依存するビュー表が削除されます。自スキーマのビュー表だけではなく，ほかのスキーマの依存するビュー表も削除されます。</td></tr>
<tr><td>RESTRICT を指定した場合</td><td>次のどれかの条件を満たす場合は，ALTER TABLE 文をエラーにします。
• 削除対象の列にインデクスが定義されている場合
• 列を削除する実表に依存するビュー表が定義されている場合
• 削除対象の列に表制約が定義されている場合</td><td>ALTER TABLE 文がエラーになるため，依存するビュー表に影響はありません。</td></tr>
</table>

実表の列を削除する場合は，マニュアル『HADB　システム構築・運用ガイド』の『実表の列を削除する方法』を参照してください。

留意事項

- 次のどちらかの条件を満たしている場合に限り，実表の列を削除できます。
  - 列を削除する実表に行を格納するセグメントが割り当てられていない状態の場合
  - 削除対象の列がALTER TABLE 文で追加した列であり，かつ列を追加したあとに次の SQL 文またはコマンドを実行していない場合
    - DELETE 文，INSERT 文，またはUPDATE 文
    - adbidxrebuild コマンド，adbimport コマンド，adbmergechunk コマンド，または adbunarchivechunk コマンド

    なお，上記の SQL 文またはコマンドを実行した場合でも，追加した列を削除できることがあります。また，列を追加したあとに更新行のカラム化機能を有効にした場合，追加した列を削除できないことがあります。
- 次のどれかの条件に該当する場合は列を削除できません。
  - 列を削除する実表に，列が 1 列だけしかない場合
  - 列を削除する実表が FIX 表であり，かつ行を格納するセグメントが割り当てられている状態の場合
  - 列を削除する実表がCREATE TABLE 文を実行した際にBRANCH ALL を指定した実表であり，かつ行を格納するセグメントが割り当てられている状態の場合
  - 列を削除する実表がアーカイブマルチチャンク表の場合

行を格納するセグメントが割り当てられている状態については，マニュアル『HADB　システム構築・運用ガイド』の『B-tree インデクスを定義する場合の注意点（B-tree インデクスの未完状態)』を参照してください。

(c)　実表の列名を変更する場合

指定形式

```
ALTER TABLE文 : : =ALTER TABLE 表名
                    RENAME COLUMN 〔FROM〕 変更前の列名 TO 変更後の列名
```

●*表名*

列名を変更する実表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2) 表名の指定形式」を参照してください。

なお，次の表は指定できません。

- ビュー表
- ディクショナリ表
- システム表

●RENAME COLUMN 〔FROM〕 *変更前の列名* TO *変更後の列名*

変更前の列名と変更後の列名を指定します。変更前の列名が，変更後の列名に変更されます。

指定規則を次に示します。

- すでに存在する列名を変更後の列名に指定できません。
- 変更前の列名と変更後の列名に同じ列名を指定できません。
- HADB によって自動的に設定される導出列名と重複する可能性があるため，変更後の列名に EXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。

> **重要**
>
> 列名を変更した場合，列名を変更した表に依存するビュー表が無効化されます。無効化されるビュー表の確認方法については，マニュアル『HADB　システム構築・運用ガイド』の『依存するビュー表を確認する方法』を参照してください。

(d)　実表の表名を変更する場合

指定形式

```
ALTER TABLE文 : : =ALTER TABLE 表名
                    RENAME 〔TABLE〕 TO 表識別子
```

●*表名*

表名を変更する実表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，次の表は指定できません。

- ビュー表
- ディクショナリ表
- システム表

●RENAME　〔TABLE〕　TO *表識別子*

変更後の表識別子を指定します。

指定規則を次に示します。

- すでに存在する表名は指定できません。
- 変更前の表識別子と同じ表識別子を指定することはできません。

> **重要**
>
> 表名を変更した場合，表名を変更した表に依存するビュー表が無効化されます。無効化されるビュー表の確認方法については，マニュアル『HADB　システム構築・運用ガイド』の『依存するビュー表を確認する方法』を参照してください。

(e)　実表の列のデータ型（VARCHAR 型のデータ長）を変更する場合

指定形式

```
ALTER TABLE文 : :=ALTER TABLE 表名
                    {CHANGE | ALTER}　〔COLUMN〕 列名 変更後のデータ型

  変更後のデータ型 : :=データ型
```

●*表名*

列のデータ型を変更する実表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，次の表は指定できません。

- ビュー表
- ディクショナリ表
- システム表

● {CHANGE | ALTER}　〔COLUMN〕 *列名 変更後のデータ型*

次のどちらかの変更ができます。

- データ長が 1～254 バイトのVARCHAR 型の列のデータ長を，最大 255 バイトまで長くする
- データ長が 256～31,999 バイトのVARCHAR 型の列のデータ長を，最大 32,000 バイトまで長くする

メモ

データ長が 255 バイトのVARCHAR 型の列のデータ長は変更できません。

列名：

データ型を変更する列の列名を指定します。

指定規則を次に示します。

- VARCHAR 型の列を指定してください。ただし，データ長が 255 バイトのVARCHAR 型の列は指定できません。
- テキストインデクスが定義されている列は指定できません。
- 外部キーの構成列は指定できません。
- ほかの表の外部キーから参照されている主キーの構成列は指定できません。
- データ長が 254 バイト以下のVARCHAR 型の列を指定する場合，次の制限事項があります。

  複数列インデクスのインデクス構成列を指定するときは，データ型を変更したあとの複数列インデクスのインデクス構成列の合計長が 255 バイト以下である必要があります。

  複数列インデクスを構成する列の長さについては，「表 3-6　複数列インデクスを構成する列の長さ」を参照してください。
- データ長が 256 バイト以上のVARCHAR 型の列を指定する場合，次の制限事項があります。

  B-tree インデクスの構成列を指定するときは，変更後のデータ型のデータ長が次に示す条件式を満たす必要があります。

  B-treeインデクスの構成列の合計長≦MIN｛（*a*÷3）-128，4036｝（単位：バイト）

  a：B-tree インデクスを格納する DB エリアのページサイズ

  単一列インデクス，または複数列インデクスを構成する列の長さについては，「表 3-5　単一列インデクスを構成する列の長さ」，または「表 3-6　複数列インデクスを構成する列の長さ」を参照してください。

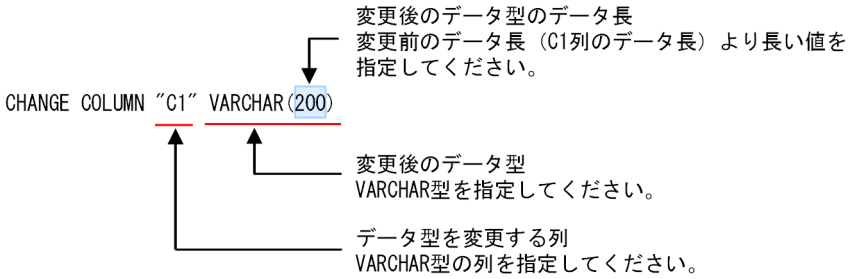
変更後のデータ型：

変更後のデータ型を指定します。

指定規則を次に示します。

- 変更後のデータ型には，VARCHAR 型を指定してください。
- 変更後のデータ型のデータ長は，変更前のデータ長より長い値を指定してください。
- 変更前のデータ型のデータ長が 254 バイト以下の場合，変更後のデータ型には，データ長が 255 バイト以下のVARCHAR 型を指定してください。
- 変更前のデータ型のデータ長が 256 バイト以上の場合，変更後のデータ型には，データ長が 32,000 バイト以下のVARCHAR 型を指定してください。

＜指定例＞



実表の列のデータ型を変更する場合は，マニュアル『HADB　システム構築・運用ガイド』の『実表の列のデータ型を変更する方法（VARCHAR 型の列のデータ長を長くする方法)』を参照してください。

重要

列のデータ型を変更した場合，列のデータ型を変更した表に依存するビュー表は，データ型を変更した列の列名をビュー表の定義時に明示的に指定しているかどうかに関係なく無効化されます。

無効化されるビュー表の確認方法については，マニュアル『HADB　システム構築・運用ガイド』の『依存するビュー表を確認する方法』を参照してください。

ビュー表の無効化を解除する方法については，マニュアル『HADB　システム構築・運用ガイド』の『ビュー表の無効化を解除する方法』の『ALTER TABLE 文で表の列のデータ型を変更したことによってビュー表が無効化された場合』を参照してください。

(f)　マルチチャンク表のチャンク数の最大値を変更する場合

指定形式

```
ALTER TABLE文 : : =ALTER TABLE 表名
                    CHANGE OPTION CHUNK=チャンク数の最大値
```

●*表名*

チャンク数の最大値を変更するマルチチャンク表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

●CHANGE OPTION CHUNK=*チャンク数の最大値*

～〈符号なし整数〉((2～30,000))（単位：個）

CREATE TABLE 文のチャンク指定で指定したチャンク数の最大値を変更する場合に，変更後のチャンク数の最大値を指定します。

注意事項を次に示します。

- このオプションは，マルチチャンク表に対してだけ指定できます。
- 表中に作成されているチャンク数よりも小さい値を指定することはできません。

(g)　配列型の列の最大要素数を大きくする場合

指定形式

```
ALTER TABLE文 : :＝ALTER TABLE 表名
                        {CHANGE | ALTER}　〔COLUMN〕 列名 ARRAY[最大要素数]
```

●*表名*

最大要素数を大きくする表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，次の表は指定できません。

- ビュー表
- ディクショナリ表
- システム表

● {CHANGE | ALTER}　〔COLUMN〕 *列名* ARRAY[*最大要素数*]

列名：

最大要素数を大きくする列の列名を指定します。

存在する配列型の列の列名を指定してください。

最大要素数：

～〈符号なし整数〉((2～30,000))

変更後の最大要素数を指定します。

変更前の最大要素数より大きい値を指定してください。

> **重要**
>
> 最大要素数を変更した場合，最大要素数を変更した表に依存するビュー表は，最大要素数を変更した列の列名をビュー表の定義時に明示的に指定しているかどうかに関係なく無効化されます。
>
> 無効化されるビュー表の確認方法については，マニュアル『HADB　システム構築・運用ガイド』の『依存するビュー表を確認する方法』を参照してください。
>
> ビュー表の無効化を解除する方法については，マニュアル『HADB　システム構築・運用ガイド』の『ビュー表の無効化を解除する方法』の『ALTER TABLE 文で配列型の列の最大要素数を変更したことによってビュー表が無効化された場合』を参照してください。

(h)　レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更する場合

指定形式

```
ALTER TABLE文 : :＝ALTER TABLE 表名
                        CHANGE OPTION CHUNK チャンクアーカイブ指定
```

```
チャンクアーカイブ指定：：＝ARCHIVABLE
                              RANGECOLUMN=列名
                              〔RANGEINDEXNAME=インデクス識別子〕
                              〔IN DBエリア名〕
                              ARCHIVEDIR=アーカイブディレクトリ名
```

●*表名*

アーカイブマルチチャンク表に変更するレギュラーマルチチャンク表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，次の表は指定できません。

- FIX 表
- シングルチャンク表
- カラムストア表

●CHANGE OPTION CHUNK *チャンクアーカイブ指定*

```
チャンクアーカイブ指定：：＝ARCHIVABLE
                              RANGECOLUMN=列名
                              〔RANGEINDEXNAME=インデクス識別子〕
                              〔IN DBエリア名〕
                              ARCHIVEDIR=アーカイブディレクトリ名
```

レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更する場合に指定するオプションです。

なお，クラウドストレージ機能を使用している場合は，このオプションは指定できません。クラウドストレージ機能については，マニュアル『HADB　システム構築・運用ガイド』の『クラウド環境で HADB サーバを使用する場合』を参照してください。

●RANGECOLUMN=*列名*

列名を指定します。ここで指定した列が，アーカイブレンジ列になります。

指定規則を次に示します。

- 次に示すデータ型の列は，アーカイブレンジ列にできません。
  - ・定義長が 33 バイト以上のCHARACTER 型
  - ・VARCHAR 型
  - ・BINARY 型
  - ・VARBINARY 型
- 非ナル値制約（NOT NULL）の指定がされている列を，アーカイブレンジ列に指定してください。

●RANGEINDEXNAME=*インデクス識別子*

ALTER TABLE 文の実行時，アーカイブレンジ列をインデクス構成列とするレンジインデクスが，HADB サーバによって自動的に定義されます。このレンジインデクスに付けるインデクス識別子を指定します。

重要

アーカイブレンジ列にすでにレンジインデクスが定義されている場合は，ALTER TABLE 文の実行時にレンジインデクスは自動的に定義されません。すでに定義されているレンジインデクスを使用します。そのため，アーカイブレンジ列にすでにレンジインデクスが定義されている場合に，RANGEINDEXNAME を指定すると，ALTER TABLE 文がエラーになります。

RANGEINDEXNAME の指定を省略した場合，HADB サーバが次の規則に従ってインデクス識別子を決定します。

```
ARCHIVE_RANGE_INDEX_nnnnnnnn
```

nnnnnnnn は，アーカイブマルチチャンク表の表 ID を 16 進数に変換した 8 桁の文字列です。

上記の規則に従って決定されたインデクス識別子が，同一スキーマ内に存在する場合，ALTER TABLE 文がエラーになります。そのため，CREATE INDEX 文でインデクスを定義する際は，上記の形式に類似した名称を使用しないことを推奨します。

メモ

自動的に定義されたレンジインデクスは，CREATE INDEX 文で定義したレンジインデクスと同じ規則が適用されます。

●IN *DB エリア名*

HADB サーバによって自動的に定義されたレンジインデクスを格納する DB エリアの名称を指定します。

なお，次に示す場合は，「IN *DB エリア名*」の指定を省略してください。

- アーカイブレンジ列にレンジインデクスがすでに定義されている場合

上記の場合は，HADB サーバによって自動的にレンジインデクスが定義されないため，「IN *DB エリア名*」の指定が不要になります。

また，次の条件をすべて満たす場合は，サーバ定義のadb_sql_default_dbarea_shared オペランドに指定した DB エリアに，HADB サーバによって自動的に定義されたレンジインデクスが格納されます。

- 「IN *DB エリア名*」の指定を省略した場合
- アーカイブレンジ列にレンジインデクスが定義されていない場合

なお，上記 2 つの条件を両方とも満たす場合に，サーバ定義のadb_sql_default_dbarea_shared オペランドが指定されていないと，ALTER TABLE 文がエラーになります。また，サーバ定義の adb_sql_default_dbarea_shared オペランドに指定した DB エリアが存在しない場合や，データ用 DB エリア以外の DB エリアを指定している場合も，ALTER TABLE 文がエラーになります。

●ARCHIVEDIR=*アーカイブディレクトリ名*

アーカイブファイルを格納するアーカイブディレクトリの名称を絶対パスで指定します。

指定規則を次に示します。

- アーカイブディレクトリ名は，文字列定数の形式で指定してください。文字列定数については，「6.3　定数」を参照してください。
- アーカイブディレクトリには，存在するディレクトリを指定してください。また，HADB 管理者に対して，読み取り権限，書き込み権限，および実行権限を与えたディレクトリを指定してください。

  さらに，アーカイブディレクトリのパスに含まれるすべてのディレクトリに，HADB 管理者に対する実行権限を付与してください。

  (例) アーカイブディレクトリが，/HADB/archive の場合

  /HADB/archive には，読み取り権限，書き込み権限，および実行権限が必要です。

  /，および/HADB には，実行権限が必要です。
- 次のディレクトリは，アーカイブディレクトリにできません。

  ・サーバディレクトリ

  ・サーバディレクトリの下位のディレクトリ

  ・下位のディレクトリにサーバディレクトリがあるディレクトリ

  ・DB ディレクトリ

  ・DB ディレクトリの下位のディレクトリ

  ・下位のディレクトリに DB ディレクトリがあるディレクトリ

  ・ルートディレクトリ

  DB ディレクトリが/HADB/db の場合，アーカイブディレクトリにできるディレクトリの例と，できないディレクトリの例を次に示します。

アーカイブディレクトリの例		理由
アーカイブディレクトリにできるディレクトリの例	/HADB/archive	なし。
アーカイブディレクトリにできないディレクトリの例	/HADB/db	左記のディレクトリは，DB ディレクトリと同一のため，アーカイブディレクトリにできません。
	/HADB/db/archive	左記のディレクトリは，DB ディレクトリの下位のディレクトリのため，アーカイブディレクトリにできません。
	/HADB	左記のディレクトリは，下位のディレクトリに DB ディレクトリがあるため，アーカイブディレクトリにできません。

- HADB サーバのインストール時にインストールデータを格納したディレクトリを，アーカイブディレクトリに指定しないようにしてください。
- アーカイブディレクトリ名は，前後の空白を除いて 1～400 バイトの長さにしてください。

> **メモ**
>
> アーカイブディレクトリ名の前後に空白がある場合，その空白は取り除かれて処理されます（空白が取り除かれた名称が仮定されます）。

- アーカイブディレクトリ名のパスに含まれる各要素は，NAME_MAX バイト以下になるようにしてください。NAME_MAX の値は，ご利用の環境によって異なります。

アーカイブディレクトリ名にシンボリックリンクを指定した場合，シンボリックリンクを解決したあとの絶対パス名が，ここで説明している規則に従っているかどうかがチェックされます。

[マルチノード機能]

マルチノード機能を使用している場合は，次のことに注意してください。

- アーカイブディレクトリは，NFS などを使用して全ノードで共有してください。また，ALTER TABLE 文の実行時点で，全ノードで共有されている必要があります。
- ALTER TABLE 文の実行時，ここで説明しているアーカイブディレクトリ名の指定規則のチェックが，プライマリノードで実行されます。セカンダリノードおよびワーカーノードでは，チェックは行われません。そのため，ALTER TABLE 文の実行後，セカンダリノードおよびワーカーノードでアーカイブディレクトリ名のチェックを行ってください。

■レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更したときに定義されるロケーション表について

ALTER TABLE 文を実行して，レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更した場合，ロケーション表とロケーション表のインデクスが HADB サーバによって自動的に定義されます。この，ロケーション表とロケーション表のインデクスは，HADB サーバが使用します。そのため，ロケーション表とロケーション表のインデクスをユーザが直接操作したり，定義変更したり，または削除したりすることはできません。ロケーション表については，マニュアル『HADB　システム構築・運用ガイド』の『アーカイブマルチチャンク表の検索』を参照してください。

ロケーション表，およびロケーション表のインデクスは，アーカイブマルチチャンク表と同じ DB エリア内に格納されます。

ロケーション表およびロケーション表のインデクスの名称は，次の表で説明している規則に従って決定されます。

表 3-2　ロケーション表およびロケーション表のインデクスの名称規則

種別	名称規則	インデクスが管理する情報	インデクス構成列
ロケーション表	"HADB"."LOCATION_TABLE_*nnnnnnnn*"	－	－
ロケーション表のインデクス	"HADB"."LOCATION_INDEX_*nnnnnnnn*_CHUNK_ID"	アーカイブファイルに対応するチャンクのチャンク ID を管理しています。	CHUNK_ID
	"HADB"."LOCATION_INDEX_*nnnnnnnn*_RANGE_01"	アーカイブファイルに格納されているデータのアーカイブレンジ列	• RANGE_MAX • RANGE_MIN

種別	名称規則	インデクスが管理する情報	インデクス構成列
		の値の範囲（上限値と下限値）を管理しています。	
	”HADB”.”LOCATION_INDEX_*nnnnnnnn*_RANGE_02”	アーカイブファイルに格納されているデータのアーカイブレンジ列の値の下限値を管理しています。	RANGE_MIN

（凡例）

－：該当しません。

注

nnnnnnnn は，アーカイブマルチチャンク表の表 ID を 16 進数に変換した 8 桁の文字列です。

ロケーション表，およびロケーション表のインデクスのスキーマ名は HADB です。

レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更する際の注意事項を次に示します。

- レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更した場合，ロケーション表，ロケーション表のインデクス，およびアーカイブレンジ列のレンジインデクスが HADB サーバによって自動的に定義されます。ただし，アーカイブレンジ列に指定した列にレンジインデクスが定義されている場合は，レンジインデクスは新たに定義されません。すでに定義されているレンジインデクスを使用します。列にレンジインデクスが定義されているかどうかを調べる方法については，マニュアル『HADB システム構築・運用ガイド』の『ディクショナリ表の検索』の『アーカイブレンジ列として指定する列にレンジインデクスが定義されているかどうかを調べる場合』を参照してください。
- レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更した場合，変更対象の表に依存するビュー表が無効化されます。無効化されるビュー表の確認方法については，マニュアル『HADB　システム構築・運用ガイド』の『依存するビュー表を確認する方法』を参照してください。

レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更する方法については，マニュアル『HADB　システム構築・運用ガイド』の『レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更する方法』を参照してください。

(i)　アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更する場合

指定形式

```
ALTER TABLE文 : : =ALTER TABLE 表名
                    CHANGE OPTION CHUNK UNARCHIVABLE
```

●*表名*

レギュラーマルチチャンク表に変更するアーカイブマルチチャンク表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

●CHANGE OPTION CHUNK UNARCHIVABLE

アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更する場合に指定するオプションです。

アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更する際の注意事項を次に示します。

- アーカイブ状態のチャンクがある場合，アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更できません。チャンクのアーカイブ状態を解除したあとに，アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更してください。
- アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更した場合，ロケーション表と，ロケーション表に定義されたインデクスが削除されますが，アーカイブレンジ列に自動的に定義されたレンジインデクスは削除されません。このレンジインデクスが不要な場合は，アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更したあとに，DROP INDEX 文でレンジインデクスを削除してください。
- アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更した場合，変更対象の表に依存するビュー表が無効化されます。無効化されるビュー表の確認方法については，マニュアル『HADB　システム構築・運用ガイド』の『依存するビュー表を確認する方法』を参照してください。

アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更する方法については，マニュアル『HADB　システム構築・運用ガイド』の『アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更する方法』を参照してください。

(2)　実行時に必要な権限

ALTER TABLE 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

(3)　規則

1. 表定義を変更できるのは，自分（HADB サーバに接続中の認可識別子の HADB ユーザ）が所有するスキーマの表だけです。ほかの HADB ユーザが所有するスキーマの表の表定義は変更できません。
2. 実表の列の長さの合計（行長）が，次の計算式を満たさない場合，列の追加または列のデータ型の変更はできません。

   ●計算式（対象の実表がローストア表の場合）

   ROWSZ（行長） ≦ *ページサイズ* － 56 （単位：バイト）

   ●計算式（対象の実表がカラムストア表の場合）

   ROWSZ（行長） ≦ *ページサイズ* － 80 （単位：バイト）

   ROWSZ（行長）を求める計算式については，マニュアル『HADB　システム構築・運用ガイド』の『行の種別ごとの格納ページ数の求め方』を参照してください。
3. コマンドの中断によって更新不可状態となった表の表定義を変更することはできません。
4. 配列型の列を定義した表は，ページサイズが 4,096 バイトのデータ用 DB エリアには格納できません。
5. 次の操作をする場合，下記の条件を満たす必要があります。
   - 配列型の列を追加する場合
   - 配列型の列を定義した表に列を追加する場合

- 配列型の列を定義した表の列のデータ型を変更する場合
- 配列型の列の最大要素数を大きくする場合

＜条件＞

1セグメントの容量（単位：バイト）≧*ARRAY_ROWSZ*（上記の操作後の表の行長）

1セグメントの容量は，adbinitコマンドまたはadbmodareaコマンドの-sオプションで指定します。

*ARRAY_ROWSZ*の求め方については，マニュアル『HADB　システム構築・運用ガイド』の『行の種別ごとの格納ページ数の求め方』の『配列型の列を定義した表の行長（変数ARRAY_ROWSZ）の求め方』を参照してください。

(4)　例題

例題1（ローストア表に列を追加する場合）

ローストア表の店舗表（SHOPSLIST）に，次の店舗メールアドレス列（EMAIL_ADDRESS）を追加します。

- 列名：EMAIL_ADDRESS
- データ型：VARCHAR(100)
- 列のデータを分岐して格納する

```
ALTER TABLE "SHOPSLIST"
    ADD COLUMN "EMAIL_ADDRESS" VARCHAR(100) BRANCH YES
```

SHOPSLIST

SHOP_CODE	RGN_CODE	SHOP_NAME	TEL_NO	ADDRESS	EMAIL_ADDRESS
S0000001	P00002	XXXXXXX	XXXXXXXXXX	XXXXXXXXXX	NULL
S0000002	P00001	XXXXXXX	XXXXXXXXXX	XXXXXXXXXX	NULL
S0000003	P00002	XXXXXXX	XXXXXXXXXX	XXXXXXXXXX	NULL

追加した列

例題2（カラムストア表に列を追加する場合）

カラムストア表のレシート表（RECEIPT）に，次の発行時刻列（ISSUE_TIME）を追加します。

- 列名：ISSUE_TIME
- データ型：TIME
- 列のデータを，差分ランレングス圧縮方式（DELTA_RUNLENGTH）で圧縮する

```
ALTER TABLE "RECEIPT"
    ADD COLUMN  "ISSUE_TIME" TIME COMPRESSION TYPE DELTA_RUNLENGTH
```

RECEIPT

RID	SHOP_CODE	RGN_CODE	EMPLOYEE_CODE	ITEM_CODE	ISSUE_TIME
XX	A0000001	R00002	XXXXXXXXXX	XXXXXXXXXX	NULL
XX	A0000002	R00001	XXXXXXXXXX	XXXXXXXXXX	NULL
XX	A0000003	R00002	XXXXXXXXXX	XXXXXXXXXX	NULL

追加した列

例題３（列を削除する場合）

店舗表（SHOPSLIST）から，店舗メールアドレス列（EMAIL_ADDRESS）を削除します。

```
ALTER TABLE "SHOPSLIST"
    DROP COLUMN "EMAIL_ADDRESS"
```

例題４（列名を変更する場合）

店舗表（SHOPSLIST）のEMAIL_ADDRESS 列の列名を，EMAIL に変更します。

```
ALTER TABLE "SHOPSLIST"
    RENAME COLUMN FROM EMAIL_ADDRESS TO EMAIL
```

例題５（実表の表名を変更する場合）

店舗表（SHOPSLIST）の表名を，STORELIST に変更します。

```
ALTER TABLE "SHOPSLIST"
    RENAME TABLE TO "STORELIST"
```

例題６（実表の列のデータ型を変更する場合）

店舗表（SHOPSLIST）の店舗メールアドレス列（EMAIL_ADDRESS）のデータ型をVARCHAR(10)から VARCHAR(20)に変更します。

```
ALTER TABLE "SHOPSLIST"
    CHANGE COLUMN "EMAIL_ADDRESS" VARCHAR(20)
```

例題７（チャンク数の最大値を変更する場合）

店舗表（SHOPSLIST）のチャンク数の最大値を300 に変更します。

```
ALTER TABLE "SHOPSLIST"
    CHANGE OPTION CHUNK=300
```

例題８（配列型の列の最大要素数を大きくする場合）

商品表（PRODUCTLIST）の配列型の列である商品コード列（PUR-CODE）の最大要素数を200 に変更します（大きくします）。

```
ALTER TABLE "PRODUCTLIST"
    CHANGE COLUMN "PUR-CODE" ARRAY[200]
```

例題９（レギュラーマルチチャンク表をアーカイブマルチチャンク表に変更する場合）

ローストア表の店舗表（SHOPSLIST）を，レギュラーマルチチャンク表からアーカイブマルチチャンク表に変更します。アーカイブレンジ列などに関する指定は，次のとおりとします。

- RECORD_DAY 列をアーカイブレンジ列とする

- /mnt/nfs/archivedir をアーカイブディレクトリとする
- HADB サーバによって自動的に定義されたレンジインデクスを格納する DB エリアを DBAREA02 とする

```
ALTER TABLE ”SHOPSLIST”
    CHANGE OPTION CHUNK ARCHIVABLE
        RANGECOLUMN=”RECORD_DAY”
        IN ”DBAREA02”
        ARCHIVEDIR=’/mnt/nfs/archivedir’
```

例題 10（アーカイブマルチチャンク表をレギュラーマルチチャンク表に変更する場合）

店舗表（SHOPSLIST）を，アーカイブマルチチャンク表からレギュラーマルチチャンク表に変更します。

```
ALTER TABLE ”SHOPSLIST”
    CHANGE OPTION CHUNK UNARCHIVABLE
```

3.2　ALTER USER（HADB ユーザの情報変更）

ここでは，ALTER USER 文の指定形式および規則について説明します。

3.2.1　ALTER USER 文の指定形式および規則

HADB ユーザの次の情報を変更します。

- パスワード
- ユーザ認証方式

(1)　指定形式

- データベース認証を使用する HADB ユーザのパスワードを変更する場合，または HADB ユーザのユーザ認証方式を PAM 認証からデータベース認証に変更する場合

```
ALTER USER文 ： ：＝ALTER USER 認可識別子 IDENTIFIED BY 変更後のパスワード
```

- HADB ユーザのユーザ認証方式をデータベース認証から PAM 認証に変更する場合

```
ALTER USER文 ： ：＝ALTER USER 認可識別子 IDENTIFIED WITH PAM
```

(2)　指定形式の説明

●*認可識別子*

ユーザ情報を変更する HADB ユーザの認可識別子を指定します。

認可識別子の指定規則については，「6.1.4　名前の指定」を参照してください。

●IDENTIFIED BY *変更後のパスワード*

変更後のパスワードを指定します。

ユーザ情報を変更する HADB ユーザがユーザ認証方式に PAM 認証を使用している場合，その HADB ユーザのユーザ認証方式はデータベース認証に変更されます。ユーザ認証方式については，マニュアル『HADB　システム構築・運用ガイド』の『ユーザ認証』を参照してください。

パスワードの指定規則を次に示します。

- パスワードに使用できる文字は，半角の英大文字，半角の英小文字，半角の数字，¥（バックスラッシュ），および次に示す半角の文字です。

  @ ` ! ” # $ % & ’ ( ) * : + ; [ ] { } , = < > | - . ^ ~ / ? _

- パスワードは文字列定数の形式で指定します。そのため，パスワードをアポストロフィで囲む必要があります。指定例を次に示します。

  （例 1）変更後のパスワードにPassword01 を指定する場合

  IDENTIFIED BY ’Password01’

（例 2）変更後のパスワードにPass'01 を指定する場合

```
IDENTIFIED BY 'Pass''01'
```

パスワードの文字列にアポストロフィがある場合，上記の例のように 1 個のアポストロフィを表すのに 2 個のアポストロフィを指定します。

文字列定数の指定規則については，「表 6-10　定数の記述形式と仮定されるデータ型」を参照してください。

- パスワードに空文字は指定できません（次の指定はできません）。

  ```
  IDENTIFIED BY ''
  ```

- パスワードは 255 文字（255 バイト）まで指定できます。

> **メモ**
>
> - JDBC ドライバを使用する場合は，パスワードに次に示す半角文字を使用しないことを推奨します。
>
>   &
>
> - ODBC ドライバを使用する場合は，パスワードに次に示す半角文字を使用しないことを推奨します。
>
>   [ ] { } ( ) , ; ? * = ! @

●IDENTIFIED WITH PAM

HADB ユーザのユーザ認証方式をデータベース認証から PAM 認証に変更する場合に指定します。

指定規則を次に示します。

- このオプションを指定する場合，認可識別子が外部ユーザ名（OS ユーザ名など）と同じである必要があります。
- すでに PAM 認証を使用している HADB ユーザには，このオプションは指定できません。
- 監査権限を持っている HADB ユーザは PAM 認証を使用できません。そのため，監査権限を持っている HADB ユーザには，このオプションは指定できません。
- 暗号管理権限を持っている HADB ユーザは PAM 認証を使用できません。そのため，暗号管理権限を持っている HADB ユーザには，このオプションは指定できません。

PAM 認証については，マニュアル『HADB　システム構築・運用ガイド』の『PAM 認証』を参照してください。

(3)　実行時に必要な権限

ALTER USER 文を実行する場合，CONNECT 権限が必要になります。

(4) 規則

1. DBA 権限を持っている HADB ユーザは，全 HADB ユーザのユーザ情報を変更できます。ただし，監査権限を持っている HADB ユーザのユーザ情報は変更できません。監査権限を持っている HADB ユーザのユーザ情報を変更できるのは，監査権限を持っている HADB ユーザ本人だけです。
2. DBA 権限を持たない HADB ユーザは，自分（HADB サーバに接続中の認可識別子の HADB ユーザ）のユーザ情報だけを変更できます。
3. ユーザ認証方式にデータベース認証を使用し，かつDBA 権限およびCONNECT 権限を持つ HADB ユーザが最低 1 人必要です。そのため，このような HADB ユーザが 1 人しかいないときに，その HADB ユーザのユーザ認証方式をデータベース認証から PAM 認証に変更できません。

(5) 例題

例題 1

データベース認証を使用している HADB ユーザADBUSER01 のパスワードを#HelloHADB_02 に変更します。

```
ALTER USER ”ADBUSER01” IDENTIFIED BY ’#HelloHADB_02’
```

例題 2

HADB ユーザOSUSER01 のユーザ認証方式をデータベース認証から PAM 認証に変更します。

```
ALTER USER ”OSUSER01” IDENTIFIED WITH PAM
```

OSUSER01 は，外部ユーザ名（OS ユーザ名など）とします。

例題 3

HADB ユーザADBUSER03 のユーザ認証方式を PAM 認証からデータベース認証に変更します。

```
ALTER USER ”ADBUSER03” IDENTIFIED BY ’#HelloHADB_03’
```

3.3　ALTER VIEW（ビュー表の再作成）

ここでは，ALTER VIEW 文の指定形式および規則について説明します。

3.3.1　ALTER VIEW 文の指定形式および規則

ビュー表を再作成します。

次に示す場合に，ALTER VIEW 文を実行してビュー表を再作成してください。

- ビュー表の無効化を解除する場合

  ビュー表の無効化要因を対策したあとに，ALTER VIEW 文を実行してビュー表を再作成すると，ビュー表の無効化が解除されます。
- ビュー表の無効化要因がわからなくなってしまった場合

  ビュー表の無効化要因を対策していない状態でALTER VIEW 文を実行すると，ALTER VIEW 文がエラーになります。そのとき出力されるエラーメッセージを参照すると，ビュー表が無効化された要因を特定できます。

> **重要**
>
> ALTER VIEW 文ではビュー表の定義を変更できません。ビュー表の定義を変更する場合は，DROP VIEW 文でいったんビュー表を削除したあとに，CREATE VIEW 文でビュー表を再定義してください。

(1)　指定形式

```
ALTER VIEW文 : :＝ALTER VIEW 表名 RECREATE
```

(2)　指定形式の説明

●*表名*

再作成するビュー表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2) 表名の指定形式」を参照してください。

次の表は指定できません。

- 実表
- ディクショナリ表
- システム表

●RECREATE

ビュー表を再作成する場合に指定します。

(3) 実行時に必要な権限

ALTER VIEW 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

(4) 規則

1. HADB サーバに接続した認可識別子と，ビュー表のスキーマ名が異なる場合，ALTER VIEW 文がエラーになります。
2. 再作成対象のビュー表に依存するビュー表が定義されている場合でも，ALTER VIEW 文を実行するとビュー表が再作成されます。このとき，再作成したビュー表に依存するビュー表は無効化されます。
3. ALTER VIEW 文でビュー表を再作成しても，再作成したビュー表に依存するビュー表に対するアクセス権限に影響はありません。
4. ALTER VIEW 文に指定したビュー表は，有効か無効かに関係なく必ず再作成されます。
5. ALTER VIEW 文でビュー表を再作成した場合，ビュー表の定義によってはビュー表の列数や列名が変わることがあります。例えば，次のようなケースが該当します。

   ビュー表V1 の定義例：

   ```
   CREATE VIEW ”V1” AS SELECT * FROM ”T1” WHERE ”C1”>100
   ```

   - ビュー表の列数が増えるケース

     1. CREATE VIEW 文でビュー表V1 を定義する
     2. ALTER TABLE 文で基表T1 に，例えばC5 列を追加する
     3. ALTER VIEW 文でビュー表V1 を再作成する

     この場合，ビュー表V1 にC5 列が追加されるため，ビュー表の列数が増えます。

   - ビュー表の列名が変わるケース

     上記の操作の 2.で，例えば，C2 列の列名をALTER TABLE 文で変更した場合，ALTER VIEW 文でビュー表V1 を再作成すると，ビュー表V1 のC2 列の列名が変わります。

6. ALTER VIEW 文でビュー表を再作成する際，基表に対するアクセス権限の内容がビュー表の定義時から変更※されている場合，再作成したビュー表に対するアクセス権限の依存権限が取り消されることがあります。

   注※

   次に示すどちらかの変更が該当します。

   - 付与権付きのアクセス権限から，付与権なしのアクセス権限に変更されている場合
   - 付与権付きのアクセス権限から，アクセス権限なしに変更されている場合

   再作成したビュー表に対するアクセス権限の依存権限が取り消される例を次に示します。

   (例)

   1.

   HADB ユーザ A は，表X.T1 に対する付与権付きのSELECT 権限を持っていて，表X.T1 を基表としたビュー表A.V1 を定義します。

2.

HADB ユーザ A は，ビュー表A.V1 に対するSELECT 権限をほかの HADB ユーザに付与します。ほかの HADB ユーザに付与したSELECT 権限は依存権限となります。

3.

HADB ユーザ A が持っている，表X.T1 に対する付与権付きのSELECT 権限が取り消されます。このとき，表X.T1 を基表としたビュー表A.V1 が無効化されます。

4.

ビュー表A.V1 の無効化を解除するために，表X.T1 に対する付与権なしのSELECT 権限を HADB ユーザ A に付与します。表X.T1 に対するアクセス権限が，1.のビュー表A.V1 の定義時に持っていた付与権付きのSELECT 権限から，付与権なしのSELECT 権限に変更されています。

5.

ALTER VIEW 文を実行してビュー表A.V1 を再作成します。

4.でSELECT 権限が付与権なしのSELECT 権限に変更されているため，2.でほかの HADB ユーザに付与した依存権限であるSELECT 権限が取り消されます。

(5) 例題

例題

店舗表のビュー表（VSHOPSLIST）が無効化されたため，ALTER VIEW 文を実行してVSHOPSLIST の無効化を解除します。

```
ALTER VIEW ”VSHOPSLIST” RECREATE
```

3.4 CREATE AUDIT（監査対象の定義）

ここでは，CREATE AUDIT 文の指定形式および規則について説明します。

なお，CREATE AUDIT 文で定義した情報を**監査対象定義**といいます。

3.4.1 CREATE AUDIT 文の指定形式および規則

監査対象の定義を行います。

> **重要**
>
> 監査証跡機能が有効な場合に，CREATE AUDIT 文を実行できます。監査証跡機能が有効かどうかは，adbauditttrail -d コマンドを実行して確認してください。

(1) 指定形式

```
CREATE AUDIT文 : : =CREATE AUDIT AUDITTYPE EVENT
                                FOR ANY OPERATION
```

(2) 指定形式の説明

●AUDITTYPE EVENT

イベントの最終結果についての監査証跡を出力する場合に指定します。

●FOR ANY OPERATION

次の表に示すイベントを監査対象とする場合に指定します。

表 3-3 監査対象とするイベント

イベントの種類	監査対象とするイベント
セッション管理イベント	CONNECT の実行（HADB サーバへの接続）
	DISCONNECT の実行（HADB サーバからの切り離し）
権限管理イベント	次に示す SQL 文の実行 • GRANT 文 • REVOKE 文 • CREATE USER 文 • DROP USER 文 • ALTER USER 文
定義系 SQL イベント	次に示す定義系 SQL の実行 • ALTER TABLE 文 • ALTER VIEW 文

イベントの種類	監査対象とするイベント
	• CREATE AUDIT 文 • CREATE INDEX 文 • CREATE SCHEMA 文 • CREATE TABLE 文 • CREATE VIEW 文 • DROP AUDIT 文 • DROP INDEX 文 • DROP SCHEMA 文 • DROP TABLE 文 • DROP VIEW 文
操作系 SQL イベント	次に示す操作系 SQL の実行 • SELECT 文 • INSERT 文 • UPDATE 文 • DELETE 文 • TRUNCATE TABLE 文 • PURGE CHUNK 文
コマンド操作イベント	次に示すコマンドの実行 • adbimport コマンド • adbexport コマンド • adbidxrebuild コマンド • adbgetcst コマンド • adbdbstatus コマンド • adbmergechunk コマンド • adbchgchunkcomment コマンド • adbchgchunkstatus コマンド • adbarchivechunk コマンド • adbunarchivechunk コマンド • adbreorgsystemdata コマンド • adbsyndict コマンド

(3) 実行時に必要な権限

CREATE AUDIT 文を実行する場合，CONNECT 権限および監査管理権限が必要になります。

(4) 規則

1. 同じ監査対象を複数定義することはできません。
2. HADB サーバは，監査証跡を出力する際の判定処理時に監査対象定義を参照します。そのため，監査証跡の出力タイミングによっては，監査対象を定義する前に実行された監査対象外の操作についての監査証跡が出力されることがあります。

(5) 例題

例題

「表 3-3　監査対象とするイベント」に示すイベントを監査対象として定義します。

```
CREATE AUDIT AUDITTYPE EVENT
             FOR ANY OPERATION
```

3.5 CREATE INDEX（インデクスの定義）

ここでは，CREATE INDEX 文の指定形式および規則について説明します。

3.5.1 CREATE INDEX 文の指定形式および規則

実表の列にインデクス（B-tree インデクス，テキストインデクス，またはレンジインデクス）を定義します。B-tree インデクス，テキストインデクス，およびレンジインデクスについては，マニュアル『HADB システム構築・運用ガイド』の『B-tree インデクス』，『テキストインデクス』，または『レンジインデクス』を参照してください。

なお，複数の列に対して B-tree インデクスを定義することができます。1 つの列に対して定義した B-tree インデクスを**単一列インデクス**といい，複数の列に対して定義した B-tree インデクスを**複数列インデクス**といいます。

重要

行を格納するセグメントが割り当てられている状態の実表に対してインデクスを定義した場合，インデクスが未完状態（インデクスデータが作成されない状態）になります。

例えば，次に示すタイミングでは，行を格納するセグメントが割り当てられていない状態です。この状態のときに実表にインデクスを定義すると，インデクスは正常に作成されます。

- 実表の定義直後
- TRUNCATE TABLE 文の実行直後

B-tree インデクスが未完状態の場合は，未完状態の B-tree インデクスを使用した検索や，表に対するINSERT，UPDATE，およびDELETE が実行できません。

テキストインデクスが未完状態の場合は，未完状態のテキストインデクスを使用した検索や，表に対するINSERT，UPDATE，およびDELETE が実行できません。

レンジインデクスが未完状態の場合は，未完状態のレンジインデクスを使用した検索や，表に対するINSERT およびUPDATE が実行できません。

インデクスの未完状態の解除方法については，マニュアル『HADB　システム構築・運用ガイド』の『B-tree インデクスが未完状態になったときの対処方法』，『テキストインデクスが未完状態になったときの対処方法』，または『レンジインデクスが未完状態になったときの対処方法』を参照してください。

行を格納するセグメントが割り当てられている状態については，マニュアル『HADB　システム構築・運用ガイド』の『B-tree インデクスを定義する場合の注意点（B-tree インデクスの未完状態)』を参照してください。

(1) 指定形式

```
CREATE INDEX文 : : =
     CREATE 〔UNIQUE〕 INDEX インデクス名
          ON 表名 (列名 〔 {ASC | DESC} 〕 〔,列名 〔 {ASC | DESC} 〕〕 …)
         〔IN DBエリア名〕
         〔PCTFREE=未使用領域の比率〕
          EMPTY
         〔INDEXTYPE {BTREE | TEXT 〔WORDCONTEXT〕 | RANGE} 〕
         〔CORRECTIONRULE〕
         〔DELIMITER {DEFAULT | ALL} 〕
         〔EXCLUDE NULL VALUES〕
```

メモ

- PCTFREE，EMPTY，INDEXTYPE，CORRECTIONRULE，DELIMITER，およびEXCLUDE NULL VALUES をまとめて**インデクスオプション**といいます。
- インデクスオプションは，どの順序で指定してもかまいません。

定義するインデクスの種類によって指定できるオプションが次の表に示すように異なります。

表 3-4　インデクス定義時に指定できるオプション

項番	CREATE INDEX の各オプション	B-tree インデクスを定義する場合	テキストインデクスを定義する場合	レンジインデクスを定義する場合
1	UNIQUE	○	×	×
2	*インデクス名*	○	○	○
3	ON *表名*	○	○	○
4	*列名*	○	○	○
5	{ASC | DESC}	○	×	×
6	IN *DB エリア名*	○	○	○
7	PCTFREE	○	○	×
8	EMPTY	○	○	○
9	INDEXTYPE	○	○	○
10	CORRECTIONRULE	×	○	×
11	DELIMITER	×	○	×
12	EXCLUDE NULL VALUES	○	×	×

(凡例)

○：指定が必要なオプション，または指定を検討するオプションです。

×：指定不要なオプションです。

> **メモ**
>
> CREATE INDEX 文で主キーを定義することはできません。主キーを定義するには，CREATE TABLE 文で一意性制約定義を指定してください。

(2) 指定形式の説明

各オプションの説明で，【B-tree インデクス】と表記されているオプションは，B-tree インデクスを定義するときに指定できるオプションです。【テキストインデクス】と表記されているオプションは，テキストインデクスを定義するときに指定できるオプションです。【レンジインデクス】と表記されているオプションは，レンジインデクスを定義するときに指定できるオプションです。【共通】と表記されているオプションは，B-tree インデクス，テキストインデクス，およびレンジインデクス共通のオプションです。

●UNIQUE　【B-tree インデクス】

B-tree インデクスをユニークインデクスとして定義する場合に指定します。ユニークインデクスとは，キー値（B-tree インデクスを定義した列の値）の重複を許さない B-tree インデクスのことです。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。

複数列インデクスの場合は，どれか 1 つの列の値が異なれば，異なるキー値となります。

UNIQUE を指定した場合，キー値が重複するようなデータの更新または追加ができません。

なお，CREATE TABLE 文でチャンク指定を指定した実表に，ユニークインデクスは定義できません。

●*インデクス名*　【共通】

定義するインデクスのインデクス名を指定します。インデクス名の指定規則については，「6.1.5　名前の修飾」の「(3)　インデクス名の指定形式」を参照してください。

なお，すでに定義されているインデクスのインデクス名は指定できません。

●ON *表名*　【共通】

インデクスを定義する実表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，*表名* にビュー表は指定できません。

●(*列名* 〔 {ASC | DESC} 〕 〔,*列名* 〔 {ASC | DESC} 〕〕…)　【共通】

・B-tree インデクスの場合

B-tree インデクスを定義する列の列名，および B-tree インデクスのキー値の並び順を指定します。

列名：

B-tree インデクスを定義する列の列名を指定します。列名は，16 個まで指定できます。列名を複数指定する場合，同じ列名を指定できません。

列名を複数指定した場合，その B-tree インデクスは複数列インデクスになります。

ASC：

B-tree インデクスのキー値を昇順に並べる場合に指定します。

DESC：

B-tree インデクスのキー値を降順に並べる場合に指定します。

単一列インデクスに対してDESC を指定しても無視されます。インデクスのキー値は，常に昇順に並べられます（ASC が指定されたと仮定されます）。

ASC およびDESC の指定を省略した場合，ASC が仮定されます。

・テキストインデクスまたはレンジインデクスの場合

テキストインデクスまたはレンジインデクスを定義する列の列名を指定します。

テキストインデクスおよびレンジインデクスの場合，列名を 2 つ以上指定することはできません。また，ASC およびDESC を指定することはできません。

そのため，テキストインデクスおよびレンジインデクスの場合，指定形式は次のようになります。

(*列名*)

●IN *DB エリア名*　【共通】

インデクスを格納する DB エリアの DB エリア名を指定します。

「IN *DB エリア名*」の指定を省略した場合，サーバ定義のadb_sql_default_dbarea_shared オペランドに指定した DB エリアに，インデクスが格納されます。

なお，次のどちらかの条件を満たす場合に，「IN *DB エリア名*」の指定を省略すると，CREATE INDEX 文がエラーになります。

- サーバ定義のadb_sql_default_dbarea_shared オペランドの指定を省略している場合
- サーバ定義のadb_sql_default_dbarea_shared オペランドに，存在しない DB エリアを指定している場合，またはデータ用 DB エリア以外の DB エリアを指定している場合

●PCTFREE=*未使用領域の比率*　【B-tree インデクス，テキストインデクス】

～〈符号なし整数〉((0～99))《30》（単位：%）

B-tree インデクスまたはテキストインデクス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。0～99（単位：%）を指定します。省略すると，30%が仮定されます。

データがインポートされてインデクスが作成される際，またはインデクスが再作成される際，ここで指定した未使用領域の比率に従って B-tree インデクスまたはテキストインデクスのデータが格納されます。

インデクスページ内の未使用領域の比率の目安については，マニュアル『HADB　システム構築・運用ガイド』の『B-tree インデクスのインデクスページ内の未使用領域の確保（PCTFREE)』，または『テキストインデクスのインデクスページ内の未使用領域の確保（PCTFREE)』を参照してください。

なお，PCTFREE は複数回指定できません。

●EMPTY　【共通】

EMPTY は必ず指定してください。EMPTY を省略すると，CREATE INDEX 文が実行できません。

EMPTY は複数回指定できません。

●INDEXTYPE {BTREE | TEXT 〔WORDCONTEXT〕 | RANGE}　【共通】

定義するインデクスの種類を指定します。

BTREE：

インデクスを B-tree インデクスとして定義する場合に指定します。

TEXT 〔WORDCONTEXT〕：

インデクスをテキストインデクスとして定義する場合に指定します。ワード検索用のテキストインデクスを定義する場合は，TEXT WORDCONTEXT と指定します。

RANGE：

インデクスをレンジインデクスとして定義する場合に指定します。

INDEXTYPE の指定を省略した場合，BTREE（B-tree インデクス）が仮定されます。

なお，INDEXTYPE は複数回指定できません。

●CORRECTIONRULE　【テキストインデクス】

表記ゆれ補正検索に対応しているテキストインデクスを定義する場合に指定します。テキストインデクスの表記ゆれ補正検索については，マニュアル『HADB　システム構築・運用ガイド』の『表記ゆれ補正検索』を参照してください。

なお，HADB サーバで使用する文字コードが Shift-JIS の場合（環境変数ADBLANG の指定値が SJIS の場合），表記ゆれ補正検索ができません。そのため，CORRECTIONRULE は指定できません。

また，CORRECTIONRULE は複数回指定できません。

> **メモ**
>
> このオプションの指定を，**テキストインデクス表記ゆれ補正指定**といいます。

●DELIMITER　{DEFAULT｜ALL}　【テキストインデクス】

ワード検索をする際の単語の区切り文字の種類を指定します。

DEFAULT：

ワード検索をする際，次の文字を区切り文字として扱います。

- 半角空白（0x20）
- タブ（0x09）
- 改行（0x0A）
- 復帰（0x0D）
- ピリオド（0x2E）
- 疑問符（0x3F）
- 感嘆符（0x21）

ALL：

ワード検索をする際，次の文字を区切り文字として扱います。

- 半角空白（0x20）
- タブ（0x09）

- 改行（0x0A）
- 復帰（0x0D）
- ピリオド，疑問符，および感嘆符を含む 1 バイトの記号（0x21～0x2F，0x3A～0x40，0x5B～0x60，0x7B～0x7E）

このオプションを指定する場合は，INDEXTYPE にTEXT WORDCONTEXT を指定している必要があります。

INDEXTYPE にTEXT WORDCONTEXT を指定している場合に，DELIMITER の指定を省略したときは，DEFAULT が仮定されます。

> **メモ**
>
> このオプションの指定を**テキストインデクス区切り文字指定**といいます。

●EXCLUDE NULL VALUES　【B-tree インデクス】

このオプションを指定すると，B-tree インデクスを作成する際，ナル値だけで構成される B-tree インデクスのキー値を作成しません。列の大半の値がナル値の列をインデクス構成列にする場合は，このオプションの指定を検討してください。

このオプションを指定すると，ナル値だけで構成される B-tree インデクスのキー値を作成しない分，B-tree インデクスの作成時間を短縮できます。そのため，データインポートに掛かる時間を短縮できたり，B-tree インデクスのデータ容量を削減できたりなどのメリットがあります。

なお，B-tree インデクスのインデクス構成列に，非ナル値制約が定義されている場合，その B-tree インデクスに対してはこのオプションを指定できません。

また，EXCLUDE NULL VALUES は複数回指定できません。

> **メモ**
>
> このオプションの指定を**ナル値除外指定**といいます。

(3)　実行時に必要な権限

CREATE INDEX 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

(4)　規則

(a)　インデクス共通の規則

1. 自分（HADB サーバに接続中の認可識別子の HADB ユーザ）が所有している実表に対してだけインデクスを定義できます。ほかの HADB ユーザが所有している実表に対してはインデクスを定義できません。
2. ビュー表にインデクスは定義できません。
3. インデクスは，B-tree インデクス，テキストインデクス，およびレンジインデクスを合わせて，1 つの表に 64 個まで定義できます。

4. インデクスは，B-tree インデクス，テキストインデクス，およびレンジインデクスを合わせて，システム内で合計 8,192 個（ディクショナリ表（実表）およびシステム表（実表）に定義されたインデクスは除く）まで定義できます。
5. 1 つの DB エリアに格納できるインデクスは 400 個までです。
6. 同一列に対して，B-tree インデクス（単一列インデクス），テキストインデクス，およびレンジインデクスを定義できます。
7. マルチチャンク表にインデクスを定義する場合は，マニュアル『HADB　システム構築・運用ガイド』の『データ用 DB エリアにマルチチャンク表を格納する場合の考慮点』を参照してください。
8. コマンドの中断によって更新不可状態となった表にインデクスを定義することはできません。

(b)　B-tree インデクスに関する規則

1. 単一列インデクスを定義する場合は，次に示す条件式を満たす必要があります。条件式を満たさない場合は単一列インデクスを定義できません。

単一列インデクスを定義する列の長さ≦MIN｛（*a*÷3）-128，4036｝（単位：バイト）

a：B-tree インデクスを格納する DB エリアのページサイズ

単一列インデクスを定義する列の長さは，次の表から求めてください。

表 3-5　単一列インデクスを構成する列の長さ

項番	列のデータ型		列の長さ（単位：バイト）
1	INTEGER		8
2	SMALLINT		4
3	DECIMAL(*m*,*n*) または NUMERIC(*m*,*n*)	1≦*m*≦4 の場合	2
		5≦*m*≦8 の場合	4
		9≦*m*≦16 の場合	8
		17≦*m*≦38 の場合	16
4	DOUBLE PRECISION またはFLOAT		8
5	CHAR(*n*)		*n*
6	VARCHAR(*n*)		*n*
7	DATE		4
8	TIME(*p*)		3＋↑*p*÷2↑
9	TIMESTAMP(*p*)		7＋↑*p*÷2↑
10	BINARY(*n*)		*n*
11	VARBINARY(*n*)		*n*

（凡例）

m，*n*：正の整数

p：0，3，6，9，または12

2. 複数列インデクスを定義する場合は，次に示す条件式を満たす必要があります。条件式を満たさない場合は複数列インデクスを定義できません。

複数列インデクスを構成する列の定義長の合計 ≦MIN｛（*a*÷3）-128，4036｝（単位：バイト）

a：B-tree インデクスを格納する DB エリアのページサイズ

複数列インデクスを構成する列の定義長の合計は，次の表から求めてください。

表 3-6　複数列インデクスを構成する列の長さ

項番	列のデータ型		複数列インデクスを構成する列の長さ（単位：バイト）※		
			各列の定義長の合計が 255 バイト以下の場合	各列の定義長の合計が 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
1	INTEGER		9	9	10
2	SMALLINT		5	5	6
3	DECIMAL(*m*,*n*) または NUMERIC(*m*,*n*)	1≦*m*≦4 の場合	3	3	4
		5≦*m*≦8 の場合	5	5	6
		9≦*m*≦16 の場合	9	9	10
		17≦*m*≦38 の場合	17	17	18
4	DOUBLE PRECISION またはFLOAT		9	9	10
5	CHARACTER(*n*)		*n* ＋ 1	*n* ＋ 1	*n* ＋ 2
6	VARCHAR(*n*)		*n* ＋ 1	－	*n* ＋ 2
7	DATE		5	5	6
8	TIME(*p*)		4 ＋ ↑*p*÷2↑	4 ＋ ↑*p*÷2↑	5 ＋ ↑*p*÷2↑
9	TIMESTAMP(*p*)		8 ＋ ↑*p*÷2↑	8 ＋ ↑*p*÷2↑	9 ＋ ↑*p*÷2↑
10	BINARY(*n*)		*n* ＋ 1	*n* ＋ 1	*n* ＋ 2
11	VARBINARY(*n*)		*n* ＋ 1	－	*n* ＋ 2

（凡例）

m，*n*：正の整数

p：0，3，6，9，または12

－：該当しません。

注※

『各列の定義長の合計が 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，『各列の定義長の合計が 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。

3. 次に示す B-tree 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 B-tree 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 B-tree インデクス
4. 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. 複数列インデクスを定義する場合，各列の指定順序が，キー値作成の優先順位になります。
6. 配列型の列を定義している表には，B-tree インデクスを定義できません。

(c) テキストインデクスに関する規則

1. 次に示すデータ型の列に対してテキストインデクスを定義できます。
   - CHARACTER 型
   - VARCHAR 型
2. インデクス構成列が同じテキストインデクスを複数定義できません。
3. カラムストア表にテキストインデクスは定義できません。
4. クラウドストレージ機能を使用している場合は，テキストインデクスは定義できません。クラウドストレージ機能については，マニュアル『HADB　システム構築・運用ガイド』の『クラウド環境で HADB サーバを使用する場合』を参照してください。

(d) レンジインデクスに関する規則

1. レンジインデクスは，次に示すデータ型の列に対しては定義できません。
   - 定義長が 33 バイト以上のCHARACTER 型
   - VARCHAR 型
   - BINARY 型
   - VARBINARY 型
2. インデクス構成列が同じレンジインデクスを複数定義できません。

(5) 例題

例題 1（B-tree インデクスを定義する場合）

店舗表（SHOPSLIST）に対して，次に示す B-tree インデクスを定義します。

- 店舗コード列（SHOP_CODE）に単一列インデクス（SHOP_CODE_IDX）を定義する

- B-tree インデクスはユニークインデクスとする
- B-tree インデクスを DB エリア（DBAREA01）に格納する
- 店舗表（SHOPSLIST）には行の追加が頻繁に発生するため，インデクスページ内の未使用領域の比率を 50 パーセントとする

```
CREATE UNIQUE INDEX "SHOP_CODE_IDX"
      ON "SHOPSLIST" ("SHOP_CODE")
    IN "DBAREA01"
    PCTFREE = 50
    EMPTY
```

例題 2（B-tree インデクスを定義する場合）

店舗表（SHOPSLIST）に対して，次に示す B-tree インデクスを定義します。

- 店舗コード列（SHOP_CODE）と地域コード列（RGN_CODE）をインデクス構成列とした複数列インデクス（SHOP_RGN_IDX）を定義する
- 店舗コードは昇順（ASC）に，地域コードは降順（DESC）にインデクスのキー値を並べる
- B-tree インデクスを DB エリア（DBAREA01）に格納する

```
CREATE INDEX "SHOP_RGN_IDX"
      ON "SHOPSLIST" ("SHOP_CODE" ASC,"RGN_CODE" DESC)
    IN "DBAREA01"
    EMPTY
```

例題 3（テキストインデクスを定義する場合）

従業員表（EMPLOYEE）に対して，次に示すテキストインデクスを定義します。

- 住所列（ADDRESS）にテキストインデクス（ADDRESS_IDX）を定義する

```
CREATE INDEX "ADDRESS_IDX"
      ON "EMPLOYEE" ("ADDRESS")
    IN "DBAREA01"
    EMPTY
    INDEXTYPE TEXT
```

表記ゆれ補正検索ができるテキストインデクスを定義する場合は，次のように下線部分の指定が必要になります。

```
CREATE INDEX "ADDRESS_IDX"
      ON "EMPLOYEE" ("ADDRESS")
    IN "DBAREA01"
    EMPTY
    INDEXTYPE TEXT
    CORRECTIONRULE
```

ワード検索用のテキストインデクスを定義する場合は，次のように下線部分の指定が必要になります。

```
CREATE INDEX "ADDRESS_IDX"
      ON "EMPLOYEE"("ADDRESS")
    IN "DBAREA01"
    EMPTY
```

```
    INDEXTYPE TEXT WORDCONTEXT
    DELIMITER DEFAULT
```

例題４（レンジインデクスを定義する場合）

店舗表（SHOPSLIST）に対して，次に示すレンジインデクスを定義します。

- 店舗コード列（SHOP_CODE）にレンジインデクス（SHOP_CODE_RIDX）を定義する
- レンジインデクスを DB エリア（DBAREA01）に格納する

```
CREATE INDEX "SHOP_CODE_RIDX"
      ON "SHOPSLIST" ("SHOP_CODE")
    IN "DBAREA01"
    EMPTY
    INDEXTYPE RANGE
```

3.6　CREATE SCHEMA（スキーマの定義）

ここでは，CREATE SCHEMA 文の指定形式および規則について説明します。

3.6.1　CREATE SCHEMA 文の指定形式および規則

スキーマを定義します。

(1)　指定形式

CREATE SCHEMA文：：＝CREATE SCHEMA 〔*スキーマ名*〕

(2)　指定形式の説明

●*スキーマ名*

定義するスキーマの*スキーマ名*を指定します。スキーマ名には，自分（HADB サーバに接続中の認可識別子の HADB ユーザ）の認可識別子を指定します。

スキーマ名を省略した場合，CREATE SCHEMA 文を実行した HADB ユーザの認可識別子が仮定されます。

スキーマ名の指定規則については，「6.1.5　名前の修飾」の「(1)　スキーマ名の指定形式」を参照してください。

なお，*スキーマ名*にALL，HADB，MASTER，およびPUBLIC は指定できません。

(3)　実行時に必要な権限

CREATE SCHEMA 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

(4)　規則

1. 1HADB ユーザが所有できるスキーマは 1 つだけです。
2. 自分（HADB サーバに接続中の認可識別子の HADB ユーザ）のスキーマだけを定義できます。ほかの HADB ユーザのスキーマは定義できません。例えば，認可識別子にADBUSER01 を指定してadbsql コマンドを実行した場合，CREATE SCHEMA 文で定義できるスキーマはADBUSER01 のスキーマだけです。

(5)　例題

例題

スキーマ名ADBUSER01 のスキーマを定義します。

```
CREATE SCHEMA ”ADBUSER01”
```

3.7 CREATE TABLE（表の定義）

ここでは，CREATE TABLE 文の指定形式および規則について説明します。

3.7.1 CREATE TABLE 文の指定形式および規則

実表を定義します。

(1) 指定形式

```
CREATE TABLE文 : : =
     CREATE 〔FIX〕 TABLE 表名(表要素 〔,表要素〕 …)
        〔IN DBエリア名〕
        〔PCTFREE=未使用領域の比率〕 ※
        〔BRANCH ALL〕 ※
        〔チャンク指定〕 ※
        〔STORAGE FORMAT {ROW | COLUMN} 〕 ※


  表要素 : : = {列定義 | 表制約}

    列定義 : : =列名 データ型 〔DEFAULT句〕 〔NOT NULL〕 〔BRANCH {YES | NO | AUTO} 〕
                                〔圧縮方式指定〕
      DEFAULT句 : : =DEFAULT 既定値選択肢
        既定値選択肢 : : = {定数 | CURRENT_DATE | CURRENT_TIME 〔(p)〕
                          | CURRENT_TIMESTAMP 〔(p)〕 | CURRENT_USER | NULL}

      圧縮方式指定 : : =COMPRESSION TYPE {AUTO | NONE | RUNLENGTH | DICTIONARY | DELTA
                                        | DELTA_RUNLENGTH}


    表制約 : : = {一意性制約定義 | 参照制約定義}
      一意性制約定義 : : = 〔CONSTRAINT 制約名〕 PRIMARY KEY (列名 〔 {ASC | DESC} 〕
                                〔,列名 〔 {ASC | DESC} 〕 〕 …)
                                〔IN DBエリア名〕
                                〔PCTFREE=未使用領域の比率〕

      参照制約定義 : : = 〔CONSTRAINT 制約名〕 FOREIGN KEY (列名 〔,列名〕 …)
                                REFERENCES 表名 DISABLE


  チャンク指定 : : =CHUNK〔=チャンク数の最大値〕
                          〔チャンクアーカイブ指定〕
    チャンクアーカイブ指定 : : =ARCHIVABLE
                                 RANGECOLUMN=列名
                                 〔RANGEINDEXNAME=インデクス識別子〕
                                 〔IN DBエリア名〕
                                 ARCHIVEDIR=アーカイブディレクトリ名
```

注※

PCTFREE，BRANCH ALL，*チャンク指定*，およびSTORAGE FORMAT は，どの順序で指定してもかまいません。

> **メモ**
>
> PCTFREE，BRANCH ALL，*チャンク指定*，およびSTORAGE FORMAT をまとめて**表オプション**といいます。

ローストア表を定義するときに指定できるオプションと，カラムストア表を定義するときに指定できるオプションを次の表に示します。

表 3-7　ローストア表またはカラムストア表の定義時に指定できるオプション

項番	CREATE TABLE の各オプション		ローストア表を定義する場合	カラムストア表を定義する場合
1	FIX		○	×
2	*表名*		○	○
3	*列定義*	*列名*	○	○
4		*データ型*	○	○
5		*DEFAULT 句*	○	○
6		NOT NULL	○	○
7		BRANCH	○	×
8		*圧縮方式指定*	×	○
9	*表制約*		○	○
10	IN *DB エリア名*		○	○
11	PCTFREE		○	×
12	BRANCH ALL		○	×
13	*チャンク指定*	*チャンク数の最大値*	○	○
14		*チャンクアーカイブ指定*	○	×
15	STORAGE FORMAT		○	○

（凡例）

○：指定が必要なオプション，または指定を検討するオプションです。

×：指定できないオプションです。

(2)　指定形式の説明

各オプションの説明で，【ローストア表】と表記されているオプションは，ローストア表を定義するときに指定できるオプションです。【カラムストア表】と表記されているオプションは，カラムストア表を定義するときに指定できるオプションです。【共通】と表記されているオプションは，ローストア表およびカラムストア表共通のオプションです。

重要

表に配列型の列を定義する場合，その表はカラムストア表として定義する必要があります。

(a) FIX 【ローストア表】

行長が固定となる実表（FIX 表）を定義します。

指定規則を次に示します。

- FIX を指定した場合，この表に対して次に示すデータ型は指定できません。
  - VARCHAR
  - VARBINARY
- FIX を指定した場合，実表のすべての列に非ナル値制約が設定されます。
- FIX 表に対してだけ，行単位（ROW 指定）の参照，更新，挿入ができます。
- アーカイブマルチチャンク表の場合，FIX は指定できません。

(b) 表名 【共通】

定義する実表の表名を指定します。すでに定義されている表と同じ表名は指定できません。表名の指定規則については，「6.1.5 名前の修飾」の「(2) 表名の指定形式」を参照してください。

(c) 表要素 【共通】

表要素：：＝｛*列定義* | *表制約*｝

表要素には，列定義または表制約のどちらかを指定します。

(d) 列定義 【共通】

列定義：：＝*列名* *データ型* 〔*DEFAULT句*〕 〔NOT NULL〕 〔BRANCH {YES | NO | AUTO}〕
〔*圧縮方式指定*〕

実表を構成する各列の定義を指定します。列定義は 1 つ以上指定する必要があります。

●*列名* 【共通】

表を構成する列の列名を指定します。1 つの表に同じ列名を指定できません。

なお，HADB によって自動的に設定される導出列名と重複する可能性があるため，列名に EXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。

●*データ型* 【共通】

列のデータ型を指定します。指定できるデータ型を次の表に示します。

表 3-8　指定できるデータ型（CREATE TABLE 文の場合）

項番	データ型	指定形式
1	INTEGER	INT またはINTEGER
2	SMALLINT	SMALLINT
3	DECIMAL	DEC〔(*m*〔,*n*〕)〕またはDECIMAL〔(*m*〔,*n*〕)〕 *m*：精度（全体の桁数） *n*：位取り（小数部の桁数） *m* を省略すると38 が仮定され，*n* を省略すると0 が仮定されます。
4	NUMERIC※1	NUMERIC〔(*m*〔,*n*〕)〕 *m*：精度（全体の桁数） *n*：位取り（小数部の桁数） *m* を省略すると38 が仮定され，*n* を省略すると0 が仮定されます。
5	DOUBLE PRECISION	DOUBLE またはDOUBLE PRECISION
6	FLOAT※2	FLOAT
7	CHARACTER	CHAR(*n*)またはCHARACTER(*n*) *n*：文字列の長さ（バイト） CHAR またはCHARACTER と指定した場合は，文字列の長さに 1 が仮定されます。
8	VARCHAR※3	VARCHAR(*n*) *n*：文字列の最大長（バイト）
9	DATE	DATE
10	TIME	TIME(*p*)またはTIME *p*：小数秒精度（小数秒の桁数） *p* に指定できる値は，0，3，6，9，または12 です。TIME と指定した場合は，*p* に0 が仮定されます。
11	TIMESTAMP	TIMESTAMP(*p*)またはTIMESTAMP *p*：小数秒精度（小数秒の桁数） *p* に指定できる値は，0，3，6，9，または12 です。TIMESTAMP と指定した場合は，*p* に0 が仮定されます。
12	BINARY	BINARY(*n*) *n*：バイナリデータの長さ（バイト数） BINARY と指定した場合は，バイナリデータの長さに 1 が仮定されます。
13	VARBINARY	VARBINARY(*n*) *n*：バイナリデータの最大長（バイト数）
14	ARRAY※4	*要素データ型* ARRAY[*最大要素数*] *要素データ型*： 配列要素のデータ型を指定します。要素データ型には，数データ，文字データ，日時データ，またはバイナリデータのどれかのデータ型を

項番	データ型	指定形式
		指定します。要素データ型の指定形式は，項番 1～13 で説明している各データ型の指定形式の規則に従います。 *最大要素数*： 配列要素の最大数を指定します。最大要素数には，2～30,000 の符号なし整数定数を指定してください。 ＜指定例＞ • 要素データ型がCHAR(5)で，最大要素数が 20 の場合 CHAR(5) ARRAY[20] • 要素データ型がINTEGER で，最大要素数が 5 の場合 INTEGER ARRAY[5]

注※1

NUMERIC 型が指定された場合，HADB はデータ型にDECIMAL 型が指定されたと見なします。

注※2

FLOAT 型が指定された場合，HADB はデータ型にDOUBLE PRECISION 型が指定されたと見なします。

注※3

列の定義長が 32,000 バイトを超えるVARCHAR 型は指定できません。

注※4

配列型のデータ型です。配列型の列を定義する際の考慮点については，マニュアル『HADB　システム構築・運用ガイド』の『配列型の列の定義【カラムストア表】』を参照してください。

> **重要**
>
> カラムストア表として定義する表に配列型の列を定義できます。ローストア表として定義する表には，配列型の列を定義できません。

データ型の詳細については，「6.2　データ型」を参照してください。

●*DEFAULT 句*　【共通】

```
DEFAULT句 : :=DEFAULT 既定値選択肢
  既定値選択肢 : := {定数 | CURRENT_DATE | CURRENT_TIME 〔(p)〕
                    | CURRENT_TIMESTAMP 〔(p)〕 | CURRENT_USER | NULL}
```

列の既定値を設定する場合にDEFAULT 句を指定します。

DEFAULT 句の指定形式，列の既定値については，「7.10　DEFAULT 句」を参照してください。

> **重要**
>
> 配列型の列にはDEFAULT 句を指定できません。

●NOT NULL　【共通】

列に非ナル値制約（ナル値を許さない制約）を定義する場合に指定します。

非ナル値制約を指定する列には，DEFAULT 句の*既定値選択肢*にNULL を指定できません。

●BRANCH　{YES｜NO｜AUTO}　　【ローストア表】

VARCHAR 型またはVARBINARY 型の列のデータの格納方法を指定します。

BRANCH にYES またはNO を指定した方がよいケースについては，マニュアル『HADB　システム構築・運用ガイド』の『可変長データ型の列データの分岐指定（BRANCH）【ローストア表】』を参照してください。

YES：

指定したVARCHAR 型またはVARBINARY 型の列のデータを分岐します。

NO：

指定したVARCHAR 型またはVARBINARY 型の列のデータを分岐しません。

AUTO：

定義長が 255 バイト以下のVARCHAR 型またはVARBINARY 型の列のデータは分岐しません。定義長が 256 バイト以上の場合で，基本行が 1 ページに収まらないときは分岐します。

BRANCH の指定を省略した場合，AUTO が仮定されます。

なお，このオプションは，次に示す表および列には指定できません。

- 表オプションBRANCH ALL を指定した表
- データ型がVARCHAR 型またはVARBINARY 型以外の列

●*圧縮方式指定*　　【カラムストア表】

圧縮方式指定：：＝COMPRESSION TYPE　{AUTO｜NONE｜RUNLENGTH｜DICTIONARY｜DELTA｜DELTA_RUNLENGTH}

カラムストア表の列のデータを圧縮する方式（列データの圧縮方式）を指定します。カラムストア表の各列に対してこのオプションを指定します。

*圧縮方式指定*の指定を省略した場合，AUTO が仮定されます。

AUTO：

カラムストア表のこの列のデータの圧縮方式を，HADB サーバが自動的に決定します。

NONE：

カラムストア表のこの列のデータを圧縮しません。

RUNLENGTH：

カラムストア表のこの列のデータをランレングス圧縮方式で圧縮します。

DICTIONARY：

カラムストア表のこの列のデータを辞書圧縮方式で圧縮します。

DELTA：

カラムストア表のこの列のデータを差分圧縮方式で圧縮します。

DELTA_RUNLENGTH：

カラムストア表のこの列のデータを差分ランレングス圧縮方式で圧縮します。

各圧縮方式の詳細については，マニュアル『HADB　システム構築・運用ガイド』の『ローストア表とカラムストア表の選択基準』の『カラムストア表の列データの圧縮方式』を参照してください。

(e)　表制約　　【共通】

表制約：：＝｛*一意性制約定義*｜*参照制約定義*｝

表制約には，一意性制約定義または参照制約定義を指定します。

(f)　一意性制約定義　　【共通】

一意性制約定義：：＝〔CONSTRAINT *制約名*〕 PRIMARY KEY（*列名* 〔 {ASC｜DESC} 〕
〔,*列名* 〔 {ASC｜DESC} 〕〕…）
〔IN *DBエリア名*〕
〔PCTFREE=*未使用領域の比率*〕

実表に主キーを定義する場合に指定します。主キーは，1 つの表に 1 つだけ定義できます。

主キーを構成する列には，一意性制約と非ナル値制約が適用されます。**一意性制約**とは，列値（または複数の列の列値の組み合わせ）が重複することを許さない制約のことです。**非ナル値制約**とは，列値にナル値を許さない制約のことです。

表中の行を一意に識別できる列，または列の組み合わせ（候補キー）の中から主キーを選択してください。

メモ

表中の行を一意に識別できる列，または列の組み合わせを**候補キー**といいます。

重要

配列型の列を定義している表には一意性制約を定義できません。

●CONSTRAINT *制約名*　　【共通】

ここで指定する一意性制約定義の制約名を指定します。制約名の指定規則については，「6.1.4　名前の指定」の「(2)　名前に使用できる文字の規則」を参照してください。

指定規則を次に示します。

- 同じ名称の制約名（参照制約定義の制約名も含む）が同一スキーマ内に存在する場合，CREATE TABLE 文がエラーになります。
- この指定を省略した場合，次の名称が制約名として仮定されます。

  PRIMARY_*nnnnnnnn*

  nnnnnnnn：主キーを定義する表の表 ID を 16 進数に変換した 8 桁の文字列

仮定された制約名が，同一スキーマ内に存在する場合，CREATE TABLE 文がエラーになります。そのため，制約名（参照制約定義の制約名も含む）およびインデクス識別子には，上記の形式に類似した名称を使用しないことを推奨します。

●*列名*　【共通】

主キーを構成する列の列名を指定します。指定規則を次に示します。

- 列名は，16 個まで指定できます。
- 複数の列名を指定する場合は，同じ列名を指定できません。

CREATE TABLE 文の実行時，列名に指定した列をインデクス構成列とする B-tree インデクスがユニークインデクスとして自動的に定義されます。定義される B-tree インデクスの規則を次に示します。

- 列名を 1 つだけ指定した場合は，単一列インデクスが定義されます。
- 複数の列名を指定した場合は，複数列インデクスが定義されます。
- インデクス識別子は制約名と同じになります。
- 制約名と同じ名称のインデクス識別子が同一スキーマ内に存在する場合，CREATE TABLE 文がエラーになります。

> **メモ**
>
> 自動的に定義された B-tree インデクスは，CREATE INDEX 文で定義した B-tree インデクスと同じ規則が適用されます。

● {ASC｜DESC}　【共通】

主キーに対応する B-tree インデクスのキー値の並び順を指定します。

ASC：

主キーに対応する B-tree インデクスのキー値を昇順に並べる場合に指定します。

DESC：

主キーに対応する B-tree インデクスのキー値を降順に並べる場合に指定します。

なお，単一列インデクスに対してDESC を指定しても無視されます。単一列インデクスのキー値は，常に昇順に並べられます（ASC が指定されたと仮定されます）。

ASC およびDESC の指定を省略した場合，ASC が仮定されます。

●IN *DB エリア名*　【共通】

主キーに対応する B-tree インデクスを格納する DB エリアの DB エリア名を指定します。

「IN *DB エリア名*」の指定を省略した場合，サーバ定義のadb_sql_default_dbarea_shared オペランドに指定した DB エリアに，主キーに対応する B-tree インデクスが格納されます。

なお，次のどちらかの条件を満たす場合に，「IN *DB エリア名*」の指定を省略すると，CREATE TABLE 文がエラーになります。

- サーバ定義のadb_sql_default_dbarea_shared オペランドの指定を省略している場合

- サーバ定義のadb_sql_default_dbarea_shared オペランドに，存在しない DB エリアを指定している場合，またはデータ用 DB エリア以外の DB エリアを指定している場合

●PCTFREE=*未使用領域の比率*　【共通】

～〈符号なし整数〉((0～99))《30》(単位：%)

主キーに対応する B-tree インデクス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。0～99（単位：%）を指定します。省略すると，30%が仮定されます。

データがインポートされて主キーに対応する B-tree インデクスが作成される際，または主キーに対応する B-tree インデクスが再作成される際，ここで指定した未使用領域の比率に従って B-tree インデクスのデータが格納されます。

インデクスページ内の未使用領域の比率の目安については，マニュアル『HADB　システム構築・運用ガイド』の『B-tree インデクスのインデクスページ内の未使用領域の確保（PCTFREE)』，または『テキストインデクスのインデクスページ内の未使用領域の確保（PCTFREE)』を参照してください。

(g)　参照制約定義　【共通】

```
参照制約定義 : : = 〔CONSTRAINT 制約名〕 FOREIGN KEY (列名 〔,列名〕…)
                        REFERENCES 表名 DISABLE
```

実表に参照制約（外部キー）を定義する場合に指定します。ほかの表の主キーを参照する列（または複数の列の組）を外部キーとして定義できます。

外部キーを定義する利点については，マニュアル『HADB　システム構築・運用ガイド』の『外部キーの指定（FOREIGN KEY)』を参照してください。

メモ

外部キーを構成する列と，主キーを構成する列は，次のすべての項目を同じにする必要があります。

- 構成列の列数
- 構成列の各列のデータ型
- 構成列の各列のデータ長

指定規則を次に示します。

- 1 つの表に対して外部キーを最大 255 個定義できます。
- 1 つの主キーを参照する外部キーは，最大 255 個定義できます。
- 同じ外部キーから，同じ主キーを参照する参照制約を複数定義することはできません。なお，ここでいう**同じ外部キー**とは，次の条件を満たす外部キーのことです。
  - 外部キーを構成する列が同じである

複数の列で外部キーを構成する場合，外部キーを定義するときの列の順番が異なっていても同じ外部キーと見なされます。

●CONSTRAINT *制約名*　【共通】

ここで指定する参照制約定義の制約名を指定します。制約名の指定規則については，「6.1.4　名前の指定」の「(2)　名前に使用できる文字の規則」を参照してください。

指定規則を次に示します。

- 同じ名称の制約名（一意性制約定義の制約名も含む）が同一スキーマ内に存在する場合，CREATE TABLE 文がエラーになります。
- この指定を省略した場合，次の名称が制約名として仮定されます。

  FOREIGN_*nnnnnnnn*_*YYYYMMDDhhmmss*th

  nnnnnnnn：外部キーを定義する表の表 ID を 16 進数に変換した 8 桁の文字列

  *YYYYMMDDhhmmss*th：外部キーを定義したときのタイムスタンプ（100 分の 1 秒まで出力）

  仮定された制約名が，同一スキーマ内に存在する場合，CREATE TABLE 文がエラーになります。そのため，制約名（一意性制約定義の制約名も含む）には，上記の形式に類似した名称を使用しないことを推奨します。

●*列名*　【共通】

外部キーを構成する列の列名を指定します。

指定規則を次に示します。

- 列名は，16 個まで指定できます。
- 複数の列名を指定する場合，同じ列名を指定できません。

●*表名*　【共通】

被参照表（主キーを定義した実表）の表名を指定します。

指定規則を次に示します。

- 外部キーを定義する表自身を被参照表にすることはできません。

●DISABLE　【共通】

このオプション（参照制約チェック抑止指定）を指定すると，外部キーによる参照制約のチェックを行いません。

このオプションは必ず指定してください。指定しないと，CREATE TABLE 文がエラーになります。

(h)　IN DB エリア名　【共通】

表を格納する DB エリアの DB エリア名を指定します。

「IN *DB エリア名*」の指定を省略した場合，サーバ定義のadb_sql_default_dbarea_shared オペランドに指定した DB エリアに，表が格納されます。

注意事項を次に示します。

- 次のどちらかの条件を満たす場合に，「IN *DB エリア名*」の指定を省略すると，CREATE TABLE 文がエラーになります。
  - サーバ定義のadb_sql_default_dbarea_shared オペランドの指定を省略している場合
  - サーバ定義のadb_sql_default_dbarea_shared オペランドに，存在しない DB エリアを指定している場合，またはデータ用 DB エリア以外の DB エリアを指定している場合
- 配列型の列を定義した表は，ページサイズが 4,096 バイトのデータ用 DB エリアには格納できません。
- 配列型の列を定義した表を格納するデータ用 DB エリアの 1 セグメントの容量は，次の条件を満たすようにしてください。

  1セグメントの容量（単位：バイト）≧*ARRAY_ROWSZ*（配列型の列を定義した表の行長）

  1 セグメントの容量は，adbinit コマンドまたはadbmodarea コマンドの-s オプションで指定します。

  ARRAY_ROWSZ の求め方については，マニュアル『HADB　システム構築・運用ガイド』の『行の種別ごとの格納ページ数の求め方』の『配列型の列を定義した表の行長（変数 ARRAY_ROWSZ）の求め方』を参照してください。

(i)　PCTFREE=未使用領域の比率　　【ローストア表】

～〈符号なし整数〉((0～99))《30》(単位：%)

データページ（表のデータを格納するページ）内の未使用領域の比率を指定します。0～99（単位：%）を指定します。省略すると，30%が仮定されます。

データがインポートされる際，ここで指定した未使用領域の比率に従って表のデータが格納されます。

なお，INSERT 文による行の追加，またはUPDATE 文による行の更新時には，ここで指定した未使用領域の比率は適用されません（未使用領域に追加データまたは更新後のデータが格納されます）。

データページ内の未使用領域の比率の目安については，マニュアル『HADB　システム構築・運用ガイド』の『データページ内の未使用領域の確保（PCTFREE）【ローストア表】』を参照してください。

なお，PCTFREE は複数回指定できません。

(j)　BRANCH ALL　　【ローストア表】

表に定義したすべてのVARCHAR 型およびVARBINARY 型の列のデータを分岐します。

BRANCH ALL を指定した方がよいケースについては，マニュアル『HADB　システム構築・運用ガイド』の『可変長データ型の列データの分岐指定（BRANCH）【ローストア表】』を参照してください。

なお，FIX を指定した場合は，BRANCH ALL を指定できません。

(k)　チャンク指定　　【共通】

チャンク指定：：＝CHUNK〔=*チャンク数の最大値*〕
〔*チャンクアーカイブ指定*〕

チャンクアーカイブ指定：：＝ARCHIVABLE
RANGECOLUMN=*列名*
〔RANGEINDEXNAME=*インデクス識別子*〕
〔IN *DBエリア名*〕
ARCHIVEDIR=*アーカイブディレクトリ名*

定義する実表をマルチチャンク表とする場合に指定します。

マルチチャンク表の設計については，マニュアル『HADB　システム構築・運用ガイド』の『マルチチャンク表を定義する場合の考慮点』を参照してください。

マルチチャンク表の種類と使用できる機能の関係を次の表に示します。

表 3-9　マルチチャンク表の種類と使用できる機能の関係

機能名	マルチチャンク表の種類	
	レギュラーマルチチャンク表	アーカイブマルチチャンク表
バックグラウンドインポート機能	○	○
PURGE CHUNK 文によるチャンク内の全行削除	○	○
チャンクアーカイブ機能	×	○
クラウドストレージ機能	○	×

(凡例)

○：使用できます。

×：使用できません。

バックグラウンドインポート機能，チャンクアーカイブ機能およびクラウドストレージ機能については，マニュアル『HADB　システム構築・運用ガイド』の『バックグラウンドインポート機能』，『チャンクアーカイブ機能（チャンク内のデータの圧縮)』および『クラウド環境で HADB サーバを使用する場合』を参照してください。

●CHUNK〔=*チャンク数の最大値*〕　【共通】

～〈符号なし整数〉((2～30,000))《256》(単位：個)

マルチチャンク表のチャンク数の最大値を指定します。CHUNK だけを指定して，*チャンク数の最大値*を省略した場合，*チャンク数の最大値*に256 が仮定されます。

レギュラーマルチチャンク表を定義する場合は，「CHUNK=*チャンク数の最大値*」だけを指定してください。

アーカイブマルチチャンク表を定義する場合は，「CHUNK=*チャンク数の最大値*」および次に説明するチャンクアーカイブ指定を指定してください。

●*チャンクアーカイブ指定*　【ロースト ア表】

チャンクアーカイブ指定：：＝ARCHIVABLE
RANGECOLUMN=*列名*
〔RANGEINDEXNAME=*インデクス識別子*〕

〔IN *DBエリア名*〕
ARCHIVEDIR=*アーカイブディレクトリ名*

定義する実表をアーカイブマルチチャンク表とする場合に指定します。

なお，クラウドストレージ機能を使用している場合は，このオプションは指定できません。

●RANGECOLUMN=*列名*

列名を指定します。ここで指定した列が，アーカイブレンジ列になります。

次に示すデータ型の列は，アーカイブレンジ列にできません。

- 定義長が 33 バイト以上のCHARACTER 型
- VARCHAR 型
- BINARY 型
- VARBINARY 型

なお，アーカイブレンジ列には，非ナル値制約が設定されます。

●RANGEINDEXNAME=*インデクス識別子*

CREATE TABLE 文の実行時，アーカイブレンジ列をインデクス構成列とするレンジインデクスが，HADB サーバによって自動的に定義されます。このレンジインデクスに付けるインデクス識別子を指定します。

RANGEINDEXNAME の指定を省略した場合，HADB サーバが次の規則に従ってインデクス識別子を決定します。

ARCHIVE_RANGE_INDEX_*nnnnnnnn*

nnnnnnnn は，アーカイブマルチチャンク表の表 ID を 16 進数に変換した 8 桁の文字列です。

上記の規則に従って決定されたインデクス識別子が，同一スキーマ内に存在する場合，CREATE TABLE 文がエラーになります。そのため，CREATE INDEX 文でインデクスを定義する際は，上記の形式に類似した名称を使用しないことを推奨します。

メモ

自動的に定義されたレンジインデクスは，CREATE INDEX 文で定義したレンジインデクスと同じ規則が適用されます。

●IN *DB エリア名*

自動的に定義されたレンジインデクスを格納する DB エリアの名称を指定します。

「IN *DB エリア名*」の指定を省略した場合，サーバ定義のadb_sql_default_dbarea_shared オペランドに指定した DB エリアに，自動的に定義されたレンジインデクスが格納されます。

なお，次のどちらかの条件を満たす場合に，「IN *DB エリア名*」の指定を省略すると，CREATE TABLE 文がエラーになります。

- サーバ定義のadb_sql_default_dbarea_shared オペランドの指定を省略している場合
- サーバ定義のadb_sql_default_dbarea_shared オペランドに，存在しない DB エリアを指定している場合

●ARCHIVEDIR＝*アーカイブディレクトリ名*

アーカイブファイルを格納するアーカイブディレクトリの名称を絶対パスで指定します。

指定規則を次に示します。

- アーカイブディレクトリ名は，文字列定数の形式で指定してください。文字列定数については，「6.3　定数」を参照してください。
- アーカイブディレクトリには，存在するディレクトリを指定してください。また，HADB 管理者に対して，読み取り権限，書き込み権限，および実行権限を与えたディレクトリを指定してください。

  さらに，アーカイブディレクトリのパスに含まれるすべてのディレクトリに，HADB 管理者に対する実行権限を付与してください。

  (例) アーカイブディレクトリが，/HADB/archive の場合

  /HADB/archive には，読み取り権限，書き込み権限，および実行権限が必要です。

  /，および/HADB には，実行権限が必要です。
- 次のディレクトリは，アーカイブディレクトリにできません。

  ・サーバディレクトリ

  ・サーバディレクトリの下位のディレクトリ

  ・下位のディレクトリにサーバディレクトリがあるディレクトリ

  ・DB ディレクトリ

  ・DB ディレクトリの下位のディレクトリ

  ・下位のディレクトリに DB ディレクトリがあるディレクトリ

  ・ルートディレクトリ

  DB ディレクトリが/HADB/db の場合，アーカイブディレクトリにできるディレクトリの例と，できないディレクトリの例を次に示します。

アーカイブディレクトリの例		理由
アーカイブディレクトリにできるディレクトリの例	/HADB/archive	なし。
アーカイブディレクトリにできないディレクトリの例	/HADB/db	左記のディレクトリは，DB ディレクトリと同一のため，アーカイブディレクトリにできません。
	/HADB/db/archive	左記のディレクトリは，DB ディレクトリの下位のディレクトリのため，アーカイブディレクトリにできません。
	/HADB	左記のディレクトリは，下位のディレクトリに DB ディレクトリがあるため，アーカイブディレクトリにできません。

- HADB サーバのインストール時にインストールデータを格納したディレクトリを，アーカイブディレクトリに指定しないようにしてください。
- アーカイブディレクトリ名は，前後の空白を除いて 1～400 バイトの長さにしてください。

> **メモ**
>
> アーカイブディレクトリ名の前後に空白がある場合，その空白は取り除かれて処理されます（空白が取り除かれた名称が仮定されます）。

- アーカイブディレクトリ名のパスに含まれる各要素は，NAME_MAX バイト以下になるようにしてください。NAME_MAX の値は，ご利用の環境によって異なります。

アーカイブディレクトリ名にシンボリックリンクを指定した場合，シンボリックリンクを解決したあとの絶対パス名が，ここで説明している規則に従っているかどうかがチェックされます。

[マルチノード機能]

マルチノード機能を使用している場合は，次のことに注意してください。

- アーカイブディレクトリは，NFS などを使用して全ノードで共有してください。また，CREATE TABLE 文の実行時点で，全ノードで共有されている必要があります。
- CREATE TABLE 文の実行時，ここで説明しているアーカイブディレクトリ名の指定規則のチェックが，プライマリノードで実行されます。セカンダリノードおよびワーカーノードでは，チェックは行われません。そのため，CREATE TABLE 文の実行後，セカンダリノードおよびワーカーノードでアーカイブディレクトリ名のチェックを行ってください。

■アーカイブマルチチャンク表を定義したときに定義されるロケーション表について

CREATE TABLE 文を実行してアーカイブマルチチャンク表を定義した場合，ロケーション表とロケーション表のインデクスが HADB サーバによって自動的に定義されます。この，ロケーション表とロケーション表のインデクスは，HADB サーバが使用します。そのため，ロケーション表とロケーション表のインデクスをユーザが直接操作したり，定義変更したり，または削除したりすることはできません。ロケーション表については，マニュアル『HADB　システム構築・運用ガイド』の『アーカイブマルチチャンク表の検索』を参照してください。

ロケーション表，およびロケーション表のインデクスは，アーカイブマルチチャンク表と同じ DB エリア内に格納されます。

ロケーション表およびロケーション表のインデクスの名称は，次の表で説明している規則に従って決定されます。

表 3-10　ロケーション表およびロケーション表のインデクスの名称規則

種別	名称規則	インデクスが管理する情報	インデクス構成列
ロケーション表	”HADB”.”LOCATION_TABLE_*nnnnnnnn*”	－	－
ロケーション表のインデクス	”HADB”.”LOCATION_INDEX_*nnnnnnnn*_CHUNK_ID”	アーカイブファイルに対応するチャンクのチャンク ID を管理しています。	CHUNK_ID
	”HADB”.”LOCATION_INDEX_*nnnnnnnn*_RANGE_01”	アーカイブファイルに格納されているデータのアーカイブレンジ列の値の範囲（上限値	• RANGE_MAX • RANGE_MIN

種別	名称規則	インデクスが管理する情報	インデクス構成列
		と下限値）を管理しています。	
	"HADB"."LOCATION_INDEX_*nnnnnnnn*_RANGE_02"	アーカイブファイルに格納されているデータのアーカイブレンジ列の値の下限値を管理しています。	RANGE_MIN

（凡例）

－：該当しません。

注

nnnnnnnn は，アーカイブマルチチャンク表の表 ID を 16 進数に変換した 8 桁の文字列です。

ロケーション表，およびロケーション表のインデクスのスキーマ名は HADB です。

(l)　STORAGE FORMAT　{ROW ｜ COLUMN}　　【共通】

定義する表の表データの格納形式を指定します。

ROW：

表データの格納形式がロースト ア形式の表を定義する場合に指定します。ROW を指定した場合，表はロースト ア表として定義されます。

COLUMN：

表データの格納形式がカラムストア形式の表を定義する場合に指定します。COLUMN を指定した場合，表はカラムストア表として定義されます。

STORAGE FORMAT の指定を省略した場合，ROW が仮定されます。

メモ

- ロースト ア表，ロースト ア形式，カラムストア表，カラムストア形式については，マニュアル『HADB　システム構築・運用ガイド』の『ロースト ア表とカラムストア表』を参照してください。
- このオプションの指定を，**表格納形式指定**といいます。

(3)　実行時に必要な権限

CREATE TABLE 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

また，参照制約（外部キー）を定義する場合は，被参照表に対するREFERENCES 権限が必要になります。

(4) 規則

1. 自分（HADB サーバに接続中の認可識別子の HADB ユーザ）が所有するスキーマに対してだけ実表を定義できます。ほかの HADB ユーザが所有するスキーマに対しては実表を定義できません。
2. システム内で合計 4,096 個（ディクショナリ表（実表）およびシステム表（実表）は除く）まで実表を定義できます。
3. 1 つの DB エリアに格納できる実表は 200 個までです。
4. 1 つの表に対して 4,000 個まで列を定義できます。
5. 実表の列の長さの合計（行長）が，次に示す条件式を満たすように列を定義してください。
   - 実表がローストア表の場合

     ROWSZ（*行長*）≦*ページサイズ*-56
   - 実表がカラムストア表の場合

     ROWSZ（*行長*）≦*ページサイズ*-80

   ROWSZ（行長）を求める計算式については，マニュアル『HADB　システム構築・運用ガイド』の『行の種別ごとの格納ページ数の求め方』を参照してください。
6. チャンク指定をした場合，主キーを定義することはできません。また，主キーを定義した場合，チャンク指定はできません。
7. B-tree インデクスを定義できない列には，主キーを定義することはできません。
8. 主キーを定義する場合，次の条件を満たす必要があります。次の条件を満たさない場合は，主キーを定義できません。
   - 主キーを構成する列が 1 列の場合

     主キーを構成する列の定義長※1≦MIN｛（*a*÷3）-128，4036｝　（単位：バイト）
   - 主キーを構成する列が 2 列以上の場合

     主キーを構成する列の定義長の合計※2≦MIN｛（*a*÷3）-128，4036｝　（単位：バイト）

   a：主キーに対応する B-tree インデクスを格納する DB エリアのページサイズ

   注※1

   列の定義長については，「表 3-5　単一列インデクスを構成する列の長さ」を参照してください。

   注※2

   各列の定義長については，「表 3-6　複数列インデクスを構成する列の長さ」を参照して，主キーを構成する列の定義長の合計を求めてください。
9. トランザクションの開始後に定義された表を，そのトランザクションからアクセスすることはできません。

(5) 例題

例題 1～例題 6 は，ローストア表の定義例です。例題 7～例題 8 は，カラムストア表の定義例です。

例題 1（FIX 表でない実表を定義する場合）

店舗表（SHOPSLIST）を定義します。店舗表の列構成と未使用領域の比率などは，次のとおりとします。

- 店舗コード（SHOP_CODE）：CHAR(8)
- 地域コード（RGN_CODE）：CHAR(6)
- 店舗名（SHOP_NAME）：VARCHAR(20)
- 店舗電話番号（TEL_NO）：CHAR(10)
- 店舗住所（ADDRESS）：VARCHAR(300)
- すべての列に非ナル値制約を定義する
- 店舗住所（ADDRESS）の列のデータを分岐して格納する
- 店舗表を DB エリア（DBAREA01）に格納する
- データページ内の未使用領域の比率を 40%とする
- チャンク数の最大値を 100 とする

```
CREATE TABLE "SHOPSLIST"
      ("SHOP_CODE" CHAR(8) NOT NULL,
       "RGN_CODE" CHAR(6) NOT NULL,
       "SHOP_NAME" VARCHAR(20) NOT NULL,
       "TEL_NO" CHAR(10) NOT NULL,
       "ADDRESS" VARCHAR(300) NOT NULL BRANCH YES)
    IN "DBAREA01"
    PCTFREE=40
    CHUNK=100
```

例題 2（FIX 表を定義する場合）

販売履歴表（SALESLIST）を定義します。販売履歴表の列構成と未使用領域の比率などは，次のとおりとします。

- 顧客 ID（USERID）：CHAR(6)
- 商品コード（PUR-CODE）：CHAR(4)
- 販売個数（PUR-NUM）：SMALLINT
- 購入日（PUR-DATE）：DATE
- 購入日（PUR-DATE）列にDEFAULT 句を指定して，列の既定値を設定する
- 販売履歴表を DB エリア（DBAREA01）に格納する
- データページ内の未使用領域の比率を 20%とする
- 行長が固定のため，FIX を指定する
- チャンク数の最大値を 200 とする

```
CREATE FIX TABLE "SALESLIST"
      ("USERID" CHAR(6),
       "PUR-CODE" CHAR(4),
       "PUR-NUM" SMALLINT,
       "PUR-DATE" DATE DEFAULT CURRENT_DATE)
    IN "DBAREA01"
    PCTFREE=20
    CHUNK=200
```

例題３（主キーを定義した実表を定義する場合）

販売履歴表（SALESLIST）を定義します。販売履歴表の列構成と未使用領域の比率などは，次のとおりとします。

- 顧客 ID（USERID）：CHAR(6)
- 商品コード（PUR-CODE）：CHAR(4)
- 販売個数（PUR-NUM）：SMALLINT
- 購入日（PUR-DATE）：DATE
- 販売履歴表を DB エリア（DBAREA01）に格納する
- データページ内の未使用領域の比率を 20%とする
- 行長が固定のため，FIX を指定する
- 主キーを定義する（顧客 ID 列（USERID）を主キーの構成列とする）
- 主キーに対応する B-tree インデクスを DB エリア（DBAREA02）に格納する
- 主キーに対応する B-tree インデクスのインデクスページ内の未使用領域の比率を 20 パーセントとする

```
CREATE FIX TABLE "SALESLIST"
      ("USERID" CHAR(6),
       "PUR-CODE" CHAR(4),
       "PUR-NUM" SMALLINT,
       "PUR-DATE" DATE,
       CONSTRAINT "PK-USERID" PRIMARY KEY ("USERID" ASC)
            IN "DBAREA02" PCTFREE=20)
    IN "DBAREA01"
    PCTFREE=20
```

下線部分が主キーの定義（一意性制約定義）の指定です。

例題４（主キーを定義した実表を定義する場合）

店舗表（SHOPSLIST）を定義します。店舗表の列構成と未使用領域の比率などは，次のとおりとします。

- 店舗コード（SHOP_CODE）：CHAR(8)
- 地域コード（RGN_CODE）：CHAR(6)
- 店舗名（SHOP_NAME）：VARCHAR(20)
- 店舗電話番号（TEL_NO）：CHAR(10)
- 店舗住所（ADDRESS）：VARCHAR(300)

- 店舗住所（ADDRESS）の列のデータを分岐して格納する
- 店舗表を DB エリア（DBAREA01）に格納する
- データページ内の未使用領域の比率を 40%とする
- 主キーを定義する（店舗コード列（SHOP_CODE）および地域コード列（RGN_CODE）を主キーの構成列とする）
- 主キーに対応する B-tree インデクスを DB エリア（DBAREA02）に格納する
- 主キーに対応する B-tree インデクスのインデクスページ内の未使用領域の比率を 20 パーセントとする

```
CREATE TABLE ”SHOPSLIST”
      (”SHOP_CODE” CHAR(8),
       ”RGN_CODE” CHAR(6),
       ”SHOP_NAME” VARCHAR(20),
       ”TEL_NO” CHAR(10),
       ”ADDRESS” VARCHAR(300) BRANCH YES,
       CONSTRAINT ”PK-CODE” PRIMARY KEY (”SHOP_CODE” ASC,”RGN_CODE” ASC)
            IN ”DBAREA02” PCTFREE=20)
    IN ”DBAREA01”
    PCTFREE=40
```

下線部分が主キーの定義（一意性制約定義）の指定です。

例題５（外部キーを定義した実表を定義する場合）

店舗表（SHOPSLIST）および従業員表（EMPLOYEE）を定義します。主キーおよび外部キーの定義は，次のとおりとします。

- 店舗表（SHOPSLIST）に主キーを定義します。主キーを構成する列は，店舗表（SHOPSLIST）の SHOP_CODE 列およびRGN_CODE 列です。
- 従業員表（EMPLOYEE）に外部キーを定義します。外部キーを構成する列は，従業員表（EMPLOYEE）のSHOP_CODE 列およびRGN_CODE 列です。

■店舗表（SHOPSLIST）

```
CREATE TABLE ”SHOPSLIST”
      (”SHOP_CODE” CHAR(8),
       ”RGN_CODE” CHAR(6),
       ”SHOP_NAME” VARCHAR(20),
       ”TEL_NO”    CHAR(10),
       ”ADDRESS”   VARCHAR(300) BRANCH YES,
       CONSTRAINT ”PK-CODE” PRIMARY KEY (”SHOP_CODE” ASC,”RGN_CODE” ASC)
              IN ”DBAREA02” PCTFREE=20)
    IN ”DBAREA01”
    PCTFREE=40
```

下線部分が主キーの定義の指定です。

■従業員表（EMPLOYEE）

```
CREATE TABLE ”EMPLOYEE”
      (”EMPLOYEE_CODE”    CHAR(8),
       ”FIRST_NAME”       VARCHAR(8),
```

```
       "FIRST_NAME_YOMI"  VARCHAR(16),
       "FAMILY_NAME"      VARCHAR(8),
       "FAMILY_NAME_YOMI" VARCHAR(16),
       "SHOP_CODE"        CHAR(8),
       "RGN_CODE"        CHAR(6),
       "EMPLOYEE_TYPE"    CHAR(1),
       "TEL_NO"           CHAR(10),
       "ADDRESS"          VARCHAR(300) BRANCH YES,
       CONSTRAINT "PK-EMPLOYEE_CODE" PRIMARY KEY ("EMPLOYEE_CODE" ASC)
             IN "DBAREA02" PCTFREE=20,
       CONSTRAINT "FK-SHOP_CODE" FOREIGN KEY ("SHOP_CODE","RGN_CODE")
             REFERENCES "SHOPSLIST" DISABLE)
    IN "DBAREA01"
    PCTFREE=40
```

下線部分が外部キーの定義（参照制約定義）の指定です。

例題６（アーカイブマルチチャンク表を定義する場合）

レシート表（RECEIPT）をアーカイブマルチチャンク表として定義します。チャンクに関する指定は，次のとおりとします。

- チャンク数の最大値を 120 とする
- RECORD_DAY 列をアーカイブレンジ列とする
- /mnt/nfs/archivedir をアーカイブディレクトリとする

```
CREATE TABLE "RECEIPT"
      ("RID" INTEGER,
       "SHOP_CODE"      CHAR(8),
       "RGN_CODE"       CHAR(6),
       "EMPLOYEE_CODE"  CHAR(8),
       "CUSTOMER_CODE"  CHAR(8),
       "RECORD_DAY"     DATE,
       "ITEM_CODE"      CHAR(8),
       "ITEM_PRICE"     INTEGER)
    IN "DBAREA01"
    PCTFREE=30
    CHUNK=120
      ARCHIVABLE RANGECOLUMN="RECORD_DAY" IN "DBAREA02"
                 ARCHIVEDIR='/mnt/nfs/archivedir'
```

下線部分がアーカイブマルチチャンク表固有の指定です。

例題７（カラムストア表を定義する場合）

レシート表（RECEIPT）をカラムストア表として定義します。

```
CREATE TABLE "RECEIPT"
      ("RID"             INTEGER,
       "SHOP_CODE"      CHAR(8),
       "RGN_CODE"       CHAR(6),
       "EMPLOYEE_CODE"  CHAR(8),
       "CUSTOMER_CODE"  CHAR(8),
       "RECORD_DAY"     DATE,
       "ITEM_CODE"      CHAR(8),
       "ITEM_PRICE"     INTEGER)
    IN "DBAREA01"
```

```
    CHUNK=120
    STORAGE FORMAT COLUMN
```

下線部分がカラムストア表固有の指定です。

例題８（配列型の列を定義した表を定義する場合）

配列型の列を定義したレシート表（RECEIPT）を定義します。

```
CREATE TABLE "RECEIPT"
      ("RID"           INTEGER,
       "SHOP_CODE"     CHAR(8),
       "AREA_CODE"     CHAR(6),
       "EMPLOYEE_CODE" CHAR(8),
       "CUSTOMER_CODE" CHAR(8),
       "RECORD_DAY"    DATE,
       "ITEM_CODE"     CHAR(8) ARRAY[20],   ...1
       "ITEM_PRICE"    INTEGER)
    IN "DBAREA01"
    CHUNK=120
    STORAGE FORMAT COLUMN                   ...2
```

[説明]

1. 商品コード列（ITEM_CODE）を配列型の列として定義します。要素データ型をCHAR(8)，最大要素数を 20 とします。
2. 配列型の列を定義した表は，カラムストア表として定義する必要があります。

3.8 CREATE USER（HADB ユーザの作成）

ここでは，CREATE USER 文の指定形式および規則について説明します。

3.8.1 CREATE USER 文の指定形式および規則

HADB ユーザを作成します。

作成した HADB ユーザには権限が付与されていないため，必要な権限をGRANT 文で HADB ユーザに付与してください。

(1) 指定形式

- データベース認証を使用する HADB ユーザを作成する場合

```
CREATE USER文 : :=CREATE USER 認可識別子 IDENTIFIED BY パスワード
```

- PAM 認証を使用する HADB ユーザを作成する場合

```
CREATE USER文 : :=CREATE USER 認可識別子 IDENTIFIED WITH PAM
```

(2) 指定形式の説明

●*認可識別子*

作成する HADB ユーザの認可識別子を指定します。

PAM 認証を使用する場合は，HADB ユーザの認可識別子を外部ユーザ名（OS ユーザ名など）と同じにしてください。

認可識別子の指定規則を次に示します。

- 認可識別子に使用できる文字は，半角の英大文字，半角の英小文字，半角の数字，¥（バックスラッシュ），#，および@です。
- 認可識別子に英小文字を使用する場合は，認可識別子を二重引用符（”）で囲んで指定してください。

  （例）CREATE USER ”ADBuser01” …

  二重引用符で囲まないと，英小文字は英大文字と見なされます。例えば，「ADBuser01」と指定した場合，「ADBUSER01」と指定したと見なされます。
- 認可識別子は名前として指定するため，二重引用符（”）で囲んで指定することを推奨します。
- ALL，HADB，MASTER，およびPUBLIC は認可識別子に指定できません。
- 認可識別子は 100 文字（100 バイト）まで指定できます。

認可識別子の指定規則の詳細については，「6.1.4 名前の指定」を参照してください。

●IDENTIFIED BY *パスワード*

作成する HADB ユーザのパスワードを指定します。

> **メモ**
>
> このオプションを指定して作成したHADBユーザのユーザ認証方式はデータベース認証になります。データベース認証については，マニュアル『HADB　システム構築・運用ガイド』の『データベース認証』を参照してください。

パスワードの指定規則を次に示します。

- パスワードに使用できる文字は，半角の英大文字，半角の英小文字，半角の数字，¥（バックスラッシュ），および次に示す半角の文字です。

  @ ` ! ” # $ % & ’ ( ) * : + ; [ ] { } , = < > | - . ^ ~ / ? _

- パスワードは文字列定数の形式で指定します。そのため，パスワードをアポストロフィで囲む必要があります。指定例を次に示します。

  （例1）パスワードにPassword01を指定する場合

  IDENTIFIED BY ’Password01’

  （例2）パスワードにPass’01を指定する場合

  IDENTIFIED BY ’Pass’’01’

  パスワードの文字列にアポストロフィがある場合，上記の例のように1個のアポストロフィを表すのに2個のアポストロフィを指定します。

  文字列定数の指定規則については，「表6-10　定数の記述形式と仮定されるデータ型」を参照してください。

- パスワードに空文字は指定できません（次の指定はできません）。

  IDENTIFIED BY ’’

- パスワードは255文字（255バイト）まで指定できます。

> **メモ**
>
> - JDBCドライバを使用する場合は，パスワードに次に示す半角文字を使用しないことを推奨します。
>
>   &
>
> - ODBCドライバを使用する場合は，パスワードに次に示す半角文字を使用しないことを推奨します。
>
>   [ ] { } ( ) , ; ? * = ! @

●IDENTIFIED WITH PAM

作成するHADBユーザのユーザ認証方式をPAM認証にする場合に指定します。PAM認証については，マニュアル『HADB　システム構築・運用ガイド』の『PAM認証』を参照してください。

(3)　実行時に必要な権限

CREATE USER文を実行する場合，DBA権限およびCONNECT権限が必要になります。

(4) 規則

HADB ユーザは，システム内に 30,000 ユーザまで作成できます。

(5) 例題

例題 1

データベース認証を使用する HADB ユーザを作成します。認可識別子およびパスワードは次のとおりとします。

- 認可識別子：ADBUSER01
- パスワード：#HelloHADB_01

```
CREATE USER ”ADBUSER01” IDENTIFIED BY ’#HelloHADB_01’
```

例題 2

PAM 認証を使用する HADB ユーザを作成します。認可識別子は，外部ユーザ名（OS ユーザ名など）と同じOSUSER01 とします。

```
CREATE USER ”OSUSER01” IDENTIFIED WITH PAM
```

3.9　CREATE VIEW（ビュー表の定義）

ここでは，CREATE VIEW 文の指定形式および規則について説明します。

3.9.1　CREATE VIEW 文の指定形式および規則

ビュー表を定義します。

(1)　指定形式

```
CREATE VIEW文 : :=CREATE VIEW 表名 〔(列名リスト)〕 AS 問合せ式 〔LIMIT句〕
  列名リスト : :=列名 〔,列名〕…
```

(2)　指定形式の説明

●*表名*

定義するビュー表の表名を指定します。すでに定義されている実表およびビュー表と同じ表名は指定できません。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

●*列名リスト*

```
列名リスト : :=列名 〔,列名〕…
```

ビュー表を構成する列を指定します。

列名：

ビュー表を構成する列の列名を指定します。1 つのビュー表に同じ列名を指定できません。

なお，HADB によって自動的に設定される導出列名と重複する可能性があるため，列名に EXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。

留意事項を次に示します。

- *列名リスト*に指定する列名の数と，問合せ式によって導出される表の列数を同じにしてください。
- *列名リスト*によって導出される列数は 4,000 以下にしてください。
- *列名リスト*を省略した場合，問合せ式によって導出される列名が，ビュー表を構成する各列の列名となります。導出される列名については，「6.9　導出列名」を参照してください。
- 次に示す場合は，*列名リスト*を指定する必要があります。
  - ・導出列名が重複している場合
  - ・導出列名が設定されていない列が 1 つでもある場合

●AS *問合せ式* 〔*LIMIT 句*〕

ビュー表の定義内容を表す問合せ式を指定します。問合せ式については，「7.1　問合せ式」を参照してください。

問合せ式中に指定した表は，ビュー表の基になる表（**基表**）になります。

問合せ式には，次に示す項目を指定できません。

- 〔*表指定*.〕ROW
- ?パラメタ

> **メモ**
>
> 問合せ式中の最も外側の問合せ指定の選択リストに，*または*表指定*.*を指定してCREATE VIEW文を実行し，そのあとで基表に列を追加しても，ビュー表にその列は追加されません。

LIMIT 句：

問合せ式の検索結果として取得する行数の最大値を指定します。

LIMIT 句については，「7.9　LIMIT 句」を参照してください。

(3)　実行時に必要な権限

CREATE VIEW 文を実行する場合，次に示すすべての権限が必要になります。

- CONNECT 権限
- スキーマ定義権限
- 問合せ式中に指定するすべての基表に対するSELECT 権限

(4)　規則

1. システム内で合計 30,000 個まで，ビュー表を定義できます。
2. CREATE VIEW 文の最大長は 64,000 バイトです。
3. CREATE VIEW 文に指定できる表参照中の表名，導出表，表関数導出表，および集まり導出表の延べ数は，最大 2,047 個になります。

   ただし，表参照中に次の指定がある場合，次の指定が内部導出表に等価変換されたあとの SQL 文に対して延べ数のチェックが行われます。

   - 問合せ名
   - ビュー表

     CREATE VIEW 文中にビュー表を指定している場合は，CREATE VIEW 文中に指定したビュー表を導出表に等価変換したあとに，延べ数のチェックが行われます。

   SQL 文中に指定されている表，導出表，表関数導出表，および集まり導出表の数え方の規則と例については，「4.4.1　SELECT 文の指定形式および規則」の「(4)　規則」を参照してください。
4. CREATE VIEW 文に指定できる問合せ指定および表値構成子の合計指定数は，最大 1,023 個になります。

5. HADB サーバに接続した認可識別子の HADB ユーザと異なるスキーマ名を持つビュー表は定義できません。

6. 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）は，CREATE VIEW 文の問合せ式の結果，導出される表の列の属性と同じになります。

7. ビュー表には，**読み取り専用ビュー表**と**更新可能ビュー表**があります。読み取り専用ビュー表に対しては，行の挿入，更新，および削除はできません。

8. ビュー表が，読み取り専用ビュー表になるか，更新可能ビュー表になるかは，「AS *問合せ式*」の指定内容で決まります。次の場合に読み取り専用ビュー表になります。

   - 最も外側の問合せ指定に，表の結合，結合表，導出表※，表関数導出表，SELECT DISTINCT，GROUP BY 句，HAVING 句，ウィンドウ関数または集合関数を指定した場合
   - 最も外側の問合せ指定の選択式に，基表の同一列を複数指定した場合
   - 最も外側の問合せ指定の選択式に，列指定以外を指定した場合
   - 最も外側の問合せ指定のFROM 句に指定した表と同じ表を，副問合せのFROM 句に指定した場合
   - 最も外側の問合せ指定のFROM 句に，読み取り専用ビュー表を指定した場合
   - 最も外側の問合せ指定を演算項に持つ集合演算を指定した場合
   - LIMIT 句を指定した場合
   - 最も外側の問合せ指定のFROM 句に再帰的問合せ名を指定した場合
   - 最も外側の問合せ指定のFROM 句に，ディクショナリ表またはシステム表を指定した場合

   注※

   導出表が展開されたことによって，読み取り専用ビュー表となる条件を満たさなくなった場合，そのビュー表は更新可能ビュー表になります。導出表の展開規則については，「7.32.3　導出表の展開規則」を参照してください。

   ディクショナリ表を検索すると，定義したビュー表が読み取り専用ビュー表か，または更新可能ビュー表かが確認できます。確認方法については，マニュアル『HADB　システム構築・運用ガイド』の『ビュー表の更新可否を調べる方法』を参照してください。

9. 定義したビュー表に対するアクセス権限は，次に示す規則に従って決まります。

   - ビュー表に対するアクセス権限は，ビュー表を定義した HADB ユーザが，すべての基表に対して持っているアクセス権限によって決まります。例えば，定義するビュー表に対してINSERT 権限を持ちたい場合は，すべての基表に対するINSERT 権限が必要になります。
   - CREATE VIEW 文の問合せ式に，次の指定をして定義したビュー表に対するアクセス権限の決定規則を説明します。
     - 表値構成子
     - 表関数導出表
     - 集まり導出表

表値構成子によって導出される導出表，表関数導出表，および集まり導出表に対しては，ビュー表を定義するHADBユーザは付与権付きのアクセス権限を持っていると仮定されます。そのため，CREATE VIEW文の問合せ式にほかの基表を指定している場合，定義したビュー表に対するアクセス権限は次のようになります。

（例）

- CREATE VIEW文の問合せ式に，表値構成子と付与権付きのアクセス権限を持っている基表を指定した場合

  定義したビュー表には，表値構成子によって導出される導出表と基表に対して持っているアクセス権限の共通部分が適用されます。したがって，この場合，ビュー表を定義したHADBユーザは，そのビュー表に対する付与権付きのアクセス権限を持ちます。

- CREATE VIEW文の問合せ式に，表値構成子とSELECT権限だけを持っている基表を指定した場合

  定義したビュー表には，表値構成子によって導出される導出表と基表に対して持っているアクセス権限の共通部分が適用されます。したがって，この場合，ビュー表を定義したHADBユーザは，そのビュー表に対するSELECT権限だけを持ちます。

10. ビュー表に対するアクセス権限をほかのHADBユーザに付与する場合は，そのビュー表の全基表に対するアクセス権限を付与権付きで持っている必要があります。

11. 基表に対するアクセス権限が新たに付与された場合，その基表に依存するビュー表に対するアクセス権限も新たに付与されます（アクセス権限の伝搬が発生します）。例えば，HADBユーザAが，表X.T1を基表としたビュー表A.V1と，ビュー表A.V1を基表としたビュー表A.V2を定義している場合に，表X.T1に対するINSERT権限がHADBユーザAに付与されると，ビュー表A.V1，A.V2に対するINSERT権限も付与されます。

12. 基表に対するアクセス権限が取り消された場合，その基表に依存するビュー表に対するアクセス権限も取り消されます。

13. CREATE VIEW文の問合せ式に，スカラ関数CONTAINS（同義語検索指定あり）を指定してビュー表を定義した場合の規則を次に示します。

    - 同義語検索指定で指定した同義語辞書を削除した場合，ビュー表へのアクセス時にエラーとなります。
    - 同義語検索指定で指定した同義語辞書を更新した場合，更新内容は定義したビュー表に反映されます。

14. CREATE VIEW文の問合せ式に指定したWITH句中の問合せ名を，CREATE VIEW文中で参照しないビュー表を定義し，そのビュー表を検索する際は，次の規則が適用されます。また，そのビュー表を別のCREATE VIEW文中に指定する際も，次の規則が適用されます。

    - CREATE VIEW文中で参照されていない問合せ名に対応する問合せ式本体の構成要素の数は，構成要素数の上限チェックの対象になりません。
    - CREATE VIEW文中で参照されていない問合せ名に対応する問合せ式本体に指定されている表については，排他が確保されません。

例を次に示します。

（例）ビュー表の定義時

```
CREATE VIEW "V1"
    AS WITH "Q1" AS (SELECT "T1"."C1","T2"."C2" FROM "T1","T2")
       SELECT * FROM "T3"
```

[説明]

ビュー表の定義時には，上記で説明した規則は適用されません。そのため，CREATE VIEW 文に指定されている問合せの数は 2（問合せ名Q1 に対応する問合せと主問合せ）となります。また，指定されている表の数は 3（実表T1，T2，およびT3）となります。

(例) ビュー表の検索時

```
SELECT * FROM "V1"
```

[説明]

- ビュー表の検索時には，上記で説明した規則が適用されます。そのため，SELECT 文に指定されている問合せの数は 2（主問合せと，ビュー表V1 を等価変換した導出表の導出問合せ）となります。
- 指定されている表の数は 2（ビュー表V1 を等価変換した導出表と，導出表の導出問合せ中の実表T3）となります。この規則に基づいて，1SQL 文中に指定できる問合せの数や表の上限値のチェックが行われます。
- 実表T1，T2 については，排他が確保されません。

15. CREATE VIEW 文の問合せ式中に副問合せを入れ子で指定する場合，入れ子の数は 31 回以内にしてください。また，FROM 句に指定した表がビュー表の場合，その基となる問合せ式を指定した内部導出表を適用したあとの副問合せの入れ子の数が 31 回以内になるようにしてください。詳細については，「7.3.1　副問合せの指定形式および規則」の「(4)　規則」の「(a)　副問合せ共通の規則」を参照してください。

(例 1)

```
CREATE VIEW "V1"
  AS SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM "T1") AS DT32
     ) AS DT31 ) AS DT30 ) AS DT29 ) AS DT28 ) AS DT27 ) AS DT26 ) AS DT25 ) AS DT24
     ) AS DT23 ) AS DT22 ) AS DT21 ) AS DT20 ) AS DT19 ) AS DT18 ) AS DT17 ) AS DT16
     ) AS DT15 ) AS DT14 ) AS DT13 ) AS DT12 ) AS DT11 ) AS DT10 ) AS DT9 ) AS DT8
     ) AS DT7 ) AS DT6 ) AS DT5 ) AS DT4 ) AS DT3 ) AS DT2 ) AS DT1 ) AS DT0
```

上記の例の場合，ビュー表V1 の副問合せの入れ子の数は 32 回となり，上限を超えているため，CREATE VIEW 文がエラーになります。

なお，T1 は実表とします。

(例 2)

```
CREATE VIEW "V2"           ← ビュー表V2
  AS SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (
     SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM "T1") AS DT30
     ) AS DT29 ) AS DT28 ) AS DT27 ) AS DT26 ) AS DT25 ) AS DT24 ) AS DT23 ) AS DT22
     ) AS DT21 ) AS DT20 ) AS DT19 ) AS DT18 ) AS DT17 ) AS DT16 ) AS DT15 ) AS DT14
     ) AS DT13 ) AS DT12 ) AS DT11 ) AS DT10 ) AS DT9 ) AS DT8 ) AS DT7 ) AS DT6
     ) AS DT5 ) AS DT4 ) AS DT3 ) AS DT2 ) AS DT1 ) AS DT0
CREATE VIEW "V3" AS SELECT * FROM "V2"       ← ビュー表V3
CREATE VIEW "V4" AS SELECT * FROM "V3"       ← ビュー表V4
```

- ビュー表V2 の副問合せの入れ子の数は 30 回となります。そのため，CREATE VIEW 文を実行できます。
- ビュー表V3 の副問合せの入れ子の数は，内部導出表が適用された結果，31 回となります。そのため，CREATE VIEW 文を実行できます。
- ビュー表V4 の副問合せの入れ子の数は，内部導出表が適用された結果，32 回となります。上限を超えているため，CREATE VIEW 文がエラーになります。

なお，T1 は実表とします。

16. CREATE VIEW 文の問合せ式中に，ビューレベルが 33 のビュー表を指定できません。
17. HADB サーバをバージョンアップする際，ディクショナリ表またはシステム表に依存するビュー表が自動的に再作成されることがあります。ビュー表が再作成される条件については，マニュアル『HADB システム構築・運用ガイド』の『バージョンアップ時の注意事項』の『バージョンアップ時に行われるビュー表の再作成』を参照してください。
18. トランザクションの開始後に定義されたビュー表を，そのトランザクションからアクセスすることはできません。

(5) 例題

例題 1

店舗表（SHOPSLIST）から，店舗住所（ADDRESS）列以外の列を検索できる店舗表のビュー表（VSHOPSLIST）を定義します。店舗表のビュー表を構成する列の順序と列名は次のとおりとします。

- 店舗コード（SHOP_CODE）
- 地域コード（RGN_CODE）
- 店舗名（SHOP_NAME）
- 店舗電話番号（TEL_NO）

```
CREATE VIEW "VSHOPSLIST" ("SHOP_CODE","RGN_CODE","SHOP_NAME","TEL_NO")
    AS SELECT "SHOP_CODE","RGN_CODE","SHOP_NAME","TEL_NO"
           FROM "SHOPSLIST"
```

店舗表のビュー表（VSHOPSLIST）は，更新可能ビュー表になります。

例題 2

販売履歴表（SALESLIST）と商品表（PRODUCTSLIST）から，売り上げ金額の最大値（QMAXSALES）を商品名（PUR-NAME）ごとに求める売り上げ表（VSALES）を，ビュー表として定義します。売り上げ表（VSALES）の列構成は次のとおりとします。

- 商品名（VPUR_NAME）
- 売り上げ金額の最大値（VQMAXSALES）

■ビュー表の定義

```
CREATE VIEW ”VSALES” (”VPUR-NAME”,”VQMAXSALES”)
    AS WITH ”QT1”(”QCODE”,”QMAXSALES”) AS (SELECT ”PUR-CODE”,MAX(”PRICE” * ”QUANTITY”)
                                             FROM ”SALESLIST”
                                               GROUP BY ”PUR-CODE”)
       SELECT ”PUR-NAME”,”QMAXSALES”
         FROM ”QT1” INNER JOIN ”PRODUCTSLIST” ON ”QCODE”=”PUR-CODE”
```

SALESLIST

USERID	PUR-CODE	PRICE	QUANTITY
U0001	P001	500	3
U0001	P002	100	10
U0002	P001	500	1
U0002	P002	100	5
U0003	P001	500	4
U0003	P002	100	1
U0003	P003	50	10

PRODUCTSLIST

PUR-CODE	PUR-NAME
P001	PRODUCT_A
P002	PRODUCT_B
P003	PRODUCT_C

■ビュー表の検索

```
SELECT * FROM ”VSALES”
```

検索結果の例

VPUR_NAME	VQMAXSALES
PRODUCT_A	2000
PRODUCT_B	1000
PRODUCT_C	500

3.10　DROP AUDIT（監査対象定義の削除）

ここでは，DROP AUDIT 文の指定形式および規則について説明します。

3.10.1　DROP AUDIT 文の指定形式および規則

CREATE AUDIT 文で定義した監査対象定義を削除します。

> **重要**
>
> 監査証跡機能が有効な場合に，DROP AUDIT 文を実行できます。監査証跡機能が有効かどうかは，adbaudittrail -d コマンドを実行して確認してください。

(1)　指定形式

```
DROP AUDIT文 : : =DROP AUDIT AUDITTYPE EVENT
                            FOR ANY OPERATION
```

(2)　指定形式の説明

●AUDITTYPE EVENT

CREATE AUDIT 文のAUDITTYPE でEVENT を指定して定義した監査対象定義を削除する場合に指定します。イベントの最終結果についての監査証跡の出力をやめる場合に指定します。

●FOR ANY OPERATION

「表 3-3　監査対象とするイベント」に示すイベントを監査対象から外す場合に指定します。

CREATE AUDIT 文で，FOR ANY OPERATION を指定して定義した監査対象定義を削除する場合に指定します。

(3)　実行時に必要な権限

DROP AUDIT 文を実行する場合，CONNECT 権限および監査管理権限が必要になります。

(4)　規則

1. CREATE AUDIT 文で定義した監査対象定義を削除できます。
2. HADB サーバは，監査証跡を出力する際の判定処理時に監査対象定義を参照します。そのため，監査証跡の出力タイミングによっては，監査対象定義を削除する前に実行された監査対象の操作についての監査証跡が出力されないことがあります。

(5) 例題

例題

CREATE AUDIT 文で定義した監査対象定義を削除します。

```
DROP AUDIT AUDITTYPE EVENT
           FOR ANY OPERATION
```

3.11　DROP INDEX（インデクスの削除）

ここでは，DROP INDEX 文の指定形式および規則について説明します。

3.11.1　DROP INDEX 文の指定形式および規則

インデクス（B-tree インデクス，テキストインデクス，またはレンジインデクス）を削除します。

(1)　指定形式

DROP INDEX文 ：：＝DROP INDEX *インデクス名*

(2)　指定形式の説明

●*インデクス名*

削除するインデクスのインデクス名を指定します。インデクス名の指定規則については，「6.1.5　名前の修飾」の「(3)　インデクス名の指定形式」を参照してください。

(3)　実行時に必要な権限

DROP INDEX 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

(4)　規則

1. 自分（HADB サーバに接続中の認可識別子の HADB ユーザ）が所有しているインデクスだけを削除できます。ほかの HADB ユーザが所有しているインデクスは削除できません。
2. 表にデータが格納されている場合でも，その表に定義されているインデクスを削除できます。このとき，表に格納されているデータは削除されません。
3. ディクショナリ表（実表）およびシステム表（実表）のインデクスは削除できません。
4. インデクスを削除した場合，削除したインデクスのコスト情報も削除されます。
5. 主キーに対応する B-tree インデクスはDROP INDEX 文で削除できません。削除する場合は，DROP TABLE 文で表と一緒に削除してください。
6. アーカイブレンジ列に定義されているレンジインデクスは，DROP INDEX 文で削除できません。削除する場合は，DROP TABLE 文で表と一緒に削除してください。

(5)　例題

例題 1

店舗表（SHOPSLIST）に定義した B-tree インデクス（SHOP_CODE_IDX）を削除します。

```
DROP INDEX ”SHOP_CODE_IDX”
```

例題 2

従業員表（EMPLOYEE）に定義したテキストインデクス（ADDRESS_IDX）を削除します。

```
DROP INDEX ”ADDRESS_IDX”
```

例題 3

店舗表（SHOPSLIST）に定義したレンジインデクス（SHOP_CODE_RIDX）を削除します。

```
DROP INDEX ”SHOP_CODE_RIDX”
```

3.12　DROP SCHEMA（スキーマの削除）

ここでは，DROP SCHEMA 文の指定形式および規則について説明します。

3.12.1　DROP SCHEMA 文の指定形式および規則

スキーマを削除します。

スキーマを削除した場合，表，インデクス，外部キーは，次のように影響を受けます。

- 削除対象のスキーマに定義されている表（実表およびビュー表）とインデクスも削除されます。
- DROP SCHEMA 文の実行によって削除される表に依存するビュー表（ほかのスキーマのビュー表）も削除されます（または無効化されます）。
- DROP SCHEMA 文の実行によって削除される表を被参照表とする外部キー（ほかのスキーマの外部キー）も削除されます。

(1)　指定形式

```
DROP SCHEMA文 : :＝DROP SCHEMA 〔スキーマ名〕 〔削除動作〕
  削除動作 : :＝ {CASCADE | RESTRICT}
```

(2)　指定形式の説明

●*スキーマ名*

削除するスキーマのスキーマ名を指定します。スキーマ名を省略した場合，DROP SCHEMA 文を実行した HADB ユーザの認可識別子が仮定されます。

スキーマ名の指定規則については，「6.1.5　名前の修飾」の「(1)　スキーマ名の指定形式」を参照してください。

なお，*スキーマ名*にALL，HADB，MASTER，およびPUBLIC は指定できません。

●*削除動作*

```
削除動作 : :＝ {CASCADE | RESTRICT}
```

削除対象のスキーマに表またはインデクスが定義されている場合に，スキーマを削除するかどうかを指定します。削除動作の各指定の説明を次の表に示します。

削除動作の指定	説明	ほかのスキーマのビュー表の扱い	ほかのスキーマの外部キーの扱い
削除動作の指定を省略した場合	削除対象のスキーマに表またはインデクスが定義されている場合でも，スキーマを削除します。この場合，スキーマに定義されている	DROP SCHEMA 文の実行によって削除される表に依存するビュー表（ほかのスキーマのビュー表）が無効化されます。	DROP SCHEMA 文の実行によって削除される表を被参照表とする外部キー（ほかのスキーマの外部キー）も削除されます。

削除動作の指定	説明	ほかのスキーマのビュー表の扱い	ほかのスキーマの外部キーの扱い
CASCADE を指定した場合	表およびインデクスも削除されます。	DROP SCHEMA 文の実行によって削除される表に依存するビュー表（ほかのスキーマのビュー表）も削除されます。	
RESTRICT を指定した場合	削除対象のスキーマに表またはインデクスが定義されている場合は，DROP SCHEMA 文をエラーにします。	DROP SCHEMA 文がエラーになるため，ほかのスキーマのビュー表に影響はありません。	DROP SCHEMA 文がエラーになるため，ほかのスキーマの外部キーに影響はありません。

(3) 実行時に必要な権限

DROP SCHEMA 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

(4) 規則

1. 自分（HADB サーバに接続中の認可識別子の HADB ユーザ）が所有しているスキーマだけを削除できます。ほかの HADB ユーザが所有しているスキーマは削除できません。例えば，認可識別子にADBUSER01を指定してadbsql コマンドを実行した場合，DROP SCHEMA 文で削除できるスキーマはADBUSER01 のスキーマだけです。
2. スキーマを削除した場合，次のコスト情報も削除されます。
   - スキーマに定義されている表のコスト情報
   - スキーマに定義されているインデクスのコスト情報
3. スキーマを削除した場合，そのスキーマ内の表に対するアクセス権限を持っているすべての HADB ユーザから，そのアクセス権限が取り消されます。アクセス権限が取り消された場合，ビュー表や，参照制約に影響を及ぼすおそれがあります。詳細については，「3.17.2　アクセス権限の取り消し」の「(4) 規則」を参照してください。

(5) 例題

例題 1

スキーマ名ADBUSER01 のスキーマを削除します。

```
DROP SCHEMA ”ADBUSER01” CASCADE
```

例題 2

スキーマ名ADBUSER01 のスキーマを削除します。ただし，スキーマに表またはインデクスが定義されている場合は，DROP SCHEMA 文をエラーにします。

```
DROP SCHEMA ”ADBUSER01” RESTRICT
```

3.13　DROP TABLE（表の削除）

ここでは，DROP TABLE 文の指定形式および規則について説明します。

3.13.1　DROP TABLE 文の指定形式および規則

実表を削除します。

実表を削除した場合，インデクス，表制約，およびビュー表は，次のように影響を受けます。

- 実表に定義されているインデクスも削除されます。
- 削除した実表に依存するビュー表も削除されます（または無効化されます）。
- 実表に定義されている表制約※も削除されます。

注※

削除対象表に定義されている主キーおよび外部キーが削除されます。また，削除対象表を被参照表とする外部キーも削除されます（ほかのスキーマの外部キーも削除されます）。

(1)　指定形式

```
DROP TABLE文 : : =DROP TABLE 表名　〔削除動作〕

  削除動作 : : = {CASCADE | RESTRICT}
```

(2)　指定形式の説明

●*表名*

削除する実表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，ビュー表の表名は指定できません。

●*削除動作*

```
削除動作 : : = {CASCADE | RESTRICT}
```

次のどれかの条件を満たす場合に，実表を削除するかどうかを指定します。

- 削除対象の実表にインデクスが定義されている場合
- 削除対象の実表に依存するビュー表が定義されている場合
- 削除対象の実表に表制約が定義されている場合

削除動作の各指定の説明を次の表に示します。

<table>
<tr><th>削除動作の指定</th><th>説明</th><th>削除した実表に依存するビュー表の扱い</th></tr>
<tr><td>削除動作の指定を省略した場合</td><td rowspan="2">次のどれかの条件を満たす場合でも，実表を削除します。
• 削除対象の実表にインデクスが定義されている場合
• 削除対象の実表に依存するビュー表が定義されている場合
• 削除対象の実表に表制約が定義されている場合
このとき，次に示すものも削除されます。
• 実表に定義されているインデクスおよび表制約
• 削除対象表を被参照表とする外部キー（ほかのスキーマの外部キーも削除されます）</td><td>削除した実表に依存するビュー表が無効化されます。自スキーマのビュー表だけではなく，ほかのスキーマの依存するビュー表も無効化されます。</td></tr>
<tr><td>CASCADE を指定した場合</td><td>削除した実表に依存するビュー表が削除されます。自スキーマのビュー表だけではなく，ほかのスキーマの依存するビュー表も削除されます。</td></tr>
<tr><td>RESTRICT を指定した場合</td><td>次のどれかの条件を満たす場合は，DROP TABLE 文をエラーにします。
• 削除対象の実表にインデクスが定義されている場合
• 削除対象の実表に依存するビュー表が定義されている場合
• 削除対象の実表に表制約が定義されている場合</td><td>DROP TABLE 文がエラーになるため，依存するビュー表に影響はありません。</td></tr>
</table>

(3)　実行時に必要な権限

DROP TABLE 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

(4)　規則

1. DROP TABLE 文に指定できる表名は，自分（HADB サーバに接続中の認可識別子の HADB ユーザ）が所有している実表だけです。ほかの HADB ユーザが所有している実表は削除できません。
2. データが格納されていても実表を削除できます。
3. ディクショナリ表（実表）およびシステム表（実表）は削除できません。
4. 表を削除した場合，次のコスト情報も削除されます。
   - 表のコスト情報
   - 表に定義されているインデクスのコスト情報
5. 表を削除した場合，その表に対するアクセス権限を持っているすべての HADB ユーザから，そのアクセス権限が取り消されます。アクセス権限が取り消された場合，ビュー表や，参照制約に影響を及ぼすおそれがあります。詳細については，「3.17.2　アクセス権限の取り消し」の「(4)　規則」を参照してください。
6. アーカイブマルチチャンク表を削除した場合，次の表とインデクスも削除されます。
   - アーカイブレンジ列に自動的に定義されたレンジインデクス
   - ロケーション表
   - ロケーション表のインデクス
7. アーカイブマルチチャンク表を削除した場合，アーカイブ状態でないチャンクに格納されたデータと，アーカイブ状態のチャンクに格納されたデータの両方が削除されます。

(5) 例題

例題

店舗表（SHOPSLIST）を削除します。

```
DROP TABLE ”SHOPSLIST” CASCADE
```

3.14　DROP USER（HADB ユーザの削除）

ここでは，DROP USER 文の指定形式および規則について説明します。

3.14.1　DROP USER 文の指定形式および規則

HADB ユーザを削除します。

(1)　指定形式

```
DROP USER文 : :＝DROP USER 認可識別子　〔削除動作〕
  削除動作 : :＝ {CASCADE | RESTRICT}
```

(2)　指定形式の説明

●*認可識別子*

削除する HADB ユーザの認可識別子を指定します。

認可識別子の指定規則を次に示します。

- 英小文字を指定する場合は，認可識別子を二重引用符（"）で囲んで指定してください。二重引用符で囲まないと，英小文字を指定しても英大文字が指定されたと見なされます。

  （例）DROP USER adbuser01 …

  この場合，認可識別子にADBUSER01 が指定されたと見なされます。
- 認可識別子は名前として指定するため，二重引用符（"）で囲んで指定することを推奨します。

認可識別子の指定規則の詳細については，「6.1.4　名前の指定」を参照してください。

●*削除動作*

```
削除動作 : :＝ {CASCADE | RESTRICT}
```

次のどちらかの条件を満たす場合に，HADB ユーザを削除するかどうかを指定します。

- 削除対象の HADB ユーザが，スキーマを所有している場合
- 削除対象の HADB ユーザが，ほかの HADB ユーザにアクセス権限を付与している場合

削除動作の各指定の説明を次の表に示します。

削除動作の指定	説明	ほかのスキーマのビュー表の扱い	ほかのスキーマの外部キーの扱い
削除動作の指定を省略した場合	次のどちらかの条件を満たす場合でも，HADB ユーザを削除します。 • 削除対象の HADB ユーザが，スキーマを所有している場合	DROP USER 文の実行によって削除される表に依存するビュー表（ほかのスキーマのビュー表）が無効化されます。	DROP USER 文の実行によって削除される表を被参照表とする外部キー（ほかのスキーマの外部キー）も削除されます。

削除動作の指定	説明	ほかのスキーマのビュー表の扱い	ほかのスキーマの外部キーの扱い
CASCADE を指定した場合	• 削除対象の HADB ユーザが，ほかの HADB ユーザにアクセス権限を付与している場合 DROP USER 文を実行した場合，削除対象の HADB ユーザが所有しているスキーマも削除されます。 また，ほかの HADB ユーザに付与したアクセス権限がすべて取り消されます。さらに，取り消されたアクセス権限のすべての依存権限が取り消されます。	DROP USER 文の実行によって削除される表に依存するビュー表（ほかのスキーマのビュー表）も削除されます。	
RESTRICT を指定した場合	次のどちらかの条件を満たす場合は，DROP USER 文をエラーにします。 • 削除対象の HADB ユーザが，スキーマを所有している場合 • 削除対象の HADB ユーザが，ほかの HADB ユーザにアクセス権限を付与している場合	DROP USER 文がエラーになるため，ほかのスキーマのビュー表に影響はありません。	DROP USER 文がエラーになるため，ほかのスキーマの外部キーに影響はありません。

(3) 実行時に必要な権限

DROP USER 文を実行する場合，DBA 権限およびCONNECT 権限が必要になります。

(4) 規則

1. 自分以外の HADB ユーザを削除できます。
2. データベース認証を使用し，かつDBA 権限およびCONNECT 権限を持つ HADB ユーザが最低 1 人必要です。そのため，このような HADB ユーザが 1 人しかいない場合，その HADB ユーザを削除することはできません。
3. HADB サーバに接続中の認可識別子の HADB ユーザは削除できません。
4. 削除された HADB ユーザが，ほかの HADB ユーザにDBA 権限，CONNECT 権限，またはスキーマ定義権限を与えていても，それらの権限は取り消されません。
5. 削除対象の HADB ユーザが，ほかの HADB ユーザにアクセス権限を付与していた場合，付与したアクセス権限はすべて取り消されます。さらに，取り消されたアクセス権限の依存権限も取り消されます。そのため，アクセス権限の取り消しに伴い，ビュー表や，参照制約に影響を及ぼすおそれがあります。詳細については，「3.17.2　アクセス権限の取り消し」の「(4)　規則」を参照してください。
6. 監査権限を持っている HADB ユーザを削除することはできません。監査権限を持っている HADB ユーザを削除する場合は，監査管理権限を持っている HADB ユーザに，削除対象の HADB ユーザの監査権限を取り消してもらってから，その HADB ユーザを削除してください。

7. DB エリア暗号化機能を使用している場合，暗号管理権限を持っている HADB ユーザが最低 1 人必要です。そのため，暗号管理権限を持っている HADB ユーザが 1 人しかいない場合，その HADB ユーザを削除することはできません。

(5) 例題

例題 1

HADB ユーザADBUSER01 を削除します。

```
DROP USER ”ADBUSER01” CASCADE
```

例題 2

HADB ユーザADBUSER01 を削除します。ただし，ADBUSER01 がスキーマを所有している場合は，DROP USER 文をエラーにします。

```
DROP USER ”ADBUSER01” RESTRICT
```

3.15　DROP VIEW（ビュー表の削除）

ここでは，DROP VIEW 文の指定形式および規則について説明します。

3.15.1　DROP VIEW 文の指定形式および規則

ビュー表を削除します。

(1)　指定形式

```
DROP VIEW文 : :＝DROP VIEW 表名　〔削除動作〕

  削除動作 : :＝ {CASCADE | RESTRICT}
```

(2)　指定形式の説明

●*表名*

削除するビュー表の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2) 表名の指定形式」を参照してください。

次の表の表名は指定できません。

- 実表
- ディクショナリ表
- システム表

●*削除動作*

```
削除動作 : :＝ {CASCADE | RESTRICT}
```

次の条件を満たす場合に，ビュー表を削除するかどうかを指定します。

- 削除対象のビュー表に依存するビュー表が存在する場合

削除動作の各指定の説明を次の表に示します。

削除動作の指定	説明	削除対象のビュー表に依存するビュー表の扱い
削除動作の指定を省略した場合	次の条件を満たす場合でも，ビュー表を削除します。 • 削除対象のビュー表に依存するビュー表が存在する場合	削除対象のビュー表に依存するビュー表が無効化されます。自スキーマのビュー表も，ほかのスキーマのビュー表も無効化されます。
CASCADE を指定した場合		削除対象のビュー表に依存するビュー表も削除されます。自スキーマのビュー表も，ほかのスキーマのビュー表も削除されます。
RESTRICT を指定した場合	次の条件を満たす場合は，DROP VIEW 文をエラーにします。	DROP VIEW 文がエラーになるため，依存するビュー表に影響はありません。

削除動作の指定	説明	削除対象のビュー表に依存するビュー表の扱い
	• 削除対象のビュー表に依存するビュー表が存在する場合	

(3) 実行時に必要な権限

DROP VIEW 文を実行する場合，CONNECT 権限およびスキーマ定義権限が必要になります。

(4) 規則

1. HADB サーバに接続した認可識別子と異なるスキーマ名を持つビュー表は削除できません。
2. ビュー表を削除した場合，そのビュー表に対するアクセス権限を持っているすべての HADB ユーザから，そのアクセス権限が取り消されます。

(5) 例題

例題

店舗ビュー表（VSHOPSLIST）を削除します。

```
DROP VIEW "VSHOPSLIST" CASCADE
```

3.16 GRANT（権限の付与）

ここでは，GRANT 文の指定形式および規則について説明します。

3.16.1 ユーザ権限，スキーマ操作権限，監査権限，および暗号管理権限の付与

HADB ユーザに次の権限を付与します。

- ユーザ権限
  - DBA 権限
  - CONNECT 権限
- スキーマ操作権限
  - スキーマ定義権限
- 監査権限
  - 監査管理権限
  - 監査参照権限
- 暗号管理権限

(1) 指定形式

```
GRANT文 : :＝GRANT 権限〔,権限〕… TO 認可識別子〔,認可識別子〕…

  権限 : :＝｛ユーザ権限｜スキーマ操作権限｜監査権限｜暗号管理権限｝
    ユーザ権限 : :＝ {DBA｜CONNECT}
    スキーマ操作権限 : :＝SCHEMA
    監査権限 : :＝ {AUDIT ADMIN｜AUDIT VIEWER}
    暗号管理権限 : :＝CRYPTO ADMIN
```

(2) 指定形式の説明

●*権限*〔,*権限*〕…

```
権限 : :＝｛ユーザ権限｜スキーマ操作権限｜監査権限｜暗号管理権限｝
```

HADB ユーザに付与する権限を指定します。同じ権限を重複して指定できません。

```
ユーザ権限 : :＝ {DBA｜CONNECT}
```

HADB ユーザにユーザ権限を付与する場合に指定します。

- DBA

HADB ユーザにDBA 権限を付与する場合に指定します。

- CONNECT

  HADB ユーザにCONNECT 権限を付与する場合に指定します。

スキーマ操作権限 ：：＝SCHEMA

HADB ユーザにスキーマ操作権限を付与する場合に指定します。

- SCHEMA

  HADB ユーザにスキーマ定義権限を付与する場合に指定します。

監査権限 ：：＝ {AUDIT ADMIN｜AUDIT VIEWER}

HADB ユーザに監査権限（監査管理権限または監査参照権限）を付与する場合に指定します。

- AUDIT ADMIN

  HADB ユーザに監査管理権限を付与する場合に指定します。

- AUDIT VIEWER

  HADB ユーザに監査参照権限を付与する場合に指定します。

暗号管理権限 ：：＝CRYPTO ADMIN

HADB ユーザに暗号管理権限を付与する場合に指定します。

- CRYPTO ADMIN

  HADB ユーザに暗号管理権限を付与する場合に指定します。

●TO *認可識別子*〔,*認可識別子*〕…

権限を付与する HADB ユーザの認可識別子を指定します。最大 128 個の認可識別子を指定できます。

認可識別子の指定規則を次に示します。

- 英小文字を指定する場合は，認可識別子を二重引用符（”）で囲んで指定してください。二重引用符で囲まないと，英小文字を指定しても英大文字が指定されたと見なされます。

  （例）GRANT DBA TO adbuser01

  この場合，認可識別子にADBUSER01 が指定されたと見なされます。

- 認可識別子は名前として指定するため，二重引用符（”）で囲んで指定することを推奨します。

認可識別子の指定規則の詳細については，「6.1.4　名前の指定」を参照してください。

(3)　実行時に必要な権限

ユーザ権限，スキーマ操作権限，監査権限，または暗号管理権限を付与するGRANT 文を実行する場合，DBA 権限およびCONNECT 権限が必要になります。

(4) 規則

1. DBA 権限を持っている HADB ユーザは，次に示す権限をほかの HADB ユーザに付与することができます。
   - ユーザ権限
   - スキーマ操作権限
   - 監査権限

   また，自分（HADB サーバに接続中の認可識別子の HADB ユーザ）に対しても，ユーザ権限，スキーマ操作権限，および監査参照権限を付与することができます。
2. DBA 権限と監査管理権限の両方の権限を持つことはできません。そのため，DBA 権限を持っている HADB ユーザに監査管理権限を付与することはできません。また，監査管理権限を持っている HADB ユーザにDBA 権限を付与することはできません。
3. 監査管理権限と暗号管理権限の両方の権限を持つことはできません。そのため，監査管理権限を持っている HADB ユーザに暗号管理権限を付与することはできません。また，暗号管理権限を持っている HADB ユーザに監査管理権限を付与することはできません。
4. DBA 権限と暗号管理権限の両方の権限を持つことはできません。そのため，DBA 権限を持っている HADB ユーザに暗号管理権限を付与することはできません。また，暗号管理権限を持っている HADB ユーザにDBA 権限を付与することはできません。
5. CONNECT 権限を持たない HADB ユーザに暗号管理権限を付与することはできません。
6. 次に示す権限は，PAM 認証を使用している HADB ユーザには付与することはできません。
   - 監査管理権限
   - 監査参照権限
   - 暗号管理権限
7. 複数の認可識別子を指定した場合，GRANT 文の実行でエラーが発生したときは，すべての HADB ユーザへの権限付与が無効になります。

(5) 例題

例題 1

HADB ユーザADBUSER01 に，DBA 権限，CONNECT 権限，およびスキーマ定義権限を付与します。

```
GRANT DBA,CONNECT,SCHEMA TO ”ADBUSER01”
```

例題 2

HADB ユーザADBUSER02，ADBUSER03 に，CONNECT 権限およびスキーマ定義権限を付与します。

```
GRANT CONNECT,SCHEMA TO ”ADBUSER02”,”ADBUSER03”
```

例題 3

HADB ユーザADBAUDITADMIN に，CONNECT 権限および監査管理権限を付与します。

```
GRANT CONNECT,AUDIT ADMIN TO ”ADBAUDITADMIN”
```

例題 4

HADB ユーザADBAUDITOR に，CONNECT 権限および監査参照権限を付与します。

```
GRANT CONNECT,AUDIT VIEWER TO ”ADBAUDITOR”
```

例題 5

HADB ユーザCADMIN に，暗号管理権限を付与します。

```
GRANT CRYPTO ADMIN TO ”CADMIN”
```

3.16.2 アクセス権限の付与

HADB ユーザにアクセス権限を付与します。

(1) 指定形式

```
GRANT文 : :＝GRANT アクセス権限 ON オブジェクト名 TO 権限受領者 〔WITH GRANT OPTION〕

  アクセス権限 : :＝ {ALL 〔PRIVILEGES〕 |動作 〔,動作〕…}
    動作 : :＝ {SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES
              | IMPORT TABLE | REBUILD INDEX | GET COSTINFO | EXPORT TABLE
              | MERGE CHUNK | CHANGE CHUNK COMMENT | CHANGE CHUNK STATUS
              | ARCHIVE CHUNK | UNARCHIVE CHUNK}

  オブジェクト名 : :＝ {〔TABLE〕 表名 | ALL TABLES}
  権限受領者 : :＝ {認可識別子 〔,認可識別子〕… | PUBLIC}
```

(2) 指定形式の説明

●アクセス権限

```
アクセス権限 : :＝ {ALL 〔PRIVILEGES〕 |動作 〔,動作〕…}
```

付与するアクセス権限の種類を指定します。

ALL 〔PRIVILEGES〕 :

すべてのアクセス権限を付与する場合に指定します。

すべてのアクセス権限とは，GRANT 文を実行する時点でサポートされているすべてのアクセス権限を意味しています。GRANT 文の実行後に，バージョンアップによって新たにサポートされたアクセス権限は対象になりません（自動的に権限が付与されるようなことはありません）。

> **重要**
>
> 付与権を持っているのが一部の種類のアクセス権限だけの場合に，ALL PRIVILEGES を指定してGRANT 文を実行しても，すべての種類のアクセス権限は付与されません。この場

合，付与権を持っているアクセス権限だけが権限受領者に付与されます。例えば，付与権を持っているのがINSERT 権限だけの場合に，ALL PRIVILEGES を指定してGRANT 文を実行すると，INSERT 権限だけが権限受領者に付与されます。

メモ

付与対象オブジェクトに対する付与権付きのアクセス権限を 1 種類も持っていない場合に，ALL PRIVILEGES を指定すると，GRANT 文がエラーになります。

動作〔,*動作*〕…：

```
動作 : := {SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES
          | IMPORT TABLE | REBUILD INDEX | GET COSTINFO | EXPORT TABLE
          | MERGE CHUNK | CHANGE CHUNK COMMENT | CHANGE CHUNK STATUS
          | ARCHIVE CHUNK | UNARCHIVE CHUNK}
```

付与するアクセス権限の種類を指定します。同じ動作を重複して指定できません。

- SELECT

  HADB ユーザにSELECT 権限を付与する場合に指定します。

- INSERT

  HADB ユーザにINSERT 権限を付与する場合に指定します。

- UPDATE

  HADB ユーザにUPDATE 権限を付与する場合に指定します。

- DELETE

  HADB ユーザにDELETE 権限を付与する場合に指定します。

- TRUNCATE

  HADB ユーザにTRUNCATE 権限を付与する場合に指定します。

- REFERENCES

  HADB ユーザにREFERENCES 権限を付与する場合に指定します。

- IMPORT TABLE

  HADB ユーザにIMPORT TABLE 権限を付与する場合に指定します。

- REBUILD INDEX

  HADB ユーザにREBUILD INDEX 権限を付与する場合に指定します。

- GET COSTINFO

  HADB ユーザにGET COSTINFO 権限を付与する場合に指定します。

- EXPORT TABLE

  HADB ユーザにEXPORT TABLE 権限を付与する場合に指定します。

- MERGE CHUNK

HADB ユーザにMERGE CHUNK 権限を付与する場合に指定します。

- CHANGE CHUNK COMMENT

  HADB ユーザにCHANGE CHUNK COMMENT 権限を付与する場合に指定します。

- CHANGE CHUNK STATUS

  HADB ユーザにCHANGE CHUNK STATUS 権限を付与する場合に指定します。

- ARCHIVE CHUNK

  HADB ユーザにARCHIVE CHUNK 権限を付与する場合に指定します。

- UNARCHIVE CHUNK

  HADB ユーザにUNARCHIVE CHUNK 権限を付与する場合に指定します。

●ON *オブジェクト名*

オブジェクト名：：＝｛〔TABLE〕 *表名*｜ALL TABLES｝

アクセス権限を付与する際の，対象オブジェクトを指定します。

なお，**オブジェクト**とは，スキーマオブジェクトのことを意味しています。

〔TABLE〕 *表名*：

ここで指定した表に対するアクセス権限を付与します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，無効化されているビュー表の表名は指定できません。

ALL TABLES：

実行ユーザのスキーマ内の全実表に対するアクセス権限を付与します。ここでいう**実行ユーザ**とは，GRANT 文を実行した HADB ユーザを意味しています。

GRANT 文を実行した HADB ユーザがスキーマを定義していない場合，またはスキーマ内に 1 つも実表を定義していない場合，アクセス権限を付与しないでGRANT 文を正常終了します。

メモ

ALL TABLES を指定した際に対象となる表は，GRANT 文の実行時点で実行ユーザが所有している実表になります。したがって，GRANT 文を実行したあとに，新たに定義した実表はALL TABLES の対象になりません（新たに定義した実表に対するアクセス権限は付与されません）。

●TO *権限受領者*

権限受領者：：＝｛*認可識別子*〔,*認可識別子*〕…｜PUBLIC｝

アクセス権限を付与する HADB ユーザを指定します。

認可識別子〔,*認可識別子*〕…：

アクセス権限を付与する HADB ユーザの認可識別子を指定します。最大 128 個の認可識別子を指定できます。

認可識別子の指定規則を次に示します。

- 英小文字を指定する場合は，認可識別子を二重引用符（”）で囲んで指定してください。二重引用符で囲まないと，英小文字を指定しても英大文字が指定されたと見なされます。
- 認可識別子は名前として指定するため，二重引用符（”）で囲んで指定することを推奨します。

認可識別子の指定規則の詳細については，「6.1.4　名前の指定」を参照してください。

PUBLIC：

指定したオブジェクトに対するアクセス権限を，全 HADB ユーザに許可する場合に指定します。ここでいう**全 HADB ユーザ**とは，PUBLIC 指定のGRANT 文の実行後に作成された HADB ユーザも含まれます。

（例）

```
GRANT SELECT,IMPORT TABLE ON ”T1” TO PUBLIC
```

上記のGRANT 文を実行した場合，全 HADB ユーザに次のアクセス権限が許可されます。

- 表T1 に対するSELECT 権限
- 表T1 に対するIMPORT TABLE 権限

PUBLIC を指定する場合は，GRANT 文の実行者が，アクセス権限を許可するオブジェクトの所有者である必要があります。

メモ

- PUBLIC キーワードは，すべての HADB ユーザを表す，システムによって暗黙的に作成されたユーザと考えることができます。
- **アクセス権限を許可する**とは，アクセス権限を使用したアクセスまたは操作を許可するという意味です。

●WITH GRANT OPTION

権限受領者にアクセス権限を付与権付きで付与する場合にこのオプションを指定します。

なお，GRANT 文の実行者は，付与するアクセス権限を付与権付きで持っている必要があります。

（例）

```
GRANT SELECT ON ”X”.”T1” TO ”ADBUSER01” WITH GRANT OPTION
```

上記のGRANT 文を実行した場合，表X.T1 に対するSELECT 権限が付与権付きでADBUSER01 に付与されます。GRANT 文を実行する HADB ユーザは，表X.T1 に対するSELECT 権限を付与権付きで持っている必要があります。

(3)　実行時に必要な権限

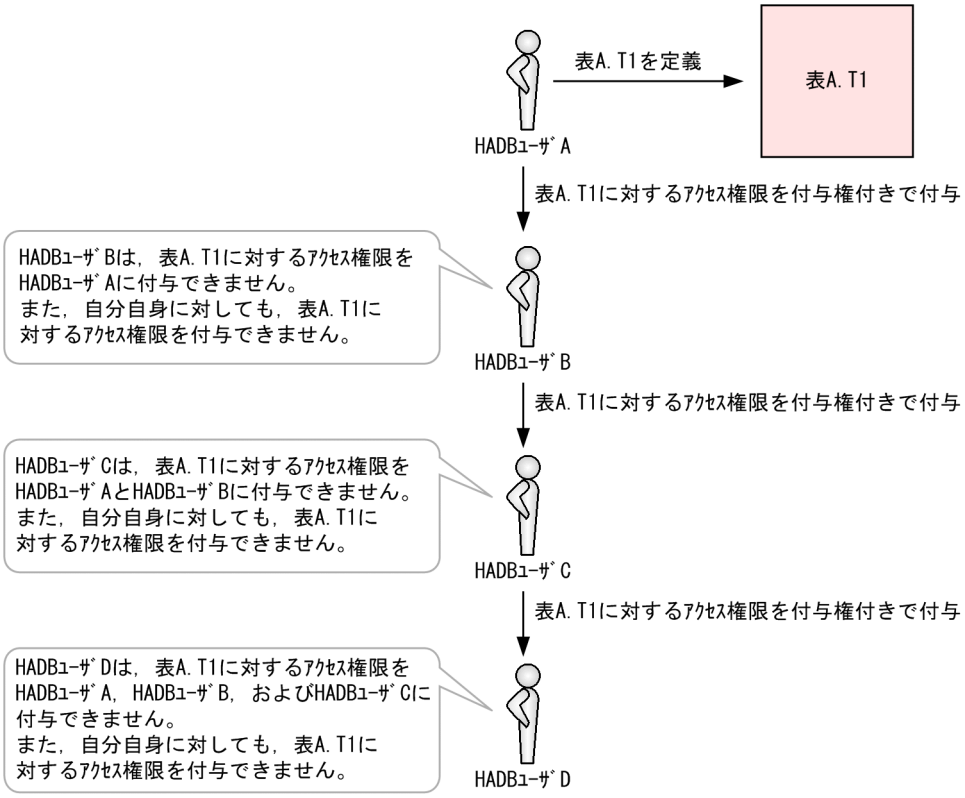
アクセス権限を付与するGRANT 文を実行する場合，次のすべての権限が必要になります。

- CONNECT 権限
- スキーマ定義権限，または付与権付きのアクセス権限

(4) 規則

1. アクセス権限をほかの HADB ユーザに付与するには，そのアクセス権限の付与権を持っている必要があります。
2. 自分が所有しているオブジェクトへのアクセス権限を，自分自身に付与することはできません。
3. アクセス権限の付与権を持っている場合でも，次に示す HADB ユーザにはアクセス権限を付与できません。
   - 付与権付きのアクセス権限を自分に付与した HADB ユーザ
   - 上記の HADB ユーザに，対象のアクセス権限を付与権付きで付与している延長上の HADB ユーザ
   - 自分自身（付与されたアクセス権限を自分に対して付与することはできません）

   （例）



4. ビュー表に対するアクセス権限をほかの HADB ユーザに付与するには，ビュー表の全基表に対するアクセス権限を付与権付きで持っている必要があります。
5. 基表に対するアクセス権限が新たに付与された場合，その基表に依存するビュー表に対するアクセス権限も新たに付与されます（アクセス権限の伝搬が発生します）。例えば，HADB ユーザ A が，表X.T1 を基表としたビュー表A.V1 と，ビュー表A.V1 を基表としたビュー表A.V2 を定義している場合に，表 X.T1 に対するINSERT 権限が HADB ユーザ A に付与されると，ビュー表A.V1，A.V2 に対するINSERT 権限も付与されます。

メモ

- アクセス権限の伝搬が発生するのは，アクセス権限を付与されたHADBユーザが定義したビュー表に限られます。
- ビュー表が無効化されている場合，そのビュー表に対してはアクセス権限の伝搬は発生しません。

重要

基表が複数ある場合に，基表に対するアクセス権限が新たに付与されても，アクセス権限がビュー表に適用される条件を満たさないときは，ビュー表に対するアクセス権限は変更されません。

(例)

HADBユーザAは，表X.T1と表X.T2に対するSELECT権限を持っていて，表X.T1と表X.T2を基表としたビュー表A.V1を定義しているとします。この場合，表X.T1に対するUPDATE権限が付与されても，ビュー表A.V1に対するUPDATE権限は付与されません。表X.T1と表X.T2の両方に対して，UPDATE権限が付与されないと，ビュー表A.V1に対するUPDATE権限は付与されません。このように，1つの基表に対してだけアクセス権限を付与しても，ビュー表に適用されるアクセス権限の条件を満たさない場合は，ビュー表に対するアクセス権限は変更されません。

6. ほかのHADBユーザに付与した付与権付きのアクセス権限から付与権だけを取り消す場合は，GRANT OPTION FORを指定したREVOKE文を実行してください。付与権付きのアクセス権限を付与したHADBユーザに対して，WITH GRANT OPTIONを指定しないでGRANT文を実行し，再度同じアクセス権限を付与しても，付与権を取り消すことはできません。
7. 権限受領者に複数の認可識別子を指定した場合，GRANT文の実行でエラーが発生したときは，すべてのHADBユーザへの権限付与が無効になります。
8. HADBサーバに接続中のHADBユーザが持っているアクセス権限を変更した場合，次のタイミングで変更後のアクセス権限が有効になります。
   - そのHADBユーザが実行する次のトランザクション以降

(5) 例題

例題1

表T1に対するSELECT権限およびINSERT権限を，HADBユーザADBUSER01に付与します。

```
GRANT SELECT,INSERT ON "T1" TO "ADBUSER01"
```

例題2

表T1に対するすべてのアクセス権限を，HADBユーザADBUSER02とADBUSER03に付与します。

```
GRANT ALL PRIVILEGES ON "T1" TO "ADBUSER02","ADBUSER03"
```

例題３

表X.T1 に対するSELECT 権限を付与権付きで，HADB ユーザADBUSER04 に付与します。

```
GRANT SELECT ON ”X”.”T1” TO ”ADBUSER04” WITH GRANT OPTION
```

3.17　REVOKE（権限の取り消し）

ここでは，REVOKE 文の指定形式および規則について説明します。

3.17.1　ユーザ権限，スキーマ操作権限，監査権限，および暗号管理権限の取り消し

HADB ユーザに付与されている次の権限を取り消します。

- ユーザ権限
  - DBA 権限
  - CONNECT 権限
- スキーマ操作権限
  - スキーマ定義権限
- 監査権限
  - 監査管理権限
  - 監査参照権限
- 暗号管理権限

(1)　指定形式

```
REVOKE文 : : =REVOKE 権限〔,権限〕…
                FROM 認可識別子〔,認可識別子〕… 〔削除動作〕

  権限 : : = {ユーザ権限 | スキーマ操作権限 | 監査権限 | 暗号管理権限}
    ユーザ権限 : : = {DBA | CONNECT}
    スキーマ操作権限 : : =SCHEMA
    監査権限 : : = {AUDIT ADMIN | AUDIT VIEWER}
    暗号管理権限 : : =CRYPTO ADMIN

  削除動作 : : = {CASCADE | RESTRICT}
```

(2)　指定形式の説明

●*権限*〔,*権限*〕…

```
権限 : : = {ユーザ権限 | スキーマ操作権限 | 監査権限 | 暗号管理権限}
```

取り消す権限を指定します。同じ権限を重複して指定できません。

ユーザ権限 : : = {DBA | CONNECT}

ユーザ権限を取り消す場合に指定します。

- DBA

  DBA 権限を取り消す場合に指定します。

- CONNECT

  CONNECT 権限を取り消す場合に指定します。

スキーマ操作権限：：＝SCHEMA

スキーマ操作権限を取り消す場合に指定します。

- SCHEMA

  スキーマ定義権限を取り消す場合に指定します。

監査権限：：＝ {AUDIT ADMIN | AUDIT VIEWER}

監査権限（監査管理権限または監査参照権限）を取り消す場合に指定します。

- AUDIT ADMIN

  監査管理権限を取り消す場合に指定します。

- AUDIT VIEWER

  監査参照権限を取り消す場合に指定します。

暗号管理権限：：＝CRYPTO ADMIN

暗号管理権限を取り消す場合に指定します。

- CRYPTO ADMIN

  暗号管理権限を取り消す場合に指定します。

●FROM *認可識別子*〔,*認可識別子*〕…

権限を取り消す HADB ユーザの認可識別子を指定します。最大 128 個の認可識別子を指定できます。

認可識別子の指定規則を次に示します。

- 英小文字を指定する場合は，認可識別子を二重引用符（”）で囲んで指定してください。二重引用符で囲まないと，英小文字を指定しても英大文字が指定されたと見なされます。

  （例）REVOKE DBA FROM adbuser01

  この場合，認可識別子にADBUSER01 が指定されたと見なされます。

- 認可識別子は名前として指定するため，二重引用符（”）で囲んで指定することを推奨します。

認可識別子の指定規則の詳細については，「6.1.4　名前の指定」を参照してください。

●*削除動作*

削除動作：：＝ {CASCADE | RESTRICT}

この指定は，スキーマ定義権限を取り消す場合に限り有効となります。

スキーマ定義権限を取り消す HADB ユーザがスキーマを所有している場合に，スキーマ定義権限を取り消すかどうかを指定します。削除動作の各指定の説明を次の表に示します。

削除動作の指定	説明	ほかのスキーマのビュー表の扱い	ほかのスキーマの外部キーの扱い
削除動作の指定を省略した場合	処理対象の HADB ユーザがスキーマを所有している場合でも，スキーマ定義権限を取り消します。このとき，処理対象の HADB ユーザが所有しているスキーマも削除されます。	REVOKE 文の実行によって削除される表に依存するビュー表（ほかのスキーマのビュー表）が無効化されます。	REVOKE 文の実行によって削除される表を被参照表とする外部キー（ほかのスキーマの外部キー）も削除されます。
CASCADE を指定した場合		REVOKE 文の実行によって削除される表に依存するビュー表（ほかのスキーマのビュー表）も削除されます。	
RESTRICT を指定した場合	処理対象の HADB ユーザがスキーマを所有している場合は，REVOKE 文をエラーにします。	REVOKE 文がエラーになるため，ほかのスキーマのビュー表に影響はありません。	REVOKE 文がエラーになるため，ほかのスキーマの外部キーに影響はありません。

(3) 実行時に必要な権限

- ユーザ権限およびスキーマ操作権限を取り消すREVOKE 文を実行する場合

  DBA 権限およびCONNECT 権限が必要になります。
- 監査権限を取り消すREVOKE 文を実行する場合

  監査管理権限およびCONNECT 権限が必要になります。
- 暗号管理権限を取り消すREVOKE 文を実行する場合

  DBA 権限およびCONNECT 権限が必要になります。

  なお，暗号管理権限およびCONNECT 権限を持っている HADB ユーザは，自分が持っている暗号管理権限を取り消すREVOKE 文を実行できます。

(4) 規則

1. HADB サーバに接続中の認可識別子の HADB ユーザのCONNECT 権限を取り消すことはできません。
2. 自分自身に付与されているDBA 権限およびCONNECT 権限を取り消すことはできません。自分自身に付与されているスキーマ定義権限を取り消すことはできます。
3. 複数の認可識別子を指定した場合，REVOKE 文の実行でエラーが発生したときは，すべての HADB ユーザの権限取り消しが無効になります。
4. 監査権限を持っている HADB ユーザのCONNECT 権限およびスキーマ定義権限を取り消すことはできません。
5. 監査管理権限を持っている HADB ユーザは，次の権限を取り消すことができます。
   - ほかの HADB ユーザが持っている監査管理権限または監査参照権限
   - 自分が持っている監査管理権限または監査参照権限
6. 監査権限を取り消す場合は，監査証跡機能が有効である必要があります。

   ただし，次に示す条件をすべて満たす場合に限り，監査証跡機能が無効なときに監査管理権限を取り消すことができます。

- 監査参照権限を持っている HADB ユーザがいない
- 監査管理権限を持っている HADB ユーザが自分だけである

7. 監査証跡機能が有効な場合に，監査管理権限とCONNECT 権限の両方を持っている HADB ユーザが 1 人しかいないときは，その HADB ユーザの監査管理権限を取り消すことはできません。
8. 暗号管理権限を持っている HADB ユーザのCONNECT 権限を取り消すことはできません。
9. 暗号管理権限を持っている HADB ユーザが 1 人しかいないときは，その HADB ユーザの暗号管理権限を取り消すことはできません。
10. データベース認証を使用し，かつDBA 権限およびCONNECT 権限を持つ HADB ユーザが 1 人しかいないときは，その HADB ユーザのDBA 権限およびCONNECT 権限を取り消すことはできません。

(5) 例題

例題 1

HADB ユーザADBUSER01 のDBA 権限，CONNECT 権限，およびスキーマ定義権限を取り消します。

```
REVOKE DBA,CONNECT,SCHEMA FROM ”ADBUSER01” CASCADE
```

例題 2

HADB ユーザADBUSER02 およびADBUSER03 のCONNECT 権限およびスキーマ定義権限を取り消します。ただし，ADBUSER02，ADBUSER03 がスキーマを所有している場合，REVOKE 文をエラーにします。

```
REVOKE CONNECT,SCHEMA FROM ”ADBUSER02”,”ADBUSER03” RESTRICT
```

例えば，ADBUSER02 はスキーマを所有していて，ADBUSER03 はスキーマを所有していない場合に上記のREVOKE 文を実行すると，ADBUSER02 およびADBUSER03 の両方に対して，REVOKE 文をエラーにします。

例題 3

HADB ユーザADBAUDITADMIN の監査管理権限を取り消します。

```
REVOKE AUDIT ADMIN FROM ”ADBAUDITADMIN”
```

例題 4

HADB ユーザADBAUDITOR の監査参照権限を取り消します。

```
REVOKE AUDIT VIEWER FROM ”ADBAUDITOR”
```

例題 5

HADB ユーザCADMIN の暗号管理権限を取り消します。

```
REVOKE CRYPTO ADMIN FROM ”CADMIN”
```

3.17.2 アクセス権限の取り消し

HADB ユーザに付与されているアクセス権限を取り消します。

(1) 指定形式

```
REVOKE文 : : =REVOKE 〔GRANT OPTION FOR〕 アクセス権限 ON オブジェクト名
                FROM 権限受領者 〔削除動作〕

  アクセス権限 : : = {ALL 〔PRIVILEGES〕 |動作 〔,動作〕…}
    動作 : : = {SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES
              | IMPORT TABLE | REBUILD INDEX | GET COSTINFO | EXPORT TABLE
              | MERGE CHUNK | CHANGE CHUNK COMMENT | CHANGE CHUNK STATUS
              | ARCHIVE CHUNK | UNARCHIVE CHUNK}

  オブジェクト名 : : = { 〔TABLE〕 表名 | ALL TABLES}

  権限受領者 : : = {認可識別子 〔,認可識別子〕 … | PUBLIC}

  削除動作 : : = {CASCADE | RESTRICT}
```

(2) 指定形式の説明

●GRANT OPTION FOR

アクセス権限の付与権だけを取り消す場合に指定します。このオプションを指定してREVOKE 文を実行した場合，アクセス権限自体は取り消されません。アクセス権限の付与権だけが取り消されます。

(例)

```
REVOKE GRANT OPTION FOR SELECT ON "X"."T1" FROM "ADBUSER01"
```

上記のREVOKE 文を実行した場合，HADB ユーザADBUSER01 が持っている，表X.T1 に対するSELECT 権限の付与権だけが取り消されます。HADB ユーザADBUSER01 が持っている，表X.T1 に対するSELECT 権限は取り消されません。

●*アクセス権限*

```
アクセス権限 : : = {ALL 〔PRIVILEGES〕 | 動作 〔,動作〕…}
```

取り消しの対象とするアクセス権限の種類を指定します。

ALL 〔PRIVILEGES〕 :

すべてのアクセス権限を取り消す場合に指定します。

> **重要**
>
> ALL PRIVILEGES を指定してREVOKE 文を実行した場合，取り消し対象となるアクセス権限は，自分が付与したアクセス権限だけとなります。ほかの HADB ユーザが付与したアクセス権限は，取り消し対象になりません。
>
> (例)
>
> HADB ユーザADBUSER01 は，表X.T1 に対して次のアクセス権限を持っているものとします。

- HADB ユーザADBUSER02 から付与されたSELECT 権限とUPDATE 権限
- HADB ユーザADBUSER03 から付与されたINSERT 権限とDELETE 権限

HADB ユーザADBUSER02 が，次のREVOKE 文を実行した場合，SELECT 権限とUPDATE 権限だけが取り消されます。

```
REVOKE ALL PRIVILEGES ON ”X”.”T1” FROM ”ADBUSER01”
```

メモ

取り消し対象オブジェクトに対する付与権付きのアクセス権限を 1 種類も持っていない場合に，ALL PRIVILEGES を指定すると，REVOKE 文がエラーになります。

動作〔,*動作*〕…：

```
動作 : := {SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES
          | IMPORT TABLE | REBUILD INDEX | GET COSTINFO | EXPORT TABLE
          | MERGE CHUNK | CHANGE CHUNK COMMENT | CHANGE CHUNK STATUS
          | ARCHIVE CHUNK | UNARCHIVE CHUNK}
```

取り消しの対象とするアクセス権限の種類を指定します。同じ動作を重複して指定できません。

- 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。

- UPDATE

  UPDATE 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。

- TRUNCATE

  TRUNCATE 権限を取り消す場合に指定します。

- REFERENCES

  REFERENCES 権限を取り消す場合に指定します。

- IMPORT TABLE

  IMPORT TABLE 権限を取り消す場合に指定します。

- REBUILD INDEX

  REBUILD INDEX 権限を取り消す場合に指定します。

- GET COSTINFO

  GET COSTINFO 権限を取り消す場合に指定します。

- EXPORT TABLE

  EXPORT TABLE 権限を取り消す場合に指定します。

- MERGE CHUNK

  MERGE CHUNK 権限を取り消す場合に指定します。

- CHANGE CHUNK COMMENT

  CHANGE CHUNK COMMENT 権限を取り消す場合に指定します。

- CHANGE CHUNK STATUS

  CHANGE CHUNK STATUS 権限を取り消す場合に指定します。

- ARCHIVE CHUNK

  ARCHIVE CHUNK 権限を取り消す場合に指定します。

- UNARCHIVE CHUNK

  UNARCHIVE CHUNK 権限を取り消す場合に指定します。

●ON *オブジェクト名*

オブジェクト名：：＝｛〔TABLE〕 *表名*｜ALL TABLES｝

アクセス権限を取り消す際の，対象オブジェクトを指定します。

なお，**オブジェクト**とは，スキーマオブジェクトのことを意味しています。

〔TABLE〕 *表名*：

ここで指定した表に対するアクセス権限を取り消します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，無効化されているビュー表の表名は指定できません。

ALL TABLES：

実行ユーザのスキーマ内の全実表に対するアクセス権限を取り消します。ここでいう**実行ユーザ**とは，REVOKE 文を実行した HADB ユーザを意味しています。

REVOKE 文を実行した HADB ユーザがスキーマを定義していない場合，またはスキーマ内に 1 つも表を定義していない場合，アクセス権限を取り消さないでREVOKE 文を正常終了します。

●FROM *権限受領者*

権限受領者：：＝｛*認可識別子*〔,*認可識別子*〕…｜PUBLIC｝

アクセス権限を取り消す HADB ユーザを指定します。

認可識別子〔,*認可識別子*〕…：

アクセス権限を取り消す HADB ユーザの認可識別子を指定します。最大 128 個の認可識別子を指定できます。

認可識別子の指定規則を次に示します。

- 英小文字を指定する場合は，認可識別子を二重引用符（”）で囲んで指定してください。二重引用符で囲まないと，英小文字を指定しても英大文字が指定されたと見なされます。

- 認可識別子は名前として指定するため，二重引用符（”）で囲んで指定することを推奨します。

認可識別子の指定規則の詳細については，「6.1.4　名前の指定」を参照してください。

PUBLIC：

GRANT 文のPUBLIC 指定で許可したアクセス権限を取り消す場合に指定します。

（例）

```
REVOKE SELECT,IMPORT TABLE ON ”T1” FROM PUBLIC
```

上記のREVOKE 文を実行した場合，GRANT 文のPUBLIC 指定で許可した次のアクセス権限が取り消されます。

- 表T1 に対するSELECT 権限
- 表T1 に対するIMPORT TABLE 権限

> **メモ**
>
> PUBLIC キーワードは，すべての HADB ユーザを表す，システムによって暗黙的に作成されたユーザと考えることができます。

●*削除動作*

削除動作：：＝ {CASCADE | RESTRICT}

この指定は，次のどちらかの場合に限り有効となります。

- SELECT 権限またはREFERENCES 権限を取り消す場合
- 依存権限が存在するアクセス権限を取り消す場合

削除動作の指定を省略した場合，CASCADE が仮定されます。

CASCADE：

次のどれかの条件を満たす場合でも，アクセス権限を取り消すときに指定します。

- 取り消し対象のSELECT 権限を使用して定義したビュー表がある場合
  この場合，該当するビュー表は無効化されます。また，無効化されるビュー表に依存するビュー表も無効化されます。
- 取り消し対象のREFERENCES 権限を使用して定義した参照制約がある場合
  この場合，該当する参照制約は削除されます。
- 取り消し対象のアクセス権限（付与権だけを取り消す場合も含む）に依存権限が存在する場合
  この場合，該当する依存権限が取り消されます。また，依存権限を使用したビュー表や参照制約が存在する場合，そのビュー表は無効化され，参照制約は削除されます。

RESTRICT：

次のどれかの条件を満たす場合は，REVOKE 文をエラーとするときに指定します。

- 取り消し対象のSELECT 権限を使用して定義したビュー表がある場合

- 取り消し対象のREFERENCES 権限を使用して定義した参照制約がある場合
- 取り消し対象のアクセス権限（付与権だけを取り消す場合も含む）に依存権限が存在する場合

(3) 実行時に必要な権限

アクセス権限を取り消すREVOKE 文を実行する場合，次のすべての権限が必要になります。

- CONNECT 権限
- スキーマ定義権限，または付与権付きのアクセス権限

(4) 規則

1. 自分が付与したアクセス権限だけを取り消すことができます。
2. 取り消し対象のアクセス権限を付与権付きで持っていない場合に，REVOKE 文を実行するとエラーになります。
3. 自分自身が持っている，自分が所有しているオブジェクトへのアクセス権限を取り消すことはできません。
4. 権限受領者に複数の認可識別子を指定した場合，REVOKE 文の実行でエラーが発生したときは，すべての HADB ユーザへの権限取り消しが無効になります。
5. 基表に対するアクセス権限が取り消された場合，その基表に依存するビュー表に対するアクセス権限も取り消されます（アクセス権限の伝搬が発生します）。

   例えば，HADB ユーザ A が，表X.T1 を基表としたビュー表A.V1 と，ビュー表A.V1 を基表としたビュー表A.V2 を定義している場合に，表X.T1 に対するINSERT 権限が取り消されると，ビュー表A.V1，A.V2 に対するINSERT 権限も取り消されます。

   ただし，ビュー表が無効化されている場合，そのビュー表に対してはアクセス権限の取り消しは伝搬しません。
6. 基表に対するSELECT 権限が取り消された場合，その基表に依存するすべてのビュー表が無効化されます。

   （例）

   HADB ユーザ A は，表X.T1 に対するSELECT 権限を持っていて，表X.T1 を基表としたビュー表A.V1 を定義しているとします。さらに，ビュー表A.V1 を基表としたビュー表A.V2 と，ビュー表A.V2 を基表としたビュー表A.V3 を定義しているとします。

   表X.T1 に対するSELECT 権限が取り消された場合，表X.T1 に依存するビュー表A.V1，A.V2，A.V3 は無効化されます。
7. 表に対するREFERENCES 権限が取り消された場合，そのREFERENCES 権限を使用して定義した参照制約が削除されます。例えば，HADB ユーザ A が持っている，表X.T1 に対するREFERENCES 権限が取り消された場合，HADB ユーザ A が表X.T1 を被参照表にして定義した参照制約が削除されます。
8. 同じ表に対するアクセス権限が，複数の HADB ユーザから付与されていたり，PUBLIC 指定でアクセス権限が許可されていたりする場合の，アクセス権限の取り消し規則を説明します。SELECT 権限の場合を例にして説明します。

（例）

下記の SQL 文を実行したとします。

```
GRANT SELECT ON ”ADBUSER01”.”T1” TO ”ADBUSER03”     ...1  ←HADBユーザADBUSER01が実行
GRANT SELECT ON ”ADBUSER01”.”T1” TO ”ADBUSER03”     ...2  ←HADBユーザADBUSER02が実行
GRANT SELECT ON ”ADBUSER01”.”T1” TO PUBLIC          ...3  ←HADBユーザADBUSER01が実行
REVOKE SELECT ON ”ADBUSER01”.”T1” FROM ”ADBUSER03”  ...4  ←HADBユーザADBUSER01が実行
REVOKE SELECT ON ”ADBUSER01”.”T1” FROM ”ADBUSER03”  ...5  ←HADBユーザADBUSER02が実行
REVOKE SELECT ON ”ADBUSER01”.”T1” FROM PUBLIC       ...6  ←HADBユーザADBUSER01が実行
```

［説明］

- 1.～3.のGRANT 文を実行し，HADB ユーザADBUSER03 に，表ADBUSER01.T1（以降表T1 と表記）に対するSELECT 権限を付与，および許可します。
- 4.のREVOKE 文を実行した場合，1.で付与されたSELECT 権限だけが取り消されます。2.で付与されたSELECT 権限と，3.で許可されたSELECT 権限は取り消されません。
- 次に，5.のREVOKE 文を実行した場合，2.で付与されたSELECT 権限だけが取り消されます。3.で許可されたSELECT 権限は取り消されません。
- 次に，6.のREVOKE 文を実行した場合，3.で許可されたSELECT 権限が取り消されます。

  このとき，表T1 に対するSELECT 権限がすべて取り消されたため，表T1 を基表としたビュー表をHADB ユーザADBUSER03 が定義している場合，この時点でビュー表が無効化されます。

なお，REFERENCES 権限の取り消しについても，上記のSELECT 権限の取り消しと同様になります。そのため，表T1 に対するREFERENCES 権限がすべて取り消されたときに参照制約が削除されます。

9. HADB サーバに接続中の HADB ユーザが持っているアクセス権限を変更した場合，次のタイミングで変更後のアクセス権限が有効になります。
   - その HADB ユーザが実行する次のトランザクション以降

(5) 例題

例題 1

HADB ユーザADBUSER01 が持っている，表T1 に対するSELECT 権限およびINSERT 権限を取り消します。

```
REVOKE SELECT,INSERT ON ”T1” FROM ”ADBUSER01”
```

上記のREVOKE 文を実行した場合，ADBUSER01 が表T1 を基表としたビュー表を定義しているときは，そのビュー表は無効化されます。また，無効化されるビュー表に依存するビュー表も無効化されます。

例題 2

HADB ユーザADBUSER02 およびADBUSER03 が持っている，表T1 に対するすべてのアクセス権限を取り消します。

```
REVOKE ALL PRIVILEGES ON ”T1” FROM ”ADBUSER02”,”ADBUSER03” RESTRICT
```

RESTRICT を指定しているため，ADBUSER02 またはADBUSER03 が，次のことを行っている場合は，REVOKE 文をエラーにします。

- 表T1 を基表としたビュー表を定義している場合
- 表T1 を被参照表にした参照制約を定義している場合

例題 3

HADB ユーザADBUSER01 が持っている，表X.T1 に対するSELECT 権限の付与権だけを取り消します。

```
REVOKE GRANT OPTION FOR SELECT ON "X"."T1" FROM "ADBUSER01"
```

上記のREVOKE 文を実行した場合，HADB ユーザADBUSER01 が持っている，表X.T1 に対するSELECT 権限は取り消されません。そのため，表X.T1 を基表としたビュー表をADBUSER01 が定義している場合でも，そのビュー表は無効化されません。ただし，次に示すような場合は，ADBUSER01 以外の HADB ユーザが定義したビュー表が無効化されます。

- ADBUSER01 が，表X.T1 に対するSELECT 権限をほかの HADB ユーザ（例えば，ADBUSER02）に付与していた場合，ADBUSER02 に付与した表X.T1 に対するSELECT 権限が取り消されます。そのため，ADBUSER02 が，表X.T1 を基表としたビュー表ADBUSER02.V1 を定義している場合，ビュー表ADBUSER02.V1 は無効化されます。
- ADBUSER01 が，表X.T1 を基表としたビュー表ADBUSER01.V1 を定義しているとします。ビュー表ADBUSER01.V1 に対するSELECT 権限をほかの HADB ユーザ（例えば，ADBUSER02）に付与していた場合，ADBUSER02 に付与したビュー表ADBUSER01.V1 に対するSELECT 権限が取り消されます。そのため，ADBUSER02 が，ビュー表ADBUSER01.V1 を基表としたビュー表ADBUSER02.V2 を定義している場合，ビュー表ADBUSER02.V2 は無効化されます。

3.18 定義系 SQL 実行時の留意事項

1. 定義系 SQL を実行した場合，定義系 SQL が正常終了した時点で自動的にCOMMIT が実行されてトランザクションが終了します。
2. 定義系 SQL はロールバックの対象になりません。
3. 次のすべての条件を満している場合，定義系 SQL がエラーになります。
   - 定義系 SQL を実行するコネクション内に，次に示す表の検索を行うカーソルがある
     ・定義系 SQL で参照，更新するディクショナリ表
     ・定義系 SQL で変更，削除する表
   - そのカーソル（上記の下線部分で示すカーソル）がオープン中である
4. 次のすべての条件を満している場合，定義系 SQL がエラーになります。
   - JDBC ドライバを使用して定義系 SQL を実行している
   - 同一コネクション内に，次に示す表の検索を行うStatement オブジェクトおよびPreparedStatement オブジェクトがある
     ・定義系 SQL で参照，更新するディクショナリ表
     ・定義系 SQL で変更，削除する表
5. JDBC ドライバを使用して定義系 SQL を実行する場合，同一コネクション内にResultSet オブジェクトがあると，定義系 SQL がエラーになります。

4 操作系 SQL

この章では，操作系 SQL の機能，指定形式，および規則について説明します。

4.1　DELETE（行の削除）

ここでは，DELETE 文の指定形式および規則について説明します。

4.1.1　DELETE 文の指定形式および規則

指定した探索条件を満たす行を削除します。

(1)　指定形式

DELETE文 : : ＝DELETE FROM *表名*　〔〔AS〕 *相関名*〕　〔WHERE *探索条件*〕

(2)　指定形式の説明

●*表名*

削除したい行がある表（削除対象表）の表名を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

指定規則を次に示します。

- 読み取り専用ビュー表は指定できません。
- 配列型の列を定義した表は指定できません。

●〔AS〕 *相関名*

削除対象表の相関名を指定します。相関名については，「6.1.5　名前の修飾」の「(4)　表指定の指定形式」を参照してください。相関名の有効範囲については，「6.8　範囲変数」を参照してください。

●WHERE *探索条件*

削除対象とする行の条件を*探索条件*に指定します。探索条件については，「7.19　探索条件」を参照してください。

WHERE *探索条件*を省略すると，指定した表のすべての行が削除されます。

指定規則を次に示します。

- 探索条件中に？パラメタを指定できます。

*表名*に更新可能ビュー表を指定した場合の留意事項を次に示します。

- 更新可能ビュー表の行を削除した場合，基表の行が削除されます。
- ビュー表を定義した際に指定した探索条件と，ここで指定した探索条件の両方を満たす行が，基表から削除されます。
- WHERE *探索条件*を省略した場合は，ビュー表を定義した際に指定した探索条件を満たす行が，基表から削除されます。

(3)　実行時に必要な権限

DELETE 文を実行する場合，次に示すすべての権限が必要になります。

- CONNECT 権限
- 行を削除する表に対するDELETE 権限
- 問合せ式本体に指定する表に対するSELECT 権限

（例）

```
DELETE FROM "T1"
    WHERE "T1"."C1" IN (SELECT "C1" FROM "T2" WHERE "C3"<=100)
```

上記のDELETE 文を実行する場合，表 T1 に対するDELETE 権限と，表 T2 に対するSELECT 権限が必要になります。

(4)　規則

1. 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。
2. DELETE 文中に指定できる表，導出表，表関数導出表，および集まり導出表の延べ数は，最大 2,048 個になります。SQL 文中に指定されている表，導出表，表関数導出表，および集まり導出表の数え方の規則と例については，「4.4.1　SELECT 文の指定形式および規則」の「(4)　規則」を参照してください。
3. DELETE 文中に指定している集合演算がすべてUNION の場合，指定できる集合演算の数は最大 1,023 個になります。ただし，指定した集合演算にEXCEPT またはINTERSECT がある場合は，指定できる集合演算の数は最大 63 個になります。
4. DELETE 文中に指定できる外結合（FULL OUTER JOIN）の数は，最大 63 個になります。
5. ディクショナリ表またはシステム表の行は削除できません。
6. 探索条件中の副問合せのFROM 句には，行の削除対象となる表を指定できません。
7. カラムストア表に対してDELETE 文を実行する場合，ローストア表とは異なる運用や設計が必要になります。詳細については，マニュアル『HADB　システム構築・運用ガイド』の『ローストア表とカラムストア表の選択基準』，『シングルチャンク表の再編成が必要かどうかを確認する方法』，および『マルチチャンク表の再編成が必要かどうかを確認する方法』を参照してください。
8. アーカイブされている行は削除できません。アーカイブされている行を削除するDELETE 文はエラーになります。アーカイブされている行を削除したい場合は，削除したい行が格納されているチャンクのアーカイブ状態をいったん解除してください。そのあとに，DELETE 文を実行して行を削除してください。
9. アーカイブされていない行は，DELETE 文で削除できます。ただし，DELETE 文を実行する際，次の条件をすべて満たす必要があります。
   - 探索条件に，アーカイブレンジ列を指定した条件を指定する
   - アーカイブレンジ列を指定した探索条件に，比較述語，IN 述語，またはBETWEEN 述語だけを指定する
   - アーカイブレンジ列を指定した探索条件に，OR 条件やNOT 条件などを指定しない

- アーカイブされている行を削除対象にしていない

上記の条件をすべて満たさないと，DELETE 文がエラーになります。

> **重要**
>
> アーカイブレンジ列を指定した探索条件に指定できる述語に制限があります。また，OR 条件やNOT 条件以外にも，探索条件に指定するとDELETE 文がエラーになる条件があります。詳細については，マニュアル『HADB　AP 開発ガイド』の『アーカイブレンジ列の日時情報を使用した検索範囲の絞り込み』を参照してください。

DELETE 文が実行できる例とできない例の代表的な例を次に示します。なお，例中のARCHIVE-T1 表はアーカイブマルチチャンク表とし，RECORD-DAY 列はアーカイブレンジ列とします。

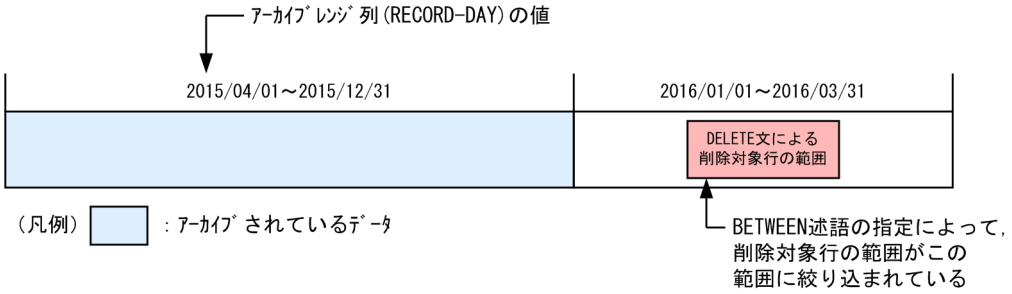
■DELETE 文が実行できる例

（例）

```
DELETE FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2016/02/01' AND DATE'2016/02/29'
      AND "CODE"='P001'
```

上記の例の場合，次の条件をすべて満たしているため，DELETE 文を実行できます。

- 探索条件に，アーカイブレンジ列（RECORD-DAY）を指定している
- アーカイブレンジ列を指定した探索条件に，BETWEEN 述語だけが指定されている
- アーカイブレンジ列を指定した探索条件に，OR 条件やNOT 条件などを指定していない
- アーカイブされている行を削除対象にしていない



> **重要**
>
> 探索条件に指定するアーカイブレンジ列に対する比較条件は，定数を指定することを推奨します。
>
> （例）推奨する指定例
>
> "RECORD-DAY" BETWEEN DATE'2016/01/01' AND DATE'2016/01/10'
>
> "RECORD-DAY" >= DATE'2016/02/10'
>
> 定数以外の指定は推奨しません。

（例）推奨しない指定例

```
"RECORD-DAY" BETWEEN ? AND ?
"RECORD-DAY" >= CURRENT_DATE
```

メモ

削除対象のデータがアーカイブされているかどうかを，HADB サーバは，アーカイブレンジ列を指定した探索条件によって判定しています。その際，アーカイブレンジ列に対する比較条件に定数を指定した場合，判定に掛かる時間を短くできます。一方，定数以外を指定した場合，判定に掛かる時間が非常に長くなるおそれがあります。

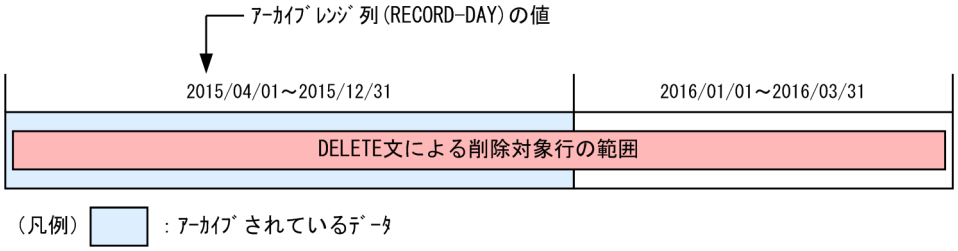
■DELETE 文が実行できない例

・探索条件にアーカイブレンジ列を指定していない

（例 1）

```
DELETE FROM "ARCHIVE-T1" _____
```

上記の例の場合，探索条件にアーカイブレンジ列（RECORD-DAY）を指定していないため，DELETE 文がエラーになります。



（例 2）

```
DELETE FROM "ARCHIVE-T1"
    WHERE "CODE"='P001'
```

上記の例の場合，探索条件にアーカイブレンジ列（RECORD-DAY）を指定していないため，DELETE 文がエラーになります。アーカイブされていない行を削除する場合でもエラーになります。

・アーカイブレンジ列を指定した探索条件にOR 条件やNOT 条件などを指定している

（例）

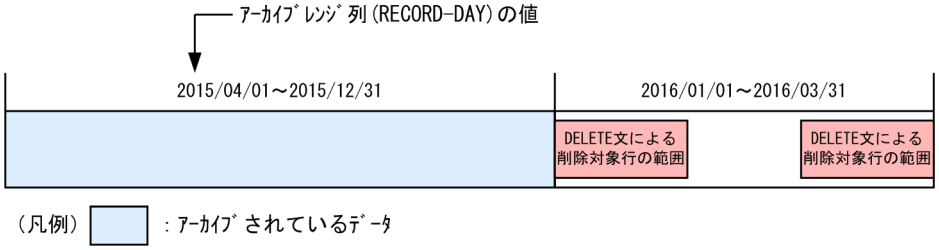
```
DELETE FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2016-01-01' AND DATE'2016-01-31'
       OR "RECORD-DAY" BETWEEN DATE'2016-03-01' AND DATE'2016-03-31'
```

上記の例の場合，アーカイブレンジ列を指定した探索条件にOR 条件が指定されているため，DELETE 文がエラーになります。上記のように，アーカイブされていない行を削除する場合でもエラーになります。

上記の例の場合，次のようにDELETE 文を 2 回に分けて実行すると，行を削除できます。

```
DELETE FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2016-01-01' AND DATE'2016-01-31'
```

```
DELETE FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2016-03-01' AND DATE'2016-03-31'
```

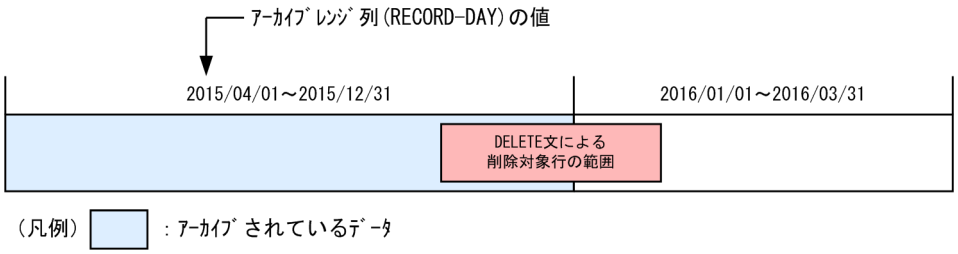


・アーカイブされている行を削除対象にしている

(例)

```
DELETE FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2015/11/01' AND DATE'2016/01/31'
```

上記の例の場合，アーカイブされている行を削除対象にしているため，DELETE 文がエラーになります。



・単独の列指定でアーカイブレンジ列を指定していない

(例)

```
DELETE FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" - 10 DAY > DATE'2016/02/01'
```

上記の例の場合，アーカイブレンジ列を使った日時演算を指定しているため，DELETE 文がエラーになります。

・アーカイブレンジ列の比較条件に日時演算を指定している

(例)

```
DELETE FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" >= CURRENT_DATE - 1 MONTH
```

上記の例の場合，アーカイブレンジ列の比較条件に日時演算を指定しているため，DELETE 文がエラーになります。

10. DELETE 文中にアーカイブマルチチャンク表を指定した場合，ロケーション表およびシステム表（STATUS_CHUNKS）に対するアクセスが発生します。その際，システム表（STATUS_CHUNKS）の排他資源が確保されます。排他制御の詳細については，マニュアル『HADB　システム構築・運用ガイド』の『排他制御』を参照してください。

(5) 例題

例題 1

顧客表（USERSLIST）から，顧客 ID（USERID）がU00212 の行を削除します。

```
DELETE FROM ”USERSLIST”
    WHERE ”USERID”=’U00212’
```

例題 2

販売履歴表（SALESLIST）から，購入日（PUR-DATE）が 2011/9/4～2011/9/5 の行を削除します。

```
DELETE FROM ”SALESLIST”
    WHERE ”PUR-DATE” BETWEEN DATE’2011-09-04’ AND DATE’2011-09-05’
```

4.2 INSERT（行の挿入）

ここでは，INSERT 文の指定形式および規則について説明します。

4.2.1 INSERT 文の指定形式および規則

表に行を挿入します。値を指定して 1 つの行を挿入したり，問合せ式本体を使用して 1 つまたは複数の行を挿入したりすることができます。

(1) 指定形式

■列単位に挿入値を設定する場合

```
INSERT文 : :＝INSERT INTO 表名 〔〔AS〕 相関名〕
                { 〔(列名 〔,列名〕…)〕
                          {問合せ式本体 | VALUES(挿入値 〔,挿入値〕…)}
                  | DEFAULT VALUES
                }

  挿入値 : :＝ {値式 | NULL | DEFAULT}
```

■行単位の挿入をする場合

```
INSERT文 : :＝INSERT INTO 表名 〔〔AS〕 相関名〕 (ROW) VALUES(行挿入値)

  行挿入値 : :＝?パラメタ
```

(2) 指定形式の説明

●*表名*

行を挿入する表（**挿入対象表**）の表名を指定します。表名の指定規則については，「6.1.5 名前の修飾」の「(2) 表名の指定形式」を参照してください。

指定規則を次に示します。

- 問合せ式本体中に指定した表と同じ表を指定できません。
- 読み取り専用ビュー表は指定できません。
- 配列型の列を定義した表は指定できません。

*表名*に更新可能ビュー表を指定した場合の留意事項を次に示します。

- 更新可能ビュー表に行を挿入した場合，基表に行が挿入されます。このとき，ビュー表を定義した際に指定した探索条件に関係なく行を挿入できます。
- 更新可能ビュー表に行を挿入する際，更新可能ビュー表の列に対応しない基表の列には，列の既定値が格納されます。列の既定値については，「7.10 DEFAULT 句」を参照してください。

  なお，DEFAULT 句で列の既定値を指定していない場合は，列の既定値としてナル値が格納されます。

重要

更新可能ビュー表の列に対応しない基表の列に，非ナル値制約（ナル値を許さない）が定義されている場合，その列にナル値を格納するような行の挿入はできません。

●〔AS〕 *相関名*

挿入対象表の相関名を指定します。相関名については，「6.1.5　名前の修飾」の「(4)　表指定の指定形式」を参照してください。

●(*列名*〔，*列名*〕…)

データを挿入する列の列名を指定します。

列名の指定を省略した列には，CREATE TABLE 文のDEFAULT 句で指定した列の既定値が格納されます。ただし，次の場合は，列の既定値としてナル値が格納されます。

- CREATE TABLE 文のDEFAULT 句で列の既定値を指定していない場合

なお，列名を 1 つも指定しない場合は，CREATE TABLE 文で表を定義したときの列の指定順序に従って，すべての列が指定されたと仮定されます。

●*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，「7.1.1　問合せ式の指定形式および規則」の「(2)　指定形式の説明」の「(b)　問合せ式本体」を参照してください。

指定規則を次に示します。

- 挿入対象表と同じ表を問合せ式本体中に指定できません。
- 問合せ式本体の選択式の値式には，配列データを指定できません。

●VALUES(*挿入値*〔，*挿入値*〕…)

挿入値 ：：＝ {*値式* | NULL | DEFAULT}

*列名*で指定した各列に対応する挿入値を指定します。*挿入値*には次のどれかを指定します。

値式：

挿入値を値式の形式で指定します。値式については，「7.21　値式」を参照してください。

指定規則を次に示します。

- *値式*中に列指定は指定できません。
- 挿入対象表と同じ表を*挿入値*中に指定できません。

NULL：

挿入値をナル値にする場合に指定します。

DEFAULT：

CREATE TABLE 文のDEFAULT 句で指定した列の既定値を，挿入値にする場合に指定します。DEFAULT 句で列の既定値を指定していない場合は，列の既定値としてナル値が仮定されます。

●DEFAULT VALUES

挿入対象表のすべての列に対して，列の既定値を挿入する場合に指定します。

DEFAULT VALUES の指定は，次の指定と同じになります。

```
VALUES(DEFAULT,DEFAULT,…)
```

上記のDEFAULT の指定数は，挿入対象表の列数と同じになります。

DEFAULT 句の指定がない表に対して，DEFAULT VALUES を指定した場合，列の既定値としてナル値が仮定されるため，すべての列にナル値が格納されます。

●ROW

行単位でデータを挿入する場合に指定します。ROW を指定すると，行全体を 1 つのデータとして挿入します。

ROW を指定する場合の規則を次に示します。

- FIX 表に対してだけ指定できます。
- 問合せ式本体は指定できません。

●VALUES(*行挿入値*)

行挿入値：：＝*?パラメタ*

行全体に挿入するデータを指定します。

？パラメタに仮定されるデータ型はCHAR 型です。また，データ長は挿入対象表の行長になります。また，構造体中に境界調整による空きがないようにしてください。行長の計算方法については，マニュアル『HADB　システム構築・運用ガイド』の『行の種別ごとの格納ページ数の求め方』の計算式*ROWSZ*を参照してください。

なお，？パラメタは 1 つだけ指定できます。

(3)　実行時に必要な権限

INSERT 文を実行する場合，次に示すすべての権限が必要になります。

- CONNECT 権限
- 行を挿入する表に対するINSERT 権限
- 問合せ式本体に指定する表に対するSELECT 権限

(例)

```
INSERT INTO ”T1”
    (”C1”,”C2”,”C3”)
    SELECT ”C1”,”C2”,”C3” FROM ”T2” WHERE ”C3”<=100
```

上記のINSERT 文を実行する場合，表 T1 に対するINSERT 権限と，表 T2 に対するSELECT 権限が必要になります。

(4) 規則

1. INSERT 文中に指定できる表，導出表，表関数導出表，および集まり導出表の延べ数は，最大 2,048 個になります。SQL 文中に指定されている表，導出表，表関数導出表，および集まり導出表の数え方の規則と例については，「4.4.1 SELECT 文の指定形式および規則」の「(4) 規則」を参照してください。
2. INSERT 文中に指定している集合演算がすべてUNION の場合，指定できる集合演算の数は最大 1,023 個になります。ただし，指定した集合演算にEXCEPT またはINTERSECT がある場合は，指定できる集合演算の数は最大 63 個になります。
3. INSERT 文中に指定できる外結合（FULL OUTER JOIN）の数は，最大 63 個になります。
4. 列単位に挿入値を設定する場合，列名および挿入値の指定数は 4,000 以下にしてください。
5. 列単位に挿入値を設定する場合，列名と挿入値の指定数を同じにしてください。また，挿入値のデータ型は，データを挿入する列のデータ型か，または変換して代入できるデータ型にする必要があります。変換して代入できるデータ型については，「6.2.2 変換，代入，比較できるデータ型」を参照してください。

   （例）

   ```
   INSERT INTO "T1" ("C1","C2","C3")
       VALUES('U00358',5,DATE'2011-09-08')
   ```

   この場合，次に示す規則を守る必要があります。

   - 列を 3 個（C1，C2，C3）指定しているため，挿入値も 3 個指定する必要があります。
   - 挿入値のデータ型は，C1，C2，C3 列のデータ型と同じにするか，または変換して代入できるデータ型にする必要があります。例えば，C3 列がDATE 型の場合，挿入値のデータもDATE 型にする必要があります。
6. 挿入値に？パラメタを指定する場合，仮定されるデータ型およびデータ長は挿入対象列のデータ型およびデータ長になります。
7. DECIMAL 型，NUMERIC 型，DOUBLE PRECISION 型，またはFLOAT 型のデータを，次に示すデータ型の列に挿入する場合，端数（小数）部分が切り捨てられます。

   - INTEGER
   - SMALLINT

   また，DECIMAL 型またはNUMERIC 型のデータを，DECIMAL 型またはNUMERIC 型の列に挿入する場合，列の位取りより下位の桁部分が切り捨てられます。DOUBLE PRECISION 型またはFLOAT 型のデータを，DECIMAL 型またはNUMERIC型の列に挿入する場合，列の位取りより下位の桁部分が丸められます（最近接偶数への丸め）。
8. 表定義時に指定した長さ以上の文字データまたはバイナリデータを挿入することはできません。
9. 列に定義されているデータ型の範囲外の数データを挿入することはできません。
10. CHAR 型の列に挿入するデータが列長より短い場合，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。

11. BINARY 型の列に挿入するデータが列長より短い場合，左詰めに挿入され，余りの部分にX'00'が設定されます。

12. ディクショナリ表またはシステム表には行を挿入できません。

13. カラムストア表に対してINSERT 文を実行する場合，ローストア表とは異なる運用や設計が必要になります。詳細については，マニュアル『HADB　システム構築・運用ガイド』の『ローストア表とカラムストア表の選択基準』，『シングルチャンク表の再編成が必要かどうかを確認する方法』，および『マルチチャンク表の再編成が必要かどうかを確認する方法』を参照してください。

(5) 例題

例題 1　VALUES 指定による行の挿入

販売履歴表（SALESLIST）に次に示すデータ（行）を挿入します。

- 顧客 ID（USERID）：U00358
- 商品コード（PUR-CODE）：P003
- 販売個数（PUR-NUM）：5
- 購入日（PUR-DATE）：2011-09-08

```
INSERT INTO "SALESLIST"
    ("USERID","PUR-CODE","PUR-NUM","PUR-DATE")
    VALUES('U00358','P003',5,DATE'2011-09-08')
```

例題 2　VALUES 指定による行の挿入（挿入値に副問合せを指定した場合）

販売履歴表（SALESLIST）に次に示すデータ（行）を挿入します。

- 商品コード（PUR-CODE）：P003
- 商品名（PUR-NAME）：商品表（PRODUCTLIST）の商品コードP003 に対応する商品名
- 商品カラー（PUR-COL）：商品表（PRODUCTLIST）の商品コードP003 に対応する商品カラー

```
INSERT INTO "SALESLIST"("PUR-CODE","PUR-NAME","PUR-COL")
    VALUES('P003',
          (SELECT "PUR-NAME" FROM "PRODUCTLIST" WHERE "PUR-CODE"='P003'),
          (SELECT "PUR-COL" FROM "PRODUCTLIST" WHERE "PUR-CODE"='P003'))
```

例題 3　問合せ式本体による行の挿入

販売履歴表（SALESLIST）に，北地区の販売履歴表（SALESLIST_N）のデータを挿入します。

- 販売履歴表（SALESLIST）と北地区の販売履歴表（SALESLIST_N）の表の列構成は同じとします。
- 北地区の販売履歴表（SALESLIST_N）の購入日（PUR-DATE_N）が，2011/9/6 以降のデータを挿入対象とします。

```
INSERT INTO "SALESLIST"
    ("USERID","PUR-CODE","PUR-NUM","PUR-DATE")
    SELECT "USERID_N","PUR-CODE_N","PUR-NUM_N","PUR-DATE_N"
        FROM "SALESLIST_N" WHERE "PUR-DATE_N">=DATE'2011-09-06'
```

例題４　ROW 指定による行の挿入

販売履歴表（SALESLIST）に新規の販売情報を追加します（ROW指定で行を挿入します）。販売履歴表の列構成は，顧客 ID（USERID），商品コード（PUR-CODE），販売個数（PUR-NUM），購入日（PUR-DATE）です。

```
INSERT INTO "SALESLIST"(ROW)
    VALUES(?)
```

4.3　PURGE CHUNK（チャンク内の全行削除）

ここでは，PURGE CHUNK 文の指定形式および規則について説明します。

4.3.1　PURGE CHUNK 文の指定形式および規則

チャンク内のすべての行を削除します。

PURGE CHUNK 文は，マルチチャンク表に対してだけ実行できます。

(1)　指定形式

```
PURGE CHUNK文 : :＝PURGE CHUNK 表名　〔〔AS〕 相関名〕
                        WHERE 探索条件
```

(2)　指定形式の説明

●*表名*

処理対象のマルチチャンク表の表名（チャンク削除対象表）を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

なお，ビュー表は指定できません。

●〔AS〕 *相関名*

チャンク削除対象表の相関名を指定します。相関名については，「6.1.5　名前の修飾」の「(4)　表指定の指定形式」を参照してください。

●WHERE *探索条件*

処理対象のチャンクのチャンク ID を指定します。

CHUNKID を指定した探索条件を指定します。探索条件については，「7.19　探索条件」を参照してください。

*探索条件*には，比較述語，IN 述語，または限定述語だけが指定できます。

比較述語：

比較述語については，「7.20.7　比較述語」を参照してください。

PURGE CHUNK 文固有の指定規則を次に示します。

- 比較演算子には，＝ だけを指定できます。
- 比較演算項 1 または比較演算項 2 のどちらかにCHUNKID を指定してください。
- CHUNKID の比較対象となる比較演算項には，整数定数，？パラメタ，またはスカラ副問合せを指定してください。スカラ副問合せについては，「7.3.1　副問合せの指定形式および規則」を参照してください。

- 比較演算項に？パラメタを指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

指定形式の例を次に示します。

```
WHERE CHUNKID=整数定数
WHERE CHUNKID=?
WHERE CHUNKID=(スカラ副問合せ)
```

重要

比較述語を使用してチャンク ID を指定する場合，チャンク ID は 1 つしか指定できません。そのため，比較述語を使用して，複数のチャンクの行を削除する場合は，PURGE CHUNK 文を複数回実行する必要があります。

なお，IN 述語または限定述語を使用すると，条件に合致した複数のチャンクの行を 1 度のPURGE CHUNK 文で削除することができます。

IN 述語：

IN 述語については，「7.20.3　IN 述語」を参照してください。

PURGE CHUNK 文固有の指定規則を次に示します。

- 表副問合せを使用したIN 述語だけが指定できます。
- IN 述語中の左側の値式にはCHUNKID を指定してください。
- NOT を指定したIN 述語は指定できません。

指定形式の例を次に示します。

```
WHERE CHUNKID IN (表副問合せ)
```

限定述語：

限定述語については，「7.20.8　限定述語」を参照してください。

PURGE CHUNK 文固有の指定規則を次に示します。

- 比較演算子には，= だけを指定できます。
- ANY またはSOME だけを指定できます。ALL は指定できません。
- 限定述語中の値式にはCHUNKID を指定してください。

指定形式の例を次に示します。

```
WHERE CHUNKID=ANY(表副問合せ)
WHERE CHUNKID=SOME(表副問合せ)
```

(3)　実行時に必要な権限

PURGE CHUNK 文を実行する場合，次に示すすべての権限が必要になります。

- CONNECT 権限

- チャンク削除対象表に対するTRUNCATE 権限

副問合せを指定した場合，FROM 句に指定したすべての表に対するSELECT 権限が必要になります。

(4) 規則

1. CHUNKID の比較相手となる値式の結果のデータ型は，INTEGER 型またはSMALLINT 型となるようにしてください。
2. 述語に対する論理演算（AND，OR，NOT）は指定できません。

   （例）エラーとなる例

   ```
   WHERE CHUNKID=1 OR CHUNKID=5
   WHERE NOT(CHUNKID=1)
   ```

3. 行値構成子要素にCHUNKID は指定できません。
4. 副問合せ中にCHUNKID は指定できません。
5. 副問合せを指定した場合，CHUNKID の比較相手となる選択式のデータ型はINTEGER 型またはSMALLINT 型である必要があります。

   （例）

   ```
   PURGE CHUNK "SALESLIST"
           WHERE CHUNKID=(
                           SELECT "CHUNK_ID"
                             FROM "MASTER"."STATUS_CHUNKS"
                               WHERE "TABLE_SCHEMA" = 'ADBUSER01'
                                 AND "TABLE_NAME" = 'SALESLIST'
                                 AND "CHUNK_COMMENT" = '2015/01/24追加データ')
   ```

   下線部分が，CHUNKID の比較相手となる選択式になります。
6. 探索条件中にチャンク削除対象表の列を指定することはできません。
7. 探索条件に指定した副問合せのFROM 句に，チャンク削除対象表は指定できません。
8. カレントチャンクのチャンク ID を指定した場合，PURGE CHUNK 文がエラーになります。
9. 存在しないチャンク ID を指定した場合，指定されたチャンク ID を無視して処理を続行します。
10. PURGE CHUNK 文中に指定できる表，導出表，表関数導出表，および集まり導出表の延べ数は，最大 2,048 個になります。SQL 文中に指定されている表，導出表，表関数導出表，および集まり導出表の数え方の規則と例については，「4.4.1　SELECT 文の指定形式および規則」の「(4)　規則」を参照してください。
11. PURGE CHUNK 文中に指定している集合演算がすべてUNION の場合，指定できる集合演算の数は最大 1,023 個になります。ただし，指定した集合演算にEXCEPT またはINTERSECT がある場合は，指定できる集合演算の数は最大 63 個になります。
12. PURGE CHUNK 文中に指定できる外結合（FULL OUTER JOIN）の数は，最大 63 個になります。

13. PURGE CHUNK 文の実行時，DB エリアに対して占有モードで排他が掛かります。そのため，チャンク削除対象表，およびチャンク削除対象表のインデクスが格納されている DB エリア内に格納されているほかの表またはインデクスの操作中に，PURGE CHUNK 文を実行することはできません。

14. PURGE CHUNK 文の処理が正常終了した場合，自動的にコミットされます。そのため，PURGE CHUNK 文の実行後にCOMMIT 文を実行する必要はありません。

15. PURGE CHUNK 文の処理が正常終了した場合，処理対象のチャンク（チャンク ID に指定したチャンク）は削除されます。

16. 次に示す条件が重なった場合，PURGE CHUNK 文がエラーになります。
    - PURGE CHUNK 文を実行するコネクション内に，チャンク削除対象表の検索を行うカーソルがある
    - そのカーソルがオープン中である

17. JDBC ドライバを使用してPURGE CHUNK 文を実行する場合，同じコネクション内にチャンク削除対象表の検索を行うStatement オブジェクトおよびPreparedStatement オブジェクトがあると，PURGE CHUNK 文がエラーになります。

(5) 例題

例題 1

販売履歴表（SALESLIST）のチャンクのうち，チャンク ID が1 のチャンクの全行を削除します。

```
PURGE CHUNK ”SALESLIST” WHERE CHUNKID=1
```

上記は，削除対象のチャンク ID を整数定数で指定している例です。

例題 2

販売履歴表（SALESLIST）のチャンクのうち，？パラメタで指定するチャンク ID のチャンクの全行を削除します。

```
PURGE CHUNK ”SALESLIST” WHERE CHUNKID=?
```

上記は，削除対象のチャンク ID を？パラメタで指定している例です。

例題 3

販売履歴表（SALESLIST）のチャンクのうち，次に示す条件を満たすチャンクの全行を削除します。

- チャンクコメントに「2015/01/24追加データ」が設定されているチャンク

```
PURGE CHUNK ”SALESLIST”
        WHERE CHUNKID=(
                    SELECT ”CHUNK_ID”
                      FROM ”MASTER”.”STATUS_CHUNKS”
                        WHERE ”TABLE_SCHEMA” = ’ADBUSER01’
                          AND ”TABLE_NAME” = ’SALESLIST’
                          AND ”CHUNK_COMMENT” = ’2015/01/24追加データ’)
```

上記は，削除対象のチャンク ID をスカラ副問合せで指定している例です。

重要

上記のPURGE CHUNK 文を実行するには，チャンクコメントに「2015/01/24追加データ」が設定されているチャンクが，1 つだけであることが前提となります。「2015/01/24 追加データ」が設定されているチャンクが複数ある場合は，IN 述語または限定述語を使用してください。

例題 4

販売履歴表（SALESLIST）のチャンクのうち，次に示す条件を満たすチャンクの全行を削除します。

- チャンクコメントに「2015*XXXX*追加データ」が設定されているチャンク

XXXX には，月，日が記述されているとします。

```
PURGE CHUNK "SALESLIST"
        WHERE CHUNKID IN (
                        SELECT "CHUNK_ID"
                          FROM "MASTER"."STATUS_CHUNKS"
                            WHERE "TABLE_SCHEMA" = 'ADBUSER01'
                              AND "TABLE_NAME" = 'SALESLIST'
                              AND "CHUNK_COMMENT" LIKE '2015%追加データ')
```

上記は，削除対象のチャンク ID をIN 述語で指定している例です。

例題 5

販売履歴表（SALESLIST）のチャンクのうち，待機状態のチャンクの全行を削除します。

```
PURGE CHUNK "SALESLIST"
        WHERE CHUNKID=ANY(
                        SELECT "CHUNK_ID"
                          FROM "MASTER"."STATUS_CHUNKS"
                            WHERE "TABLE_SCHEMA" = 'ADBUSER01'
                              AND "TABLE_NAME" = 'SALESLIST'
                              AND "CHUNK_STATUS" = 'Wait')
```

上記は，削除対象のチャンク ID を限定述語で指定している例です。

4.4 SELECT（行の検索）

ここでは，SELECT 文の指定形式および規則について説明します。

4.4.1 SELECT 文の指定形式および規則

表のデータを検索します。

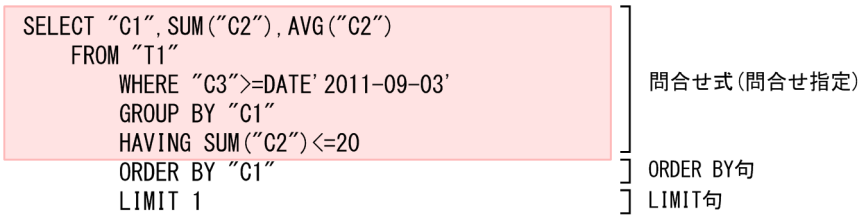
(1) 指定形式

```
SELECT文 ：：＝〔SQLパラレル実行指定〕 問合せ式 〔ORDER BY句〕 〔LIMIT句〕

  SQLパラレル実行指定 ：：＝/*>> SQL PARALLEL EXECUTION <<*/
  ORDER BY句 ：：＝ORDER BY ソート指定リスト
```

SELECT 文は，SQL パラレル実行指定，問合せ式，および句（ORDER BY 句とLIMIT 句）で構成されています。SELECT 文の構成例を次の図に示します。

図 4-1 SELECT 文の構成例



注

上記の例には，SQL パラレル実行指定は記載していません。

なお，SELECT 文を使用した行の取り出し方法については，マニュアル『HADB　AP 開発ガイド』の次に示す個所を参照してください。

- JDBC ドライバを使用する場合：『データを検索する場合（SELECT 文を実行する場合)』の『データの検索方法』
- CLI 関数を使用する場合：『CLI 関数の使い方』の『データを参照する場合』

(2) 指定形式の説明

●*SQL パラレル実行指定*

```
SQLパラレル実行指定 ：：＝/*>> SQL PARALLEL EXECUTION <<*/
```

SQL パラレル実行機能を検索系 SQL に適用する場合に，SQL パラレル実行指定を指定します。SQL パラレル実行機能については，マニュアル『HADB　システム構築・運用ガイド』の『SQL パラレル実行機能』を参照してください。

ただし，SQL パラレル実行指定を指定しても，マニュアル『HADB　AP 開発ガイド』の『SQL パラレル実行機能が適用される条件』に記載されている条件をすべて満たさないと，検索系 SQL に SQL パラレル実行機能は適用されません。

●*問合せ式*

問合せ式を指定します。問合せ式については，「7.1　問合せ式」を参照してください。

問合せ指定を指定するか，または問合せ指定によって導出される表の和集合を求めるために指定します。

●*ORDER BY 句*

ORDER BY句：：＝ORDER BY *ソート指定リスト*

問合せ式の結果を昇順または降順に並べ替える場合に指定します。ORDER BY 句を省略した場合，問合せ式の結果は昇順または降順に並びません。

ソート指定リストには，ソートキーおよびソート結果の並び順を指定します。ソート指定リストについては，「7.25　ソート指定リスト」を参照してください。

留意事項を次に示します。

- ソートキーが文字データの場合，サーバ定義，クライアント定義，またはコネクション属性に指定した文字データの並び替え順序に従って，バイトコード順またはソートコード順に並び替えられます。
- 文字データをソートコード順に並び替える場合は，各文字に割り当てられた ISO/IEC14651:2011 規格のソートコード<S0000>～<S2FFFF>とサブコード<T0000>～<TFFFF>を使用して並び替えられます。ソートコードが割り当てられていない文字同士の並び替えは，バイトコード順に並び替えられます。
- 文字データをソートコード順に並び替える場合は，Unicode（UTF-8）のビットパターンとして不正な文字は，1 バイトの 1 文字として扱われ，いちばん後ろに並べられます。
- ORDER BY 句を指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。

●*LIMIT 句*

問合せ式の結果から取得する行数の最大値を指定します。

LIMIT 句については，「7.9　LIMIT 句」を参照してください。

(3)　実行時に必要な権限

SELECT 文を実行する場合，次に示すすべての権限が必要になります。

- CONNECT 権限

- SELECT 文中のすべての問合せ指定中に指定する表に対するSELECT 権限

(4) 規則

1. SELECT 文中の全表参照に指定できる表，導出表，表関数導出表，および集まり導出表の延べ数は，最大 2,048 個になります。ただし，SQL 文中に次の指定がある場合，次の指定が内部導出表に等価変換されたあとの SQL 文に対して延べ数のチェックが行われます。

   - ビュー表

     CREATE VIEW 文中にビュー表を指定している場合は，CREATE VIEW 文中に指定したビュー表を導出表に等価変換したあとに，延べ数のチェックが行われます。

   - 問合せ名
   - アーカイブマルチチャンク表

   > **メモ**
   >
   > アーカイブマルチチャンク表に関する等価変換については，マニュアル『HADB　AP 開発ガイド』の『アーカイブマルチチャンク表を検索する SQL 文の等価変換』を参照してください。

   SQL 文中に指定した表，導出表，表関数導出表，および集まり導出表の数え方の例を次に示します。

   (例)

```
WITH "Q1" AS (SELECT * FROM "T6","T7")
SELECT * FROM
  "T1",                                                    ...[a]
  "T2" LEFT OUTER JOIN "T3" ON "T2"."C1"="T3"."C1",        ...[b]
  (SELECT * FROM "T4","T5") W1,                            ...[c]
  "Q1",                                                    ...[d]
  TABLE(ADB_CSVREAD(MULTISET['/tmp/data.gz'],'COMPRESSION_FORMAT=GZIP;'))
  AS W2 ("C1" INTEGER),                                    ...[e]
  "V1",                                                    ...[f]
  "V2",                                                    ...[g]
  "T001",                                                  ...[h]
  UNNEST("T1"."C1") "CDT1"                                 ...[i]
```

   [説明]

   a. 表（T1）を指定しているため，表の指定数は 1 となります。

   b. 結合表を指定しているため，表の指定数は，結合表に指定した表（T2 とT3）の合計数の 2 となります。

   c. 導出表を指定しているため，表の指定数は，導出表の数が 1，導出問合せに含まれる表（T4 とT5）が 2 となり，合計 3 となります。

   d. 問合せ名を指定しているため，表の指定数は，問合せ名を等価変換した導出表の数が 1，導出問合せに含まれる表（T6 とT7）が 2 となり，合計 3 となります。

   e. 表関数導出表を指定しているため，表の指定数は 1 となります。

f. ビュー表を指定しているため，表の指定数は，ビュー表（V1）を等価変換した導出表の数が 1，導出問合せに含まれる表（T8 とT9）が 2 となり，合計 3 となります。

g. ビュー表を指定しているため，表の指定数は，ビュー表（V2）を等価変換した導出表の数が 1，導出問合せに含まれるビュー表（V1）を等価変換した導出表の数が 1，導出問合せに含まれる表（T8 とT9）2 となり，合計 4 となります。

h. アーカイブマルチチャンク表を指定しているため，アーカイブマルチチャンク表を等価変換した導出表の数が 1，導出問合せに含まれる表が 4 となり，合計 5 となります。T001 は，導出表に等価変換されるアーカイブマルチチャンク表とします。

i. 集まり導出表（CDT1）を指定しているため，表の指定数は 1 となります。

上記の例の場合，SQL 文中に指定されている表，導出表，表関数導出表，および集まり導出表の延べ数は 23 になります。

なお，V1，V2 は，次に示すCREATE VIEW 文で定義されたビュー表とします。

```
CREATE VIEW ”V1” AS SELECT * FROM ”T8”,”T9”
CREATE VIEW ”V2” AS SELECT * FROM ”V1”
```

2. SELECT 文中に指定している集合演算がすべてUNION の場合，指定できる集合演算の数は最大 1,023 個になります。ただし，指定した集合演算にEXCEPT またはINTERSECT がある場合は，指定できる集合演算の数は最大 63 個になります。

3. SELECT 文中に指定できる外結合（FULL OUTER JOIN）の数は，最大 63 個になります。

4. 問合せ式の結果，導出される列の名前を検索項目列名といいます。問合せ式の結果，導出される列の名前がない場合（列名長が 0 の場合），検索項目列名は次のように決まります。

   EXP*nnnn*_NO_NAME

   （凡例）*nnnn*：0001～4000 の符号なし整数

   （例）

```
SELECT ”C1”,MAX(”C2”),MIN(”C2”)
    FROM ”T1” GROUP BY ”C1”
```

   上記のSELECT 文を実行した場合，検索項目列名はC1，EXP0001_NO_NAME，EXP0002_NO_NAME となります。

5. アーカイブマルチチャンク表を検索する場合，SELECT 文中の探索条件の指定に注意する必要があります。詳細については，マニュアル『HADB　AP 開発ガイド』の『アーカイブマルチチャンク表を検索する際の考慮点』を参照してください。アーカイブマルチチャンク表を検索するSELECT 文を指定する場合は，必ず参照してください。

(5) 例題

例題 1

販売履歴表（SALESLIST）から，2013/9/6 以降に商品コードP002 の商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT ”USERID”,”PUR-CODE”,”PUR-DATE”
    FROM ”SALESLIST”
```

```
        WHERE "PUR-DATE">=DATE'2013-09-06'
        AND "PUR-CODE"='P002'
```

例題 2

社員表（EMPLIST）から，社員の平均年齢（AGE）を部門（SCODE）ごとに求めます。

```
SELECT "SCODE",AVG("AGE")
    FROM "EMPLIST"
        GROUP BY "SCODE"
```

例題 3

社員表（EMPLIST）から，社員の平均年齢（AGE）を部門（SCODE）ごとに求めます。このとき，SQL パラレル実行機能を適用します。

```
/*>> SQL PARALLEL EXECUTION <<*/
SELECT "SCODE",AVG("AGE")
    FROM "EMPLIST"
        GROUP BY "SCODE"
```

SELECT 文の基本的な例題については，「1. SELECT 文の例題集」にまとめて記載しています。そちらも参照してください。

ORDER BY 句を指定したSELECT 文の例については，「7.25.4 例題」の「(1) ORDER BY 句にソート指定リストを指定した例」を参照してください。

LIMIT 句を指定したSELECT 文の例については，「7.9.1 LIMIT 句の指定形式および規則」の「(4) 例題」を参照してください。

4.5 TRUNCATE TABLE（実表の全行削除）

ここでは，TRUNCATE TABLE 文の指定形式および規則について説明します。

4.5.1 TRUNCATE TABLE 文の指定形式および規則

実表内のすべての行を削除します。

(1) 指定形式

TRUNCATE TABLE文 ： ：＝TRUNCATE TABLE *表名*

(2) 指定形式の説明

●*表名*

行を削除する実表の表名を指定します。表名の指定規則については，「6.1.5 名前の修飾」の「(2) 表名の指定形式」を参照してください。

なお，次の表は指定できません。

- ビュー表
- ディクショナリ表
- システム表

メモ

アーカイブマルチチャンク表を指定した場合，アーカイブ状態ではないチャンクに格納されたデータと，アーカイブ状態のチャンクに格納されたデータの両方が削除されます。

(3) 実行時に必要な権限

TRUNCATE TABLE 文を実行する場合，次に示すすべての権限が必要になります。

- CONNECT 権限
- 表に対するTRUNCATE 権限

(4) 規則

1. TRUNCATE TABLE 文の実行時，DB エリアに対して占有モードで排他が掛かります。そのため，処理対象表および処理対象表のインデクスが格納されている DB エリア内に格納されているほかの表またはインデクスの更新中に，TRUNCATE TABLE 文を実行することはできません。

2. TRUNCATE TABLE 文の処理が正常終了した場合，処理完了と同時にコミットされます。そのため，TRUNCATE TABLE 文の実行後にCOMMIT 文を実行する必要はありません。
3. 次のすべての条件を満たしている場合，TRUNCATE TABLE 文がエラーになります。
   - TRUNCATE TABLE 文を実行するコネクション内に，TRUNCATE TABLE 文の実行対象表の検索を行うカーソルがある
   - そのカーソルがオープン中である
4. 次のすべての条件を満たしている場合，TRUNCATE TABLE 文がエラーになります。
   - JDBC ドライバを使用してTRUNCATE TABLE 文を実行している
   - 同一コネクション内に，TRUNCATE TABLE 文の実行対象となる表の検索を行うStatement オブジェクトおよびPreparedStatement オブジェクトがある

(5) 例題

例題

販売履歴表（SALESLIST）内の全行を削除します。

```
TRUNCATE TABLE ”SALESLIST”
```

4.6 UPDATE（行の更新）

ここでは，UPDATE 文の指定形式および規則について説明します。

4.6.1 UPDATE 文の指定形式および規則

行の値を更新します。

(1) 指定形式

■更新対象列名を指定して行を更新する場合

```
UPDATE文 ： ：＝UPDATE 表名 〔〔AS〕 相関名〕
                SET 更新対象列名=更新値 〔,更新対象列名=更新値〕…
                  〔WHERE 探索条件〕

  更新値 ： ： ＝ {値式 | NULL | DEFAULT}
```

■ROW 指定で行単位の更新をする場合

```
UPDATE文 ： ：＝UPDATE 表名 〔〔AS〕 相関名〕
                SET ROW=行更新値
                  〔WHERE 探索条件〕

  行更新値 ： ：＝?パラメタ
```

(2) 指定形式の説明

●*表名*

更新対象表の表名を指定します。表名の指定規則については，「6.1.5 名前の修飾」の「(2) 表名の指定形式」を参照してください。

指定規則を次に示します。

- 読み取り専用ビュー表は指定できません。
- 配列型の列を定義した表は指定できません。

●〔AS〕 *相関名*

更新対象表の相関名を指定します。相関名については，「6.1.5 名前の修飾」の「(4) 表指定の指定形式」を参照してください。相関名の有効範囲については，「6.8 範囲変数」を参照してください。

●*更新対象列名=更新値* 〔,*更新対象列名=更新値*〕…

```
更新値 ： ： ＝ {値式 | NULL | DEFAULT}
```

更新対象の列と，更新値（更新後の値）を指定します。

更新対象列名 は，列指定の形式で指定できます。列指定の指定形式については，「6.1.5 名前の修飾」の「(5) 列指定の指定形式」を参照してください。

*更新値*には次のどれかを指定します。

値式：

更新後の値を値式の形式で指定します。値式については，「7.21　値式」を参照してください。

NULL：

更新後の値をナル値にする場合に指定します。

DEFAULT：

CREATE TABLE 文のDEFAULT 句で指定した列の既定値を，更新後の値とする場合に指定します。列の既定値については，「7.10　DEFAULT 句」を参照してください。

DEFAULT 句で列の既定値を指定していない場合は，列の既定値としてナル値が仮定されます。

●WHERE *探索条件*

更新する行を選択する条件を指定します。WHERE *探索条件*を省略すると，指定した表のすべての行が更新されます。

探索条件については，「7.19　探索条件」を参照してください。

指定規則を次に示します。

- 探索条件中に？パラメタを指定できます。

*表名*に更新可能ビュー表を指定した場合の留意事項を次に示します。

- 更新可能ビュー表の行を更新した場合，基表の行が更新されます。
- ビュー表を定義した際に指定した探索条件と，ここで指定した探索条件の両方を満たす基表の行が更新されます。
- WHERE *探索条件*を省略した場合は，ビュー表を定義した際に指定した探索条件を満たす基表の行が更新されます。

●ROW=*行更新値*

行更新値：：＝*？パラメタ*

行単位でデータを更新する場合に指定します。ROW は FIX 表に対してだけ指定できます。ROW を指定すると，行全体を 1 つのデータとして更新します。

？パラメタに仮定されるデータ型はCHAR 型です。また，データ長は更新対象表の行長になります。また，構造体中に境界調整による空きがないようにしてください。行長の計算方法については，マニュアル『HADB　システム構築・運用ガイド』の『行の種別ごとの格納ページ数の求め方』の計算式*ROWSZ*を参照してください。

なお，？パラメタは 1 つだけ指定できます。

(3)　実行時に必要な権限

UPDATE 文を実行する場合，次に示すすべての権限が必要になります。

- CONNECT 権限
- 行を更新する表に対するUPDATE 権限

- 問合せ式本体に指定する表に対するSELECT 権限

（例）

```
UPDATE "T1"
    SET "C1"='P001'
       WHERE "T1"."C2" IN (SELECT "C2" FROM "T2" WHERE "C3"<=100)
```

上記のUPDATE 文を実行する場合，表 T1 に対するUPDATE 権限と，表 T2 に対するSELECT 権限が必要になります。

(4) 規則

1. UPDATE 文中に指定できる表，導出表，表関数導出表，および集まり導出表の延べ数は，最大 2,048 個になります。SQL 文中に指定されている表，導出表，表関数導出表，および集まり導出表の数え方の規則と例については，「4.4.1 SELECT 文の指定形式および規則」の「(4) 規則」を参照してください。
2. UPDATE 文中に指定している集合演算がすべてUNION の場合，指定できる集合演算の数は最大 1,023 個になります。ただし，指定した集合演算にEXCEPT またはINTERSECT がある場合は，指定できる集合演算の数は最大 63 個になります。
3. UPDATE 文中に指定できる外結合（FULL OUTER JOIN）の数は，最大 63 個になります。
4. 探索条件中および更新値中の副問合せのFROM 句には，更新対象表を指定できません。
5. 更新値のデータ型は，更新対象列のデータ型と変換または代入できるデータ型にしてください。変換または代入できるデータ型については，「6.2.2 変換，代入，比較できるデータ型」を参照してください。
6. 更新値に？パラメタを単独で指定する場合，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。
7. DECIMAL 型，NUMERIC 型，DOUBLE PRECISION 型，またはFLOAT 型のデータで，次のデータ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。
   - INTEGER
   - SMALLINT

   また，DECIMAL 型またはNUMERIC 型のデータで，DECIMAL 型またはNUMERIC 型の列を更新する場合，列の位取りより下位の桁部分が切り捨てられます。

   DOUBLE PRECISION 型またはFLOAT 型のデータで，DECIMAL 型またはNUMERIC型の列を更新する場合，列の位取りより下位の桁部分が丸められます（最近接偶数への丸め）。
8. CHAR 型，VARCHAR 型，BINARY 型，またはVARBINARY 型の列を更新する際，更新値のデータ長が列の定義長より長い場合，表の更新ができません。
9. CHAR 型の列を更新する際，更新値のデータ長が列の定義長より短い場合，データが左詰めに格納されて，余りの部分には半角空白が埋められます。
10. BINARY 型の列を更新する際，更新値のデータ長が列の定義長より短い場合，データが左詰めに格納されて，余りの部分にはX'00' が埋められます。

11. INTEGER 型，SMALLINT 型，DECIMAL 型，NUMERIC 型，DOUBLE PRECISION 型，またはFLOAT 型の列を更新する際，更新値が各データ型の範囲外の場合，表の更新ができません。

12. SET 句には更新対象列名を 4,000 個まで指定できます。

13. 更新対象の行がない場合は，SQLCODE に100 が設定されます。

14. 更新対象列に同じ列名を重複して指定できません。

15. ROW 指定をした場合，SET 句を 2 つ以上指定できません。

16. ディクショナリ表またはシステム表の行は更新できません。

17. カラムストア表に対してUPDATE 文を実行する場合，ローストア表とは異なる運用や設計が必要になります。詳細については，マニュアル『HADB　システム構築・運用ガイド』の『ローストア表とカラムストア表の選択基準』，『シングルチャンク表の再編成が必要かどうかを確認する方法』，および『マルチチャンク表の再編成が必要かどうかを確認する方法』を参照してください。

18. アーカイブされている行は更新できません。アーカイブされている行を更新するUPDATE 文はエラーになります。アーカイブされている行を更新したい場合は，更新したい行が格納されているチャンクのアーカイブ状態をいったん解除してください。そのあとに，UPDATE 文を実行して行を更新してください。

19. アーカイブされていない行は，UPDATE 文で更新できます。ただし，UPDATE 文を実行する際，次の条件をすべて満たす必要があります。

    - 探索条件に，アーカイブレンジ列を指定した条件を指定する
    - アーカイブレンジ列を指定した探索条件に，比較述語，IN 述語，またはBETWEEN 述語だけを指定する
    - アーカイブレンジ列を指定した探索条件に，OR 条件やNOT 条件などを指定しない
    - アーカイブされている行を更新対象にしていない

    上記の条件をすべて満たさないと，UPDATE 文がエラーになります。

> **重要**
>
> アーカイブレンジ列を指定した探索条件に指定できる述語に制限があります。また，OR 条件やNOT 条件以外にも，探索条件に指定するとUPDATE 文がエラーになる条件があります。詳細については，マニュアル『HADB　AP 開発ガイド』の『アーカイブレンジ列の日時情報を使用した検索範囲の絞り込み』を参照してください。

UPDATE 文が実行できる例とできない例の代表的な例を次に示します。なお，例中のARCHIVE-T1 表はアーカイブマルチチャンク表とし，RECORD-DAY 列はアーカイブレンジ列とします。

■UPDATE 文が実行できる例

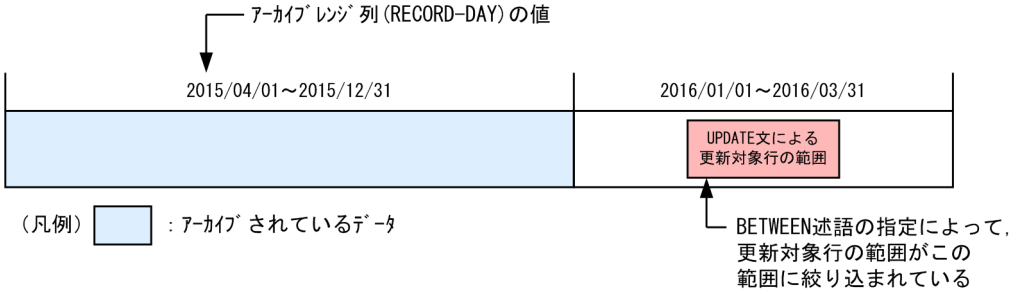
（例）

```
UPDATE "ARCHIVE-T1" SET "NUMBER"=100
    WHERE "RECORD-DAY" BETWEEN DATE'2016/02/01' AND DATE'2016/02/29'
      AND "CODE"='P001'
```

上記の例の場合，次の条件をすべて満たしているため，UPDATE 文を実行できます。

- 探索条件に，アーカイブレンジ列（RECORD-DAY）を指定している

- アーカイブレンジ列を指定した探索条件に，BETWEEN 述語だけが指定されている
- アーカイブレンジ列を指定した探索条件に，OR 条件やNOT 条件などを指定していない
- アーカイブされている行を更新対象にしていない



重要

探索条件に指定するアーカイブレンジ列に対する比較条件は，定数を指定することを推奨します。

（例）推奨する指定例

```
"RECORD-DAY" BETWEEN DATE'2016/01/01' AND DATE'2016/01/10'
"RECORD-DAY" >= DATE'2016/02/10'
```

定数以外の指定は推奨しません。

（例）推奨しない指定例

```
"RECORD-DAY" BETWEEN ? AND ?
"RECORD-DAY" >= CURRENT DATE
```

メモ

更新対象のデータがアーカイブされているかどうかを，HADB サーバは，アーカイブレンジ列を指定した探索条件によって判定しています。その際，アーカイブレンジ列に対する比較条件に定数を指定した場合，判定に掛かる時間を短くできます。一方，定数以外を指定した場合，判定に掛かる時間が非常に長くなるおそれがあります。

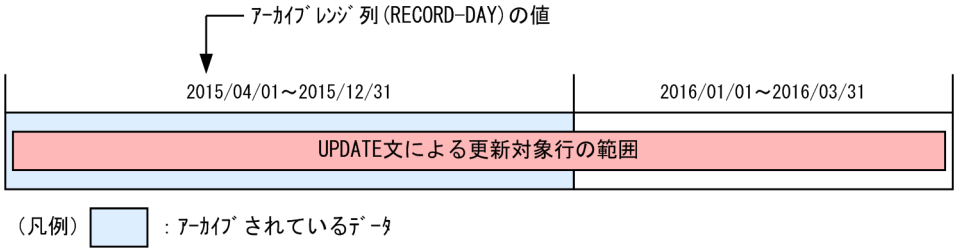
■UPDATE 文が実行できない例

・探索条件にアーカイブレンジ列を指定していない

(例 1)

```
UPDATE "ARCHIVE-T1" SET "NUMBER"=100 _____
```

上記の例の場合，探索条件にアーカイブレンジ列（RECORD-DAY）を指定していないため，UPDATE 文がエラーになります。



（例 2）

```
UPDATE ”ARCHIVE-T1” SET ”NUMBER”=100
    WHERE ”CODE”='P001'
```

上記の例の場合，探索条件にアーカイブレンジ列（RECORD-DAY）を指定していないため，UPDATE 文がエラーになります。アーカイブされていない行を更新する場合でもエラーになります。

・アーカイブレンジ列を指定した探索条件に OR 条件や NOT 条件などを指定している

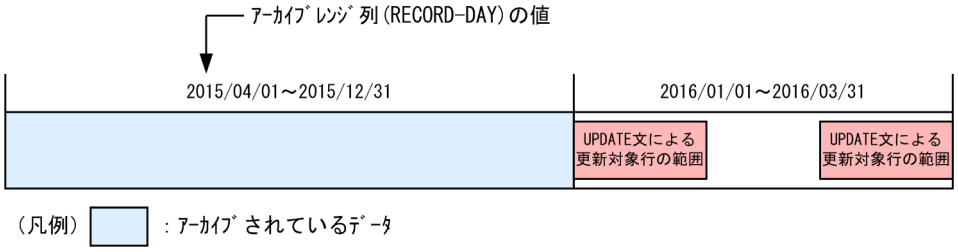
（例）

```
UPDATE ”ARCHIVE-T1” SET ”NUMBER”=100
    WHERE ”RECORD-DAY” BETWEEN DATE'2016-01-01' AND DATE'2016-01-31'
       OR ”RECORD-DAY” BETWEEN DATE'2016-03-01' AND DATE'2016-03-31'
```

上記の例の場合，アーカイブレンジ列を指定した探索条件にOR 条件が指定されているため，UPDATE 文がエラーになります。上記のように，アーカイブされていない行を更新する場合でもエラーになります。

上記の例の場合，次のようにUPDATE 文を 2 回に分けて実行すると，行を更新できます。

```
UPDATE ”ARCHIVE-T1” SET ”NUMBER”=100
    WHERE ”RECORD-DAY” BETWEEN DATE'2016-01-01' AND DATE'2016-01-31'
UPDATE ”ARCHIVE-T1” SET ”NUMBER”=100
    WHERE ”RECORD-DAY” BETWEEN DATE'2016-03-01' AND DATE'2016-03-31'
```

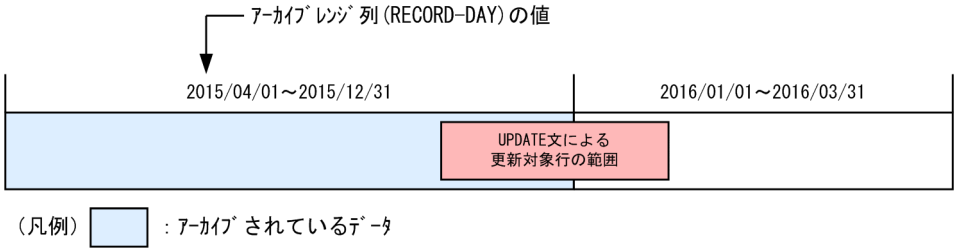


・アーカイブされている行を更新対象にしている

（例）

```
UPDATE ”ARCHIVE-T1” SET ”NUMBER”=100
    WHERE ”RECORD-DAY” BETWEEN DATE'2015/11/01' AND DATE'2016/01/31'
```

上記の例の場合，アーカイブされている行を更新対象にしているため，UPDATE 文がエラーになります。



・単独の列指定でアーカイブレンジ列を指定していない

（例）

```
UPDATE ”ARCHIVE-T1” SET ”NUMBER”=100
    WHERE ”RECORD-DAY” - 10 DAY > DATE’2016/02/01’
```

上記の例の場合，アーカイブレンジ列を使った日時演算を指定しているため，UPDATE 文がエラーになります。

・アーカイブレンジ列の比較条件に日時演算を指定している

（例）

```
UPDATE ”ARCHIVE-T1” SET ”NUMBER”=100
    WHERE ”RECORD-DAY” >= CURRENT_DATE - 1 MONTH
```

上記の例の場合，アーカイブレンジ列の比較条件に日時演算を指定しているため，UPDATE 文がエラーになります。

20. UPDATE 文中にアーカイブマルチチャンク表を指定した場合，ロケーション表およびシステム表（STATUS_CHUNKS）に対するアクセスが発生します。その際，システム表（STATUS_CHUNKS）の排他資源が確保されます。排他制御の詳細については，マニュアル『HADB　システム構築・運用ガイド』の『排他制御』を参照してください。

(5)　例題

例題 1　更新対象列名を指定した行の更新

販売履歴表（SALESLIST）の次に示す条件を満たす行の販売個数（PUR-NUM）を 6 に更新します。

- 顧客 ID（USERID）：U00358
- 商品コード（PUR-CODE）：P003
- 購入日（PUR-DATE）：2011-09-08

```
UPDATE ”SALESLIST”
    SET ”PUR-NUM”=6
    WHERE ”USERID”=’U00358’
      AND ”PUR-CODE”=’P003’
      AND ”PUR-DATE”=DATE’2011-09-08’
```

例題 2　更新対象列名を指定した行の更新（更新値に副問合せを指定した場合）

販売履歴表（SALESLIST）の商品コード（PUR-CODE）列がP003 の商品カラー（PUR-COL）を，商品表（PRODUCTLIST）の商品コードがP003 の商品と同じ商品カラーに変更します。

```
UPDATE "SALESLIST"
    SET "PUR-COL" = (SELECT "PUR-COL" FROM "PRODUCTLIST" WHERE "PUR-CODE"='P003')
    WHERE "PUR-CODE"='P003'
```

例題３　ROW 指定による行の更新

販売履歴表（SALESLIST）の販売情報を更新します（ROW 指定で行全体を更新します）。販売履歴表の列構成は，顧客 ID（USERID），商品コード（PUR-CODE），販売個数（PUR-NUM），購入日（PUR-DATE）です。

```
UPDATE "SALESLIST"
    SET ROW=?
    WHERE "USERID"=?
```

5 制御系 SQL

この章では，制御系 SQL の機能，指定形式，および規則について説明します。

5.1 COMMIT（トランザクションの正常終了）

ここでは，COMMIT 文の指定形式について説明します。

5.1.1 COMMIT 文の指定形式

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て，トランザクションを正常終了します。

(1) 指定形式

adbsql コマンドで SQL 文を実行する際にCOMMIT 文を指定できます。AP 中にはCOMMIT 文を指定できません。

adbsql コマンドでのCOMMIT 文の指定形式を次に示します。

```
COMMIT文 : : =COMMIT
```

(2) 実行時に必要な権限

COMMIT 文を実行する場合，CONNECT 権限が必要になります。

5.2　ROLLBACK（トランザクションの取り消し）

ここでは，ROLLBACK 文の指定形式について説明します。

5.2.1　ROLLBACK 文の指定形式

トランザクションが更新したデータベースの内容を無効にして，トランザクションを取り消します。

(1)　指定形式

adbsql コマンドで SQL 文を実行する際にROLLBACK 文を指定できます。AP 中にはROLLBACK 文を指定できません。

adbsql コマンドでのROLLBACK 文の指定形式を次に示します。

```
ROLLBACK文 : : =ROLLBACK
```

(2)　実行時に必要な権限

ROLLBACK 文を実行する場合，CONNECT 権限が必要になります。

6 基本項目

この章では，SQL の基本項目について説明します。

6.1　SQL の記述規則

ここでは，SQL の記述規則について説明します。

6.1.1　SQL の書き方の規則

(1)　オプションの指定順序

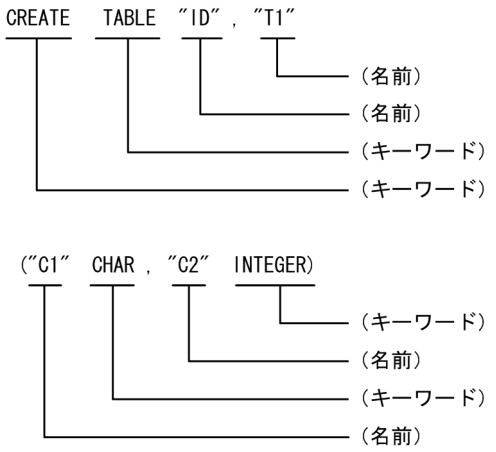
各 SQL の指定形式で説明している順序に従ってオプションを指定してください。

(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るため，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。キーワードおよび名前の例を次の図に示します。

図 6-1　キーワードおよび名前の例



次に示す指定を名前として指定します。

- インデクス識別子
- 相関名
- 問合せ名
- 認可識別子
- スキーマ識別子
- 表識別子

- 列名
- 制約名
- DB エリア名

予約語については，「6.10　予約語」を参照してください。

(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てください。数定数以外の数値を次に示します。

- 未使用領域の比率（表およびインデクス定義の未使用領域の比率）
- 長さ，最大長（文字データまたはバイナリデータの長さおよび最大長）
- 精度（10 進数データの桁数）
- 位取り（10 進数データの小数点以下の桁数）
- 小数秒精度（時刻データ，時刻印データの小数秒の桁数）

(4)　SQL の最大長

1 つの SQL 文は，16,000,000 バイトまで記述できます。

なお，ビュー定義（CREATE VIEW 文）の場合は，64,000 バイトまで記述できます。

6.1.2　分離符号に関する規則

(1)　分離符号とは

SQL を記述する際，キーワードとキーワードの間や，キーワードと名前の間などに，次の分離符号を入れる必要があります。

- 空白
  次に示す文字が空白として使用できます。
  - 半角空白
  - 全角空白
  - CR（Carriage Return）：復帰
  - NL（New Line）：改行
  - TAB
- 注釈

(2) 分離符号の記述位置

次に示す個所に分離符号を記述します。

- 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

分離符号の記述位置の例を次の図に示します。

図 6-2 分離符号の記述位置の例

（凡例）▲：分離符号

(3) 分離符号を記述できない位置

次に示す個所には分離符号を記述できません。

- 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（ただし，先頭に指定した符号の後ろを除く）
- 16 進形式バイナリ定数を表す「X’…’」の，X と次の「 ’ 」の間
- 2 進形式バイナリ定数を表す「B’…’」の，B と次の「 ’ 」の間
- 演算子の中（2 文字から成る比較演算子の中）

分離符号を記述できない位置の例を次の図に示します。

図 6-3　分離符号を記述できない位置の例

（凡例）▲：分離符号

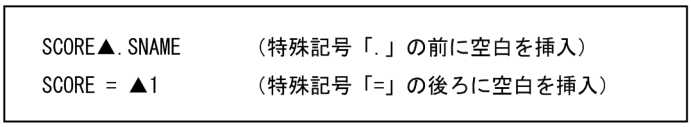
(4)　分離符号を記述してもよい位置

分離符号を記述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)　分離符号を記述できない位置」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」

分離符号を記述してもよい位置の例を次の図に示します。

図 6-4　分離符号を記述してもよい位置の例



（凡例）▲：分離符号

(5)　注釈

SQL 文中の分離符号を記述してもよい位置に注釈を記述できます。注釈の記述例を次に示します。

（例）

```
SELECT "C1","C2" FROM "T1"     /* comment1 */
    ORDER BY "C1" ASC          /* comment2 */
```

下線部分が注釈です。上記のように「/*～*/」の間が注釈と見なされます。

注釈を記述する際の注意事項を次に示します。

- 識別子中または文字列定数中に注釈は記述できません。
- 注釈は入れ子にできません。
  （例）

```
SELECT * FROM  "T1" /* /* comment1 */ comment2 */
```

上記のように注釈を入れ子で指定した場合，構文解析エラーになります。

- 二重引用符（"）またはアポストロフィ（'）で囲まれた「/*～*/」は注釈と見なされません。
  （例）

```
SELECT * FROM "T1" WHERE "C1"='/* comment */'
```

上記の下線部分は，注釈ではなく，文字列定数と見なされます。

- 「/*>>～<<*/」と指定した場合，「/*>>～<<*/」は注釈と見なされません。

  （例）

```
SELECT * FROM "T1" /*>> WITHOUT INDEX <<*/
```

  上記の下線部分は，注釈ではなく，インデクス指定と見なされます。インデクス指定については，「7.14　インデクス指定」を参照してください。

  なお，「/*>>～<<*/」のインデクス指定中に注釈を記述できます。

  （例）

```
SELECT * FROM "T1" /*>> WITHOUT INDEX /* WITH INDEX(INDEXNAME)*/ <<*/
```

  上記の下線部分は，注釈と見なされます。したがって，下記の SQL 文が指定されたと見なされます。

```
SELECT * FROM "T1" /*>> WITHOUT INDEX <<*/
```

- 注釈に「/*>>～<<*/」の形式の指定はできません。例えば，注釈中にインデクス指定は指定できません。

  （例）

```
SELECT * FROM "T1" /* /*>> WITHOUT INDEX <<*/ */
```

  注釈中に上記の下線部分（この例の場合はインデクス指定）を指定すると，この SQL 文は構文解析エラーになります。

6.1.3　SQL 文中に記述できる文字

SQL 文中に記述できる文字を次の表に示します。

表 6-1　SQL 文中に記述できる文字

項番	種別	SQL 文中に記述できる文字
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
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 下線文字（ _ ） カタカナ文字 • 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（ . ） ハイフンまたは負符号（-）

項番	種別	SQL 文中に記述できる文字
		正符号（+） アスタリスク（*） 引用符（ ’ ） 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（!） スラッシュ（/） 疑問符（?） パーセント（%） 垂直棒（|） 左角括弧（[） 右角括弧（] ） TAB NL CR

(1) SQL 文中に記述できる文字コード

SQL 文中に記述できる文字コードは，HADB で使用する文字コードによって異なります。HADB で使用する文字コードと SQL 文中に記述できる文字コードの関係を次の表に示します。

表 6-2 HADB で使用する文字コードと SQL 文中に記述できる文字コードの関係

HADB で使用する文字コード	SQL 文中に記述できる文字コード
Unicode（UTF-8）	JISX0221
Shift-JIS	JISX0201 および JISX0208

(2) 文字の扱い

文字データ中の各文字の構成バイト数は，次の表に示す文字コードの範囲と構成バイト数の関係に従って決定されます。文字データの終端までのバイト数が構成バイト数に満たない場合は，先頭の 1 バイトを 1 バイトで構成する文字として扱います。次の文字は後続するバイトから始まるものと仮定します。

表 6-3　文字コードの範囲と構成バイト数の関係

HADB で使用する文字コード	先頭 1 バイトの範囲	2 バイト目または 2 バイト目以降の範囲	構成バイト数（バイト）
Unicode（UTF-8）	0x00～0x7F	－	1
	0xC0～0xDF	×	2
	0xE0～0xEF	×	3
	0xF0～0xF7	×	4
	0xF8～0xFB	×	5
	0xFC～0xFD	×	6
	上記以外	－	1
Shift-JIS	0x00～0x7F	－	1
	0x81～0x9F	0x40～0x7E または0x80～0xFC	2
		上記以外	1
	0xA1～0xDF	－	1
	0xE0～0xFC	0x40～0x7E または0x80～0xFC	2
		上記以外	1
	上記以外	－	1

（凡例）

×：参照しません。

－：対象外です。

6.1.4　名前の指定

(1)　名前とは

次に示す指定は名前として指定します。

- インデクス識別子
- 相関名
- 問合せ名
- 認可識別子
- スキーマ識別子

- 表識別子
- 列名
- 制約名
- DB エリア名

名前を指定する場合，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います。

> **メモ**
>
> 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 の機能拡張に伴って，予約語が追加される可能性があるため，あらかじめ名前を二重引用符で囲んでおくと，あとから追加された予約語と重複する問題が発生しません。

なお，名前は**識別子**として指定します。識別子には，二重引用符（”）で囲まない指定形式の**通常識別子**と，二重引用符（”）で囲む指定形式の**区切り識別子**があります。通常識別子と区切り識別子の指定例を次に示します。

- 表識別子を通常識別子の形式で指定する場合
  （例）table01
- 表識別子を区切り識別子の形式で指定する場合
  （例）”table01”

(2) 名前に使用できる文字の規則

- 名前の先頭の文字は，半角英大文字，半角英小文字，半角カタカナ，または全角文字のどれかにしてください。ただし，最初の問合せ指定中（SELECT 文のWITH 句中の問合せ指定を除く）の次の個所では，名前の先頭の文字に半角括弧（左括弧または右括弧）を指定できます。
  - 「*選択式* AS *列名*」の列名
  - ORDER BY 句中の列名（副問合せ中に指定したORDER BY 句を除く）

  なお，列名中に半角括弧がある場合は，列名を二重引用符（”）で囲む必要があります（区切り識別子として指定する必要があります）。
- 名前に使用できる文字，および名前の長さの上限を次の表に示します。

表 6-4　名前に使用できる文字，および名前の長さの上限

項番	名前の種類	長さの上限（バイト）	半角文字の使用の可否※1						全角文字の使用の可否※1，※4
			英大文字，数字	英小文字※2	カタカナ	下線（_）	空白※3	ハイフン（-）	
1	インデクス識別子	100	○	○	○	○	△	△	○
2	相関名	100	○	○	○	○	△	△	○
3	問合せ名	100	○	○	○	○	△	△	○
4	認可識別子※5	100	○	○	×	×	×	×	×
5	スキーマ識別子	100	○	○	×	×	×	×	×
6	表識別子	100	○	○	○	○	△	△	○
7	列名	100	○	○	○	○	△	△	○
8	制約名	100	○	○	○	○	△	△	○
9	DB エリア名	30	○※6	○	×	○	×	△	×

（凡例）

○：使用できます。

△：名前を区切り識別子の形式で指定する場合に使用できます。名前を通常識別子の形式で指定する場合は使用できません。

×：使用できません。

注※1

名前に使用する文字は，半角文字と全角文字を混在できます。

注※2

名前を通常識別子の形式で指定する場合，半角英小文字は半角英大文字として扱われます。

名前を区切り識別子の形式で指定する場合，半角英小文字と半角英大文字は区別されます。

注※3

名前を区切り識別子の形式で指定する場合，名前の最後の文字に半角空白は使用できません。

注※4

全角空白は使用できません。

注※5

ALL，HADB，MASTER，およびPUBLIC は認可識別子として指定できません。

注※6

#，@，¥は使用できません。

(3) 名前がSQL文の予約語と重複したときの対応

名前がSQL文の予約語と重複する場合は，予約語と重複する名前を二重引用符（”）で囲んで，区切り識別子の形式で指定してください。

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ため注意してください。

6.1.5 名前の修飾

名前の修飾は，ピリオド（.）によってスキーマ名，表識別子などを連結して，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) スキーマ名の指定形式

形式

```
スキーマ名::=スキーマ識別子
```

スキーマ名には，スキーマ識別子を指定します。

SQL中に指定する表またはインデクスの所有者が自分（HADBサーバに接続中の認可識別子のHADBユーザ）の場合は，自分の認可識別子を指定します。

SQL中でスキーマ名を省略した場合，HADBサーバに接続するときに指定した認可識別子がスキーマ名として仮定されます。

なお，ディクショナリ表またはシステム表を検索する場合は，スキーマ名にMASTERを指定してください。

(2) 表名の指定形式

形式

```
表名::=〔スキーマ名.〕表識別子
```

表名は，表識別子をスキーマ名で修飾した形式で指定します。表識別子には，実表またはビュー表の名前を指定します。

SQL中でスキーマ名を省略した場合，HADBサーバに接続するときに指定した認可識別子がスキーマ名として仮定されます。

(3)　インデクス名の指定形式

形式

インデクス名：：＝〔*スキーマ名*.〕*インデクス識別子*

インデクス名は，インデクス識別子をスキーマ名で修飾した形式で指定します。

SQL 中でスキーマ名を省略した場合，HADB サーバに接続するときに指定した認可識別子がスキーマ名として仮定されます。

(4)　表指定の指定形式

表指定は，1 つの SQL 文中に 2 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，表名，問合せ名，または相関名を指定します。

問合せ名については，「7.1.1　問合せ式の指定形式および規則」の「(2)　指定形式の説明」の「(a)　WITH 句」を参照してください。

相関名とは，次に示す場合に指定するそれらの表の代替名として使用します。

- 同じ表同士を結合する場合
- 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

相関名を指定することで，1 つの表をあたかも 2 つの異なる表として扱うことができます。

形式

表指定：：＝｛*表名* | *問合せ名* | *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"
```

下線部分が表指定での修飾例です。

(5)　列指定の指定形式

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，指定する表名および問合せ名の有効範囲の中でなければ，列名をその表名および問合せ名で修飾できません。表名および問合せ名の有効範囲については，「6.8　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が有効範囲に含まれている表または導出される表に存在しなければなりません。導出される表の列名の規則については，「6.9　導出列名」を参照してください。

列名は構文上修飾できる場合と修飾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，列名を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，次に示す場合は，列名を表指定で修飾しなければなりません。

- 1 つのFROM 句に複数の表を指定した検索（2 つ以上の表の結合）をするときに，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わかりません）。

6.2 データ型

ここでは，HADB でサポートしているデータ型について説明します。

6.2.1 データ型の種類

データ型の種類を次の表に示します。

表 6-5 データ型の種類

項番	分類	データ型	データ型コード※1		データ格納長（単位：バイト）	データ形式
			10 進	16 進		
1	数データ	INTEGER	241	F1	8	整数（8 バイト）
2		SMALLINT	245	F5	4	整数（4 バイト）
3		DECIMAL(*m*,*n*)	229	E5	• 1≦*m*≦4 の場合：2 • 5≦*m*≦8 の場合：4 • 9≦*m*≦16 の場合：8 • 17≦*m*≦38 の場合：16	固定小数点数
4		NUMERIC(*m*,*n*)				
5		DOUBLE PRECISION	225	E1	8	倍精度浮動小数点数
6		FLOAT				
7	文字データ	CHARACTER(*n*)	197	C5	*n*	固定長文字列
8		VARCHAR(*n*)	193	C1	*n*＋2	可変長文字列
9	日時データ	DATE	113	71	4	年，月，日の情報を持つ日付のデータ型
10		TIME(*p*)	121	79	3 ＋ ↑*p*÷2↑	時，分，秒の情報を持つ時刻のデータ型
11		TIMESTAMP(*p*)	125	7D	7 ＋ ↑*p*÷2↑	年，月，日，時，分，秒の情報を持つ時刻印のデータ型
12	バイナリデータ	BINARY(*n*)	149	95	*n*	固定長バイナリデータ
13		VARBINARY(*n*)	145	91	*n* ＋ 2	可変長バイナリデータ
14	配列データ	ARRAY	なし	なし	要素データ型に従う	順序付けられた一連のデータを要素とする一次元の配列データ
15	行データ	ROW	69	45	行長※2	行インタフェースで使用するデータ型

注※1

検索結果列のデータ型を表すコードです。

データ型コードは，CLI 関数使用時にa_rdb_SQLDataType_t 構造体に格納されます。

注※2

各列のデータ格納長の合計が行長になります。

(1) 数データ

■INTEGER

- 値の範囲が-9,223,372,036,854,775,808～9,223,372,036,854,775,807 の整数を扱うデータ型です。
- データ型を指定する際の記述形式を次に示します。

  INTまたはINTEGER
- データは，8 バイトの 2 進形式です。
- 定数は，100，200 という形式で記述します。定数については，「6.3　定数」を参照してください。

■SMALLINT

- 値の範囲が-2,147,483,648～2,147,483,647 の整数を扱うデータ型です。
- データ型を指定する際の記述形式を次に示します。

  SMALLINT
- データは，4 バイトの 2 進形式です。
- SMALLINT 型を使用すると，INTEGER 型を使用するよりも，データベースの容量を少なくできます。

■DECIMAL，NUMERIC

- 固定小数点数を扱うデータ型です。
- データ型を指定する際の記述形式を次に示します。
  - DECIMAL 型の場合

    DEC〔(*m*〔,*n*〕)〕またはDECIMAL〔(*m*〔,*n*〕)〕
  - NUMERIC 型の場合

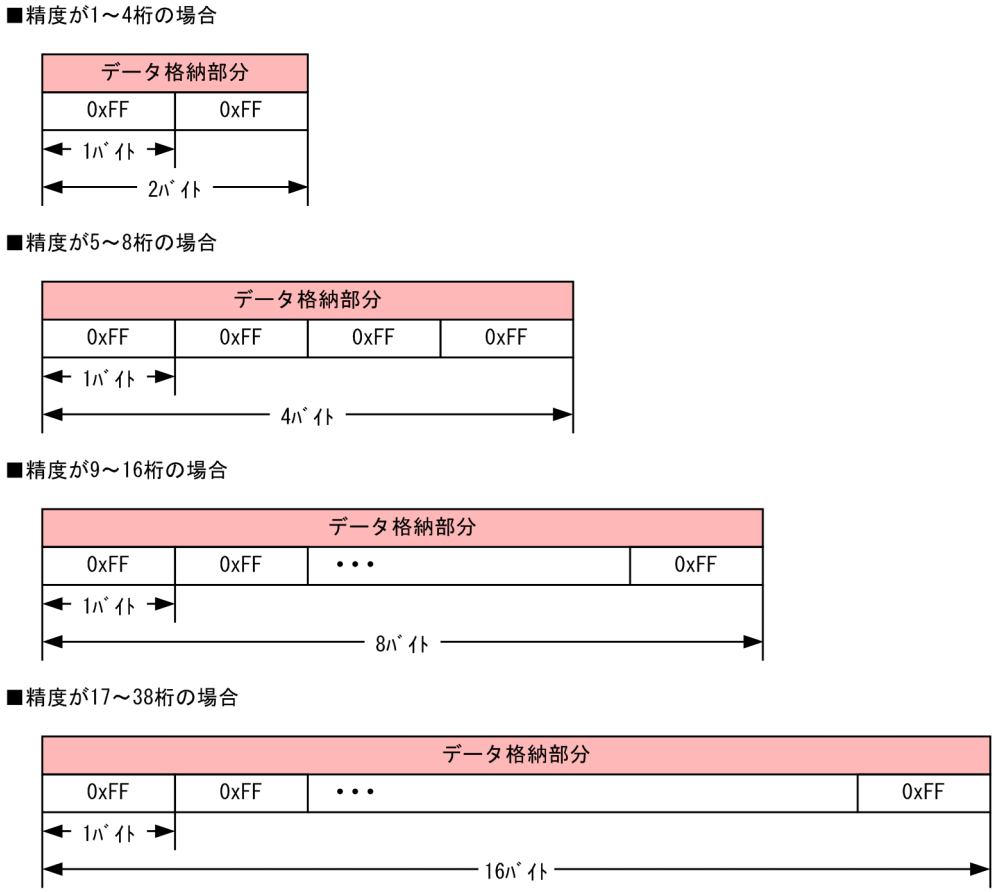
    NUMERIC〔(*m*〔,*n*〕)〕

  > **重要**
  >
  > データ型にNUMERIC 型が指定された場合，HADB はデータ型にDECIMAL 型が指定されたと見なします。

- 精度（全体の桁数）を*m*，位取り（小数部の桁数）を*n* で指定します。
- *m*，*n* は正整数で，1≦*m*≦38，0≦*n*≦38，*n*≦*m* です。

- *m* を省略すると38 が仮定され，*n* を省略すると0 が仮定されます。
- データは 2 進形式で，位取りに合わせた値が格納されます。
- 負の値は 2 の補数で表します。
- データは，次の図に示すように精度によって 2～16 バイトの整数データとして格納されます。

図 6-5　DECIMAL 型のデータ形式



- 定数は，123.4，12.345 という形式で記述します。定数については，「6.3　定数」を参照してください。

■DOUBLE PRECISION，FLOAT

- 値の範囲は，約-1.7×10^{308}～-2.3×10^{-308}，0，および約2.3×10^{-308}～1.7×10^{308}の倍精度浮動小数点数を扱うデータ型です。

  なお，値の範囲は，ハードウェア表現に依存します。
- データ型を指定する際の記述形式を次に示します。
  - DOUBLE PRECISION 型の場合

    DOUBLEまたはDOUBLE PRECISION
  - FLOAT 型の場合

    FLOAT

> **重要**
>
> データ型にFLOAT 型が指定された場合，HADB はデータ型にDOUBLE PRECISION 型が指定されたと見なします。

- データは，8 バイトの浮動小数点数データです。
- 定数は，1.0e2，-3.4E-1 のように，仮数を整数定数または 10 進定数で表し，指数を 3 桁以下の整数で記述します。定数については，「6.3　定数」を参照してください。
- NaN（非数），および無限大となる値を扱うことはできません。
- -0 は＋0 に変換されます。
- 非正規化数は＋0 に変換されます。
- 浮動小数点データの丸めが発生する場合は，最近接偶数への丸めが適用されます。

(2)　文字データ

■CHARACTER

- 固定長文字列を扱うデータ型です。
- データ型を指定する際の記述形式を次に示します。

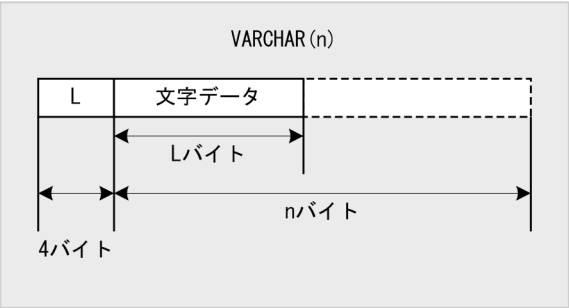
  CHAR，CHAR(*n*)，CHARACTER，またはCHARACTER(*n*)
- 文字列の長さ（バイト数）を*n* で指定します。*n* は 1～32,000 の整数で，*n* を省略すると1 が仮定されます。
- 定数は，'char'，'Ｃｈａｒ'，'charＣｈａｒ'という形式で記述します。定数については，「6.3 定数」を参照してください。
- 半角文字，全角文字の両方を扱うことができます。
- 文字データの比較を行う場合，文字コードの大小が比較するデータの大小となります。

■VARCHAR

- 可変長文字列を扱うデータ型です。
- データ型を指定する際の記述形式を次に示します。

  VARCHAR(*n*)
- 文字列の最大長（バイト数）を*n* で指定します。*n* は 1～64,000 の整数でなければなりません。*n* は省略できません。
- VARCHAR 型のデータ形式を次の図に示します。

図 6-6　VARCHAR 型のデータ形式



文字データの長さ（*L*）を 4 バイトで表します。

- 半角文字，全角文字の両方を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。
- 次に示す個所に，データ長が 32,000 バイトを超えるVARCHAR 型は指定できません。
  - ALTER TABLE 文の列定義に指定するデータ型
  - CREATE TABLE 文の列定義に指定するデータ型
  - 表関数列リストに指定するデータ型
  - スカラ関数CAST に指定する変換後のデータ型
  - スカラ関数CONVERT に指定する変換後のデータ型

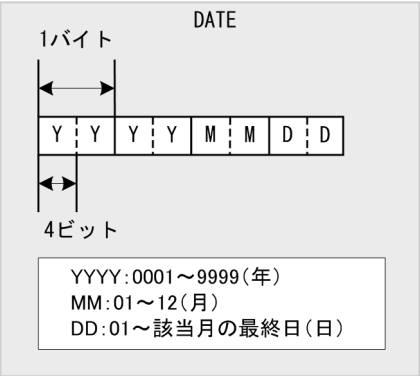
(3)　日時データ

■DATE

- 年，月，日の情報を持つ日付のデータ型です。
- データ型を指定する際の記述形式を次に示します。

  DATE
- 値の範囲が，0001 年 1 月 1 日～9999 年 12 月 31 日の日付を扱います。
- データ長は 4 バイトになります。入力するデータ長は，この長さである必要があります。
- DATE 型のデータ形式を次の図に示します。

図 6-7　DATE 型のデータ形式



- 定数は，DATE'2012-03-30'またはDATE'2012/03/30'という形式で記述します。定数については，「6.3　定数」を参照してください。

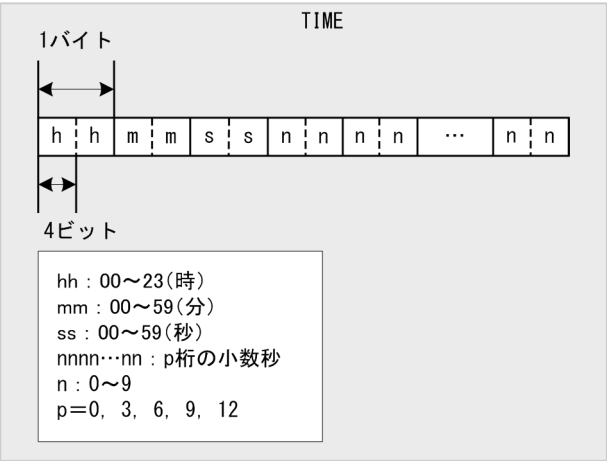
■TIME

- 時，分，秒の情報を持つ時刻のデータ型です。
- データ型を指定する際の記述形式を次に示します。

  TIME(p)，またはTIME

  小数秒精度（小数秒の桁数）をpで指定します。pに指定できる値は，0，3，6，9，または12です。TIMEと指定した場合は，pに0が仮定されます。
- 値の範囲が，0時0分0.000000000000秒～23時59分59.999999999999秒の時刻を扱います。
- データ長は，3＋↑p÷2↑バイトになります。入力するデータ長は，この長さである必要があります。
- TIME型のデータ形式を次の図に示します。

図 6-8　TIME 型のデータ形式



4ビットで1桁を表します。小数秒精度が奇数の場合は，末尾の4ビットに0が格納されます。

- 定数は，TIME'11:03:58.123456'という形式で記述します。定数については，「6.3　定数」を参照してください。

■TIMESTAMP

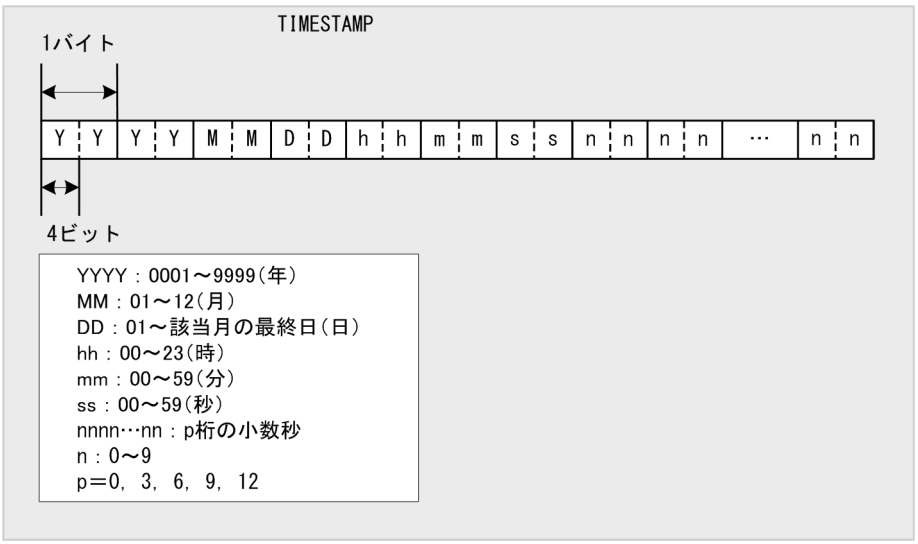
- 年，月，日，時，分，秒の情報を持つ時刻印のデータ型です。

- データ型を指定する際の記述形式を次に示します。

  TIMESTAMP(*p*)，またはTIMESTAMP

  小数秒精度（小数秒の桁数）を*p* で指定します。*p* に指定できる値は，0，3，6，9，または12 です。TIMESTAMP と指定した場合は，*p* に0 が仮定されます。
- 値の範囲が，0001 年 1 月 1 日 0 時 0 分 0.000000000000 秒～9999 年 12 月 31 日 23 時 59 分 59.999999999999 秒の時刻印を扱います。
- データ長は，7 ＋ ↑*p*÷2↑バイトになります。入力するデータ長は，この長さである必要があります。
- TIMESTAMP 型のデータ形式を次の図に示します。

  図 6-9　TIMESTAMP 型のデータ形式



  4 ビットで 1 桁を表します。小数秒精度が奇数の場合は，末尾の 4 ビットに 0 が格納されます。
- 定数は，TIMESTAMP'2012-03-30 11:03:58.123456'，またはTIMESTAMP'2012/03/30 11:03:58.123456'という形式で記述します。定数については，「6.3　定数」を参照してください。

(4)　バイナリデータ

■BINARY

- 固定長バイナリデータを扱うデータ型です。
- データ型を指定する際の記述形式を次に示します。

  BINARY(*n*)，またはBINARY
- バイナリデータの長さ（バイト数）を*n* で指定します。*n* は 1～32,000 の整数で，*n* を省略すると 1 が仮定されます。
- 定数は，X'0A38ef92'という形式で記述します。定数については，「6.3　定数」を参照してください。
- BINARY 型のデータ形式を次の図に示します。

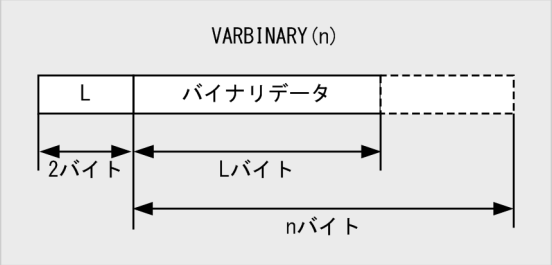
図 6-10　BINARY 型のデータ形式

■VARBINARY

- 可変長バイナリデータを扱うデータ型です。
- データ型を指定する際の記述形式を次に示します。

  VARBINARY(*n*)
- バイナリデータの最大長（バイト数）を *n* で指定します。*n* は 1～32,000 の整数でなければなりません。*n* は省略できません。
- 定数は，X'0A38ef92' という形式で記述します。定数については，「6.3　定数」を参照してください。
- バイナリデータの長さが 0 バイトのデータを扱うこともできます。
- VARBINARY 型のデータ形式を次の図に示します。

図 6-11　VARBINARY 型のデータ形式



バイナリデータの長さ（*L*）を 2 バイトで表します。

(5)　配列データ

■ARRAY（配列型）

- 配列型は，順序付けられた一連のデータを要素とする一次元の配列データを扱うデータ型です。
- データ型を指定する際の記述形式を次に示します。

  要素データ型 ARRAY[*最大要素数*]

  要素データ型：

  配列要素のデータ型を指定します。

  要素データ型には，数データ，文字データ，日時データ，またはバイナリデータのどれかのデータ型を指定してください。要素データ型の記述形式は，各データ型の記述形式の規則に従います。

  最大要素数：

  配列要素の最大数を指定します。

最大要素数には，2～30,000 の符号なし整数定数を指定してください。

＜指定例＞

- 要素データ型がCHAR(5)で，最大要素数が 20 の場合

```
CHAR(5) ARRAY[20]
```

- 要素データ型がINTEGER で，最大要素数が 5 の場合

```
INTEGER ARRAY[5]
```

- 配列データの各要素を，**配列要素**といいます。各配列要素には，**要素番号**が割り当てられます。先頭の配列要素の要素番号を 1 とし，順に 2，3，…となる符号なし整数定数の要素番号が各配列要素に割り当てられます。
- 配列要素数が 0 の配列データを，**空の配列データ**といいます。
- 配列型の列を定義する際の考慮点については，マニュアル『HADB　システム構築・運用ガイド』の『配列型の列の定義【カラムストア表】』を参照してください。

6.2.2　変換，代入，比較できるデータ型

(1)　比較できるデータ型

比較できるデータ型の組み合わせを次の表に示します。

表 6-6　比較できるデータ型の組み合わせ

データ型		データ型の比較可否					
		数データ	文字データ	日時データ			バイナリデータ
				DATE	TIME	TIMESTAMP	
数データ		○	×	×	×	×	×
文字データ		×	○	○	○	○	×
日時データ	DATE	×	○	○	×	○	×
	TIME			×	○	×	×
	TIMESTAMP			○	×	○	×
バイナリデータ		×	×	×	×	×	○

(凡例)

○：比較できます。

×：比較できません。

■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，半角空白を補ってから比較されます。

■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。範囲の広さを次に示します。

DOUBLE PRECISION またはFLOAT ＞DECIMAL またはNUMERIC ＞INTEGER ＞SMALLINT

■日時データと文字データの比較

文字データが既定の入力表現で記述された定数の場合に限り，日時データと文字データを比較できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。

- 日付データと日付データの既定の入力表現で記述された文字列の比較ができます。日付データの既定の入力表現を，文字データから日付データに変換します。その後，日付データ同士の比較を行います。
- 日付データと時刻印データの既定の入力表現で記述された文字列の比較ができます。日付データに時刻 0 時 0 分 0 秒を補って時刻印データに変換します。また，時刻印データの既定の入力表現を，文字データから時刻印データに変換します。その後，時刻印データ同士の比較を行います。
- 時刻データと時刻データの既定の入力表現で記述された文字列の比較ができます。時刻データの既定の入力表現を，文字データから時刻データに変換します。その後，時刻データ同士の比較を行います。
- 時刻印データと時刻印データの既定の入力表現で記述された文字列の比較ができます。時刻印データの既定の入力表現を，文字データから時刻印データに変換します。その後，時刻印データ同士の比較を行います。
- 時刻印データと日付データの既定の入力表現で記述された文字列の比較ができます。日付データの既定の入力表現に時刻 0 時 0 分 0 秒を補って時刻印データに変換します。その後，時刻印データ同士の比較を行います。

ただし，副問合せの選択式と，その選択式の比較相手となる値式は，比較することはできません。

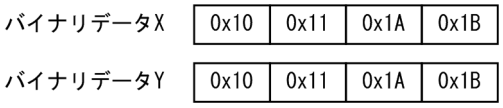
■日時データの比較

- 日付データと時刻印データを比較する場合は，日付データを時刻印データに変換します。その際，時刻 0 時 0 分 0 秒が補われます。
- 小数秒の桁数が異なる場合は，精度の高い方にそろえて，拡張した小数秒部分に 0 が補われます。

■バイナリデータの比較

- 比較するデータのデータ長が同じ場合は，すべてのバイト値が一致したときに，データが等しいと見なされます。

  （例）



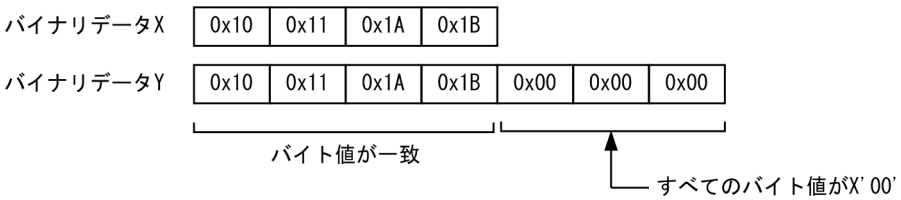
上記の場合，「バイナリデータ X ＝バイナリデータ Y」となります。

- 比較するデータのデータ長が異なる場合は，次の 2 つの条件を満たす場合に，データが等しいと見なされます。

  ・データ長が短い方のデータに合わせて先頭バイトから比較した結果，すべてのバイト値が一致している

  ・データ長が長い部分のバイト値が，すべてX'00'である

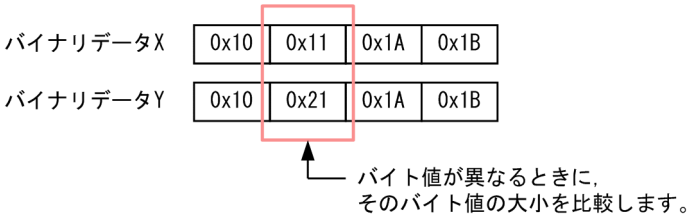
  （例）



上記の場合，「バイナリデータ X ＝バイナリデータ Y」となります。

- データの先頭バイトから順に比較し，バイト値が異なるときにそのバイト値の大小を比較します。その結果によって，データの大小を決定します。

  （例）

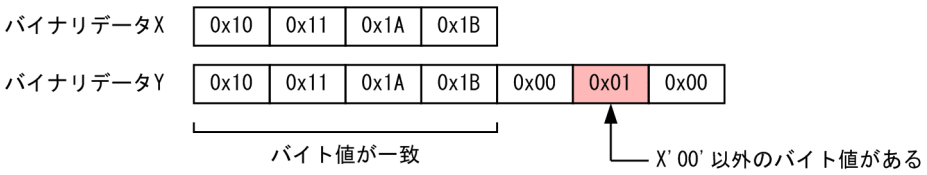


上記の場合，「バイナリデータ X ＜バイナリデータ Y」となります。

- 比較するデータのデータ長が異なる場合で，かつ先頭バイトから短い方のデータ長分のバイト値が一致するときは，次のようにデータの大小を決定します。

  データ長が短いデータを X とし，データ長が長いデータを Y とします。このとき，Y のデータの長い部分に，X'00'以外のバイト値が 1 つ以上ある場合，X ＜ Y となります。

  （例）



上記の場合，「バイナリデータ X ＜バイナリデータ Y」となります。

(2)　格納代入できるデータ型

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，？パラメタ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。？パラメタについては，「6.6　変数（？パラメタ）」を参照してください。

表 6-7　格納代入できるデータ型の組み合わせ

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	日時データ			バイナリデータ	行データ
				DATE	TIME	TIMESTAMP		
数データ		○	×	×	×	×	×	×
文字データ		×	○	○	○	○	×	○※
日時データ	DATE	×	×	○	×	○	×	×
	TIME			×	○	×	×	
	TIMESTAMP			○	×	○	×	
バイナリデータ		×	×	×	×	×	○	×

（凡例）

○：格納代入できます。

×：格納代入できません。

注※

CHAR 型から行データ（ROW）への代入ができます。

■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。
- 代入先がCHAR 型で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します。

■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 型またはSMALLINT 型で，代入元がDECIMAL 型，NUMERIC 型，DOUBLE PRECISION 型，またはFLOAT 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 型またはNUMERIC 型の場合，代入先の位取りより下位の桁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 を補って格納します。
- 代入元がDOUBLE PRECISION 型またはFLOAT 型で，代入先がDECIMAL 型またはNUMERIC 型の場合，代入先の位取りより下位の桁部分は，丸められます（最近接偶数への丸め）。代入先の位取りより小さい場合は，小数部分に 0 を補って格納します。

■日時データへの文字データの格納代入

文字データが既定の入力表現で記述された定数の場合に限り，文字データから日時データへの格納代入ができます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。

ただし，INSERT 文の挿入列が日時データの場合，問合せ指定の選択式に既定の入力表現で記述した定数を指定しても格納代入は行えません。

- 日付データに対して，日付データの既定の入力表現で記述した文字列を格納代入できます。日付データの既定の入力表現の文字データを日付データに変換し，日付データとして格納代入します。
- 日付データに対して，時刻印データの既定の入力表現で記述した文字列を格納代入できます。このとき，時刻印データの既定の入力表現の文字データを時刻印データに変換し，時刻印データの日付部分だけを格納代入します。
- 時刻データに対して，時刻データの既定の入力表現で記述した文字列を格納代入できます。時刻データの既定の文字列表現の文字データを時刻データに変換し，時刻データとして格納代入します。
- 時刻印データに対して，時刻印データの既定の入力表現で記述した文字列を格納代入できます。時刻印データの既定の入力表現の文字データを時刻印データに変換し，時刻印データとして格納代入します。
- 時刻印データに対して，日付データの既定の入力表現で記述した文字列を格納代入できます。このとき，日付データの既定の文字列表現に時刻 0 時 0 分 0 秒を補って時刻印データに変換します。その後，時刻印データとして格納代入します。

■日時データの格納代入

- 日付データを時刻印データに格納代入する場合，日付データに時刻 0 時 0 分 0 秒を補います。その後，時刻印データに変換して格納代入します。
- 時刻印データを日付データに格納代入する場合，時刻印データの日付部分だけを格納代入します。
- 「代入元の小数秒の桁数＞代入先の小数秒の桁数」の場合，代入できない部分は切り捨てます。
- 「代入元の小数秒の桁数＜代入先の小数秒の桁数」の場合，足りない部分に 0 を補って格納代入します。

■バイナリデータの格納代入

- 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。
- 代入先がBINARY 型で，代入元のデータ長が代入先のデータ長よりも短い場合，末尾にX'00'を補って格納します。

■行データの格納代入

代入先の行データが仮定するデータ長（挿入対象表または更新対象表の行長）と，代入元のデータ長を一致させてください。

(3)　検索代入できるデータ型

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。

選択式にROW（行データ）を指定している場合，検索結果を受け取る代入先のデータ型にCHAR 型の変数を指定できます。

(4) 表関数導出表の列への格納代入（ADB_CSVREAD 関数の場合）

ここでは，CSV ファイル中のフィールドデータを表関数導出表の列に格納代入する際の規則について説明します。また，フィールドデータの記述形式の規則についても説明します。

なお，ここでいう表関数導出表は，ADB_CSVREAD 関数によって導出される表関数導出表を意味しています。

> **メモ**
>
> **表関数導出表**とは，ADB_AUDITREAD 関数またはADB_CSVREAD 関数によって導出された表形式のデータ集合のことです。ADB_AUDITREAD 関数については，「7.15.2 ADB_AUDITREAD 関数」を参照してください。ADB_CSVREAD 関数については，「7.15.3 ADB_CSVREAD 関数」を参照してください。

表関数導出表の列のデータ型とフィールドデータの記述形式に互換性がある必要があります。表関数導出表の列のデータ型とフィールドデータの記述形式の関係を次の表に示します。

表 6-8 表関数導出表の列のデータ型とフィールドデータの記述形式の関係

項番	表関数導出表の列のデータ型		フィールドデータの記述形式			
			形式	記述例	注意事項	ナル値が格納代入される記述例
1	数データ	INTEGER	〔 {+|-} 〕*a…a* +，-：符号 *a*..*a*：数値（*a* は0～9）	• 100 • -123 • 000 • 0657	• 符号と数値を合わせて 20 文字まで記述できます。 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	• …,*,… • …,”*”,… • …,,… • …,””,… • …,””””,… ただし，囲み文字指定オプションにNONE を指定した場合，囲み文字を使用した例は指定できません。
2		SMALLINT	〔 {+|-} 〕*a…a* +，-：符号 *a*..*a*：数値（*a* は0～9）	• 100 • -0123 • 0 • +0657	• 符号と数値を合わせて 11 文字まで記述できます。 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空	項番 1 と同じ。

項番	表関数導出表の列のデータ型		フィールドデータの記述形式			
			形式	記述例	注意事項	ナル値が格納代入される記述例
					白またはタブを挿入できます。※1	
3		DECIMAL または NUMERIC	〔{+|-}〕 {*a...a*〔.〔*b...b*〕〕|.*b...b*} +，-：符号 *a...a*：整数部（*a* は0～9） *b...b*：小数部（*b* は0～9）※2	• 100 • -123.00 • △.00 • 012. • -1.56 • +.560	• 整数部と小数部を合わせて 38 文字（格納先の列の精度と位取りが一致している場合は，整数部（0）を省略しないときだけ 39 文字）まで記述できます。 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	項番 1 と同じ。
4		DOUBLE PRECISION または FLOAT	〔{+|-}〕 {*a...a*〔.〔*b...b*〕〕|.*b...b*}〔{E|e}〔〔{+|-}〕*c...c*〕〕 +，-：符号 *a...a*：仮数の整数部（*a* は0～9） *b...b*：仮数の小数部（*b* は0～9） *c...c*：指数部（*c* は0～9）※3 E，e：浮動小数点数定数	• 100 • -△123 • 0.△ • -1.5600 • .56 • -02.4e+9 • 000e • 2.4E+009	• 509 文字まで記述できます。※4 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	項番 1 と同じ。
5	文字データ	CHARACTER	*a...a* *a...a*：1 文字以上のデータ	• abcdef△△ • ABCDEF • △	• 格納先の列の定義長までの文字数が記述できます。 • データの先頭を除き，末尾まで連続する半角空白は省略できます。※5	• …,,… • …,””,… 半角空白およびタブは挿入できません。 囲み文字指定オプションにNONEを指定した場合，囲み文字を使用

項番	表関数導出表の列のデータ型		フィールドデータの記述形式			
			形式	記述例	注意事項	ナル値が格納代入される記述例
						した例は指定できません。
6		VARCHAR	*a...a* *a..a*：1 文字以上のデータ	• abcdef△△ • ABCDEF • △△	• 格納先の列の定義長までの文字数が記述できます。	• …,,… 囲み文字は指定できません。また，半角空白およびタブは挿入できません。 **■長さ 0 のデータを指定する場合** • …,"",… ただし，囲み文字指定オプションにNONE を指定した場合，長さ 0 のデータは指定できません。
7	日時データ	DATE	{*YYYY-MM-DD* | *YYYY/MM/DD*} *YYYY*：年（0001～9999） *MM*：月（01～12） *DD*：日（01～その月の最終日）	• 2013-06-10 • 2013/06/10	• 形式に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	項番 1 と同じ。
8		TIME	*hh*:*mm*:*ss*〔.〔*nn...n*〕〕 *hh*：時（00～23） *mm*：分（00～59） *ss*：秒（00～59） *nn..n*：小数秒（*n* は0～9）	• 11:03:58 • 11:03:58. • 11:03:58 △.1234	• 小数秒（*nn..n*）は 12 文字まで記述できます。※6 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	項番 1 と同じ。
9		TIMESTAMP	{*YYYY-MM-DD* | *YYYY/MM/DD*} △*hh*:*mm*:*ss*〔.〔*nn...n*〕〕 *YYYY*：年（0001～9999） *MM*：月（01～12）	• 2013-06-10△ 11:03:58 • 2013-06-10△ 11:03:58 △.1234	• 小数秒（*nn..n*）は 12 文字まで記述できます。※6 • 形式および文字数の制限に関係なく，すべての文字の前後に 1	項番 1 と同じ。

項番	表関数導出表の列のデータ型		フィールドデータの記述形式			
			形式	記述例	注意事項	ナル値が格納代入される記述例
			DD：日（01～その月の最終日） *hh*：時（00～23） *mm*：分（00～59） *ss*：秒（00～59） *nn…n*：小数秒（*n* は0～9）		個以上の半角空白またはタブを挿入できます。※1	
10	バイナリデータ	BINARY	16 進文字列の場合 *a…a* *a*：0～9，A～F，またはa～f	• 12340000 • 90△AB • 90ab△CDEF	• 格納先の列の定義長×2 までの 2 の倍数の文字数を記述できます。※7 • データの先頭を除き，末尾まで連続する「00」は省略できます。※8 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	項番 1 と同じ。
11			2 進文字列の場合 *a…a* *a*：0 または1	• 01010101 • 0101△0101	• 格納先の列の定義長×8 までの 8 の倍数の文字数を記述できます。※7 • データの先頭を除き，末尾まで連続する「00000000」は省略できます。※8 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	項番 1 と同じ。
12		VARBINARY	16 進文字列の場合	• 12340000 • 90△AB	• 格納先の列の定義長×2 までの 2	• …,*,… • …,”*”,…

項番	表関数導出表の列のデータ型		フィールドデータの記述形式			
			形式	記述例	注意事項	ナル値が格納代入される記述例
			a…a *a*：0～9，A～F，またはa～f	• 90ab△CDEF	の倍数の文字数を記述できます。※7 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	• …,,… ただし，囲み文字指定オプションにNONE を指定した場合，囲み文字を使用した例は指定できません。 **■長さ 0 のデータを指定する場合** • …,””,… • …,”””””,… ただし，囲み文字指定オプションにNONE を指定した場合，長さ 0 のデータは指定できません。
13			2 進文字列の場合 *a…a* *a*：0 または1	• 01010101 • 0101△0101	• 格納先の列の定義長×8 までの 8 の倍数の文字数を記述できます。※7 • 形式および文字数の制限に関係なく，すべての文字の前後に 1 個以上の半角空白またはタブを挿入できます。※1	項番 12 と同じ。

（凡例）

△：1 個以上の半角空白，またはタブ

，：区切り文字

”：囲み文字

注※1

文字の前後に 1 個以上の半角空白（0x20）やタブ（0x09）がある場合，その半角空白およびタブは削除されます。

（例）△1△23△△4△△△ → 1234

なお，データがすべて削除された場合は，ナル値として扱われます。

注※2

格納先の位取りより下位の桁部分は切り捨てられます。

注※3

指数を省略した場合，指数として+0 を仮定します。

注※4

指定する値によっては，桁落ちが生じる場合があります。

注※5

入力データが定義長に満たない場合は，半角空白が残りの部分に格納されます。

注※6

小数秒（*nn...n*）の桁数が表のデータ型の小数秒精度に満たない場合は，右側に 0 が補われます。

小数秒（*nn...n*）の桁数が表のデータ型の小数秒精度を超える場合は，入力データは切り捨てられます。

注※7

16 進文字列の文字数が 2 の倍数でない場合，エラーとなります。

2 進文字列の文字数が 8 の倍数でない場合，エラーとなります。

注※8

入力データが定義長に満たない場合は，**0x00** が残りの部分に格納されます。

6.3 定数

プログラム中で値を変更できないデータを**定数**といいます。

6.3.1 定数の種類

定数には，数定数と一般定数があります。定数の種類を次の表に示します。

表 6-9 定数の種類

項番	定数の種類	説明	定数の種類		説明
1	数定数	数値を表す定数です。数定数には，右記に示す定数があります。	整数定数		整数を表す定数です。
2			10 進数定数		小数点以下がある数を表す定数です。
3			浮動小数点数定数		小数点以下がある数を表す定数です。
4	一般定数	文字，日時，バイナリデータを表す定数です。一般定数には，右記に示す定数があります。	文字列定数		文字を表す定数です。
5			日付定数		日付を表す定数です。
6			時刻定数		時刻を表す定数です。
7			時刻印定数		日付および時刻を表す定数です。
8			バイナリ定数	16 進形式バイナリ定数	16 進形式で表すバイナリ定数です。
9				2 進形式バイナリ定数	2 進形式で表すバイナリ定数です。

6.3.2 定数の記述形式

定数の記述形式と仮定されるデータ型を次の表に示します。

表 6-10 定数の記述形式と仮定されるデータ型

項番	定数の種類		記述形式	仮定されるデータ型
1	数定数	整数定数	• 記述形式 〔*符号*〕*符号なし整数* • 記述例 45，6789，-123 • 説明	INTEGER

項番	定数の種類		記述形式	仮定されるデータ型
			*符号*部分には，＋または-を記述します。＋は省略できます。	
2		10 進数定数	• 記述形式 〔*符号*〕*整数部.小数部* • 記述例 12.3，-456.，.789 • 説明 符号部分には，＋または-を記述します。＋は省略できます。 整数部と小数部には，符号なし整数を記述します。整数と小数のどちらかを記述する必要があります。また，小数点は必ず付けてください。	DECIMAL（*m*〔，*n*〕） *m*，*n*：記述した桁数
3		浮動小数点数定数	• 記述形式 *仮数*E*指数*，または*仮数*e*指数* • 記述例 +1.0E+1，1.0E2，-3.4e-01，.5E+67 • 説明 仮数は，整数定数または 10 進数定数の形式で記述します。 指数は 1～3 桁の整数定数の形式で記述します。指数は 10 のべき乗を意味しています。 文字E またはe は必ず記述してください。	DOUBLE PRECISION
4	一般定数	文字列定数	• 記述形式 '*文字列*' • 記述例 'HITACHI'，'88'，''''95.7.30' • 説明 文字列をアポストロフィ（'）で囲んで記述します。半角文字または全角文字を記述できます。 なお，「'95.7.30」のように，文字列中にアポストロフィがある場合，上記の記述例のように 1 個のアポストロフィを表すのに 2 個のアポストロフィを記述してください。	• *n* ＞ 0 の場合 CHAR(*n*) • *n* ＝ 0 の場合 VARCHAR(1) 実長は 0 です。 *n*：表記した文字列長
5		日付定数	• 記述形式 DATE'*YYYY-MM-DD*'，または DATE'*YYYY/MM/DD*' • 記述例 DATE'2012-03-30' DATE'2012/03/30' • 説明 年（*YYYY*）は 4 桁，月（*MM*）および日（*DD*）は 2 桁で記述します。桁数が足りない場	DATE

項番	定数の種類		記述形式	仮定されるデータ型
			合は，足りない分，左側に 0 を記述してください。 なお，*YYYY*，*MM*，*DD* には，DATE 型で有効となる値（例えば，*MM* は01～12）を指定してください。 '*YYYY-MM-DD*'，または'*YYYY/MM/DD*'の中に分離符号は記述できません。	
6		時刻定数	• 記述形式 TIME'*hh:mm:ss.nn...n*' • 記述例 TIME'11:03:58' TIME'11:03:58.123' TIME'11:03:58.123456' TIME'11:03:58.123456789' TIME'11:03:58.123456789012' TIME'11:03:58.' • 説明 時（*hh*），分（*mm*）および秒（*ss*）は 2 桁で記述します。桁数が足りない場合は，左側に 0 を記述してください。 *nn...n* は小数秒を意味しています。*nn...n* は，3，6，9，または 12 桁で表します。 小数秒を使用する場合は，秒と小数秒精度の間にピリオドを記述してください。 小数秒を省略し，ピリオドだけを指定した場合は，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 13 以上の場合は，エラーになります。 なお，*hh*，*mm*，*ss*，*nn...n* には，TIME 型で有効となる値（例えば，*hh* は00～23）を指定してください。 '*hh*:*mm*:*ss*.*nn...n*'の中に分離符号は記述できません。	TIME〔(*p*)〕 *p*：小数秒精度
7		時刻印定数	• 記述形式 TIMESTAMP'*YYYY-MM-DD hh*:*mm*:*ss*.*nn...n*'，または TIMESTAMP'*YYYY/MM/DD hh*:*mm*:*ss*.*nn...n*' • 記述例 TIMESTAMP'2012-03-30 11:03:58' TIMESTAMP'2012/03/30 11:03:58' TIMESTAMP'2014-07-30 11:03:58.123' TIMESTAMP'2014/07/30 11:03:58.123456789012'	TIMESTAMP〔(*p*)〕 *p*：小数秒精度

項番	定数の種類		記述形式	仮定されるデータ型
			TIMESTAMP'2014-07-30 11:03:58.' • 説明 *YYYY-MM-DD*（または*YYYY/MM/DD*）と*hh*:*mm*:*ss* の間に半角空白を記述します。 年（*YYYY*）は 4 桁，月（*MM*）および日（*DD*）は 2 桁で記述します。桁数に満たない場合は，足りない分，左側に 0 を記述してください。 同様に，時（*hh*），分（*mm*），秒（*ss*）の桁数が足りない場合も，左側に 0 を記述してください。 *nn...n* は小数秒を意味しています。*nn...n* は，3，6，9，または 12 桁で表します。 小数秒を使用する場合は，秒と小数秒精度の間にピリオドを記述してください。 小数秒を省略し，ピリオドだけを指定した場合は，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 13 以上の場合は，エラーになります。 なお，*YYYY*，*MM*，*DD*，*hh*，*mm*，*ss*，*nn...n* には，TIMESTAMP 型で有効となる値（例えば，*hh* は00～23）を指定してください。 '*YYYY-MM-DD hh*:*mm*:*ss*.*nn...n*'，または'*YYYY/MM/DD hh*:*mm*:*ss*.*nn...n*' の中に分離符号は記述できません。	
8		16 進形式バイナリ定数	• 記述形式 X'*16 進文字列*'，またはx'*16 進文字列*' • 記述例 X'82A0' X'82a0' x'82A0' • 説明 *16 進文字列*は，0～9 およびA～F（またはa～f）で表します。 *16 進文字列*の文字数は 2 の倍数にしてください。16 進文字 2 文字で 1 バイトになります。 *16 進文字列*の文字数は，64,000 文字以内にしてください。 *16 進文字列*の中に分離符号は記述できません。	• *n* ＞ 0 の場合 BINARY(*n* ÷ 2) • *n* ＝ 0 の場合 VARBINARY(1) 実長は 0 です。 *n*：表記した 16 進文字列長
9		2 進形式バイナリ定数	• 記述形式 B'*2 進文字列*'，またはb'*2 進文字列*' • 記述例 B'01010101' b'01010101'	• *n* ＞ 0 の場合 BINARY(*n* ÷ 8) • *n* ＝ 0 の場合 VARBINARY(1) 実長は 0 です。

項番	定数の種類		記述形式	仮定されるデータ型
			• 説明 *2進文字列*は，0と1で表します。 *2進文字列*の文字数は，8の倍数にしてください。2進文字8文字で1バイトになります。 *2進文字列*の文字数は，256,000文字以内にしてください。 *2進文字列*の中に分離符号は記述できません。	*n*：表記した2進文字列長

数定数の使用上の制限を次の表に示します。

表 6-11　数定数の使用上の制限

項番	数定数	範囲	指定できる桁数の最大値（上位の無効数字0の桁数を含む）
1	整数定数※1	-9,223,372,036,854,775,808～ 9,223,372,036,854,775,807	19桁
2	10進数定数	$-(10^{38}-1)$～-10^{-38}，0， および10^{-38}～$(10^{38}-1)$	38桁
3	浮動小数点数定数※2	約-1.7×10^{308}～-2.3×10^{-308}，0，および約2.3×10^{-308}～1.7×10^{308}	仮数部：17桁 指数部：3桁

注※1

整数定数の値の範囲を超える定数が，整数定数の表記方法で記述された場合，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

値の範囲は，ハードウェア表現に依存します。

6.3.3　既定の文字列表現

既定の文字列表現の形式に従った文字列定数を，次に示す定数として使用できます。

- 日付を表す定数
- 時刻を表す定数
- 時刻印を表す定数

ここでは，上記の既定の文字列表現の形式について説明します。

(1)　日付を表す既定の文字列表現

日付を表す既定の文字列表現には，既定の入力表現と既定の出力表現があります。

(a) 既定の入力表現

既定の入力表現の形式に従った文字列定数を，日付を表す定数として使用できます。日付を表す既定の入力表現の形式を次に示します。

既定の入力表現の形式：

```
'YYYY-MM-DD'または'YYYY/MM/DD'
```

- 年（*YYYY*）は4桁，月（*MM*）および日（*DD*）は2桁で表します。桁数が足りない場合は，足りない分，左側に0を補ってください。
- *YYYY*，*MM*，*DD* には，データ型がDATE型のときに有効となる値（例えば，*MM* は01～12）を指定してください。

例：

2013年7月30日を表す場合，次のようになります。

- 文字列定数（既定の入力表現）の場合：'2013-07-30'または'2013/07/30'
- 日付定数の場合：DATE'2013-07-30'またはDATE'2013/07/30'

(b) 既定の出力表現

adbsqlコマンドなどで日付データを検索した結果は，既定の出力表現に従った形式で出力されます。

既定の出力表現の形式：

```
'YYYY-MM-DD'
```

年（*YYYY*）は4桁，月（*MM*）および日（*DD*）は2桁で出力されます。桁数が足りない場合は，足りない分，左側に0が補われます。

例：

日付データがX'20130730'の場合，既定の出力表現は次のようになります。

```
'2013-07-30'
```

(2) 時刻を表す既定の文字列表現

時刻を表す既定の文字列表現には，既定の入力表現と既定の出力表現があります。

(a) 既定の入力表現

既定の入力表現の形式に従った文字列定数を，時刻を表す定数として使用できます。時刻を表す既定の入力表現の形式を次に示します。

既定の入力表現の形式：

```
'hh:mm:ss.nn...n'
```

- 時（*hh*），分（*mm*），および秒（*ss*）は 2 桁で表します。桁数が足りない場合は，足りない分，左側に 0 を補ってください。
- 小数秒を使用する場合は，「.*nn...n*」を指定してください。*nn...n* は，3，6，9，または 12 桁で表します。*nn...n* が，3，6，9，または 12 桁以外の場合，次のように小数秒精度が仮定されます。その際，足りない桁数分だけ右側に 0 が補われます。

nn...n の桁数	仮定される小数秒精度
1，2	3
4，5	6
7，8	9
10，11	12

- *nn...n* が 13 桁以上の場合は，エラーになります。
- 秒と小数秒の間にピリオドが必要です。
- 小数秒を使用しない場合は，「.*nn...n*」の指定は不要です。
- *nn...n* を省略し，ピリオドだけを指定した場合は，小数秒精度が 0 のデータとして扱われます。
- *hh*，*mm*，*ss*，*nn...n* には，データ型がTIME 型のときに有効となる値（例えば，hh は00～23）を指定してください。

例：

11 時 3 分 58.123456 秒を表す場合，次のようになります。

- 文字列定数（既定の入力表現）の場合：'11:03:58.123456'
- 時刻定数の場合：TIME'11:03:58.123456'

(b) 既定の出力表現

adbsql コマンドなどで時刻データを検索した結果は，既定の出力表現に従った形式で出力されます。

既定の出力表現の形式：

```
'hh:mm:ss.nn...n'
```

- 時（*hh*），分（*mm*），および秒（*ss*）は 2 桁で出力されます。桁数が足りない場合は，足りない分，左側に 0 が補われます。
- *nn...n* には小数秒が表示されます。時刻データの小数秒精度の指定に従って小数秒の桁数が決まります。
- 小数秒精度が 0 の場合は，「.*nn...n*」の部分は表示されません。

例：

時刻データがX'110358123'の場合，既定の出力表現は次のようになります。

```
'11:03:58.123'
```

(3) 時刻印を表す既定の文字列表現

時刻印を表す既定の文字列表現には，既定の入力表現と既定の出力表現があります。

(a) 既定の入力表現

既定の入力表現の形式に従った文字列定数を，時刻印を表す定数として使用できます。時刻印を表す既定の入力表現の形式を次に示します。

既定の入力表現の形式：

'YYYY-MM-DD hh:mm:ss.nn...n' または *'YYYY/MM/DD hh:mm:ss.nn...n'*

- 年（*YYYY*）は 4 桁で表します。月（*MM*），日（*DD*），時（*hh*），分（*mm*），および秒（*ss*）は 2 桁で表します。桁数が足りない場合は，足りない分，左側に 0 を補ってください。
- *YYYY-MM-DD* と *hh:mm:ss* の間に半角空白が必要です。
- 小数秒を使用する場合は，「*.nn...n*」を指定してください。*nn...n* は，3，6，9，または 12 桁で表します。*nn...n* が，3，6，9，または 12 桁以外の場合，次のように小数秒精度が仮定されます。その際，足りない桁数分だけ右側に 0 が補われます。

nn...n の桁数	仮定される小数秒精度
1，2	3
4，5	6
7，8	9
10，11	12

- *nn...n* が 13 桁以上の場合は，エラーになります。
- 秒と小数秒の間にピリオドが必要です。
- 小数秒を使用しない場合は，「*.nn...n*」の指定は不要です。
- *nn...n* を省略し，ピリオドだけを指定した場合は，小数秒精度が 0 のデータとして扱われます。
- *YYYY*，*MM*，*DD*，*hh*，*mm*，*ss*，*nn...n* には，データ型がTIMESTAMP 型のときに有効となる値（例えば，*hh* は00～23）を指定してください。

例：

2013 年 7 月 30 日 11 時 3 分 58.123456 秒を表す場合，次のようになります。

- 文字列定数（既定の入力表現）の場合：'2013-07-30 11:03:58.123456' または '2013/07/30 11:03:58.123456'
- 時刻印定数の場合：TIMESTAMP'2013-07-30 11:03:58.123456' またはTIMESTAMP'2013/07/30 11:03:58.123456'

(b) 既定の出力表現

adbsql コマンドなどで時刻印データを検索した結果は，既定の出力表現に従った形式で出力されます。

既定の出力表現の形式：

```
'YYYY-MM-DD hh:mm:ss.nn...n'
```

- 年（*YYYY*）は 4 桁，月（*MM*），日（*DD*），時（*hh*），分（*mm*），および秒（*ss*）は 2 桁で出力されます。桁数が足りない場合は，足りない分，左側に 0 が補われます。
- *nn..n* には小数秒が表示されます。時刻印データの小数秒精度の指定に従って小数秒の桁数が決まります。
- 小数秒精度が 0 の場合は，「.*nn..n*」の部分は表示されません。

例：

時刻印データがX'20130730110358123'の場合，既定の出力表現は次のようになります。

```
'2013-07-30 11:03:58.123'
```

6.4　日時情報取得関数

日時情報取得関数には次に示す関数があります。

- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIMESTAMP

ここでは，各関数について説明します。

6.4.1　CURRENT_DATE

現在の日付を返します。

(1)　指定形式

日時情報取得関数CURRENT_DATE：：＝CURRENT_DATE

(2)　規則

1. 実行結果のデータ型はDATE 型になります。
2. 1 つの SQL 文中にCURRENT_DATE を複数指定した場合，すべて同じ日付になります。
3. HADB サーバで SQL 文を実行するときにCURRENT_DATE の値を取得します。SQL 文を実行するときのインタフェースおよび実行方法については，マニュアル『HADB　AP 開発ガイド』の JDBC の API，ODBC 関数，または CLI 関数の説明を参照してください。
4. CURRENT_DATE は，値指定が指定できる個所に指定できます。

(3)　例題

例題 1

販売履歴表（SALESLIST）から，本日商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE）を検索します。

```
SELECT ”USERID”,”PUR-CODE”
    FROM ”SALESLIST”
        WHERE ”PUR-DATE”=CURRENT_DATE
```

例題 2

販売履歴表（SALESLIST）に次に示すデータ（行）を挿入します。

- 顧客 ID（USERID）：U00358
- 商品コード（PUR-CODE）：P003

- 販売個数（PUR-NUM）：5
- 購入日（PUR-DATE）：今日の日付

```
INSERT INTO "SALESLIST"
      ("USERID","PUR-CODE","PUR-NUM","PUR-DATE")
    VALUES('U00358','P003',5,CURRENT_DATE)
```

6.4.2 CURRENT_TIME

現在の時刻を返します。

(1) 指定形式

日時情報取得関数CURRENT_TIME：：＝CURRENT_TIME〔(*p*)〕

(2) 規則

1. *p* には小数秒精度（小数秒の桁数）を指定します。p に指定できる値は，0，3，6，9，または12 のどれかです。例えば，*p* に3 を指定した場合，CURRENT_TIME の実行結果の小数秒の桁数が 3 桁になります。
2. (*p*)を省略した場合，*p* ＝ 0 が仮定されます。
3. 実行結果のデータ型はTIME 型になります。
4. 1 つの SQL 文中にCURRENT_TIME を複数指定した場合，すべて同じ時刻になります。
5. HADB サーバで SQL 文を実行するときにCURRENT_TIME の値を取得します。SQL 文を実行するときのインタフェースおよび実行方法については，マニュアル『HADB　AP 開発ガイド』の JDBC の API，ODBC 関数，または CLI 関数の説明を参照してください。
6. CURRENT_TIME によって取得される小数秒の精度は，ハードウェアの性能に依存します。例えば，CURRENT_TIME(12)を指定しても，使用しているハードウェアによっては 12 桁の小数秒精度が取得できないことがあります。

   （例）

   10:35:55.123456000000

   小数秒精度が 6 桁までしか取得できない場合，上記のように 7 桁目以降は 0 になります。
7. CURRENT_TIME は，値指定が指定できる個所に指定できます。

(3) 例題

例題

日ごとの販売履歴表（SALESLIST_DAY）に，商品の販売情報を追加します。販売履歴表の列構成は，次のとおりです。商品の販売時刻（SALE_TIME）には，現在の時刻を格納します。

- 支店コード（SCODE）

- 商品コード（GCODE）
- 顧客の性別（SEX）
- 商品の販売時刻（SALE_TIME）

```
INSERT INTO ”SALESLIST_DAY”
      (”SCODE”,”GCODE”,”SEX”,”SALE_TIME”)
    VALUES('S001','G03542','M',CURRENT_TIME)
```

6.4.3 CURRENT_TIMESTAMP

現在の時刻印（日付と時刻）を返します。

(1) 指定形式

```
日時情報取得関数CURRENT_TIMESTAMP : : =CURRENT_TIMESTAMP〔(p)〕
```

(2) 規則

1. p には小数秒精度（小数秒の桁数）を指定します。p に指定できる値は，0，3，6，9，または12 のどれかです。例えば，p に3 を指定した場合，CURRENT_TIMESTAMP の実行結果の小数秒の桁数が 3 桁になります。
2. (p)を省略した場合，p ＝ 0 が仮定されます。
3. 実行結果のデータ型はTIMESTAMP 型になります。
4. 1 つの SQL 文中にCURRENT_TIMESTAMP を複数指定した場合，すべて同じ日付，同じ時刻になります。
5. HADB サーバで SQL 文を実行するときにCURRENT_TIMESTAMP の値を取得します。SQL 文を実行するときのインタフェースおよび実行方法については，マニュアル『HADB　AP 開発ガイド』の JDBC の API，ODBC 関数，または CLI 関数の説明を参照してください。
6. CURRENT_TIMESTAMP によって取得される小数秒の精度は，ハードウェアの性能に依存します。例えば，CURRENT_TIMESTAMP(12)を指定しても，使用しているハードウェアによっては 12 桁の小数秒精度が取得できないことがあります。

   (例)

   2014-09-25 10:35:55.123456000000

   小数秒精度が 6 桁までしか取得できない場合，上記のように 7 桁目以降は 0 になります。
7. CURRENT_TIMESTAMP は，値指定が指定できる個所に指定できます。

(3) 例題

例題

顧客表（USERSLIST）に新規の顧客情報を追加します。顧客表の列構成は，次のとおりです。

- 顧客 ID（USERID）
- 名前（NAME）
- 性別（SEX）
- 顧客情報の最終更新日時（LAST_UPDATE_TIME）

```
INSERT INTO ”USERSLIST”
      (”USERID”,”NAME”,”SEX”,”LAST_UPDATE_TIME”)
    VALUES('U00887','Edward Connelly','M',CURRENT_TIMESTAMP)
```

6.5　ユーザ情報取得関数

ここでは，次に示すユーザ情報取得関数について説明します。

- CURRENT_USER

6.5.1　CURRENT_USER

実行中の HADB ユーザの認可識別子を返します。

(1)　指定形式

ユーザ情報取得関数CURRENT_USER：：＝CURRENT_USER

(2)　規則

1. 実行結果のデータ型はVARCHAR 型になります。
2. 1 つの SQL 文中にCURRENT_USER を複数指定した場合，すべて同じ値になります。
3. HADB サーバで SQL 文を実行するときにCURRENT_USER の値を取得します。SQL 文を実行するときのインタフェースおよび実行方法については，マニュアル『HADB　AP 開発ガイド』の JDBC の API，ODBC 関数，または CLI 関数の説明を参照してください。
4. CURRENT_USER は，値指定が指定できる個所に指定できます。

(3)　例題

例題

自分（HADB サーバに接続中の認可識別子の HADB ユーザ）が所有する表の情報の一覧（SQL_TABLES の内容）を検索します。

```
SELECT * FROM "MASTER"."SQL_TABLES"
    WHERE TABLE_SCHEMA=CURRENT_USER
```

6.6　変数（？パラメタ）

？パラメタとは，SQL に値を渡す変数のことです。SQL に対して値を渡す際，値を渡す個所に「？」と指定します。この「？」が？パラメタです。

6.6.1　？パラメタ指定時の規則

1. 指定した？パラメタに仮定されるデータ型およびデータ長は，？パラメタを指定した個所によって変わります。？パラメタ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 6-12　？パラメタに仮定されるデータ型およびデータ長

項番	？パラメタを指定した個所	仮定されるデータ型およびデータ長
1	述語中（LIKE 述語，NULL 述語，および LIKE_REGEX 述語を除く）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	• INSERT 文のVALUES に指定する挿入値に単独で指定した場合 • UPDATE のSET に指定する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 中に指定できる？パラメタは最大 1,000 個になります。
3. ？パラメタに値を渡す場合，仮定されるデータ型およびデータ長の値を渡してください。
4. SQL 文中に指定する？パラメタのデータ長の合計が，32,000,000 バイト以下になるようにしてください。

6.6.2　？パラメタを指定できる個所

？パラメタを指定できる個所を次の表に示します。

表 6-13　？パラメタを指定できる個所

項番	SQL 文	？パラメタを指定できる個所
1	SELECT	選択式※1
2		探索条件中で定数が指定できる個所※2
3		ORDER BY 句※1
4		LIMIT 句
5	INSERT	挿入値，および行挿入値
6	UPDATE	更新値，および行更新値

項番	SQL 文	？パラメタを指定できる個所
7		探索条件中で定数が指定できる個所※2
8	DELETE	探索条件中で定数が指定できる個所※2
9	PURGE CHUNK	

注※1

？パラメタを単独では指定できません。

注※2

次の個所には指定できません。

- 比較述語の両側
- BETWEEN 述語の左側
- ビュー定義（CREATE VIEW）
- WITH 句中

6.6.3 留意事項

スカラ演算中に指定した？パラメタのデータは，操作対象となる各行に対して演算を行います。列を含まないスカラ演算中に？パラメタを指定している部分については，値が固定のときは，SQL 文中に定数で指定することを検討してください。

6.7　ナル値

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特殊な値です。領域に値がないか，または設定されていない場合は，ナル値が設定されます。ここでは，ナル値の扱いについて説明します。

検索結果の列の値の受け取り

- JDBC ドライバを使用している場合

  ResultSet インタフェースのwasNull メソッドを使用して，取得した列値がナル値かどうかを判定します。

- ODBC ドライバを使用している場合

  検索結果の列の値がナル値の場合，SQLBindCol またはSQLGetData の引数StrLen_or_IndPtr に SQL_NULL_DATA が設定されます。

- CLI 関数を使用している場合

  ナル値の識別にはインジケータを使用します。詳細については，マニュアル『HADB　AP 開発ガイド』の『a_rdb_SQLInd_t（インジケータ)』を参照してください。

比較時の扱い

次に示す項目以外に指定した値式，列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。

- NULL 述語の左側の値式
- LIKE 述語の「ESCAPE エスケープ文字」に指定した値式

なお，スカラ関数DECODE でのナル値の比較時の扱いについては，「8.16.1　DECODE」を参照してください。

ソート時の扱い

ソート指定リストのナル値ソート順指定に従ってナル値を並べ替えます。ナル値ソート順指定については，「7.25.1　ソート指定リストの指定形式」を参照してください。

グループ分け時の扱い

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

重複排除時の扱い

ナル値同士は，重複するものとして扱われます。

集合関数での扱い

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，ナル値かどうかに関係なく条件を満たすすべての行を計算します。

ウィンドウ関数での扱い

ウィンドウ関数の場合，ウィンドウ指定に指定した値式の結果がナル値となる行があるときは，ナル値同士を同じ値として扱います。

インデクスでの扱い

ナル値がある列にインデクスを定義できます。

6.8　範囲変数

ここでは，範囲変数となる識別子の種類と，範囲変数の有効範囲について説明します。

6.8.1　範囲変数とは

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，有効範囲と名称を持ちます。

相関名が指定された場合，相関名の指定された表名および問合せ名は有効範囲がなくなります。

表指定の位置に有効範囲を持つ範囲変数のうち，表指定と同じ名称で最も内側（いちばん近い）の問合せ指定（更新対象表または削除対象表）の範囲変数が，その表指定の範囲変数になります。

範囲変数となる識別子の種類を次の表に示します。

表 6-14　範囲変数となる識別子の種類

項番	範囲変数		扱い
1	相関名		○
2	表名および問合せ名	相関名の指定がない場合	○
3		相関名の指定がある場合	×
4	UPDATE 文に指定した更新対象表，または DELETE 文に指定した削除対象表	相関名の指定がない場合	○
5		相関名の指定がある場合	×

（凡例）

○：範囲変数となります。

×：範囲変数となりません。

6.8.2　範囲変数の名称

範囲変数の名称の例を次の表に示します。

表 6-15　範囲変数の名称の例

項番	SQL 例	範囲変数の名称
1	WITH Q1 AS	－
2	(SELECT * FROM	
3	T0),	A.T0
4	Q2 AS	－

項番	SQL 例	範囲変数の名称
5	(SELECT * FROM	
6	Q1)	Q1
7	SELECT "T1"."C1","X"."C1","Y"."C1" FROM	－
8	"T1"	A.T1
9	,"A"."T1" "X"	X
10	,"A"."T2" "Y"	Y

(凡例)

－：該当しません。

A：スキーマ名

T0, T1, T2：表識別子

X, Y：相関名

C1：列名

Q1, Q2：問合せ名

6.8.3 範囲変数の有効範囲

範囲変数の有効範囲の例を次の表に示します。

表 6-16 範囲変数の有効範囲の例

項番	SQL 例	範囲変数							
		A.T1	X	A.T2	A.T3	Y	A.T4	A.T5	Z
1	SELECT "X"."C1","T2"."C2"	×	○	○	×	○	×	×	×
2	FROM "A"."T1" "X",	×	○	○	×	○	×	×	×
3	"A"."T2",	×	○	○	×	○	×	×	×
4	(SELECT * FROM "T3"	×	×	×	○	×	×	×	×
5	WHERE "T3"."C1"=100) "Y"	×	×	×	○	×	×	×	×
6	WHERE "X"."C1"=100 AND	×	○	○	×	○	×	×	×
7	"X"."C1"=ANY(	×	○	○	×	○	×	×	×
8	SELECT "T4"."C1" FROM "T4",	×	×	×	×	×	○	×	○
9	(SELECT *	×	×	×	×	×	×	○	×
10	FROM "T5"	×	×	×	×	×	×	○	×
11	WHERE "T5"."C1"="X"."C1")"Z"	×	○	○	×	○	×	○	×

項番	SQL 例	範囲変数							
		A.T1	X	A.T2	A.T3	Y	A.T4	A.T5	Z
12	WHERE ”T4”.”C2”=”A”.”T2”.”C2”)	×	○	○	×	○	○	×	○

（凡例）

○：有効範囲内です。

×：有効範囲外です。

A：スキーマ名

T1～T5：表識別子

X，Y，Z：相関名

C1，C2：列名

SELECT 文，UPDATE 文およびDELETE 文の範囲変数の有効範囲について説明します。

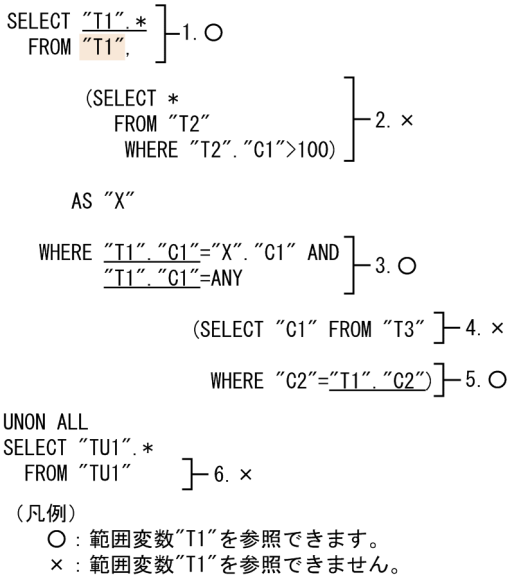
(1) SELECT 文のFROM 句に指定した範囲変数の有効範囲

範囲変数となる識別子を直接FROM 句に含む問合せ指定と，その問合せ指定に含まれる副問合せ中の探索条件が，範囲変数の有効範囲になります。ただし，次に示す導出表中または集まり導出表中では，範囲変数は有効範囲を持ちません。

- 範囲変数となる識別子を直接含むFROM 句と同じFROM 句に指定した導出表中
- 範囲変数となる識別子を直接含むFROM 句と同じFROM 句中で，かつ範囲変数となる識別子の左側に指定した集まり導出表中

例を次の図に示します。

図 6-12 FROM 句に指定した範囲変数"T1"の有効範囲の例（その 1）



[説明]

1. 範囲変数をFROM 句に直接含む問合せ指定のため，範囲変数”T1”は参照できます。
2. 範囲変数となる識別子を直接含むFROM 句と同じFROM 句に指定した導出表のため，範囲変数"T1"は参照できません。
3. 範囲変数をFROM 句に直接含む問合せ指定のため，範囲変数"T1"は参照できます。
4. 副問合せの探索条件以外のため，範囲変数"T1"は参照できません。
5. 範囲変数をFROM 句に直接含む問合せ指定に含まれる副問合せの探索条件のため，範囲変数"T1"は参照できます。
6. 外側の問合せのため，範囲変数"T1"は参照できません。

図 6-13　FROM 句に指定した範囲変数"T1"の有効範囲の例（その 2）

```
SELECT "T1".*     ┐1.○
  FROM ("T1"

         LEFT JOIN
         "T2"
          ON "T1"."C1"="T2"."C1"   ┐2.○
        )
        LEFT JOIN
       ("T3"
        LEFT JOIN
        "T4"
         ON "T3"."C1"="T4"."C1")   ┐3.×
        )
        ON "T1"."C2"="T3"."C2"                ┐
         AND EXISTS                           ├4.○
           (SELECT * FROM "T5"                │
              WHERE "T5"."C3"="T1"."C3")      ┘
```

（凡例）
○：範囲変数"T1"を参照できます。
×：範囲変数"T1"を参照できません。

[説明]

結合表の結合指定の例です。

1. 範囲変数をFROM 句に直接含む問合せ指定のため，範囲変数"T1"は参照できます。
2. 結合指定を含む結合表の表参照に"T1"が指定されているため，範囲変数"T1"は参照できます。
3. 結合指定を含む結合表の表参照に"T1"が指定されていないため，範囲変数"T1"は参照できません。
4. 結合指定を含む結合表の表参照に"T1"が指定されているため，範囲変数"T1"は参照できます。

図 6-14　範囲変数の有効範囲の例（問合せ名を指定している場合）

```
WITH                                                                    ┐
  "Q1"("C1","C2") AS (SELECT "C1,"C2" FROM "T1"                         │問合せ名Q1を
                      UNION ALL                                         │参照できます。
                      SELECT "Q1"."C1"+1,"T2"."C2" FROM "Q1","T2"       │
                        WHERE "Q1"."C1"<5)                              │
  "Q2"("C1","C2") AS (SELECT "T2"."C1,"T2"."C2" FROM "T2","Q1"          │  ┐問合せ名Q2を
                        WHERE "T2"."C1"="Q1"."C1"),                     │  │参照できます。
  "Q3"("C1","C2") AS (SELECT "Q1"."C1", "Q2"."C2" FROM "Q1","Q2"        │  │
                        WHERE "Q1"."C1"="Q2"."C1")                      │  │ ┐問合せ名Q3を
SELECT * FROM "Q3"                                                      ┘  ┘ ┘参照できます。
```

[説明]

問合せ名は，スキーマ名で修飾できません。スキーマ名で修飾した場合，問合せ名ではなく表識別子として見なされます。また，問合せ名と同じ名称の表識別子が存在する場合，問合せ名の有効範囲内では，問合せ名として見なされます。問合せ名の有効範囲外では，表識別子として見なされます。そのため，表識別子としてFROM 句に範囲変数を指定する場合は，スキーマ名で修飾してください。

図 6-15　範囲変数の有効範囲の例（集まり導出表を指定している場合）

```
SELECT "T3".*                          ┐1.○
  FROM "T1",
       UNNEST("T1"."C2")                 ┐2.×
         AS "CDT2",
       "T3",
       UNNEST("T3"."C4")                 ┐3.○
         AS "CDT4"
 WHERE "T3"."C1" = "CDT2"."C2"           ┐4.○
```

[説明]

範囲変数T3 についての有効範囲を説明しています。

1. 範囲変数をFROM 句に直接含む問合せ指定のため，範囲変数T3 を参照できます。
2. 範囲変数となる識別子を直接含むFROM 句と同じFROM 句中では，範囲変数となる識別子の左側に指定した集まり導出表中であるため，範囲変数T3 を参照できません。
3. 範囲変数となる識別子を直接含むFROM 句と同じFROM 句中であり，範囲変数となる識別子の左側に指定した集まり導出表中ではないため，範囲変数T3 を参照できます。
4. 範囲変数をFROM 句に直接含む問合せ指定のため，範囲変数T3 を参照できます。

(2)　UPDATE 文の更新対象表（範囲変数）の有効範囲

UPDATE 文のSET 句，探索条件，および探索条件に含まれる副問合せ中の探索条件が，範囲変数の有効範囲となります。例を次の図に示します。

図 6-16　UPDATE 文に指定した範囲変数"T1"の有効範囲の例

```
UPDATE "T1"                    ┐
   SET "C1"=100                ├1.○
     ,"C2"=                    ┘
          (SELECT "C1" FROM "T4"       ┐2.×
               WHERE "C2"="T1"."C2"    ├3.○
           )                           ┘
   WHERE "T1"."C1"=ANY                          ┐4.○
                 (SELECT "C1" FROM "T3"         ┐5.×
                      WHERE "C2"="T1"."C2"      ├6.○
                  )                             ┘
  (凡例)
     ○：範囲変数"T1"を参照できます。
     ×：範囲変数"T1"を参照できません。
```

[説明]

1. UPDATE 文のSET 句，探索条件のため，範囲変数"T1"は参照できます。
2. 副問合せの探索条件以外のため，範囲変数"T1"は参照できません。
3. UPDATE 文のSET 句に含まれる副問合せの探索条件のため，範囲変数"T1"は参照できます。

4. UPDATE 文のSET 句，探索条件のため，範囲変数"T1"は参照できます。

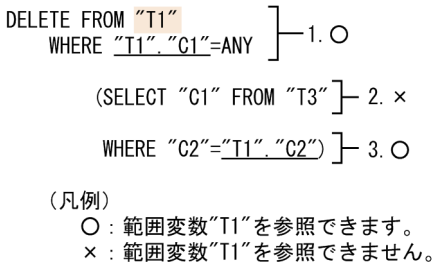
5. 副問合せの探索条件以外のため，範囲変数"T1"は参照できません。

6. UPDATE 文の探索条件に含まれる副問合せの探索条件のため，範囲変数"T1"は参照できます。

(3) DELETE 文の削除対象表（範囲変数）の有効範囲

DELETE 文の探索条件，および探索条件に含まれる副問合せ中の探索条件が，範囲変数の有効範囲となります。例を次の図に示します。

図 6-17 DELETE 文に指定した範囲変数"T1"の有効範囲の例



[説明]

1. DELETE 文の探索条件のため，範囲変数"T1"は参照できます。

2. 副問合せの探索条件以外のため，範囲変数"T1"は参照できません。

3. DELETE 文の探索条件に含まれる副問合せの探索条件のため，範囲変数"T1"は参照できます。

(4) INSERT 文の挿入対象表（範囲変数ではない）の有効範囲

INSERT 文の挿入対象表は，挿入値中（副問合せも含みます），および問合せ式本体中のどこにも有効範囲を持ちません。例を次に示します。

(例)

```
INSERT INTO "T1"     ...1
    VALUES(
            (SELECT "C1" FROM "T3"
                WHERE "C2">="C3"     ...2
            )
           )
```

[説明]

1. 下線で示している挿入対象表は，INSERT 文中のどこにも有効範囲を持ちません。

2. 副問合せの探索条件であっても挿入対象表T1 は参照できません。

6.9　導出列名

問合せ指定中の各句によって導出される表の列を**導出列**といい，導出列名はここで説明する規則に従って決まります。

6.9.1　問合せ指定中の導出列名の決定規則

問合せ指定中の導出列名は，次の規則に従って決まります。

- FROM 句の場合

  表参照中に指定した表の列名が導出列名となります。

  (例)

```
SELECT "C1","C2" FROM "T1"
```

  上記の例の場合，表T1 の各列の列名が導出列名になります。

- GROUP BY 句の場合

  グループ化列の列名が導出列名となります。

  (例)

```
SELECT "C1","GC2",SUM("C3") FROM "T1"
    GROUP BY "C1",SUBSTR("C2",5,2) AS "GC2"
```

  上記の例の場合，導出列名は"C1", "GC2"になります。

6.9.2　問合せの結果の導出列名の決定規則

問合せの結果の導出列名は，次の規則に従って決まります。

(1)　問合せ式の場合

1 番目に指定した問合せ一次子の結果によって導出される列名が**導出列名**になります。

(2)　問合せ指定または副問合せの場合

■i 番目の選択式にAS 句の指定がない場合

- i 番目の選択式に指定された値式が列指定の場合

  その列名が i 番目の導出列名になります。

- i 番目の選択式に指定された値式が副問合せの場合

  副問合せの結果によって導出される列名が i 番目の導出列名になります。

- 上記以外

導出列に列名は設定されません。

■i 番目の選択式にAS 句の指定がある場合

i 番目の選択式に指定したAS 句の列名が，i 番目の導出列名になります。

導出列名の決定規則の例を次に示します。

(例 1)

```
SELECT ”C1”,”C2”,”C3” FROM ”T1”
```

上記のSELECT 文を実行した場合，導出列名は”C1”，”C2”，”C3”になります。導出列の列順もこの順序になります。

(例 2)

```
SELECT ”C1”,”C2” AS ”X2”,”C3” FROM ”T1”
```

上記のSELECT 文を実行した場合，導出列名は”C1”，”X2”，”C3”になります。導出列の列順もこの順序になります。

(例 3)

```
SELECT ”C1”,SUM(”C2”) AS ”SUM-C2”,AVG(”C2”) FROM ”T1”
    WHERE ”C3”>=DATE’2011-09-03’
    GROUP BY ”C1”
```

上記のSELECT 文を実行した場合，導出列名は”C1”，”SUM-C2”，"列名なし"になります。導出列の列順もこの順序になります。

(3) 導出表の場合

■導出列リストの指定がある場合

i 番目の導出列リスト中の列名が，i 番目の導出列名になります。

■導出列リストの指定がない場合

- 導出表として表副問合せを指定する場合

  副問合せの結果，導出される表の列名が導出列名になります。

  導出列名が設定されていない場合，EXP*nnnn*_NO_NAME が導出列名になります。*nnnn* は，0001～4000 の符号なし整数です。*nnnn* には，導出列名が設定されていない列に対して，先頭から順番に0001，0002，…が仮定されます。

- 導出表として表値構成子を指定する場合

  導出列名は，EXP*nnnn*_NO_NAME になります。*nnnn* は，0001～4000 の符号なし整数です。*nnnn* には，導出列の先頭から順番に0001，0002，…が仮定されます。

(4)　表関数導出表の場合

■表関数列リストの指定がある場合

表関数列リストに指定したi番目の列名が，i番目の導出列名になります。

■表関数列リストの指定がない場合

表関数導出表に指定したシステム定義関数によって導出される列名が，導出列名になります。

(5)　集まり導出表の場合

■導出列リストの指定がある場合

導出列リストに指定したi番目の列名が，i番目の導出列名になります。

■導出列リストの指定がない場合

集まり導出表中に指定したi番目の列指定の列名が，i番目の導出列名になります。

6.9.3　導出列名の有効範囲

導出列名の有効範囲の例を次の表に示します。

表 6-17　導出列名の有効範囲の例（GROUP BY 句の指定がない場合）

SQL 文の例	有効範囲				
	C1	C2	C3	DC3	C4
SELECT ”C1”,”C2”	○	○	×	○	×
FROM ”T1”,	○	○	×	○	×
(SELECT *	×	×	○	×	×
FROM ”T2”	×	×	○	×	×
WHERE ”C3”=100)	×	×	○	×	×
”Y”(”DC3”)	×	×	×	○	×
WHERE ”C1”=100 AND	○	○	×	○	×
”C2”=ANY	○	○	×	○	×
(SELECT ”C4”	×	×	×	×	○
FROM ”T3”	○	○	×	×	○
WHERE ”DC3”=”C4”)	○	○	×	○	○

（凡例）

○：有効範囲内です。

×：有効範囲外です。

T1，T2，T3：表識別子

C1，C2：T1 の列名

C3：T2 の列名

C4：T3 の列名

Y：相関名

DC3：導出表Y の導出列名

表 6-18　導出列名の有効範囲の例（GROUP BY 句の指定がある場合）

SQL 文の例	有効範囲						
	C1	C2	GC2	C3	DC3	C4	C5
SELECT ”C1”,”GC2”	○	△	○	×	△	×	×
FROM ”T1”,	○	○	×	×	○	×	×
(SELECT *	×	×	×	○	×	×	×
FROM ”T2”	×	×	×	○	×	×	×
WHERE ”C3”=100)	×	×	×	○	×	×	×
”Y”(”DC3”)	×	×	×	×	○	×	×
WHERE ”C1”=100 AND	○	○	×	×	○	×	×
”C2”=ANY	○	○	×	×	○	×	×
(SELECT ”C4”	×	×	×	×	×	○	×
FROM ”T3”	○	○	×	×	○	○	×
WHERE ”DC3”=”C4”)	○	○	×	×	○	○	×
GROUP BY ”C1”,”C2”+100 AS ”GC2”	○	○	×	×	○	×	×
HAVING ”C1”=100 AND	○	△	○	×	△	×	×
”GC2”=ANY	○	△	○	×	△	×	×
(SELECT ”C5”	×	×	×	×	×	×	○
FROM ”T4”	○	×	×	×	×	×	○
WHERE SUM(”DC3”)=”C5”)	○	△	×	×	△	×	○

（凡例）

○：有効範囲内です。

△：有効範囲内ですが，グループ化列ではないため，集合関数の引数中に限り指定できます。

×：有効範囲外です。

T1，T2，T3，T4：表識別子

C1，C2：T1 の列名

C3：T2 の列名

C4：T3 の列名

C5：T4 の列名

GC2：グループ化列の列名

Y：相関名

DC3：導出表 Y の導出列名

6.10　予約語

ここでは，HADB の予約語，および名前が予約語と重複した場合の対応について説明します。

6.10.1　予約語の一覧

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指定できません。予約語と同じ文字列を名前に指定したい場合は，「6.10.2　名前が予約語と重複した場合の対応」を参照してください。

> **メモ**
>
> HADB の予約語は，SQL92（ISO 9075－1992 Database Language SQL）で規定されている予約語に準拠しています。

HADB の予約語を次の表に示します。

表 6-19　HADB の予約語

頭文字	予約語
A	ABS, ABSOLUTE, ACCESS, ACTION, ADD, ADMIN, AFTER, AGGREGATE, AGGREGATES, ALIAS, ALL, ALLOCATE, **ALTER**, AND, ANDNOT, ANY, ARE, ARRAY, ARRAY_AGG, ARRAY_MAX_CARDINALITY, AS, ASC, ASENSITIVE, ASSERTION, **ASSIGN**, ASYMMETRIC, AT, ATOMIC, AUTHORIZATION, AVG
B	BEFORE, BEGIN, BEGIN_FRAME, BEGIN_PARTITION, BETWEEN, BIGINT, BINARY, BIT, BIT_AND_TEST, BIT_LENGTH, BLOB, BOOLEAN, BOTH, BREADTH, BY
C	CALL, CALLED, CARDINALITY, CASCADE, CASCADED, CASE, CAST, CATALOG, CEIL, CEILING, CHANGE, CHAR, CHAR_LENGTH, CHARACTER, CHARACTER_LENGTH, CHECK, CHUNK, CHUNKID, CLASS, CLASSIFIER, CLOB, **CLOSE**, CLUSTER, COALESCE, COLLATE, COLLATION, COLLECT, COLUMN, COLUMNS, COMMENT, **COMMIT**, COMPLETION, CONDITION, CONNECT, CONNECTION, CONSTRAINT, CONSTRAINTS, CONSTRUCTOR, CONTINUE, CONVERT, CORR, CORRESPONDING, COUNT, COUNT_FLOAT, COVAR_POP, COVAR_SAMP, **CREATE**, CROSS, CUBE, CUME_DIST, CURRENT, CURRENT_CATALOG, CURRENT_DATE, CURRENT_PATH, CURRENT_ROLE, CURRENT_ROW, CURRENT_SCHEMA, CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_USER, **CURRENT_USER_IS_DBA**, CURSOR, CYCLE
D	DATA, DATALINK, **DATE**, DAY, DAYS, DBA, DEALLOCATE, DEC, DECIMAL, **DECLARE**, DEFAULT, DEFERRABLE, DEFERRED, DEFINE, **DELETE**, DENSE_RANK, DEPTH, DEREF, DESC, DESCRIBE, DESCRIPTOR, DESTROY, DESTRUCTOR, DETERMINISTIC, DIAGNOSTICS, DICTIONARY, DIGITS, DISCONNECT, DISTINCT, DLNEWCOPY, DLPREVIOUSCOPY, DLURLCOMPLETE, DLURLCOMPLETEONLY, DLURLCOMPLETEWRITE, DLURLPATH, DLURLPATHONLY, DLURLPATHWRITE, DLURLSCHEME, DLURLSERVER, DLVALUE, DO, DOMAIN, DOUBLE, **DROP**, DYNAMIC
E	EACH, ELEMENT, ELSE, ELSEIF, END, END_FRAME, END_PARTITION, END-EXEC, EQUALS, ESCAPE, EVERY, EXCEPT, EXCEPTION, EXCLUSIVE, EXEC, **EXECUTE**, EXISTS, EXIT, EXP, EXTERNAL, EXTRACT
F	**FALSE**, **FETCH**, FILTER, FIRST, FIRST_VALUE, FIX, FLAT, FLOAT, FLOOR, FOR, FOREIGN, FOUND, FRAME_ROW, FREE, FROM, FULL, FUNCTION, FUSION
G	GENERAL, GET, GLOBAL, GO, GOTO, **GRANT**, GROUP, GROUPING, GROUPS

頭文字	予約語
H	HANDLER, HASH, HAVING, HEX, HOLD, HOST, HOUR, HOURS
I	IDENTIFIED, IDENTITY, IF, IGNORE, IMMEDIATE, IMPORT, IN, INCREMENTAL, INDEX, INDICATOR, INITIAL, INITIALIZE, INITIALLY, INNER, INOUT, INPUT, INSENSITIVE, **INSERT**, INT, INTEGER, INTERSECT, INTERSECTION, INTERVAL, INTO, IS, ISOLATION, ITERATE
J	JAR, JOIN
K	KEY
L	LAG, LANGUAGE, LARGE, LAST, LAST_VALUE, LATERAL, LEAD, LEADING, LEAVE, LEFT, LENGTH, LESS, LEVEL, LIKE, LIKE_REGEX, LIMIT, LIST, LISTAGG, LN, LOCAL, LOCALTIME, LOCALTIMESTAMP, LOCATOR, LOCK, LONG, LOOP, LOWER
M	MAP, MATCH, MATCH_NUMBER, MATCH_RECOGNIZE, MAX, MAXIMAL, MCHAR, MEASURES, MEMBER, MERGE, METHOD, MICROSECOND, MICROSECONDS, MILLISECOND, MILLISECONDS, MIN, MINUTE, MINUTES, MOD, MODE, MODIFIES, MODIFY, MODULE, MONTH, MONTHS, MULTISET, MVARCHAR
N	NAMES, NANOSECOND, NANOSECONDS, NATIONAL, NATURAL, NCHAR, NCLOB, NESTING, NEW, NEXT, NO, NONE, NORMALIZE, NOT, NOWAIT, NTH_VALUE, NTILE, **NULL**, NULLIF, NUMERIC, NVARCHAR
O	OBJECT, OCCURRENCES_REGEX, OCTET_LENGTH, OF, **OFF**, OFFSET, OLD, **ON**, ONE, ONLY, OPEN, OPERATION, OPTIMIZE, OPTION, OR, ORDER, ORDINALITY, OUT, OUTER, OUTPUT, OVER, OVERLAPS, OVERLAY
P	PAD, PAGE, PARAMETER, PARAMETERS, PARTIAL, PARTITION, PARTITIONED, PATH, PATTERN, PCTFREE, PER, PERCENT, PERCENT_RANK, PERCENTILE_CONT, PERCENTILE_DISC, PERIOD, PICOSECOND, PICOSECONDS, PORTION, POSITION, POSITION_REGEX, POSTFIX, POWER, PRECISION, PREFIX, PREORDER, **PREPARE**, PRESERVE, PRIMARY, PRIOR, PRIVATE, PRIVILEGES, PROCEDURE, PROGRAM, PROTECTED, PUBLIC, **PURGE**
R	RANGE, RANK, READ, READS, REAL, RECOVERY, RECURSIVE, REDO, REF, REFERENCES, REFERENCING, REGR_AVGX, REGR_AVGY, REGR_COUNT, REGR_INTERCEPT, REGR_R2, REGR_SLOPE, REGR_SXX, REGR_SXY, REGR_SYY, RELATIVE, RELEASE, REPEAT, RESIGNAL, RESTRICT, RESULT, RETURN, RETURNS, **REVOKE**, RIGHT, ROLE, **ROLLBACK**, ROLLUP, ROUTINE, ROW, ROW_NUMBER, ROWID, ROWS
S	SAVEPOINT, SCHEMA, SCOPE, SCROLL, SEARCH, SECOND, SECONDS, SECTION, SEEK, **SELECT**, SENSITIVE, SEQUENCE, SESSION, SESSION_USER, SET, SETS, SHARE, SIGNAL, SIMILAR, SIZE, SKIP, SMALLFLT, SMALLINT, SOME, SPACE, SPECIFIC, SPECIFICTYPE, SQL, SQLCODE, SQLCODE_OF_LAST_CONDITION, SQLCOUNT, SQLERRM_OF_LAST_CONDITION, SQLERROR, SQLEXCEPTION, SQLSTATE, SQLWARNING, SQRT, START, STATE, STATEMENT, STATIC, STDDEV_POP, STDDEV_SAMP, STRUCTURE, SUBMULTISET, SUBSE, SUBSTR, SUBSTRING, SUBSTRING_REGEX, SUM, SUPPRESS, SYMMETRIC, SYSTEM, SYSTEM_TIME, SYSTEM_USER
T	TABLE, TABLESAMPLE, TEMPORARY, TERMINATE, TEST, THAN, THEN, **TIME**, **TIMESTAMP**, TIMESTAMP_FORMAT, TIMEZONE_HOUR, TIMEZONE_MINUTE, TO, TRAILING, TRANSACTION, TRANSLATE, TRANSLATE_REGEX, TRANSLATION, TREAT, TRIGGER, TRIM, TRIM_ARRAY, **TRUE**, **TRUNCATE**, TYPE
U	UESCAPE, UNDER, UNDO, UNION, UNIQUE, **UNKNOWN**, UNNEST, UNTIL, **UPDATE**, UPPER, USAGE, USER, USING
V	VALUE, VALUE_OF, VALUES, VAR_POP, VAR_SAMP, VARBINARY, VARCHAR, VARCHAR_FORMAT, VARIABLE, VARYING, VERSIONING, VIEW
W	WAIT, WHEN, WHENEVER, WHERE, WHILE, WIDTH_BUCKET, WINDOW, **WITH**, WITHIN, WITHOUT, WORK, WRITE
X	XLIKE, XML, XMLAGG, XMLATTRIBUTES, XMLBINARY, XMLCAST, XMLCOMMENT, XMLCONCAT, XMLDOCUMENT, XMLELEMENT, XMLEXISTS, XMLFOREST, XMLITERATE, XMLNAMESPACES, XMLPARSE, XMLPI, XMLQUERY, XMLSERIALIZE, XMLTABLE, XMLTEXT, XMLVALIDATE

頭文字	予約語
Y	YEAR，YEARS
Z	ZONE

注

- 下線の予約語は，SQL92 で規定されている予約語です。
- 網掛けが付いている予約語（色が付いている予約語）は，「6.10.2　名前が予約語と重複した場合の対応」の「(2)　重複した予約語を削除する」の方法で削除できない予約語です。

6.10.2　名前が予約語と重複した場合の対応

名前が予約語と重複した場合は，次に示すどちらかの方法で対応してください。

- 名前を二重引用符で囲む
- 予約語を削除する

基本的には，名前を二重引用符（”）で囲むように SQL 文を修正してください。作業量の問題などで SQL 文を修正できないときは，予約語を削除してください。

(1)　名前を二重引用符で囲む

予約語と同じ文字列を二重引用符（”）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。

(2)　重複した予約語を削除する

予約語を削除すると，削除した予約語は二重引用符（”）で囲まないで名前に指定できるようになります。ただし，削除できない予約語もあるため注意してください。削除できない予約語については，「6.10.1　予約語の一覧」を参照してください。

重要

予約語を削除すると，削除した予約語を含む SQL 文が実行できなくなります。

なお，表の定義後に予約語を削除した場合は，定義時の予約語が有効となります。そのため，表の定義後に予約語を削除すると，再定義時にエラーとなることがあります。

予約語を削除するには，サーバ定義のadb_sql_prep_delrsvd_words オペランドに削除する予約語を指定してください。adb_sql_prep_delrsvd_words オペランドについては，マニュアル『HADB　システム構築・運用ガイド』の『サーバ定義の設計』の『サーバ定義のオペランドの内容』の『SQL 文に関するオペランド（set 形式)』を参照してください。

サーバ定義のadb_sql_prep_delrsvd_words オペランドで指定した予約語の削除を，実行する AP によって無効にしたい場合は，クライアント定義のadb_sql_prep_delrsvd_use_srvdef オペランドにN を指定してください。adb_sql_prep_delrsvd_use_srvdef オペランドについては，マニュアル『HADB　AP 開発ガイド』の『クライアント定義の設計』の『クライアント定義のオペランドの内容』の『SQL に関するオペランド』を参照してください。

なお，サーバ定義のadb_sql_prep_delrsvd_words オペランドに指定した削除対象の予約語は，HADB サーバに接続する認可識別子にも適用されます。例えば，adb_sql_prep_delrsvd_words オペランドにABS を指定すると，adbsql コマンドの-u オプションにABS を指定して，認可識別子ABS で HADB サーバに接続することができます。

7 構成要素

この章では，SQL の構成要素について説明します。

7.1　問合せ式

ここでは，問合せ式について説明します。

7.1.1　問合せ式の指定形式および規則

問合せ式には，WITH 句と問合せ式本体の組み合わせを指定します。

問合せ式本体には，問合せ指定，または 2 つの問合せ式本体によって導出される表の集合（和集合，差集合，または積集合）を求める集合演算を指定します。和集合を求める場合はUNION を，差集合を求める場合はEXCEPT を，積集合を求める場合はINTERSECT を指定します。

WITH 句を指定した場合，問合せ式本体の結果によって得られる導出表に問合せ名を付けて，問合せ式本体に指定できます。

また，WITH リスト要素に指定した問合せ名を，そのWITH リスト要素内の問合せ式本体から参照できます(再帰的な検索ができます)。このとき，WITH リスト要素に指定した問合せ名を**再帰的問合せ名**といい，WITH リスト要素に指定した問合せ式本体を**再帰的問合せ**といいます。

(1)　指定形式

```
問合せ式 : :=〔WITH句〕 問合せ式本体


  WITH句 : :=WITH WITHリスト要素〔,WITHリスト要素〕…
    WITHリスト要素 : :=問合せ名 〔(WITH列リスト)〕 AS (問合せ式本体 〔LIMIT句〕) 〔最大再帰
数指定〕
      WITH列リスト : :=列名〔,列名〕…
      最大再帰数指定 : :=/*>> MAX RECURSION 最大再帰数 <<*/

  問合せ式本体 : := {問合せ項
                    |問合せ式本体 {UNION | EXCEPT} 〔 {ALL | DISTINCT} 〕 〔集合演算方式指
定〕 問合せ項}
    問合せ項 : := {問合せ一次子
                  |問合せ項 INTERSECT 〔 {ALL | DISTINCT} 〕 問合せ一次子}
      問合せ一次子 : := {問合せ指定 | (問合せ式本体)}
    集合演算方式指定 : :=/*>> SET OPERATION NOT BY HASH <<*/
```

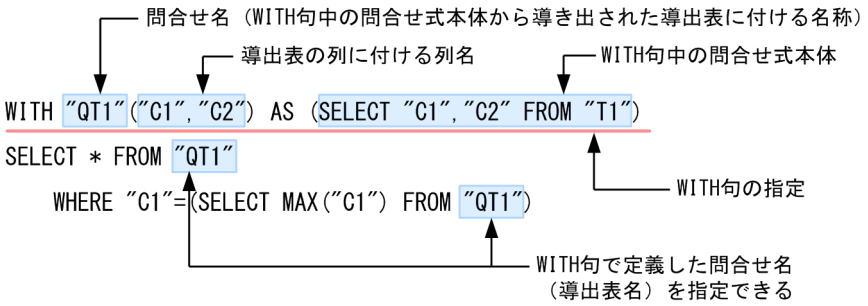
(2)　指定形式の説明

(a)　WITH 句

```
WITH句 : :=WITH  WITHリスト要素〔,WITHリスト要素〕…
  WITHリスト要素 : :=問合せ名 〔(WITH列リスト)〕 AS (問合せ式本体 〔LIMIT句〕) 〔最大再帰数
指定〕
    WITH列リスト : :=列名〔,列名〕…
    最大再帰数指定 : :=/*>> MAX RECURSION 最大再帰数 <<*/
```

AS（*問合せ式本体*）の結果を一時的な導出表として定義する場合に指定します。WITH 句の指定例を次の図に示します。

図 7-1　WITH 句の指定例



問合せ名：

導出表の名称を指定します。ここで指定した名称が問合せ名として定義されます。WITH 句中で，同じ問合せ名を指定することはできません。

WITH 列リスト：

問合せ名（導出表）の各列の列名を指定します。

*WITH 列リスト*に指定する列名の数は，対応する「AS（*問合せ式本体*）」に指定した問合せ式本体のうち，最も外側の問合せの結果によって導出される列数と同じにしてください。

*WITH 列リスト*を省略した場合は，対応する「AS（*問合せ式本体*）」に指定した問合せ式本体のうち，最も外側の問合せの結果によって導出される列の列名が，*問合せ名*の列の列名となります。導出される列名の規則については，「6.9　導出列名」を参照してください。

注意事項を次に示します。

- *WITH 列リスト*中に，同じ列名を指定できません。
- *WITH 列リスト*を省略した場合，問合せ式本体によって導出される列名が重複しないようにしてください。
- HADB によって自動的に設定される導出列名と重複する可能性があるため，*WITH 列リスト*の列名に EXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。
- 対応する「AS（*問合せ式本体*）」に指定した問合せ式本体のうち，最も外側の問合せの結果によって導出される列の数が 4,000 以下になるようにしてください。

AS（*問合せ式本体*　〔*LIMIT 句*〕）：

問合せ式本体を指定します。

ここで指定した問合せ式本体から導出表が作成されます。*問合せ名*に指定した名称が，導出表の名称になります。

なお，問合せ式本体中には，？パラメタを指定できません。

LIMIT 句：

問合せ式本体の検索結果として取得する行数の最大値を指定します。

LIMIT 句については，「7.9　LIMIT 句」を参照してください。

なお，再帰的問合せに対しては，LIMIT 句を指定できません。

最大再帰数指定：

```
最大再帰数指定 ： ：＝/*>> MAX RECURSION 最大再帰数 <<*/
```

再帰的問合せをする際の最大再帰数を指定します。最大再帰数には，再帰可能な回数の上限を符号なし整数定数の形式で指定します。指定規則を次に示します。

- 最大再帰数指定を省略した場合，最大再帰数に100 が仮定されます。
- 最大再帰数には，0～32,767 の符号なし整数定数を指定してください。
- 再帰数が最大再帰数の値を超えた場合，SQL 文がエラーになります。
- 最大再帰数に0 を指定した場合，再帰可能な回数は無制限になります。そのため，0 を指定した場合，SQL 文の実行が無限に繰り返されるおそれがあります。
- 再帰的問合せが指定されていない場合，最大再帰数指定の指定は無効になります。

(b) 問合せ式本体

```
問合せ式本体 ： ：＝ {問合せ項
                    |問合せ式本体 {UNION | EXCEPT} 〔 {ALL | DISTINCT} 〕 〔集合演算方式指定〕
 問合せ項}
  問合せ項 ： ：＝ {問合せ一次子 | 問合せ項 INTERSECT 〔 {ALL | DISTINCT} 〕 問合せ一次子}
    問合せ一次子 ： ：＝ {問合せ指定 | (問合せ式本体)}
  集合演算方式指定 ： ：＝/*>> SET OPERATION NOT BY HASH <<*/
```

*問合せ式本体*には，次のどちらかを指定します。

- *問合せ項*
- *問合せ式本体*と*問合せ項*によって導出される表の集合（和集合または差集合）を求める集合演算

なお，UNION ALL，UNION DISTINCT，EXCEPT ALL，EXCEPT DISTINCT，INTERSECT ALL，およびINTERSECT DISTINCT を**集合演算子**といいます。

問合せ項：

*問合せ項*には，次のどちらかを指定します。

- *問合せ一次子*
- *問合せ項*と*問合せ一次子*の積集合を求める集合演算

{UNION | EXCEPT} ：

和集合を求める場合はUNION を，差集合を求める場合はEXCEPT を指定します。

{ALL | DISTINCT} ：

集合演算の結果に重複した行がある場合，重複した行を排除するかどうかを指定します。

ALL：集合演算の結果に重複した行があっても，重複した行を排除しません。

DISTINCT：集合演算の集合演算項や，集合演算の結果に重複した行がある場合，重複した行を排除して1 つの行にします。

ALL およびDISTINCT の指定を省略した場合，DISTINCT が仮定されます。

問合せ一次子：

*問合せ一次子*には，*問合せ指定*，または(*問合せ式本体*)を指定します。

INTERSECT：

積集合を求めるときの指定です。

問合せ指定：

問合せ指定を指定します。問合せ指定については，「7.2　問合せ指定」を参照してください。

(*問合せ式本体*)：

問合せ式本体を指定します。

集合演算方式指定：

集合演算方式指定を指定した場合，集合演算の処理方式にハッシュ実行以外の方式が適用されます。集合演算の処理方式については，マニュアル『HADB　AP 開発ガイド』の『集合演算の処理方式』を参照してください。

なお，通常は，この指定をする必要はありません。集合演算方式指定を省略した場合，HADB が集合演算の処理方式を決定します。

集合演算方式指定の規則を次に示します。

- 集合演算EXCEPT に集合演算方式指定を指定しても，その指定は無視されます。
- 集合演算方式指定は，問合せ式本体中のすべての集合演算に対して（UNION DISTINCT またはUNION ALL に対して）適用されます。集合演算方式指定が適用されたかどうかは，アクセスパス情報で確認できます。アクセスパス情報の詳細については，マニュアル『HADB　AP 開発ガイド』の『集合演算方式指定』を参照してください。

(例)

```
SELECT ”C1” FROM ”T1”
UNION                                          ...1
SELECT ”C1” FROM ”T2”
UNION /*>> SET OPERATION NOT BY HASH <<*/      ...2
SELECT ”C1” FROM ”T3”
```

上記の例のように，2．の集合演算に集合演算方式指定を指定した場合，1．の下線部分の集合演算にも集合演算方式指定が適用されます。

(3)　規則

(a)　WITH 句の規則

1. 問合せ名が 1 つの場合，問合せ名の有効範囲は，WITH 句の後ろの問合せ式本体です。問合せ名が 2 つ以上の場合，問合せ名ごとに有効範囲が異なります。問合せ名の有効範囲の例については，「6.8.3　範囲変数の有効範囲」の「(1)　SELECT 文のFROM 句に指定した範囲変数の有効範囲」を参照してください。

2. WITH リスト要素の問合せ式本体中に副問合せを入れ子で指定する場合，入れ子の数は 31 回以内にしてください。また，FROM 句に指定した表がビュー表または問合せ名の場合，ビュー表または問合せ名に内部導出表を適用したあとの副問合せの入れ子の数が 31 回以内になるようにしてください。詳細については，「7.3.1　副問合せの指定形式および規則」の「(4)　規則」の「(a)　副問合せ共通の規則」を参照してください。

(例 1)

```
WITH "Q1" AS
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM "T1") AS DT32
  ) AS DT31 ) AS DT30 ) AS DT29 ) AS DT28 ) AS DT27 ) AS DT26 ) AS DT25 ) AS DT24
  ) AS DT23 ) AS DT22 ) AS DT21 ) AS DT20 ) AS DT19 ) AS DT18 ) AS DT17 ) AS DT16
  ) AS DT15 ) AS DT14 ) AS DT13 ) AS DT12 ) AS DT11 ) AS DT10 ) AS DT9 ) AS DT8
  ) AS DT7 ) AS DT6 ) AS DT5 ) AS DT4 ) AS DT3 ) AS DT2 ) AS DT1 ) AS DT0 )
SELECT * FROM "Q1"
```

上記の例の場合，問合せ名Q1 の副問合せの入れ子の数は 32 回となり，上限を超えているため，SELECT 文がエラーになります。

なお，T1 は実表とします。

(例 2)

```
WITH "Q2" AS
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
  (SELECT * FROM (SELECT * FROM "T1") AS DT30
  ) AS DT29 ) AS DT28 ) AS DT27 ) AS DT26 ) AS DT25 ) AS DT24 ) AS DT23 ) AS DT22
  ) AS DT21 ) AS DT20 ) AS DT19 ) AS DT18 ) AS DT17 ) AS DT16 ) AS DT15 ) AS DT14
  ) AS DT13 ) AS DT12 ) AS DT11 ) AS DT10 ) AS DT9 ) AS DT8 ) AS DT7 ) AS DT6
  ) AS DT5 ) AS DT4 ) AS DT3 ) AS DT2 ) AS DT1 ) AS DT0 ),
  "Q3" AS (SELECT * FROM "Q2"),
  "Q4" AS (SELECT * FROM "Q3")
SELECT * FROM "Q4"
```

- 問合せ名Q2 の副問合せの入れ子の数は 30 回で，上限を超えていません。
- 問合せ名Q3 の副問合せの入れ子の数は，内部導出表が適用された結果，31 回となりますが，上限を超えていません。
- 問合せ名Q4 の副問合せの入れ子の数は，内部導出表が適用された結果，32 回となります。上限を超えているため，SELECT 文がエラーになります。

なお，T1 は実表とします。

(例 3)

```
WITH "Q5" AS (SELECT "C1" FROM (SELECT "C1" FROM "T1") AS DT
              UNION ALL
              SELECT "C1"+1 FROM "Q5" WHERE "C1"+1 < 5),
     "Q6" AS (SELECT * FROM
              (SELECT * FROM
               (SELECT * FROM
                (SELECT * FROM
                 (SELECT * FROM
                  (SELECT * FROM "Q5") AS DT4
                 ) AS DT3
                ) AS DT2
               ) AS DT1
              ) AS DT0),
      "Q7" AS
      (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
      (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
      (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
      (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
      (SELECT * FROM (SELECT * FROM (SELECT * FROM (SELECT * FROM
      (SELECT * FROM "Q6") AS DT23
      ) AS DT22 ) AS DT21 ) AS DT20 ) AS DT19 ) AS DT18
      ) AS DT17 ) AS DT16 ) AS DT15 ) AS DT14 ) AS DT13
      ) AS DT12 ) AS DT11 ) AS DT10 ) AS DT9 ) AS DT8
      ) AS DT7 ) AS DT6 ) AS DT5 ) AS DT4 ) AS DT3
      ) AS DT2 ) AS DT1 ) AS DT0 )
SELECT * FROM "Q7"
```

- 問合せ名Q5 の副問合せの入れ子の数は 1 回で，上限を超えていません。
- 問合せ名Q6 の副問合せの入れ子の数は，内部導出表が適用された結果，7 回となりますが，上限を超えていません。
- 問合せ名Q7 の副問合せの入れ子の数は，内部導出表が適用された結果，32 回となります。上限を超えているため，SELECT 文がエラーになります。

なお，T1 は実表とします。

(b) 再帰的問合せの規則

1. 再帰的問合せには，その再帰的問合せを参照する再帰的問合せ名を含まない問合せ指定※と，再帰的問合せを参照する再帰的問合せ名を含む問合せ指定※を，それぞれ 1 つ以上指定する必要があります。このとき，再帰的問合せを参照する再帰的問合せ名を含まない問合せ指定※を**アンカーメンバ**といい，再帰的問合せを参照する再帰的問合せ名を含む問合せ指定※を**再帰的メンバ**といいます。

   注※　副問合せを除きます。

2. 再帰的メンバを複数指定する場合は，各再帰的メンバを集合演算（UNION ALL）の演算項となるように指定してください。

3. 再帰的問合せ中で最後に指定するアンカーメンバと最初に指定する再帰的メンバは，集合演算（UNION ALL）の演算項となるように指定してください。アンカーメンバと再帰的メンバの指定例を次に示します。

   （例）

```
WITH "Q1"("C1","C2")
    AS (SELECT "C1","C2" FROM "T1" WHERE "C1" > 0                    ...1
```

```
        UNION ALL
        SELECT "Q1"."C1"+1,"T1"."C2" FROM "Q1","T1" WHERE "Q1"."C1" < 5)   ...2
SELECT * FROM "Q1"
```

[説明]

1. 下線部分がアンカーメンバです。
2. 下線部分が再帰的メンバです。

4. すべてのアンカーメンバは，再帰的問合せ中で最初に指定した再帰的メンバよりも前に指定してください。
5. 再帰的問合せの再帰的メンバのFROM 句に，その再帰的問合せを参照する再帰的問合せ名を 2 つ以上指定できません。
6. 再帰的問合せ中の副問合せに，その再帰的問合せを参照する再帰的問合せ名は指定できません。
7. 再帰的メンバ中には，次の指定はできません。また，再帰的メンバ中の副問合せにも，次の指定はできません。
   - SELECT DISTINCT
   - GROUP BY 句
   - HAVING 句
   - LIMIT 句
   - 集合関数
   - LEFT OUTER JOIN
   - RIGHT OUTER JOIN
   - FULL OUTER JOIN

   ただし，次の個所には上記の指定ができます。
   - 再帰的メンバ中に指定したビュー表
   - 再帰的メンバ中の副問合せに指定したビュー表
   - 問合せ名から導出される導出問合せ
8. 次に示す 2 つの項目のデータ型とデータ長を同じにしてください。
   - 再帰的問合せ中に指定したすべての再帰的メンバを集合演算した結果から導出される表の各構成列※1 のデータ型とデータ長
   - すべてのアンカーメンバを集合演算した結果から導出される表の各構成列※2 のデータ型とデータ長

   注※1　再帰的メンバが 1 つの場合は，再帰的メンバの選択式が該当します。

   注※2　アンカーメンバが 1 つの場合は，アンカーメンバの選択式が該当します。
9. 次に示す条件下での再帰的問合せの検索の流れについて説明します。
   - 再帰的問合せ中に指定したすべてのアンカーメンバを集合演算した問合せ式本体をQ_0 とする
   - Q_0 の検索結果をX_0 とする

- 再帰的問合せ中に指定したすべての再帰的メンバを集合演算した問合せ式本体をQi とする
- Qi の検索結果をXi とする
- 再帰数をi とする

再帰的問合せの検索の流れ

1. Q_0 を検索する（検索結果はX_0）
2. 検索結果X_0 を再帰的問合せの結果とする
3. 前回の検索結果X_{i-1} を基にQi を検索する（検索結果はXi）
4. 検索結果Xi によって次のどちらかの処理を行う
   - 検索結果Xi が空行でない場合は，検索結果Xi を再帰的問合せの結果として 3.に戻る
   - 検索結果Xi が空行の場合は，再帰的問合せを終了する

(例)

```
WITH "REC"("VAL") AS (
    SELECT * FROM (VALUES(1))                          ←アンカーメンバ
    UNION ALL
    SELECT "VAL" + 1 FROM "REC" WHERE "VAL" + 1 <= 5   ←再帰的メンバ
    )
SELECT "VAL" FROM "REC"
```

上記の SQL 文の実行結果の例

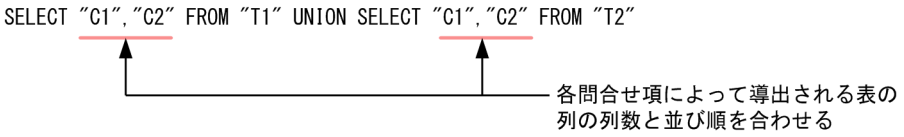
VAL		
1	←$X_0 = Q_0( )$	アンカーメンバ Q_0
2	←$X_1 = Q_1(X_0)$	再帰的メンバ Q_1(再帰1回目)
3	←$X_2 = Q_2(X_1)$	再帰的メンバ Q_2(再帰2回目)
4	←$X_3 = Q_3(X_2)$	再帰的メンバ Q_3(再帰3回目)
5	←$X_4 = Q_4(X_3)$	再帰的メンバ Q_4(再帰4回目)

上記の SQL 文の場合，再帰数は 4 回になります。

(c) 集合演算の規則

1. 集合演算の評価順序は，括弧内，INTERSECT，UNION またはEXCEPT の順になります。
2. 集合演算の対象となる*問合せ項と問合せ項*，*問合せ項と問合せ一次子*，または*問合せ一次子と問合せ一次子*によって導出される表を，それぞれ行の集合と見なし，集合演算が実行されます。
3. 集合演算の対象となる表（各問合せ項によって導出された表）の列数と並び順が，同じである必要があります。

   (例)



また，対応する列のデータ型は，比較できるデータ型である必要があります。上記の例の場合，表T1のC1 列と表T2 のC1 列，表T1 のC2 列と表T2 のC2 列が比較できるデータ型である必要があります。

比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

ただし，次のデータは，集合演算の場合は比較できません。

- DATE 型のデータと，日付を表す既定の入力表現の文字データ
- TIME 型のデータと，時刻を表す既定の入力表現の文字データ
- TIMESTAMP 型のデータと，時刻印を表す既定の入力表現の文字データ

既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。

4. 集合演算によって導出される表の列名は，集合演算に指定した問合せ項によって導出される表の列名によって決まります。集合演算によって導出される表の列名規則については，「6.9.2　問合せの結果の導出列名の決定規則」の「(1)　問合せ式の場合」を参照してください。
5. 集合演算によって導出される表の列数と並び順は，集合演算の対象とした表（各問合せ項によって導出された表）の構成列と同じになります。なお，対応する列が 1 つでも非ナル値制約なし（ナル値を許す）の場合は，導出された表のすべての列が非ナル値制約なし（ナル値を許す）として集合演算が実行されます。
6. 集合演算によって導出される表の列のデータ型とデータ長は，集合演算の対象とした表（各問合せ項によって導出された表）の対応する構成列のデータ型およびデータ長によって決まります。詳細については，「7.21.2　値式の結果のデータ型」を参照してください。
7. *Q1* および *Q2* を集合演算の集合演算項とします。この場合，「*Q1 集合演算 Q2*」の結果の行数は，次の表のようになります。

表 7-1　集合演算の結果の行数

集合演算の指定	集合演算の結果の行数	
	ALL の指定がない場合	ALL の指定がある場合
UNION	• 0（$m = 0$ かつ $n = 0$ の場合） • 1（$m > 0$ または $n > 0$ の場合）	$m + n$
EXCEPT	• 0（$m = 0$ または $n > 0$ の場合） • 1（$m > 0$ かつ $n = 0$ の場合）	MAX（$m - n$，0）
INTERSECT	• 0（$m = 0$ または $n = 0$ の場合） • 1（$m > 0$ かつ $n > 0$ の場合）	MIN（m，n）

注

ある行 *R* が *Q1* 中に含まれている行数を m，*Q2* 中に含まれている行数を n としています。

8. 問合せ式本体を指定した SQL 文中に指定している集合演算がすべてUNION の場合，指定できる集合演算の数は最大 1,023 個になります。ただし，指定した集合演算にEXCEPT またはINTERSECT がある場合は，指定できる集合演算の数は最大 63 個になります。

なお，SQL 文中にビュー表を指定している場合は，CREATE VIEW 文で指定した問合せ式に基づいた内部導出表が適用されます。集合演算の最大数の規則は，この内部導出表に対して適用されます。

9. 問合せ式本体を指定した SQL 文中に指定できる外結合（FULL OUTER JOIN）の数は，最大 63 個になります。

10. DISTINCT の指定がある集合演算中に指定された集合演算は，DISTINCT の指定があるものとして処理されることがあります。

11. 集合演算の集合演算項に指定した問合せ指定の選択式の値式には，配列データを指定できません。

(4) 例題

例題 1（WITH 句の例）

販売履歴表（SALESLIST）と商品表（PRODUCTSLIST）から，売り上げ金額の最大値（QMAXSALES）を商品名（PUR-NAME）ごとに求めます。

```
WITH "QT1"("QCODE","QMAXSALES") AS
    (SELECT "PUR-CODE",MAX("PRICE"*"QUANTITY") FROM "SALESLIST"
        GROUP BY "PUR-CODE")
SELECT "PUR-NAME","QMAXSALES" FROM "QT1"
    INNER JOIN "PRODUCTSLIST" ON "QCODE"="PUR-CODE"
```

下線部分がWITH 句の指定です。

例題 2（集合演算の和集合の例）

支店 A の販売履歴表（SALESLIST_A）と支店 B の販売履歴表（SALESLIST_B）から，支店 A と支店 B のすべての販売履歴を取得します。

```
SELECT "A"."USERID","A"."PUR-CODE","A"."PUR-NUM"
    FROM "SALESLIST_A" "A"
UNION ALL
SELECT "B"."USERID","B"."PUR-CODE","B"."PUR-NUM"
    FROM "SALESLIST_B" "B"
```

■支店Aの販売履歴表（SALESLIST_A）

USERID	PUR-CODE	PUR-NUM
U00555	P001	1
U00555	P003	2
U00358	P001	3
U00614	P001	1

■支店Bの販売履歴表（SALESLIST_B）

USERID	PUR-CODE	PUR-NUM
U00358	P002	6
U00212	P005	2
U00614	P001	1

■検索結果

USERID	PUR-CODE	PUR-NUM
U00555	P001	1
U00555	P003	2
U00358	P001	3
U00614	P001	1
U00358	P002	6
U00212	P005	2
U00614	P001	1

例題３（集合演算の和集合の例）

支店Ａの販売履歴表（SALESLIST_A）と支店Ｂの販売履歴表（SALESLIST_B）から，支店Ａと支店Ｂの少なくとも一方で買い物をしたことがある顧客の顧客ID（USERID）を取得します。

```
SELECT "A"."USERID"
    FROM "SALESLIST_A" "A"
UNION DISTINCT
SELECT "B"."USERID"
    FROM "SALESLIST_B" "B"
```

■支店Aの販売履歴表（SALESLIST_A）

USERID	PUR-CODE	PUR-NUM
U00555	P001	1
U00555	P003	2
U00358	P001	3
U00614	P001	1

■支店Bの販売履歴表（SALESLIST_B）

USERID	PUR-CODE	PUR-NUM
U00358	P002	6
U00212	P005	2
U00614	P001	1

■検索結果

USERID
U00212
U00358
U00555
U00614

例題４（集合演算の差集合の例）

支店Ａの販売履歴表（SALESLIST_A）と支店Ｂの販売履歴表（SALESLIST_B）から，支店Ａでは買い物をしたことがあるが，支店Ｂでは買い物をしたことがない顧客の顧客ID（USERID）を取得します。

```
SELECT "A"."USERID"
    FROM "SALESLIST_A" "A"
EXCEPT
SELECT "B"."USERID"
    FROM "SALESLIST_B" "B"
```

■支店Aの販売履歴表（SALESLIST_A）

USERID	PUR-CODE	PUR-NUM
U00555	P001	1
U00555	P003	2
U00358	P001	3
U00614	P001	1

■支店Bの販売履歴表（SALESLIST_B）

USERID	PUR-CODE	PUR-NUM
U00358	P002	6
U00212	P005	2
U00614	P001	1

■検索結果

USERID
U00555

例題５（集合演算の積集合の例）

支店Ａの販売履歴表（SALESLIST_A）と支店Ｂの販売履歴表（SALESLIST_B）から，支店Ａと支店Ｂの両方で買い物をしたことがある顧客の顧客ID（USERID）を取得します。

```
SELECT "A"."USERID"
    FROM "SALESLIST_A" "A"
```

```
INTERSECT
SELECT "B"."USERID"
    FROM "SALESLIST_B" "B"
```

■支店Aの販売履歴表（SALESLIST_A）

USERID	PUR-CODE	PUR-NUM
U00555	P001	1
U00555	P003	2
U00358	P001	3
U00614	P001	1

■支店Bの販売履歴表（SALESLIST_B）

USERID	PUR-CODE	PUR-NUM
U00358	P002	6
U00212	P005	2
U00614	P001	1

■検索結果

USERID
U00358
U00614

例題６（再帰的問合せの例）

部品Parts_B を製造するために必要な部品のうち，在庫がない部品を求めます。

```
WITH "V1"("ID","PARENT","NAME","QUANTITY") AS (
  SELECT "A"."ID","A"."PARENT","A"."NAME","A"."QUANTITY"
      FROM "BOMS" "A" WHERE "A"."ID"=2
  UNION ALL
  SELECT "A"."ID","A"."PARENT","A"."NAME","A"."QUANTITY"
      FROM "V1", "BOMS" "A" WHERE "A"."PARENT" = "V1"."ID"
  )
SELECT "NAME","QUANTITY" FROM "V1" WHERE "QUANTITY"=0
```

■検索結果

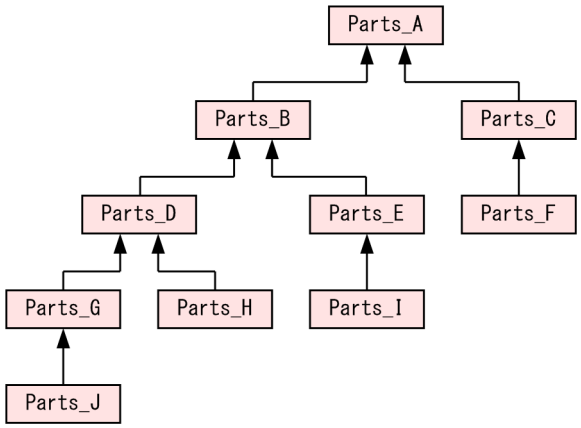
NAME	QUANTITY
Parts_D	0
Parts_E	0

部品の在庫数などが格納されている部品表（BOMS）の構成と，部品の階層構造は次のとおりです。

■部品表（BOMS）

ID （部品ID）	PARENT （親ID）	NAME （部品名）	QUANTITY （在庫数）
1	NULL	Parts_A	1
2	1	Parts_B	150
3	1	Parts_C	250
4	2	Parts_D	0
5	2	Parts_E	0
6	3	Parts_F	0
7	4	Parts_G	11
8	4	Parts_H	15000
9	5	Parts_I	2000
10	7	Parts_J	100

■部品の階層構造



(5) 留意事項

1. 集合演算を指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。
2. 集合演算の処理方式にハッシュ実行が適用された場合，適切な大きさのハッシュテーブル領域が必要になります。ハッシュテーブル領域の大きさは，サーバ定義またはクライアント定義の adb_sql_exe_hashtbl_area_size オペランドで指定します。集合演算の処理方式については，マニュアル『HADB　AP 開発ガイド』の『集合演算の処理方式』を参照してください。
3. 集合演算の処理方式にハッシュ実行が適用された場合，導出表が作成されます。導出表の相関名は，HADB が次の名称規則に従って決定します。

   ```
   ##DRVTBL_xxxxxxxxxx
   ```

   xxxxxxxxxx は，10 桁の整数です。
4. 次の述語を B-tree インデクスを使用して評価する場合，表副問合せに指定された集合演算は，DISTINCT の指定があるものとして処理されることがあります。

- 表副問合せが指定されたIN 述語
- 限定述語（=ANY 指定または=SOME 指定）

7.2　問合せ指定

ここでは，問合せ指定について説明します。

7.2.1　問合せ指定の指定形式および規則

問合せ指定には，検索結果として出力する項目（選択リスト）と表の検索条件（表式）を指定します。

(1)　指定形式

```
問合せ指定：：＝SELECT 〔 {ALL｜DISTINCT} 〕〔SELECT重複排除方式指定〕 選択リスト 表式

  SELECT重複排除方式指定：：＝/*>> SELECT DISTINCT NOT BY HASH <<*/
  選択リスト：：＝ {*｜選択式〔,選択式〕…}
    選択式：：＝ {値式 〔AS句〕 ｜NULL 〔AS句〕 ｜表指定.*｜〔表指定.〕ROW}
      AS句：：＝〔AS〕 列名
```

(2)　指定形式の説明

(a)　{ALL ｜ DISTINCT}

検索結果に重複行があった場合，検索結果から重複行を排除するかどうかを指定します。

ALL：

検索結果に重複行があっても，そのまま出力します。

DISTINCT：

検索結果に重複行があった場合，重複を排除して検索結果を出力します。

DISTINCT を指定した場合と指定しない場合の検索結果の違いについては，「1.10.1　例（商品を購入した顧客を検索する）」を参照してください。

注意事項を次に示します。

- DISTINCT を指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。
- SELECT DISTINCT の処理方式にハッシュ実行が適用された場合，適切な大きさのハッシュテーブル領域が必要になります。ハッシュテーブル領域の大きさはサーバ定義またはクライアント定義の adb_sql_exe_hashtbl_area_size オペランドで指定します。SELECT DISTINCT の処理方式については，マニュアル『HADB　AP 開発ガイド』の『SELECT DISTINCT の処理方式』を参照してください。

ALL およびDISTINCT の指定を省略した場合，ALL が仮定されます。

(b)　SELECT 重複排除方式指定

SELECT 重複排除方式指定を指定した場合，SELECT DISTINCT の処理方式にハッシュ実行以外の処理方式が適用されます。SELECT DISTINCT の処理方式については，マニュアル『HADB　AP 開発ガイド』の『SELECT DISTINCT の処理方式』を参照してください。

なお，通常は，この指定をする必要はありません。SELECT 重複排除方式指定を省略した場合，HADB が SELECT DISTINCT の処理方式を決定します。

(c)　選択リスト

選択リスト：：＝｛*｜*選択式*〔,*選択式*〕…｝

選択リストには，検索結果として出力する項目を指定します。

＊：

表のすべての列を検索結果として出力する場合に指定します。

*を指定した場合，FROM 句に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したと見なされます。各表中の列の順序は，表定義時に指定した順序となります。

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。

(d)　表式

検索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），グループ分け，およびグループを選択する条件が指定できます。表式の詳細については，「7.4　表式」を参照してください。

(e)　選択式

選択式：：＝｛*値式*　〔*AS句*〕｜NULL　〔*AS句*〕｜*表指定*.＊｜〔*表指定*.〕ROW｝
　AS句：：＝〔AS〕　*列名*

選択式には，検索結果として出力する項目を指定します。

選択式中に外への参照列を指定することはできません。外への参照列については，「7.3.1　副問合せの指定形式および規則」の「(4)　規則」の「(a)　副問合せ共通の規則」を参照してください。

値式　〔*AS 句*〕：

検索結果として出力する項目を値式の形式で指定します。

検索結果の列名を変更する場合はAS 句を指定します。

検索結果の列名および列順については，「6.9　導出列名」を参照してください。

指定規則を次に示します。

- 最初の問合せ指定（SELECT 文のWITH 句中の問合せ指定を除く）の場合，AS 句中の列名に半角括弧（左括弧または右括弧）を指定できます。2 番目以降の問合せ指定の場合は，AS 句中の列名に半角括弧を指定できません。
- HADB によって自動的に設定される導出列名と重複する可能性があるため，AS 句の列名に EXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。
- DISTINCT を指定する場合，選択式の値式には配列データを指定できません。

■GROUP BY 句，HAVING 句，または集合関数を指定する場合の注意事項

GROUP BY 句，HAVING 句，または集合関数を指定する場合，選択式中の値式に指定する列指定は，次のどれかの条件を満たす必要があります。

- グループ化列名を指定する

  （例）正しい SQL 文の例

```
SELECT "C1" FROM "T1" GROUP BY "C1" HAVING "C1">100
```

  上記の例では，GROUP BY 句に指定したグループ化列名を，選択式中の値式に指定しています。

- 集合関数の引数に指定する

  （例）正しい SQL 文の例

```
SELECT COUNT("C2") FROM "T1" HAVING MAX("C1")>100
```

  上記の例では，集合関数の引数に列指定を指定しています。

  （例）エラーになる SQL 文の例

```
SELECT COUNT("C1")+"C1" FROM "T1"
```

  上記の例では，集合関数の引数以外の個所に列指定を指定しています。

- グループ化指定に指定した値式（列指定を含む値式）と同じ値式を指定する

  （例）正しい SQL 文の例

```
SELECT "C1"+"C2" FROM "T1" GROUP BY "C1"+"C2"
```

  上記の例では，GROUP BY 句のグループ化指定に指定した値式（列指定を含む値式）と同じ値式を，選択式中の値式に指定しています。

  なお，選択式に表指定ありの列指定を指定した場合，グループ化列名に同じ名称の列名があっても，そのグループ化列は参照できません。

  （例）エラーになる SQL 文の例

```
SELECT "T1"."C2" FROM "T1" GROUP BY "C1"+1 AS "C2"
```

  上記の例では，選択式に表指定ありの列指定（"T1."C2"）を指定しているため，グループ化列名に同じ列名の「"C1"+1 AS "C2"」があっても参照できません。そのため，上記の SQL 文はエラーになります。

NULL 〔*AS 句*〕：

検索結果にナル値を出力する場合に指定します。

検索結果に列名を付与する場合はAS 句を指定します。

指定規則を次に示します。

- SELECT 文の最も外側の問合せ指定の選択式にだけNULL を指定できます。
- 集合演算の対象となる問合せ指定の選択式にはNULL を指定できません。

  （例）エラーになる SQL 文の例

```
SELECT NULL FROM ”T1” UNION SELECT ”C1” FROM ”T1”
```

- WITH 句中の問合せ指定の選択式にはNULL を指定できません。

検索結果については，次の規則が適用されます。

- NULL の結果のデータ型はINTEGER 型になります。
- NULL の結果は，非ナル値制約なし（ナル値を許す）となります。

表指定.*：

この指定をした場合，指定した表の全列が検索結果として出力されます。検索結果の列順は，指定した表の列順と同じになります。

この指定をした場合，その問合せ指定中にGROUP BY 句，HAVING 句，または集合関数を指定したときは，選択式中に含まれる列指定は次のようにしてください。

- グループ化列

〔*表指定*.〕ROW：

この指定をした場合，行全体を 1 つのデータとして 1 つの領域に取り出します。ROW は行全体を意味しています。

取り出されるデータに指定する変数は，各列のデータ型に関係なく，ROW に対応するCHAR 型の変数を指定します。ただし，変数中に境界調整による空きがないようにしてください。また，データ長は行長（各列のデータ長の合計）になります。行長の計算方法については，マニュアル『HADB　システム構築・運用ガイド』の『行の種別ごとの格納ページ数の求め方』の計算式『ROWSZ』を参照してください。

ROW を指定する場合の規則を次に示します。

- FIX 表に対してだけ指定できます。
- ROW を指定する場合，その問合せ指定中に次の指定はできません。
  - ・集合関数
  - ・ウィンドウ関数
  - ・GROUP BY 句
  - ・SELECT DISTINCT
  - ・集合演算
- FROM 句に結合表のLEFT OUTER JOIN を指定した場合，右側の表にはROW を指定できません。
- FROM 句に結合表のRIGHT OUTER JOIN を指定した場合，左側の表にはROW を指定できません。
- FROM 句に結合表のFULL OUTER JOIN を指定した場合，両側の表にはROW を指定できません。

- WITH 句中の問合せ指定にROW を指定できません。

(3) 実行時に必要な権限

問合せ指定を実行する場合，問合せ指定中で指定する表に対するSELECT 権限が必要になります。

(4) 規則

1. 問合せ指定の検索結果の列の数は，4,000 以下にしてください。
2. 選択式に*表指定*.*，または*表指定*.ROW を指定する場合，その選択式を有効範囲に含む範囲変数と等価な表指定としてください。
3. 問合せ指定中にアーカイブマルチチャンク表を指定する場合，探索条件の指定に注意する必要があります。詳細については，マニュアル『HADB　AP 開発ガイド』の『アーカイブマルチチャンク表を検索する際の考慮点』を参照してください。問合せ指定中にアーカイブマルチチャンク表を指定する場合は，必ず参照してください。
4. 次の述語を B-tree インデクスを使用して評価する場合，表副問合せに指定された問合せ指定は，SELECT DISTINCT の指定があるものとして処理されることがあります。
   - 表副問合せが指定されたIN 述語
   - 限定述語（=ANY 指定または=SOME 指定）
5. 集合演算にDISTINCT を指定した場合，その集合演算中の問合せ指定は，SELECT DISTINCT の指定があるものとして処理されることがあります。

(5) 例題

問合せ指定の指定例を例題を使って説明します。

例題

販売履歴表（SALESLIST）から，2011/9/3 以降の商品コード（PUR-CODE）ごとの販売個数の合計値，平均値を求めます。その際，販売個数の合計値が 20 個以下の商品コードだけを検索対象にします。

```
SELECT "PUR-CODE",SUM("PUR-NUM"),AVG("PUR-NUM")        ...1
    FROM "SALESLIST"                                   ...2
        WHERE "PUR-DATE">=DATE'2011-09-03'             ...2
        GROUP BY "PUR-CODE"                            ...2
        HAVING SUM("PUR-NUM")<=20                      ...2
```

説明

この例では，SELECT 文全体が問合せ指定です。

1. 下線部分が選択リストの指定です。
2. 下線部分が表式の指定です。

7.3 副問合せ

ここでは，副問合せについて説明します。

7.3.1 副問合せの指定形式および規則

内側の問合せ指定を**副問合せ**といいます。副問合せには，次の 2 種類があります。

- スカラ副問合せ
  問合せの結果の列数が 1 列で，行数が 1 行以下の副問合せのことです。
- 表副問合せ
  問合せの結果の列数が 1 列以上で，行数が 0 行以上の副問合せのことです。

(1) 指定形式

```
副問合せ ： ：＝(〔副問合せ処理方式指定〕 問合せ式本体 〔LIMIT句〕)

  副問合せ処理方式指定 ： ：＝/*>> SUBQUERY NOT BY HASH〔副問合せ処理委譲指定〕<<*/
    副問合せ処理委譲指定 ： ： = (DELEGATION)
```

(2) 指定形式の説明

副問合せ処理方式指定：

副問合せ処理方式指定を指定した場合，副問合せの処理方式にハッシュ実行以外の方式が適用されます。副問合せの処理方式については，マニュアル『HADB　AP 開発ガイド』の『副問合せの処理方式』を参照してください。

なお，通常は，この指定をする必要はありません。副問合せ処理方式指定を省略した場合，HADB が副問合せの処理方式を決定します。

また，副問合せ処理方式指定は次の個所に指定できません。

- 導出表の表副問合せ
- 問合せによるマルチ集合値構成子の表副問合せ

副問合せ処理委譲指定：

副問合せ処理委譲指定を指定した場合，ほかの処理で使用する SQL 処理リアルスレッドを，外への参照列を含む副問合せの検索処理に振り分けることができます。

なお，通常は，この指定をする必要はありません。外への参照列を含む副問合せの結果を得るために多くの検索処理が必要な場合に，副問合せ処理委譲指定を指定することを検討してください。

(例)

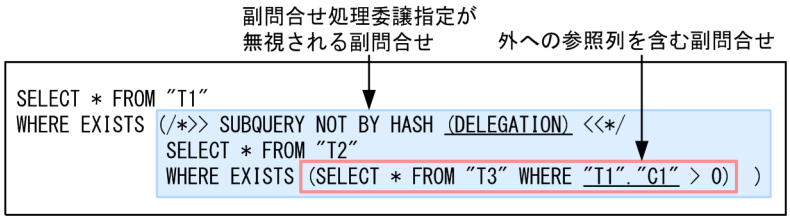
```
SELECT COUNT(*) FROM "T1" WHERE "T1"."C1" = ANY(
  SELECT "T2"."C1" FROM "T2" WHERE "T1"."C2" = "T2"."C2")
```

上記の SQL 文の場合，表T1 の結果 1 件ごとに表T2 の検索が行われます。探索条件「"T1"."C2" = "T2"."C2"」を満たす表T2 の行数が多い場合，副問合せ処理委譲指定を指定すると，検索性能が向上することがあります。ただし，副問合せ処理委譲指定を指定した場合，外への参照列を含む副問合せの検索処理をほかの SQL 処理リアルスレッドに振り分ける分のオーバヘッドが掛かります。そのため，検索条件によっては，検索性能が低下するおそれもあります。

副問合せ処理委譲指定に関する規則を次に示します。

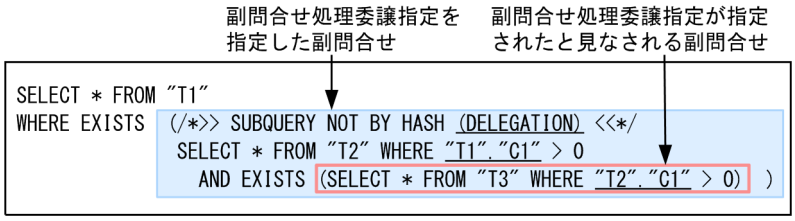
- 次のどちらかの条件を満たす副問合せに副問合せ処理委譲指定を指定した場合，その副問合せ処理委譲指定は無視されます。
  - 外への参照列を含まない副問合せの場合
  - 副問合せを指定した SQL 文に非順序実行方式が適用されない場合

  （例）副問合せ処理委譲指定が無視される例

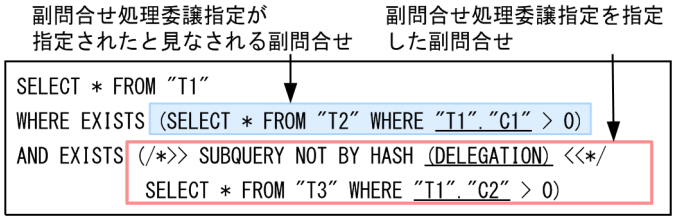


  上記の例では，外への参照列を含まない副問合せに，副問合せ処理委譲指定を指定していますが，外への参照列を含む副問合せには，副問合せ処理委譲指定を指定していません。この場合，外への参照列を含む副問合せに，副問合せ処理委譲指定が指定されていないため，副問合せ処理委譲指定が無視されます。

- 外への参照列を含む副問合せを入れ子で指定した場合，どれか 1 つでも副問合せ処理委譲指定を指定していると，入れ子中のすべての外への参照列を含む副問合せに，副問合せ処理委譲指定が指定されたと見なされます。外への参照列を含む副問合せを入れ子で指定している例を次に示します。この例では，外への参照列を含む副問合せが 2 つ指定されていて，片方の副問合せにだけ副問合せ処理委譲指定が指定されています。この場合，2 つの副問合せに対して，副問合せ処理委譲指定が指定されたと見なされます。



- 外への参照列を含む副問合せに副問合せ処理委譲指定を指定した場合，同じ表を外への参照列として参照する副問合せにも，副問合せ処理委譲指定が指定されたと見なされます。例を次に示します。この例では，同じ表を外への参照列として参照する副問合せが 2 つ指定されていて，片方の副問合せにだけ副問合せ処理委譲指定が指定されています。この場合，2 つの副問合せに対して，副問合せ処理委譲指定が指定されたと見なされます。



問合せ式本体：

*問合せ式本体*の詳細については，「7.1.1　問合せ式の指定形式および規則」の「(2)　指定形式の説明」を参照してください。

LIMIT 句：

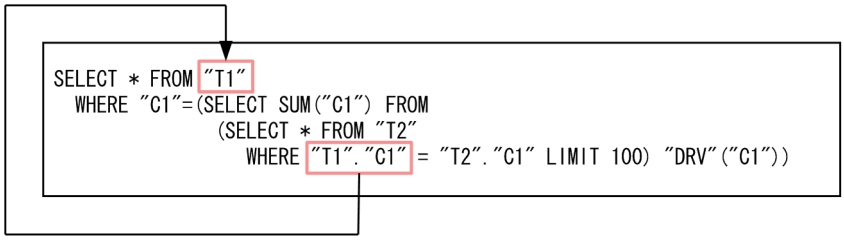
問合せ式本体の検索結果として取得する行数の最大値を指定します。

LIMIT 句については，「7.9　LIMIT 句」を参照してください。

なお，LIMIT 句を指定できるのは，導出表の表副問合せか，またはスカラ副問合せの場合だけです。ただし，次の場合は，導出表であっても，LIMIT 句を指定できません。

- LIMIT 句を指定した導出表を超えて，外への参照を行っている導出表

  （例）エラーになる SQL 文の例



この場合，"T1"."C1"が，LIMIT 句を指定した導出表（相関名：DRV）を超えて外への参照を行っています。そのため，LIMIT 句を指定することはできません。

導出表については，「7.11.1　表参照の指定形式」を参照してください。

(3)　実行時に必要な権限

副問合せを実行する場合，副問合せ中で参照する表に対するSELECT 権限が必要になります。

(4)　規則

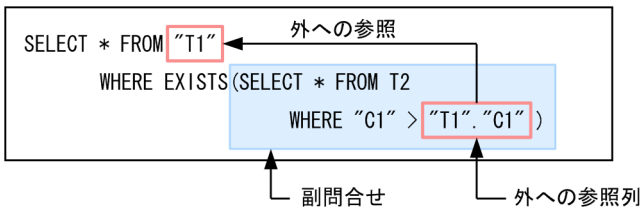
(a)　副問合せ共通の規則

1. 副問合せの結果のデータ型は，問合せ式本体の結果のデータ型と同じになります。
2. 副問合せの結果，導出される表の列名については，「6.9　導出列名」を参照してください。
3. 副問合せ中の選択式に次の指定はできません。
   - 外への参照列
   - 〔*表指定*.〕ROW

■外への参照列

外側の問合せのFROM 句に指定した表を，副問合せ中の探索条件で参照している場合，これを**外への参照**といいます。また，参照している列を**外への参照列**といいます。外への参照列の例を次の図に示します。

図 7-2　外への参照列の例



4. 副問合せを 32 回（ビュー定義およびWITH 句の場合は 31 回）まで入れ子で指定できます。さらに次の規則があります。

- FROM 句に指定した表がビュー表または問合せ名の場合

  ビュー表または問合せ名に内部導出表を適用したあとの副問合せの入れ子の数が 32 回（ビュー定義およびWITH 句の場合は 31 回）以内となるようにしてください。

- FROM 句に指定した表が，FROM 句に指定した表を含む再帰的問合せを参照する再帰的問合せ名の場合

  再帰的問合せ名に対応する内部導出表を 1 度適用したあとの副問合せの入れ子の数が 32 回（ビュー定義およびWITH 句の場合は 31 回）以内となるようにしてください。

- FROM 句にアーカイブマルチチャンク表を指定した場合

  アーカイブマルチチャンク表は内部導出表に等価変換されます。内部導出表に等価変換されたあとの副問合せの入れ子の数が 33 個以上となった場合，その SQL 文はエラーになります。アーカイブマルチチャンク表に関する等価変換については，マニュアル『HADB　AP 開発ガイド』の『アーカイブマルチチャンク表を検索する SQL 文の等価変換』を参照してください。

- FROM 句に表値構成子によって導出される導出表を指定した場合

  副問合せの入れ子の数が 1 つ加算されます。

副問合せの入れ子の数の数え方の例を次に説明します。

(例 1)

次のSELECT 文は，副問合せを 8 回入れ子にして指定しています。

```
SELECT * FROM "TT" WHERE EXISTS(
    SELECT * FROM "T0" WHERE EXISTS(
    SELECT * FROM "T1" WHERE EXISTS(     ←1つ目の入れ子
    SELECT * FROM "T2" WHERE EXISTS(     ←2つ目の入れ子
    SELECT * FROM "T3" WHERE EXISTS(     ←3つ目の入れ子
    SELECT * FROM "T4" WHERE EXISTS(     ←4つ目の入れ子
    SELECT * FROM "T5" WHERE EXISTS(     ←5つ目の入れ子
    SELECT * FROM "T6" WHERE EXISTS(     ←6つ目の入れ子
    SELECT * FROM "T7" WHERE EXISTS(     ←7つ目の入れ子
    SELECT * FROM "T8"                   ←8つ目の入れ子
    )))))))))
```

(例 2)

次のCREATE VIEW 文は，副問合せを 7 回入れ子にして指定しています。

```
CREATE VIEW ”V1” AS SELECT * FROM ”TT” WHERE EXISTS(
    SELECT * FROM ”T0” WHERE EXISTS(
    SELECT * FROM ”T1” WHERE EXISTS(     ←1つ目の入れ子
    SELECT * FROM ”T2” WHERE EXISTS(     ←2つ目の入れ子
    SELECT * FROM ”T3” WHERE EXISTS(     ←3つ目の入れ子
    SELECT * FROM ”T4” WHERE EXISTS(     ←4つ目の入れ子
    SELECT * FROM ”T5” WHERE EXISTS(     ←5つ目の入れ子
    SELECT * FROM ”T6” WHERE EXISTS(     ←6つ目の入れ子
    SELECT * FROM ”T7”                   ←7つ目の入れ子
    ))))))))
```

次のSELECT 文を実行した場合，内部導出表が適用されて，副問合せの入れ子の数は 8 回になります。

```
SELECT * FROM ”V1”
```

(例 3)

次のSELECT 文を実行した場合，内部導出表が適用されて，副問合せの入れ子の数は 8 回になります。

```
WITH ”Q1” AS (SELECT * FROM ”TT” WHERE EXISTS(
    SELECT * FROM ”T0” WHERE EXISTS(
    SELECT * FROM ”T1” WHERE EXISTS(     ←1つ目の入れ子
    SELECT * FROM ”T2” WHERE EXISTS(     ←2つ目の入れ子
    SELECT * FROM ”T3” WHERE EXISTS(     ←3つ目の入れ子
    SELECT * FROM ”T4” WHERE EXISTS(     ←4つ目の入れ子
    SELECT * FROM ”T5” WHERE EXISTS(     ←5つ目の入れ子
    SELECT * FROM ”T6” WHERE EXISTS(     ←6つ目の入れ子
    SELECT * FROM ”T7”                   ←7つ目の入れ子
    )))))))))
SELECT * FROM ”Q1”                  ←内部導出表が適用された結果，8つ目の入れ子となる
```

(例 4)

次のSELECT 文のFROM 句には，再帰的問合せ名Q1 が指定されています。次のSELECT 文を実行した場合，内部導出表が適用されて，副問合せの入れ子の数は 8 回になります。

実行する SQL 文：

```
WITH ”Q1” AS (SELECT ”C1” FROM ”TT”
              UNION ALL
              SELECT ”C1”+1 FROM ”Q1” WHERE ”C1”+1 < 5)
    SELECT * FROM ”TT” WHERE EXISTS(
    SELECT * FROM ”T0” WHERE EXISTS(
    SELECT * FROM ”T1” WHERE EXISTS(
    SELECT * FROM ”T2” WHERE EXISTS(
    SELECT * FROM ”T3” WHERE EXISTS(
    SELECT * FROM ”T4” WHERE EXISTS(
    SELECT * FROM ”T5” WHERE EXISTS(
    SELECT * FROM ”Q1”)))))))
```

内部導出表が適用されたあとの SQL 文：

```
SELECT * FROM ”TT” WHERE EXISTS(
SELECT * FROM ”T0” WHERE EXISTS(
SELECT * FROM ”T1” WHERE EXISTS(     ←1つ目の入れ子
```

```
SELECT * FROM "T2" WHERE EXISTS(     ←2つ目の入れ子
SELECT * FROM "T3" WHERE EXISTS(     ←3つ目の入れ子
SELECT * FROM "T4" WHERE EXISTS(     ←4つ目の入れ子
SELECT * FROM "T5" WHERE EXISTS(     ←5つ目の入れ子
SELECT * FROM                        ←6つ目の入れ子
              (SELECT "C1" FROM "TT"                          ←7つ目の入れ子
               UNION ALL
               SELECT "C1"+1 FROM
                                 (SELECT "C1" FROM "TT"   ←8つ目の入れ子
                                  UNION ALL
                                  SELECT "C1"+1 FROM "Q1" WHERE "C1"+1 < 5)"Q1"
               WHERE "C1"+1 < 5)"Q1")))))))
```

(例 5)

次のSELECT 文は，副問合せを 7 回入れ子にして指定しています。また，FROM 句に表値構成子によって導出される導出表を指定しているため，副問合せの入れ子の数が 1 つ加算されます。したがって，合計 8 回の入れ子が指定されていると見なされます。

```
SELECT * FROM "TT" WHERE EXISTS(
    SELECT * FROM "T0" WHERE EXISTS(
    SELECT * FROM "T1" WHERE EXISTS(      ←1つ目の入れ子
    SELECT * FROM "T2" WHERE EXISTS(      ←2つ目の入れ子
    SELECT * FROM "T3" WHERE EXISTS(      ←3つ目の入れ子
    SELECT * FROM "T4" WHERE EXISTS(      ←4つ目の入れ子
    SELECT * FROM "T5" WHERE EXISTS(      ←5つ目の入れ子
    SELECT * FROM "T6" WHERE EXISTS(      ←6つ目の入れ子
    SELECT * FROM (VALUES (1,2,3)) "T7"   ←7～8つ目の入れ子
    ))))))))
```

5. 集合関数中に副問合せは指定できません。
6. ウィンドウ関数中に副問合せは指定できません。
7. GROUP BY 句のグループ化指定に副問合せは指定できません。
8. FULL OUTER JOIN を指定した結合表のON 探索条件に，副問合せは指定できません。
9. 行値構成子要素中に副問合せは指定できません。
10. FULL OUTER JOIN を指定した結合表中の表参照を参照先とする，外への参照を行う列は指定できません。

（例）下線の部分が誤っている外への参照列です。

```
SELECT * FROM ("T1" LEFT OUTER JOIN "T2" ON "T1"."C1"="T2"."C1")
                    FULL OUTER JOIN "T3" ON "T1"."C2"="T3"."C2"
    WHERE "T1"."C3">(SELECT MAX(C3) FROM "T4"
                        WHERE "C1"="T1"."C1"
                          AND "C2"="T3"."C2")
```

(b) スカラ副問合せの規則

1. スカラ副問合せの結果の列数は 1 にしてください。
2. スカラ副問合せの結果の行数は 1 以下にしてください。結果の行数が 2 以上の場合，SQL エラーとなります。
3. スカラ副問合せの結果の行数が 0 の場合，その結果はナル値となります。

4. スカラ副問合せの結果は，非ナル値制約なし（ナル値を許す）となります。

5. スカラ副問合せの選択式の値式には，配列データを指定できません。

(c) 表副問合せの規則

1. 表副問合せの結果の列数の最大値を次に示します。
   - 導出表に指定した表副問合せの場合：4,000
   - IN 述語，および限定述語の右側に指定した表副問合せの場合：1
   - EXISTS 述語に指定した表副問合せの場合：4,000

(5) 例題

例題 1

最も給与が高い社員の名前（NAME），給与（SAL）を検索します。

```
SELECT "NAME","SAL"
    FROM "SALARYLIST"
       WHERE "SAL"=(SELECT MAX("SAL") FROM "SALARYLIST")
```

下線部分が副問合せの指定です。

例題 2

全社員の平均給与より，平均給与が高い部門（SCODE）の平均給与額を検索します。

```
SELECT "SCODE",AVG("SAL")
    FROM "SALARYLIST"
      GROUP BY "SCODE"
      HAVING AVG("SAL")>(SELECT AVG("SAL") FROM "SALARYLIST")
```

下線部分が副問合せの指定です。

例題 3

販売履歴表（SALESLIST）から任意の 100 行を取得し，その取得結果に対して商品コード（PUR-CODE）ごとに，販売個数（PUR-NUM）の合計値を求めます。

```
SELECT "PUR-CODE",SUM("PUR-NUM")
    FROM (SELECT * FROM "SALESLIST" LIMIT 100) "SALESLIST"
        GROUP BY "PUR-CODE"
```

下線部分が副問合せの指定です。

(6) 留意事項

1. 副問合せを指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。

2. 副問合せの処理方式にハッシュ実行が適用された場合，適切な大きさのハッシュテーブル領域が必要になります。ハッシュテーブル領域の大きさはサーバ定義またはクライアント定義の adb_sql_exe_hashtbl_area_size オペランドで指定します。また，副問合せの処理方式にハッシュ実行が適用された場合，ハッシュフィルタを格納するハッシュフィルタ領域も必要になります。ハッシュフィルタ領域の大きさは，サーバ定義またはクライアント定義のadb_sql_exe_hashflt_area_size オペランドで指定します。副問合せの処理方式については，マニュアル『HADB　AP 開発ガイド』の『副問合せの処理方式』を参照してください。

7.4　表式

ここでは，表式について説明します。

7.4.1　表式の指定形式および規則

表式とは，FROM 句，WHERE 句，GROUP BY 句，およびHAVING 句の総称です。表式は問合せ指定中に指定します。

(1)　指定形式

```
表式 : :=FROM句
        〔WHERE句〕
        〔GROUP BY句〕
        〔HAVING句〕
```

(2)　指定形式の説明

FROM 句：

FROM 句には，検索対象となる表を指定します。FROM 句の詳細については，「7.5　FROM 句」を参照してください。

WHERE 句：

WHERE 句には探索条件を指定します。WHERE 句の詳細については，「7.6　WHERE 句」を参照してください。

GROUP BY 句：

グループごとに検索データを集計する場合にGROUP BY 句を指定します。GROUP BY 句の詳細については，「7.7　GROUP BY 句」を参照してください。

HAVING 句：

GROUP BY 句によってグループごとに集計されたデータの抽出条件を指定する場合にHAVING 句を指定します。HAVING 句の詳細については，「7.8　HAVING 句」を参照してください。

(3)　規則

1. 表式の結果の列を列指定として参照することができます。
2. WHERE 句，GROUP BY 句，およびHAVING 句すべてを省略した場合，表式の結果はFROM 句の結果となります。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ます。表式の結果は，最後に指定した句の結果となります。

   次に示すSELECT 文を実行する場合を例にして具体的に説明します。

   (例)

販売履歴表（SALESLIST）から，2011/9/3 以降の商品コード（PUR-CODE）ごとの販売個数の合計値を求めます。その際，販売個数の合計値が 20 個以下の商品コードだけを検索対象にします。

```
SELECT "PUR-CODE",SUM("PUR-NUM")
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-03'
        GROUP BY "PUR-CODE"
        HAVING SUM("PUR-NUM")<=20
```

説明

下線部分が表式の指定です。上記のSELECT 文が実行された場合，次に示す流れで表式の結果が導き出されます。

1．FROM 句の結果がWHERE 句に適用されます。その結果，SALESLIST 表中のPUR-DATE 列が 2011-09-03 以降のデータ（PUR-CODE 列のデータとPUR-NUM 列のデータ）が抽出されます。

2．1 の抽出結果がGROUP BY 句に適用されてグループ化されます。その結果，PUR-CODE ごとに，1 の抽出結果のデータが集計されます。

3．2 の集計結果がHAVING 句に適用されます。その結果，PUR-NUM 列の値の合計が 20 以下のデータだけが集計されます。この集計結果が表式の結果となります。

(4) 例題

表式の指定例を例題を使って説明します。

例題

販売履歴表（SALESLIST）から，2011/9/6 以降に商品コードP002 の商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-06'
        AND "PUR-CODE"='P002'
```

下線部分が表式の指定です。

7.5 FROM 句

ここでは，FROM 句について説明します。

7.5.1 FROM 句の指定形式および規則

FROM 句には，検索対象とする表を指定します。

(1) 指定形式

FROM句 ：：＝FROM *表参照*〔,*表参照*〕…

(2) 指定形式の説明

表参照：

検索対象となる表を表参照の形式で指定します。表参照の詳細については，「7.11 表参照」を参照してください。

複数の表参照を指定した問合せ（FROM 句に表名，問合せ名，導出表，表関数導出表，および集まり導出表を複数含む問合せ）を実行する場合，これを**表の結合**といいます。

また，複数の表参照をコンマ（ , ）で区切って指定する結合を**コンマ結合**といいます。

(3) 規則

1. FROM 句に指定するすべての表参照中に指定できる表名，問合せ名，導出表，表関数導出表，および集まり導出表の数は，最大 64 個になります。表名の指定数の算出方法を次に示します。
   - 表参照に表名を指定した場合：1
   - 表参照に導出表を指定した場合：1
   - 表参照に結合表を指定した場合：結合表に指定した表名，および導出表の合計数
   - WITH 句に指定した問合せ名を指定した場合：1
   - 表参照に表関数導出表を指定した場合：1
   - 表参照に集まり導出表を指定した場合：1

   表名の指定数の算出例を次に示します。

   （例）

```
WITH "Q1" AS (SELECT * FROM "T6","T7")
SELECT * FROM "T1",                                          ...[a]
    "T2" LEFT OUTER JOIN "T3" ON "T2"."C1"="T3"."C1",        ...[b]
    (SELECT * FROM "T4","T5") "W1",                          ...[c]
    "Q1",                                                    ...[d]
    TABLE(ADB_CSVREAD(MULTISET['/dir/file.csv.gz'],
                      'COMPRESSION_FORMAT=GZIP;'))
```

```
    AS "W2" ("C1" INT),                                    ...[e]
    UNNEST("T1"."C1") AS "W3" ("C1")                       ...[f]
```

[説明]

a. 表名（T1）を指定しているため，表名の数は 1 つとなります。

b. 結合表を指定しているため，表名の数は結合表に指定した表名（T2 およびT3）の合計で 2 つとなります。

c. 導出表（W1）を指定しているため，導出表の数は 1 つとなります。

d. WITH 句に指定した問合せ名（Q1）を指定しているため，問合せ名の数は 1 つとなります。

e. 表関数導出表を指定しているため，1 と計算します。

f. 集まり導出表を指定しているため，1 と計算します。

したがって，上記の例の場合，すべての表参照に指定された表名の指定数は，合計 7 になります。

2. FROM 句の結果の列の記述子は，FROM 句に指定した表の列の記述子と同じになります。また，FROM 句の結果の列の順序は，FROM 句に指定した表の列の順序になります。次に示すSELECT 文を実行した場合を例にして具体的に説明します。

(例)

```
SELECT * FROM "T1","T2"
```

表T1 にC1 列とC2 列が，表T2 にC3 列とC4 列が定義されているとします。この場合，FROM 句の結果の列の順序は，C1，C2，C3，C4 となります。

> **メモ**
>
> 列の記述子とは，列の名称，データ型，データ長，ナル値の有無，列 ID（先頭の列からの番号）から成る列の属性情報のことです。

(4) 例題

FROM 句の指定例を例題を使って説明します。

例題

販売履歴表（SALESLIST）から，2011/9/6 以降に商品コードP002 の商品を購入した顧客の，顧客 ID (USERID)，商品コード（PUR-CODE)，購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-06'
        AND "PUR-CODE"='P002'
```

下線部分がFROM 句の指定です。

(5) 留意事項

次に示す場合は，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。

- 1 つのFROM 句に，複数の表参照を指定した場合
- 1 つの SQL 文中に，同じビュー表名を複数の個所に指定した場合
- 1 つの SQL 文中に，WITH 句に指定した問合せ名を複数の個所に指定した場合

作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。

7.6　WHERE 句

ここでは，WHERE 句について説明します。

7.6.1　WHERE 句の指定形式

WHERE 句には探索条件を指定します。

(1)　指定形式

WHERE句：：＝WHERE *探索条件*

(2)　指定形式の説明

探索条件：

探索条件の詳細については，「7.19　探索条件」を参照してください。

(3)　例題

WHERE 句の指定例を例題を使って説明します。

例題

販売履歴表（SALESLIST）から，2011/9/6 以降に商品コード P002 の商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-06'
        AND "PUR-CODE"='P002'
```

下線部分がWHERE 句の指定です。

7.7 GROUP BY 句

ここでは，GROUP BY 句について説明します。

7.7.1 GROUP BY 句の指定形式および規則

グループごとに検索データを集計する場合にGROUP BY 句を指定します。

(1) 指定形式

```
GROUP BY句 : : =GROUP BY 〔グループ化方式指定〕 グループ化指定 〔,グループ化指定〕…

  グループ化方式指定 : : =/*>> WITHOUT GLOBAL HASH GROUPING <<*/
  グループ化指定 : : =値式 〔〔AS〕 列名〕
```

(2) 指定形式の説明

●*グループ化方式指定*

```
グループ化方式指定 : : =/*>> WITHOUT GLOBAL HASH GROUPING <<*/
```

*グループ化方式指定*を指定した場合，グループ化の処理方式にグローバルハッシュグループ化が適用されなくなります。

通常は，この指定をする必要はありません。

グループ化の処理方式については，マニュアル『HADB　AP 開発ガイド』の『グループ化の処理方式』を参照してください。

なお，/*>>と<<*/で囲んだ文字列は注釈になりません。グループ化方式指定以外を記述するとエラーになります。

●*グループ化指定*

```
グループ化指定 : : =値式 〔〔AS〕 列名〕
```

検索データを集計する際のグループを値式の形式で指定します。値式については，「7.21　値式」を参照してください。

GROUP BY 句の指定例を次に示します。

（例 1）商品コード（PUR-CODE）ごとにデータを集計する場合

```
GROUP BY "PUR-CODE"
```

（例 2）月単位でデータを集計する場合

```
GROUP BY EXTRACT(MONTH FROM "SALE-DAY") AS "GMONTH"
```

SALE-DAY 列には，商品の販売日がDATE 型の形式で格納されています。スカラ関数EXTRACT を使用して，SALE-DAY 列の月の部分を抽出しています。

〔AS〕 *列名*：

ここで指定した列名がグループ化列名になります。

（例）

```
GROUP BY SUBSTR(”C1”,5,2) AS ”GC1”
```

上記の例の場合，GC1 がグループ化列名になります。

> **メモ**
>
> GROUP BY 句の結果，導出される列を**グループ化列**といいます。また，グループ化列に付けられた列名を**グループ化列名**といいます。

（例 1）

```
GROUP BY ”C1”
```

上記の例の場合，下線部分がグループ化列になり，グループ化列名はC1 になります。

（例 2）

```
GROUP BY ”T1”.”C1”
```

上記の例の場合，下線部分がグループ化列になり，グループ化列名はC1 になります。

（例 3）

```
GROUP BY ”C1” AS ”GC1”
```

上記の例の場合，下線部分がグループ化列になり，グループ化列名はGC1 になります。

（例 4）

```
GROUP BY SUBSTR(”C1”,5,2) AS ”GC1”
```

上記の例の場合，下線部分がグループ化列になり，グループ化列名はGC1 になります。

（例 5）

```
GROUP BY SUBSTR(”C1”,5,2)
```

上記の例の場合，下線部分がグループ化列になり，グループ化列名はなしとなります。

（例 6）

```
GROUP BY ”C1”,”C2”
```

上記の例の場合，下線部分がグループ化列になります（グループ化列が 2 つになります）。グループ化列名はC1 とC2 になります。

（例 7）

```
GROUP BY 1 AS ”GC1”
```

上記の例の場合，下線部分がグループ化列になり，グループ化列名はGC1 になります。

（例 8）

```
GROUP BY 1
```

上記の例の場合，下線部分がグループ化列になり，グループ化列名はなしとなります。

(3) 規則

1. グループ化列の数の上限は 64 個になります。
2. 値式中に集合関数は指定できません。
3. 値式中に副問合せは指定できません。
4. 値式中に？パラメタは指定できません。
5. 値式には配列データを指定できません。
6. ほかのグループ化列に単独で指定した列指定と同じ列名を，「AS *列名*」に指定できません。

   （例）エラーになる指定例

```
GROUP BY "C1","C3" AS "C1"
```

```
GROUP BY "C1" AS "C2","C3" AS "C1"
```

7. HADB によって自動的に設定される導出列名と重複する可能性があるため，グループ化指定の「AS *列名*」の列名にEXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。
8. 「AS *列名*」に指定する列名は，重複して指定できません。

   （例）エラーになる指定例

```
GROUP BY "C1"+1 AS "GC1","C2"+1 AS "GC1"
```

9. 「AS *列名*」に指定した列名を，選択式中の副問合せ，またはHAVING 句中の副問合せから参照することはできません。

   （例）エラーになる指定例

```
SELECT "GC1",MAX("C2") FROM "T1"
    GROUP BY SUBSTR("C1",5,2) AS "GC1"
    HAVING EXISTS(SELECT * FROM "T2" WHERE "T2"."C1"="GC1")
```

10. GROUP BY 句を指定した場合，次に示す項目だけが選択式に指定できます。
    1. グループ化列名
    2. 集合関数
    3. 値指定
    4. スカラ副問合せ
    5. 1.～4.を指定した値式
    6. グループ化指定に指定した値式（列指定を含む値式）と同じ値式

    （例）正しい指定例

```
SELECT "C1","C2",COUNT(*)  ←選択式にグループ化列名と集合関数が指定されている
    FROM "T1"
        GROUP BY "C1","C2"
```

（例）エラーになる指定例

```
SELECT "C1","C2",COUNT(*)  ←選択式にグループ化列名ではないC2列が指定されている
    FROM "T1"
        GROUP BY "C1"
```

11. GROUP BY 句中に指定する列指定は，次の条件を満たす必要があります。

- GROUP BY 句を指定した表式中のFROM 句に指定している表の列を指定する
- 列名は一意に識別できるように指定する

次に示すSELECT 文を実行した場合を例にして具体的に説明します。

（例）

```
SELECT "SALESLIST"."USERID",SUM("PUR-NUM")
    FROM "SALESLIST","USERSLIST"
        WHERE "PUR-CODE"='P002'
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
        GROUP BY "SALESLIST"."USERID"
```

販売履歴表（SALESLIST）と顧客表（USERSLIST）には，同じ列名のUSERID 列があります。このようなケースで，GROUP BY 句にUSERID 列を指定する場合，どちらのUSERID 列のことなのか一意に識別できるように指定する必要があります。そのため，GROUP BY "USERID"とは指定できません。GROUP BY "SALESLIST"."USERID"のように表名で修飾するようにしてください。

12. 選択式の値式中，またはHAVING 句の値式中に指定した列が参照するグループ化列は，次に示す優先順位に従って決定されます。番号が小さいほど優先順位が高くなります（1.がいちばん優先順位が高くなります）。

1. グループ化列名に同じ列名がある場合

（例）正しい指定例

```
SELECT "C1"+"C2" FROM "T1" GROUP BY "C1"+"C2" AS "C1","C2"
```

上記の例の場合，選択式中のC1 は，グループ化列名に同じ名称のC1 があるため，グループ化列「"C1"+"C2" AS "C1"」を参照します。

選択式中のC2 は，グループ化列名に同じ名称のC2 があるため，グループ化列C2 を参照します。

2. 単独の列指定のグループ化列がある場合，または値式の形式が同じであるグループ化列がある場合

（例 1）正しい指定例

```
SELECT "C1"+"C2" FROM "T1" GROUP BY "C1"+"C2" AS "C3"
```

上記の例の場合，選択式中のC1 とC2 は，値式の形式が同じであるグループ化列「"C1"+"C2" AS "C3"」があるため，グループ化列「"C1"+"C2" AS "C3"」を参照します。

（例 2）正しい指定例

```
SELECT "GC1","C1" FROM "T1" GROUP BY "C1" AS "GC1"
```

上記の場合，選択式中のC1 は値式の形式が同じであるグループ化列「"C1" AS "GC1"」を参照します。

また，選択式中のGC1 はグループ化列名に同じ名称のGC1 があるため，上記の指定例は「1. グループ化列名に同じ列名がある場合」の条件にも該当します。そのため，GC1 も，グループ化列「"C1" AS "GC1"」を参照します。

3. グループ化列の指定順（前方優先）

（例）正しい指定例

```
SELECT "C1"+"C2" FROM "T1" GROUP BY "C1"+"C2" AS "C3","C1"+"C2"
```

上記の例の場合，グループ化列の値式の形式が「"C1"+"C2" AS "C3"」と「"C1"+"C2"」で同じになります。この場合，前方優先となるため，選択式中のC1 とC2 は，グループ化列「"C1"+"C2" AS "C3"」を参照します。

■エラーになる指定例

```
SELECT "C1"+"C2" FROM "T1" GROUP BY "C1"+"C2" AS "C1"
```

上記の例の場合，選択式中のC1 は，下線部分のグループ化列名C1 に対応するグループ化列を参照します。一方，選択式中のC2 は，同じ名称のグループ化列名がありません。そのため，上記の SQL 文はエラーになります。

13. グループ化指定の値式に指定した列名と同じ名称を，同じグループ化指定のグループ化列名に指定しないことを推奨します。グループ化指定の値式に指定した列名とグループ化列名が同じ場合，意図しないグループ化列を参照するおそれがあります。

    （例）

    ```
    SELECT "C1"+1 FROM "T1" GROUP BY "C1"+1 AS "C1"
    ```

    上記の例の場合，グループ化指定の値式中にC1 を指定し，グループ化列名にもC1 を指定しています。この場合，選択式中のC1 は，グループ化列「"C1"+1 AS "C1"」を参照します。

14. 選択式中の副問合せ，またはHAVING 句中の副問合せからは，グループ化列として指定した列指定を含む値式を参照できません。

    （例）エラーになる指定例

    ```
    SELECT "T1"."C1"+"T1"."C2" FROM "T1"
        GROUP BY "T1"."C1"+"T1"."C2"
        HAVING (SELECT "T2"."C1" FROM "T2"
                    WHERE "T2"."C1" > "T1"."C1"+"T1"."C2") > 0
    ```

    HAVING 句中の副問合せに指定した「"T1"."C1"+"T1"."C2"」は，グループ化列「"T1"."C1"+"T1"."C2"」を参照できません。そのため，上記の SQL 文はエラーになります。

15. WHERE 句の結果がGROUP BY 句に適用されてグループ化されます。表式の結果が導き出される順序については，「7.4.1　表式の指定形式および規則」の「(3)　規則」を参照してください。

16. 先行するWHERE 句の結果を *T* とします（WHERE 句が指定されていない場合は，先行するFROM 句の結果を *T* とします）。

- GROUP BY 句を指定した場合，集合 *T* が複数のグループに分けられます（グループ化列の値が同じである集合を 1 つのグループとします)。その際，各グループのグループ化列の重複行が排除されるため，GROUP BY 句の結果の行は，作成されたグループ数と同じになります。

  なお，グループ化列にナル値がある場合は，ナル値同士を同じ値として 1 つのグループにします。
- GROUP BY 句を省略して，その問合せ指定中にHAVING 句または集合関数を指定した場合，*T* 全体で構成される 1 つのグループが作成されます。

(4) 例題

例題 1

販売履歴表（SALESLIST）から，顧客ごとの商品購入回数の一覧を求めます。

```
SELECT "USERID",COUNT(*) AS "COUNT"
    FROM "SALESLIST"
        GROUP BY "USERID"
```

下線部分がGROUP BY 句の指定です。

実行結果の例

USERID	COUNT
U00212	4
U00358	5
U00555	1

例題 2

販売履歴表（SALESLIST）から，2011/9/3 以降の商品コード（PUR-CODE）ごとの販売個数の合計値，平均値を求めます。

```
SELECT "PUR-CODE",SUM("PUR-NUM") AS "SUM",AVG("PUR-NUM") AS "AVG"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-03'
        GROUP BY "PUR-CODE"
```

下線部分がGROUP BY 句の指定です。

実行結果の例

PUR-CODE	SUM	AVG
P001	16	5
P002	37	6
P003	17	5

例題 3

販売履歴表（SALESLIST）と顧客表（USERSLIST）から，2011/9/4 以降の商品コードP002 の販売個数（PUR-NUM）の合計値を顧客ごとに求めます。

```
SELECT "NAME",SUM("PUR-NUM") AS "SUM"
    FROM "SALESLIST","USERSLIST"
        WHERE "PUR-DATE">=DATE'2011-09-04'
        AND "PUR-CODE"='P002'
```

```
        AND "SALESLIST"."USERID"="USERSLIST"."USERID"
        GROUP BY "NAME"
```

下線部分がGROUP BY 句の指定です。

実行結果の例

NAME	SUM
Maria Gomez	12
Nancy White	9
Mike Johnson	5

例題 4

社員表（EMPLIST）から，年齢を 10 歳単位のグループに分けて，グループごとの社員数を求めます。60 歳以上は 60 歳のグループに属するようにします。

```
SELECT "GAGE",COUNT(*) AS "COUNT"
    FROM "EMPLIST"
        GROUP BY CASE WHEN "AGE">=60 THEN 60
                      ELSE TRUNC("AGE",-1)
                 END AS "GAGE"
```

下線部分がGROUP BY 句の指定です。

実行結果の例

GAGE	COUNT
20	212
30	375
40	186
50	113
60	35

例題 5

販売履歴表（SALESLIST）から，2013 年の売り上げ金額を月単位で求めます。

- SALE-DAY 列には，商品の販売日がDATE 型の形式で格納されています。
- AMOUNT 列には，顧客が商品を購入した際の購入金額が格納されています。

```
SELECT "GMONTH",SUM("AMOUNT") AS "SUM"
    FROM "SALESLIST"
        WHERE EXTRACT(YEAR FROM "SALE-DAY")=2013
        GROUP BY EXTRACT(MONTH FROM "SALE-DAY") AS "GMONTH"
```

下線部分がGROUP BY 句の指定です。

実行結果の例

GMONTH	SUM
1	302245
2	258764
3	378847
4	402213
5	437022
⋮	⋮

(5) 留意事項

1. GROUP BY 句を指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。
2. グループ化の処理方式にグローバルハッシュグループ化が適用された場合，適切な大きさのハッシュテーブル領域が必要になります。ハッシュテーブル領域の大きさはサーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドで指定します。

   また，グループ化の処理方式にローカルハッシュグループ化が適用された場合，適切な大きさのハッシュグループ化領域が必要になります。ハッシュグループ化領域の大きさはサーバ定義またはクライアント定義のadb_sql_exe_hashgrp_area_size オペランドで指定します。

   グループ化の処理方式については，マニュアル『HADB　AP 開発ガイド』を参照してください。

7.8 HAVING 句

ここでは，HAVING 句について説明します。

7.8.1 HAVING 句の指定形式および規則

先行するGROUP BY 句によってグループ化されたグループの選択条件を，HAVING 句で指定します。

GROUP BY 句の指定がない場合，先行するWHERE 句またはFROM 句の結果を，1 つのグループと見なして処理を行います。

(1) 指定形式

HAVING句 : : = HAVING *探索条件*

(2) 指定形式の説明

探索条件：

探索条件の詳細については，「7.19 探索条件」を参照してください。

(3) 規則

1. HAVING 句の*探索条件*中に指定する列指定は，次に示すどれかの条件を満たす必要があります。

- グループ化列名を指定する

  （例 1）

```
SELECT COUNT(”C2”) FROM ”T1” GROUP BY ”C1” HAVING ”C1”>100
```

  下線部分には，同じ列名（グループ化列名）を指定します。

  （例 2）

```
SELECT ”GC1”,COUNT(*) FROM ”MEMBERS”
    GROUP BY CASE WHEN ”AGE”>=90 THEN 90 ELSE TRUNC(”AGE”,-1) END AS ”GC1”
    HAVING ”GC1”>=20
```

  下線部分には，同じ列名（グループ化列名）を指定します。

- グループ化指定に指定した値式（列指定を含む値式）と同じ値式を指定する

  （例）

```
SELECT ”C1”+”C2” FROM ”T1” GROUP BY ”C1”+”C2” HAVING ”C1”+”C2” > 100
```

  下線部分に同じ値式（列指定を含む値式）を指定する必要があります。

- 集合関数の引数に指定する

（例）

```
SELECT COUNT(”C2”) FROM ”T1” HAVING MAX(”C1”)>100
```

下線部分が集合関数の引数の指定です。

- 外への参照列を指定する

（例）

```
SELECT * FROM ”T1”
    WHERE EXISTS(SELECT * FROM ”T2” HAVING MAX(”C1”)<”T1”.”C1”)
```

下線部分が外への参照列の指定です。

2. *探索条件*中の副問合せ中に含まれ，かつ先行するFROM 句に指定した表参照の列を参照する列指定は，次に示す条件を満たす必要があります。

- 先行するGROUP BY 句に指定した単独の列指定を指定する（*グループ化指定*の「AS *列名*」の指定有無は関係ありません）

（例）

```
SELECT ”C1” FROM ”T1”
    GROUP BY ”C1”
    HAVING EXISTS(SELECT * FROM ”T2” WHERE ”C1”<”T1”.”C1”)
```

下線部分には，先行するGROUP BY 句のグループ化列に指定した単独の列指定と同じ列指定を指定します。

- 集合関数の引数に指定する

（例）

```
SELECT COUNT(”C1”) FROM ”T1”
    HAVING EXISTS(SELECT * FROM ”T2” WHERE ”C1”<MAX(”T1”.”C1”))
```

下線部分が外への参照列を引数に指定した集合関数です。

3. GROUP BY 句の結果に対して，HAVING 句で指定した探索条件が適用されます。表式の結果が導き出される順序については，「7.4.1　表式の指定形式および規則」の「(3)　規則」を参照してください。

(4)　例題

例題 1

販売履歴表（SALESLIST）から，2011/9/3 以降の商品コード（PUR-CODE）ごとの販売個数の合計値，平均値を求めます。

その際，販売個数の合計値が 20 個以下の商品コードのグループだけを検索対象にします。

```
SELECT ”PUR-CODE”,SUM(”PUR-NUM”) AS ”SUM”,AVG(”PUR-NUM”) AS ”AVG”
    FROM ”SALESLIST”
        WHERE ”PUR-DATE”>=DATE’2011-09-03’
        GROUP BY ”PUR-CODE”
        HAVING SUM(”PUR-NUM”)<=20
```

下線部分がHAVING 句の指定です。

実行結果の例

PUR-CODE	SUM	AVG
P001	16	5
P003	17	5

例題 2

全社員の平均年齢より，平均年齢が低い部門（SCODE）とその平均年齢を求めます。

```
SELECT "SCODE",AVG("AGE") AS "AVG"
    FROM "EMPLIST"
       GROUP BY "SCODE"
       HAVING AVG("AGE")<(SELECT AVG("AGE") FROM "EMPLIST")
```

下線部分がHAVING 句の指定です。

実行結果の例

SCODE	AVG
S001	28
S003	26

7.9 LIMIT 句

ここでは，LIMIT 句について説明します。

7.9.1 LIMIT 句の指定形式および規則

LIMIT 句には，問合せ式または問合せ式本体の検索結果として取得する行数の上限を指定します。

LIMIT 句は次の個所に指定できます。

- SELECT 文の最も外側の問合せ指定または問合せ式本体
- 導出表※
  ただし，再帰的メンバ中の導出表にはLIMIT 句を指定できません。
- スカラ副問合せ
  ただし，再帰的メンバ中のスカラ副問合せにはLIMIT 句を指定できません。
- WITH 句のWITH リスト要素
  ただし，再帰的問合せに対応するWITH リスト要素にはLIMIT 句を指定できません。
- CREATE VIEW 文

注※

表副問合せによる導出表に限ります。「7.9.1 LIMIT 句の指定形式および規則」中で表記されている導出表は，表副問合せによる導出表のことです。

(1) 指定形式

■SELECT 文の最も外側の問合せ指定または問合せ式本体にLIMIT 句を指定する場合

```
LIMIT句 : :=LIMIT 〔オフセット行数,〕リミット行数

  オフセット行数 : :=値指定
  リミット行数 : :=値指定
```

■導出表，スカラ副問合せ，WITH 句，またはCREATE VIEW 文にLIMIT 句を指定する場合

```
LIMIT句 : :=LIMIT リミット行数

  リミット行数 : :=値指定
```

(2) 指定形式の説明

オフセット行数：

問合せ式の検索結果のうち，先頭行から読み飛ばす行数を指定します。例えば，「LIMIT 10,5」と指定した場合（*オフセット行数*が10，*リミット行数*が5），問合せ式の検索結果の最初の 10 行が読み飛ばされ，11 行目～15 行目が取得されます。

指定規則を次に示します。

- オフセット行数は，SELECT 文の最も外側の問合せ指定または問合せ式本体に指定するLIMIT 句にだけ指定できます。導出表，スカラ副問合せ，WITH 句，またはCREATE VIEW 文に指定するLIMIT 句には，オフセット行数を指定することはできません。
- オフセット行数は，値指定の形式で指定します。値指定については，「7.22　値指定」を参照してください。
- *オフセット行数*には，0～2,147,483,647 の整数（INTEGER 型のデータ）を指定してください。
- *オフセット行数*に0 を指定した場合，オフセット行数の指定がないときと同様に，問合せ式の検索結果の先頭行から，*リミット行数*に指定した行数が取得されます。
- *オフセット行数*に？パラメタを指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。
- *オフセット行数*にナル値は指定できません。

リミット行数：

問合せ式または問合せ式本体の検索結果から取得する行数の最大値を指定します。

指定規則を次に示します。

- *リミット行数*は，値指定の形式で指定します。値指定については，「7.22　値指定」を参照してください。
- *リミット行数*には，0～2,147,483,647 の整数（INTEGER 型のデータ）を指定してください。
- *リミット行数*に0 を指定した場合，検索結果は0 行となります。
- *リミット行数*に？パラメタを指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。
- *リミット行数*にナル値は指定できません。

(3) 規則

(a) SELECT 文の最も外側の問合せ指定または問合せ式本体に LIMIT 句を指定する場合の規則

1. LIMIT 句を指定した場合，問合せ式の検索結果の行数は次のようになります。

   MAX｛MIN（*LIMIT 句を指定しない場合の問合せ式の検索結果の行数－オフセット行数*，*リミット行数*），0｝

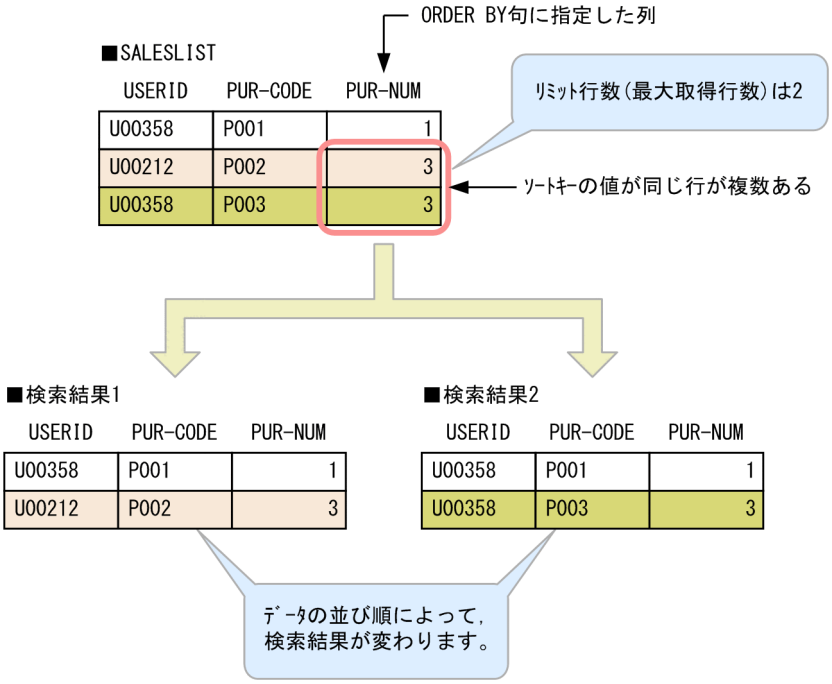
2. 問合せ式の検索結果の行数が，*オフセット行数*および*リミット行数*を超えた場合，次のケースでは検索結果が一意に定まりません。

- ORDER BY 句の指定がない場合
- ORDER BY 句の指定はあるが，LIMIT 句によって取得される結果の最終行のソートキーの値と同じ値を持つ行がほかにもある場合（例 1 を参照）
- ORDER BY 句の指定はあるが，オフセット行数の指定によって読み飛ばされた最終行のソートキーの値と同じ値を持つ行がほかにもある場合（例 2 を参照）

(例 1)

*リミット行数*に2 を指定して，次のSELECT 文を実行して販売履歴表（SALESLIST）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-NUM"
    FROM "SALESLIST"
        ORDER BY "PUR-NUM" ASC
        LIMIT 2
```



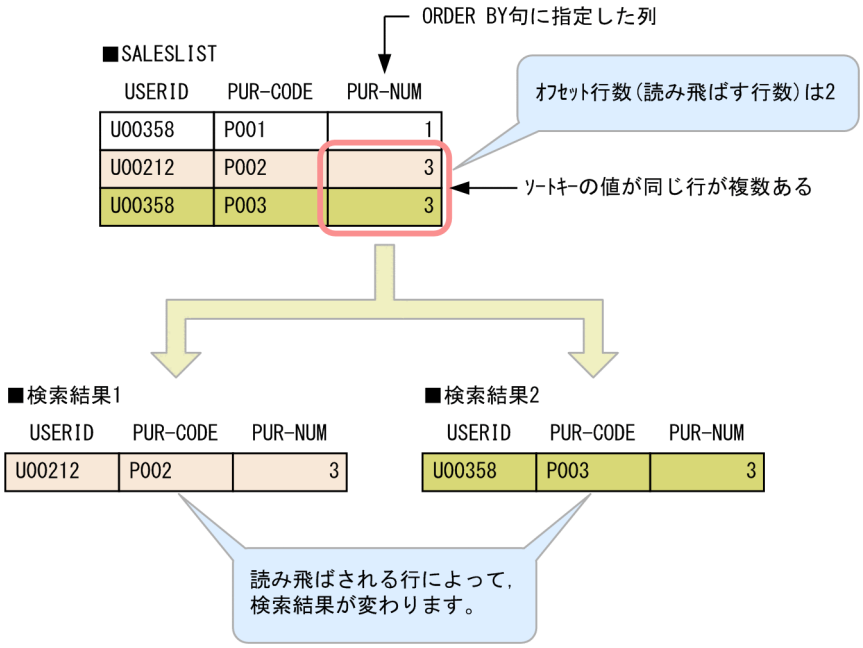
[説明]

- ORDER BY 句の指定によって，PUR-NUM 列の値をソートキーとして，問合せ式の結果を昇順に並べます。
- *リミット行数*の指定によって，最初の 2 行が検索結果として取得されます。
- 最終行（2 行目）のソートキーの値（3）が同じである行が 2 つあるため，検索結果が不定になります。

(例 2)

*オフセット行数*に2，*リミット行数*に1 を指定して，次のSELECT 文を実行して販売履歴表（SALESLIST）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-NUM"
    FROM "SALESLIST"
       ORDER BY "PUR-NUM" ASC
       LIMIT 2,1
```



[説明]

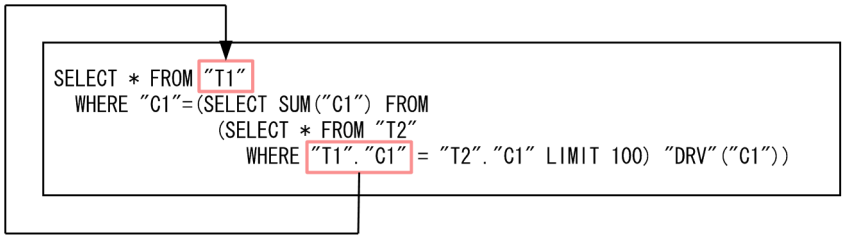
- ORDER BY 句の指定によって，PUR-NUM 列の値をソートキーとして，問合せ式の結果を昇順に並べます。
- *オフセット行数*の指定によって，最初の 2 行が読み飛ばされます。
- 読み飛ばされた最終行（2 行目）のソートキーの値（3）が同じである行が 2 つあるため，読み飛ばされる行によって検索結果が変わります。

(b) 導出表，スカラ副問合せ，WITH 句，または CREATE VIEW 文に LIMIT 句を指定する場合の規則

1. 次の導出表にはLIMIT 句を指定できません。

- LIMIT 句を指定した導出表を超えて，外への参照を行っている導出表

  （例）エラーになる SQL 文の例



この場合，"T1"."C1"が，LIMIT 句を指定した導出表（相関名：DRV）を超えて外への参照を行っています。そのため，LIMIT 句を指定することはできません。

導出表については，「7.11.1　表参照の指定形式」を参照してください。

2. LIMIT 句を指定した場合，問合せ式本体の結果行数は次のようになります。

   MIN（*LIMIT 句を指定しない場合の問合せ式本体の結果行数*，*リミット行数*）

3. 問合せ式本体の結果行数が*リミット行数*を超えた場合，次のケースでは検索結果が一意に定まりません。

   - 導出表，スカラ副問合せ，またはWITH 句にLIMIT 句を指定した場合（これらの指定にはORDER BY 句を指定できないため）
   - LIMIT 句を指定したCREATE VIEW 文で定義したビュー表を検索した場合（CREATE VIEW 文にはORDER BY 句を指定できないため）

   (例)

```
CREATE VIEW "SALESLIST_VIEW" AS SELECT * FROM "SALESLIST" LIMIT 2
SELECT * FROM "SALESLIST_VIEW" ORDER BY "USERID"
```

   上記のSELECT 文を実行した場合，検索結果は次のように一意に定まりません。

■SALESLIST

USERID	PUR-CODE	PUR-NUM
U00358	P001	1
U00212	P002	8
U00555	P003	5

■検索結果1

USERID	PUR-CODE	PUR-NUM
U00358	P001	1
U00212	P002	8

■検索結果2

USERID	PUR-CODE	PUR-NUM
U00555	P003	5
U00212	P002	8

   上記では検索結果を 2 例だけ説明していますが，上記以外の検索結果も表示されることがあります。

4. 外側の問合せの列を参照している問合せ式本体にLIMIT 句を指定した場合，その問合せ式本体の結果の全体行数は，LIMIT 句の対象になりません。この場合，外側の問合せの列の 1 つの値に対する問合せ式本体の結果の行数が，LIMIT 句の対象になります。

   (例)

```
SELECT (SELECT "PRODUCTLIST"."PUR-NAME" FROM "PRODUCTLIST"
          WHERE "SALESLIST"."PUR-CODE"="PRODUCTLIST"."PUR-CODE" LIMIT 1)
       ,"SALESLIST"."PUR-NUM"
    FROM "SALESLIST"
```

   上記の例の場合，下線部分の"SALESLIST"."PUR-CODE"が，外側の問合せの列を参照しています。

   上記のSELECT 文を実行した場合，検索結果は次のようになります。

■SALESLIST

PUR-CODE	PUR-NUM
P001	1
P002	8
P003	5

■検索結果

PUR-NAME	PUR-NUM
File	1
Pen	8
Highlighter	5

(4) 例題

例題 1（問合せ指定にLIMIT 句を指定した例）

支店表（BRANCHESLIST）から，売上高（SALES）上位 10 位の支店を検索します。

```
SELECT "BRANCH-CODE","RGN-CODE","BRANCH-NAME","SALES"
    FROM "BRANCHESLIST"
    ORDER BY "SALES" DESC
    LIMIT 10
```

下線部分がLIMIT 句の指定です。

例題 2（オフセット行数を指定した例）

支店表（BRANCHESLIST）から，売上高（SALES）21 位～30 位の支店を検索します。

```
SELECT "BRANCH-CODE","RGN-CODE","BRANCH-NAME","SALES"
    FROM "BRANCHESLIST"
    ORDER BY "SALES" DESC
    LIMIT 20,10
```

下線部分がLIMIT 句の指定です。

例題 3（導出表にLIMIT 句を指定した例）

販売履歴表（SALESLIST）から任意の 100 行を取得し，その取得結果に対して商品コード（PUR-CODE）ごとに，販売個数（PUR-NUM）の合計値を求めます。

```
SELECT "PUR-CODE",SUM("PUR-NUM")
    FROM (SELECT * FROM "SALESLIST" LIMIT 100) "SALESLIST"
        GROUP BY "PUR-CODE"
```

下線部分がLIMIT 句の指定です。

上記のSELECT 文は，販売履歴表（SALESLIST）から任意の 100 行を取得し，検索結果を求めています。任意の 100 行は検索のたびに変わることがあるため，上記のSELECT 文の結果は検索のたびに変わることがあります。

例題 4（導出表にLIMIT 句を指定した例）

販売履歴表（SALESLIST）の購入日（PUR-DATE）に条件を指定して，検索結果行数を求めます。LIMIT 句を指定して，導出表の行数が 1,000 を超えた時点で検索を終了し，検索結果を返します。

```
SELECT COUNT(*)
  FROM (SELECT 1 FROM "SALESLIST"
        WHERE "PUR-DATE" BETWEEN ? AND ? LIMIT 1000) "SALESLIST"("PUR-DATE")
```

下線部分がLIMIT 句の指定です。

導出表にLIMIT 句を指定することによって（導出表の行数の上限を決めることによって），SELECT 文の実行時間を一定時間内に抑えることができます。検索結果が 1,000 行未満になるまで絞り込むような場合に，上記のSELECT 文を実行します。上記の SELECT 文を実行した結果，検索結果が 1,000 の場合は検索条件を満たす行が 1,000 行以上あります。この場合，検索結果が 1,000 未満になるまで，？パラメタの値を変えてSELECT 文を繰り返し実行します。

例題５（スカラ副問合せにLIMIT 句を指定した例）

販売履歴表（SALESLIST）から，販売個数（PUR-NUM）が最も多い日の購入日（PUR-DATE）と商品コード（PUR-CODE）を求めます。

```
SELECT DISTINCT "PUR-DATE","PUR-CODE"
  FROM "SALESLIST"
    WHERE "PUR-DATE"=(SELECT "PUR-DATE" FROM "SALESLIST"
                        WHERE "PUR-NUM"=(SELECT MAX("PUR-NUM")
                                           FROM "SALESLIST") LIMIT 1)
```

下線部分がLIMIT 句の指定です。

販売個数（PUR-NUM）が最も多い日が複数ある場合，そのうちの任意の購入日（PUR-DATE）が検索対象になるため，上記のSELECT 文の結果は検索のたびに変わることがあります。

7.10 DEFAULT 句

ここでは，DEFAULT 句について説明します。

7.10.1 DEFAULT 句の指定形式および規則

DEFAULT 句には，列の既定値を指定します。列の既定値とは，次のときに列に格納されるデフォルトの列値のことです。

- INSERT 文による行の挿入時

  次の場合に列の既定値が格納されます。

  - 挿入値にDEFAULT を指定した場合
  - DEFAULT VALUES を指定した場合
  - データを挿入する列の列名の指定を省略した場合（すべての列名を省略した場合を除きます）
  - ビュー表に行を挿入した場合（ビュー表の列に対応しない基表の列には，列の既定値が格納されます）
- UPDATE 文による列値の更新時

  更新値にDEFAULT を指定した場合に，列の既定値が格納されます。
- adbimport コマンドによるデータインポート時

  adbimport コマンドでデータをインポートする際，入力データファイルのフィールドデータが空文字列の場合に，列の既定値が格納されます。

  ただし，インポートオプションのadb_import_null_string にNULL を指定した場合は，列の既定値ではなくナル値が格納されます。

(1) 指定形式

```
DEFAULT句 : :＝DEFAULT 既定値選択肢
  既定値選択肢 : : = {定数 | CURRENT_DATE | CURRENT_TIME 〔(p)〕
                    | CURRENT_TIMESTAMP 〔(p)〕 | CURRENT_USER | NULL}
```

(2) 指定形式の説明

定数：

列の既定値を定数の形式で指定します。定数については，「6.3 定数」を参照してください。

列の既定値を設定する列のデータ型と，*既定値選択肢*に指定できる定数の種類を次の表に示します。

表 7-2　列の既定値を設定する列のデータ型と，既定値選択肢に指定できる定数の種類

列の既定値を設定する列のデータ型		既定値選択肢に指定できる定数					
		数定数	文字列定数	日付定数	時刻定数	時刻印定数	バイナリ定数
数データ		○	×	×	×	×	×
文字データ		×	○※1	×	×	×	×
日時データ	DATE 型	×	○※2	○	×	○	×
	TIME 型	×	○※2	×	○	×	×
	TIMESTAMP 型	×	○※2	○	×	○	×
バイナリデータ		×	×	×	×	×	○※1

（凡例）

○：指定できます。ただし，格納代入の規則が適用されます。※3

×：指定できません。

注※1

1,024 バイト以上の文字列定数またはバイナリ定数は指定できません。

注※2

既定の入力表現の形式で文字列定数を記述した場合に限り指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。

注※3

格納代入の規則の詳細については，「6.2.2　変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」を参照してください。

格納代入の規則が適用されるため，例えば，「列の既定値として指定した文字列定数のデータ長＞DEFAULT 句を指定した列のデータ長」の場合，CREATE TABLE 文がエラーになります。

CURRENT_DATE：

INSERT 文またはUPDATE 文を実行した日付，またはadbimport コマンドを起動した日付が，列の既定値として設定されます。

CURRENT_DATE は，DATE 型またはTIMESTAMP 型の列に指定できます。

CURRENT_DATE の指定規則の詳細については，「6.4.1　CURRENT_DATE」を参照してください。

CURRENT_TIME〔(*p*)〕：

INSERT 文またはUPDATE 文を実行した時刻，またはadbimport コマンドを起動した時刻が，列の既定値として設定されます。

p には小数秒精度（小数秒の桁数）を指定します。(*p*)を省略した場合，*p* ＝ 0 が仮定されます。

CURRENT_TIME は，TIME 型の列に指定できます。

CURRENT_TIME の指定規則の詳細については，「6.4.2　CURRENT_TIME」を参照してください。

CURRENT_TIMESTAMP〔(*p*)〕：

INSERT 文またはUPDATE 文を実行した日付と時刻，またはadbimport コマンドを起動した日付と時刻が，列の既定値として設定されます。

p には小数秒精度（小数秒の桁数）を指定します。(*p*)を省略した場合，*p* ＝ 0 が仮定されます。

CURRENT_TIMESTAMP は，DATE 型またはTIMESTAMP 型の列に指定できます。

CURRENT_TIMESTAMP の指定規則の詳細については，「6.4.3　CURRENT_TIMESTAMP」を参照してください。

CURRENT_USER：

INSERT 文，UPDATE 文，またはadbimport コマンドを実行したユーザの認可識別子が，列の既定値として設定されます。

CURRENT_USER は，CHARACTER 型またはVARCHAR 型の列に指定できます。

CURRENT_USER の指定規則の詳細については，「6.5.1　CURRENT_USER」を参照してください。

NULL：

列の既定値としてナル値を設定する場合に指定します。

非ナル値制約（ナル値を許さない制約）を定義している列には，NULL を指定できません。

メモ

- CURRENT_DATE を指定した際の日付，CURRENT_TIME〔(*p*)〕を指定した際の時刻，またはCURRENT_TIMESTAMP〔(*p*)〕を指定した際の日付と時刻は，HADB サーバで取得されます。
- 1 つの SQL 文で複数の行に列の既定値を格納する場合，CURRENT_DATE を指定しているときは，すべて同じ日付が格納されます。CURRENT_TIME〔(*p*)〕を指定しているときは，すべて同じ時刻が格納され，CURRENT_TIMESTAMP〔(*p*)〕を指定しているときは，すべて同じ日付と時刻が格納されます。

(3)　規則

1. DEFAULT 句を省略した場合，ナル値が列の既定値になります。
2. 列の既定値を設定した列にデータを格納する際，代入規則に従って格納されます。例えば，TIMESTAMP 型の列にCURRENT_DATE を指定した場合，代入規則に従って時刻00:00:00 が付与されてデータが格納されます。代入規則の詳細については，「6.2.2　変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」を参照してください。
3. CURRENT_TIME(*p*)またはCURRENT_TIMESTAMP(*p*)によって取得される小数秒の精度は，ハードウェアの性能に依存します。例えば，CURRENT_TIME(12)を指定しても，使用しているハードウェアによっては 12 桁の小数秒精度が取得できないことがあります。

   （例）

   10:35:55.123456000000

   小数秒精度が 6 桁までしか取得できない場合，上記のように 7 桁目以降は 0 になります。

(4) 例題

例題

販売履歴表（SALESLIST）を定義します。購入日（PUR-DATE）列にDEFAULT 句を指定して，列の既定値を設定します。

```
CREATE FIX TABLE ”SALESLIST”
      (”USERID” CHAR(6),
       ”PUR-CODE” CHAR(4),
       ”PUR-NUM” SMALLINT,
       ”PUR-DATE” DATE DEFAULT CURRENT_DATE)
    IN ”DBAREA01”
    PCTFREE=20
    CHUNK=200
```

下線部分がDEFAULT 句の指定です。

7.11　表参照

ここでは，表参照について説明します。

7.11.1　表参照の指定形式

表参照には検索対象の表を指定します。表参照はFROM 句中に指定します。

なお，同じ表を結合した検索を行う場合は，相関名を指定します。

(1)　指定形式

```
表参照：：＝｛表一次子｜結合表｝

  表一次子：：＝｛表名　〔〔AS〕 相関名〕〔インデクス指定〕
                |問合せ名　〔〔AS〕 相関名〕
                |導出表　〔〔AS〕 相関名　〔(導出列リスト)〕〕
                |表関数導出表　〔AS〕 相関名　〔(表関数列リスト)〕
                |集まり導出表　〔AS〕 相関名　〔(導出列リスト)〕
                |(結合表)｝

    導出表：：＝｛表副問合せ｜(表値構成子)｝
    表関数導出表：：＝TABLE(システム定義関数)
    集まり導出表：：＝UNNEST（列指定〔,列指定〕…）

    導出列リスト：：＝列名〔,列名〕…
    表関数列リスト：：＝列名 データ型〔,列名 データ型〕…
```

(2)　指定形式の説明

表名：

検索対象の表を指定します。表名の指定規則については，「6.1.5　名前の修飾」の「(2)　表名の指定形式」を参照してください。

ディクショナリ表またはシステム表を検索する場合は，スキーマ名にMASTER を指定してください。

なお，アーカイブマルチチャンク表を指定した場合，ロケーション表およびシステム表（STATUS_CHUNKS）に対するアクセスが発生します。その際，システム表（STATUS_CHUNKS）の排他資源が確保されます。排他制御の詳細については，マニュアル『HADB　システム構築・運用ガイド』の『排他制御』を参照してください。

〔AS〕 *相関名*：

次に示す場合に，表同士を区別するために付ける名称を指定します。

- 同じ表を結合する場合
- 内側の副問合せで同じ表の列を参照する場合

留意事項を次に示します。

- 表関数導出表または集まり導出表を指定する場合は，相関名を指定する必要があります。
- 1 つのFROM 句中で同じ範囲変数を複数回指定する場合は，相関名を指定してそれぞれの範囲変数が一意に識別できるようにしてください。
- 相関名には，1 つのFROM 句中に指定したほかの範囲変数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。範囲変数の有効範囲については，「6.8　範囲変数」を参照してください。
- AS の指定のありなしに関係なく結果は同じになります。
- 導出表の相関名を省略した場合，次の名称規則に従って自動的に相関名が付けられます。

  ##ADD_DRVTBL_*xxxxxxxxxx*

  xxxxxxxxxx は 10 桁の整数です。この相関名は，アクセスパスの実行結果で表示されます。

  有効範囲が同じになる表参照では，##ADD_DRVTBL_から始まる表名，および相関名を指定しないでください。

  > **メモ**
  >
  > HADB サーバによって自動的に付けられた相関名が重複することがあります。

インデクス指定：

実表の検索時に使用する B-tree インデクスまたはテキストインデクスを指定します。または，B-tree インデクスまたはテキストインデクスの使用抑止を指定します。インデクス指定については，「7.14 インデクス指定」を参照してください。

問合せ名：

問合せ名を指定します。問合せ名については，「7.1.1　問合せ式の指定形式および規則」の「(2)　指定形式の説明」の「(a)　WITH 句」を参照してください。

導出表：

導出表を表副問合せまたは表値構成子の形式で指定します。表副問合せについては，「7.3　副問合せ」を参照してください。表値構成子については，「7.17　表値構成子」を参照してください。

導出表とは，表副問合せまたは表値構成子の結果，導出された表のことです。表副問合せまたは表値構成子の*n* 番目の列が，導出表の*n* 番目の列になります。

導出表を含む問合せ指定は，その導出表を含まない等価な問合せ指定に変換されます。

導出表の相関名（相関名を省略した場合は HADB サーバが生成した相関名）を導出問合せ名，導出表の問合せ式を導出問合せ式とした場合，内部導出表の適用規則によって，その導出問合せ式が内部導出表として扱われます。内部導出表の適用規則については，「7.32　内部導出表」を参照してください。

留意事項を次に示します。

- 導出表の相関名を省略した場合，その導出表の範囲変数は有効範囲を持ちますが，名称がありません（HADB サーバが内部的に名称を生成するため，ユーザはその名称を確認できません）。したがって，同じ有効範囲内に同じ列名を持つ表参照が複数ある場合は，相関名を指定するようにしてください。

  （例）エラーになる SQL 文の例

```
SELECT "C1" FROM "T1",(SELECT "C1" FROM "T1")
```

下線部分の列C1 は，同じ有効範囲内に同じ列名を持つ表参照が複数あります（表T1 と導出表）。上記の場合，下線部分の列C1 が，表T1 の列C1 なのか，導出表の列C1 なのかが特定できないため，SQL 文がエラーになります。そのため，導出表の列を参照する場合は，導出表に相関名を指定し，その相関名で列名を修飾してください。例を次に示します。

（例）正しい SQL 文の例

```
SELECT "DT1"."C1" FROM "T1",(SELECT "C1" FROM "T1") AS "DT1"
```

- 導出表に対して行インタフェース（ROW）は指定できません。

導出列リスト：

導出表または集まり導出表の各列の列名を指定します。*導出列リスト*は，次に示す形式で指定します。

```
列名〔,列名〕…
```

*導出列リスト*を指定した場合と，*導出列リスト*の指定を省略した場合で，導出される表の列名が異なります。導出される列名の規則については，「6.9　導出列名」を参照してください。

留意事項を次に示します。

- *導出列リスト*の指定を省略した場合，導出される列名が重複しないようにしてください。
- *導出列リスト*中に，同じ列名は指定しないでください。
- HADB によって自動的に設定される導出列名と重複する可能性があるため，*導出列リスト*の列名にEXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。
- *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，導出表または集まり導出表によって導出される表の列数と同じになるようにしてください。
- *導出列リスト*に指定する列数は，4,000 以下にしてください。

表関数導出表：

```
表関数導出表：：＝TABLE(システム定義関数)
```

表関数導出表とは，システム定義関数によって導出される表形式のデータ集合のことです。システム定義関数については，「7.15　システム定義関数」を参照してください。

表関数導出表の指定規則を次に示します。

- 表参照に表関数導出表を指定した場合，表関数導出表に対する相関名を指定してください。
- 表関数導出表に対して行インタフェース（ROW）は指定できません。

表関数列リスト：

```
表関数列リスト：：＝列名　データ型〔,列名　データ型〕…
```

表関数導出表の各列の列名とデータ型を指定します。

表関数列リストの指定規則を次に示します。

- 表関数導出表にADB_AUDITREAD 関数を指定した場合，表関数列リストは指定できません。

- 表関数導出表にADB_CSVREAD関数を指定した場合，表関数列リストを必ず指定してください。
- データ型の指定形式については，「表 3-8　指定できるデータ型（CREATE TABLE 文の場合）」を参照してください。
- 表関数列リスト中に同じ列名は指定できません。
- HADB によって自動的に設定される導出列名と重複する可能性があるため，表関数列リストの列名にEXP*nnnn*_NO_NAME を指定しないでください。*nnnn* は，0000～9999 の符号なし整数です。
- 表関数列リストの列名の数は，4,000 個以下にしてください。
- 表関数列リストのデータ型には，データ長が 32,000 バイトを超えるVARCHAR 型を指定できません。
- 表関数列リストのデータ型には，配列型を指定できません。
- 表関数列リストのデータ型にNUMERIC 型が指定された場合，HADB はDECIMAL 型が指定されたと見なします。
- 表関数列リストのデータ型にFLOAT 型が指定された場合，HADB はDOUBLE PRECISION 型が指定されたと見なします。

導出される列名の規則については，「6.9.2　問合せの結果の導出列名の決定規則」の「(4)　表関数導出表の場合」を参照してください。

集まり導出表：

集まり導出表：：＝UNNEST（*列指定*〔,*列指定*〕…）

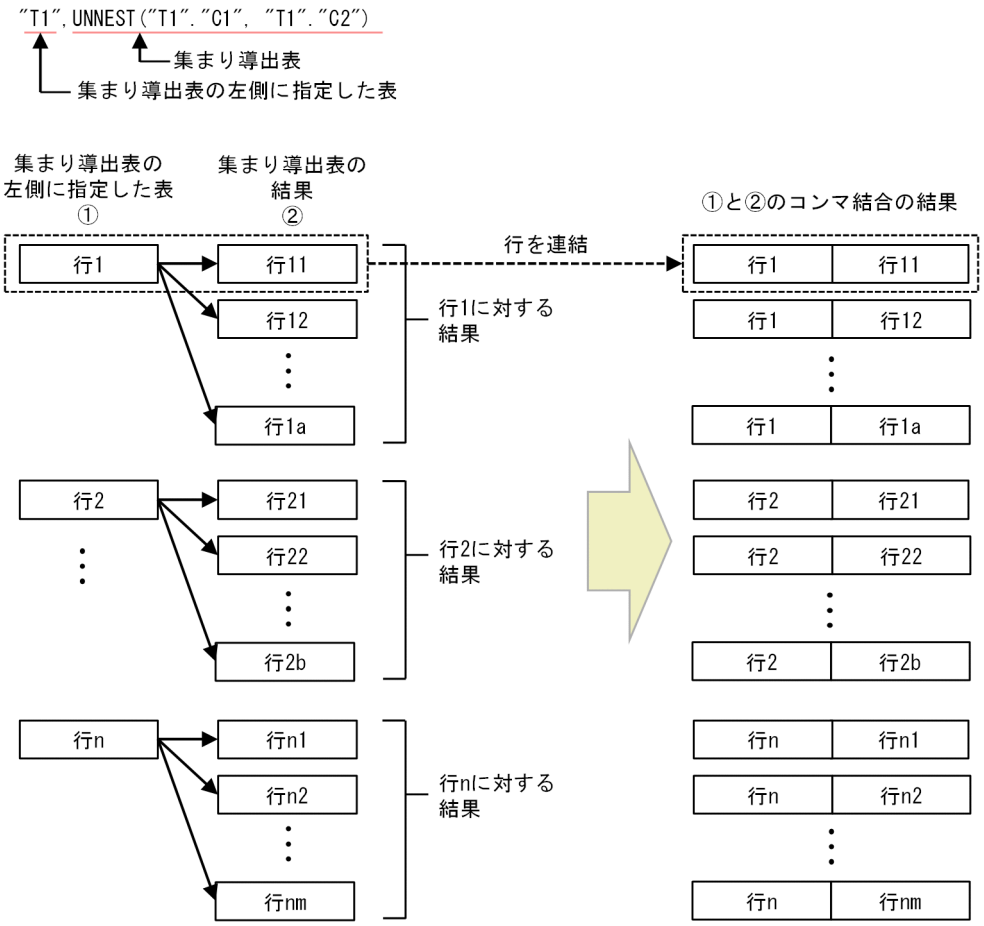
集まり導出表には，列指定を指定します。列指定に指定した配列データの各配列要素の値を列値とする行が生成され，その行の集合が集まり導出表の結果になります。集まり導出表の結果の*n* 番目の列が，集まり導出表の*n* 番目の列になります。列指定の指定形式については，「6.1.5　名前の修飾」の「(5) 列指定の指定形式」を参照してください。

指定規則を次に示します。

- 列指定には，配列型の列を指定してください。
- 列指定の数は 4,000 以下にしてください。
- 列指定には，次のすべての条件を満たす実表の列を指定してください。
  - 集まり導出表と同じFROM 句に指定している実表
  - 集まり導出表とコンマ結合している実表
  - 集まり導出表よりも左側に指定している実表
- 列指定を複数指定する場合，異なる表の列は指定できません。

集まり導出表の左側に指定した表と集まり導出表のコンマ結合の結果は，左側に指定した表の各行に対して，集まり導出表の結果の各行を組み合わせて連結した行の集合になります。

（例）

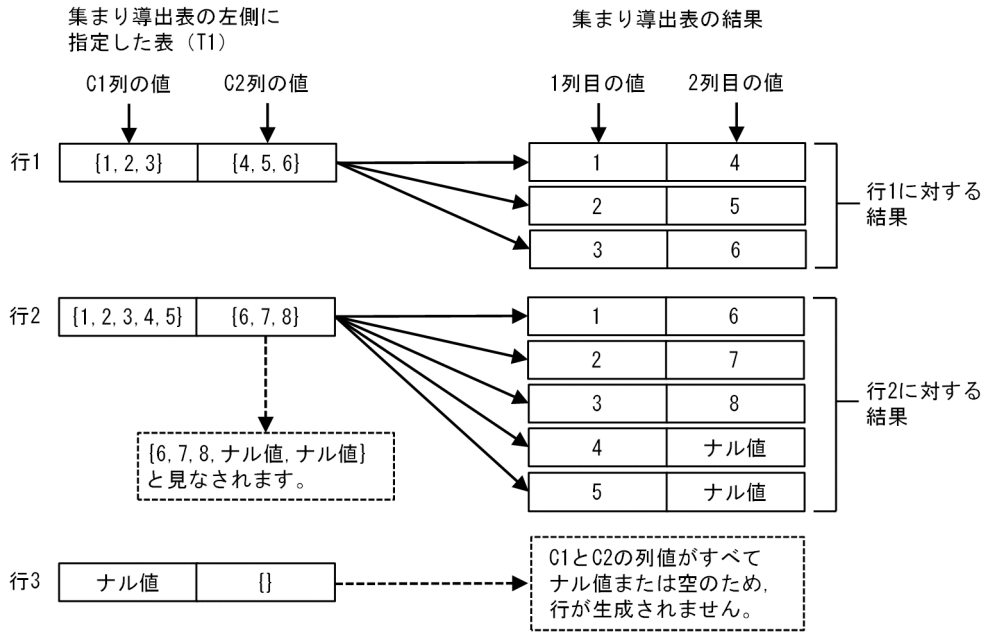


このとき，集まり導出表の結果は，次の流れで導出されます。

1. *i* 番目の列指定が参照する配列データの各配列要素に対して，それを *i* 番目の列値とする行を，列指定の中で配列要素数が最大となる配列データの配列要素数分だけ生成します。このとき，次の規則が適用されます。
   - 列指定の中で配列要素数が最大となる配列要素数を N_{max} とします。配列要素数が N_{max} 未満となる *j* 番目の列指定の配列要素数を N_j とする場合，*j* 番目の列指定の配列データの「N_{j+1}～N_{max}」番目の配列要素にはナル値が補完されます。
   - すべての列指定が参照する配列データが，ナル値または空の配列データの場合，行は生成されません。
2. 生成された行の集合が，集まり導出表の結果になります。

   （例）"T1", UNNEST("T1"."C1","T1"."C2")と指定した場合

   C1 列とC2 列は，要素データ型がINTEGER 型で，最大要素数が 10 の配列型の列とします。



集まり導出表の左側に指定した表と集まり導出表のコンマ結合の結果の例を次に示します。

例で使用する表T1 は次のとおりとします。

表T1

C1列	C2列	C3列
1	{1, 2, 3}	{1, 2}
2	{4, ナル値, 5}	{3, ナル値, 4}
3	{6, 7}	ナル値
4	{8, 9}	{}
5	ナル値	{5, 6}
6	ナル値	ナル値
7	ナル値	{}
8	{}	{7, 8}
9	{}	ナル値
10	{}	{}

INTEGER

INTEGER ARRAY[5]

- 集まり導出表に指定する列指定が 1 つの場合

  （例 1）

```
FROM "T1",UNNEST("T1"."C2") AS "T2"("DC2")
```

  下線部分が集まり導出表の指定です。

  ＜上記のFROM 句の結果＞

T1.C1	T1.C2	T1.C3	T2.DC2
1	{1,2,3}	{1,2}	1
1	{1,2,3}	{1,2}	2
1	{1,2,3}	{1,2}	3
2	{4,ナル値,5}	{3,ナル値,4}	4
2	{4,ナル値,5}	{3,ナル値,4}	ナル値
2	{4,ナル値,5}	{3,ナル値,4}	5
3	{6,7}	ナル値	6
3	{6,7}	ナル値	7
4	{8,9}	{}	8
4	{8,9}	{}	9

(例 2)

```
FROM "T1",UNNEST("T1"."C3") AS "T2"("DC3")
```

下線部分が集まり導出表の指定です。

＜上記のFROM 句の結果＞

T1.C1	T1.C2	T1.C3	T2.DC3
1	{1,2,3}	{1,2}	1
1	{1,2,3}	{1,2}	2
2	{4,ナル値,5}	{3,ナル値,4}	3
2	{4,ナル値,5}	{3,ナル値,4}	ナル値
2	{4,ナル値,5}	{3,ナル値,4}	4
5	ナル値	{5,6}	5
5	ナル値	{5,6}	6
8	{}	{7,8}	7
8	{}	{7,8}	8

(例 3)

```
FROM "T1",UNNEST("T1"."C2") AS "T2"("DC2"),UNNEST("T1"."C3") AS "T3"("DC3")
```

下線部分が集まり導出表の指定です。

＜上記のFROM 句の結果＞

T1.C1	T1.C2	T1.C3	T2.DC2	T3.DC3
1	{1,2,3}	{1,2}	1	1
1	{1,2,3}	{1,2}	1	2
1	{1,2,3}	{1,2}	2	1
1	{1,2,3}	{1,2}	2	2
1	{1,2,3}	{1,2}	3	1
1	{1,2,3}	{1,2}	3	2
2	{4,ナル値,5}	{3,ナル値,4}	4	3
2	{4,ナル値,5}	{3,ナル値,4}	4	ナル値
2	{4,ナル値,5}	{3,ナル値,4}	4	4
2	{4,ナル値,5}	{3,ナル値,4}	ナル値	3
2	{4,ナル値,5}	{3,ナル値,4}	ナル値	ナル値
2	{4,ナル値,5}	{3,ナル値,4}	ナル値	4
2	{4,ナル値,5}	{3,ナル値,4}	5	3
2	{4,ナル値,5}	{3,ナル値,4}	5	ナル値
2	{4,ナル値,5}	{3,ナル値,4}	5	4

- 集まり導出表に指定する列指定が複数の場合

  (例)

```
FROM "T1",UNNEST("T1"."C2","T1"."C3") AS "T2" ("DC2","DC3")
```

  下線部分が集まり導出表の指定です。

  <上記のFROM 句の結果>

T1.C1	T1.C2	T1.C3	T2.DC2	T2.DC3
1	{1,2,3}	{1,2}	1	1
1	{1,2,3}	{1,2}	2	2
1	{1,2,3}	{1,2}	3	ナル値
2	{4,ナル値,5}	{3,ナル値,4}	4	3
2	{4,ナル値,5}	{3,ナル値,4}	ナル値	ナル値
2	{4,ナル値,5}	{3,ナル値,4}	5	4
3	{6,7}	ナル値	6	ナル値
3	{6,7}	ナル値	7	ナル値
4	{8,9}	{}	8	ナル値
4	{8,9}	{}	9	ナル値
5	ナル値	{5,6}	ナル値	5
5	ナル値	{5,6}	ナル値	6
8	{}	{7,8}	ナル値	7
8	{}	{7,8}	ナル値	8

結合表：

結合表を指定します。結合表については,「7.12　結合表」を参照してください。

(3)　例題

表参照の指定例を例題を使って説明します。

例題 1

販売履歴表（SALESLIST）から，2011/9/6 以降に商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-06'
```

下線部分が表参照の指定です。

例題 2

販売履歴表（SALESLIST）に定義されているインデクス名（INDEX_NAME）を調べるためにディクショナリ表（SQL_INDEXES）を検索します。

```
SELECT "INDEX_NAME"
    FROM "MASTER"."SQL_INDEXES"
        WHERE "TABLE_NAME"='SALESLIST'
```

下線部分が表参照の指定です。ディクショナリ表を検索する場合は，スキーマ名MASTER で表名を修飾する必要があります。

例題 3

販売履歴表（SALESLIST）に定義されているインデクス名（INDEX_NAME）を調べるためにディクショナリ表（SQL_INDEXES）を検索します。相関名としてIDX を使用しています。

```
SELECT "IDX"."INDEX_NAME"
    FROM "MASTER"."SQL_INDEXES" AS "IDX"
        WHERE "IDX"."TABLE_NAME"='SALESLIST'
```

下線部分が表参照の指定です。

例題 4

顧客 ID 列（USERID）を探索条件にした販売履歴表（SALESLIST）および顧客表（USERSLIST）の結合表から，顧客 ID（USERID），商品コード（PUR-CODE），顧客名（NAME）および性別（SEX）を検索します。

```
SELECT "SALESLIST"."USERID","PUR-CODE","NAME","SEX"
    FROM ("SALESLIST" JOIN "USERSLIST"
          ON "USERSLIST"."USERID"="SALESLIST"."USERID")
```

下線部分が表参照の指定です。

例題 5

GZIP 形式で圧縮した CSV ファイル（/dir/file.csv.gz）から，次のデータを取り出します。

- 顧客 ID（USERID）
- 顧客名（NAME）
- 年齢（AGE）

```
SELECT "USERID","NAME","AGE"
    FROM TABLE(ADB_CSVREAD(MULTISET ['/dir/file.csv.gz'],
                           'COMPRESSION FORMAT=GZIP;'))
        AS "USERLIST" ("USERID" CHAR(5),
```

```
                    "NAME" VARCHAR(100),
                    "AGE" INTEGER,
                    "COUNTRY" VARCHAR(100),
                    "INFORMATION" VARBINARY(10))
```

下線部分が表参照の指定です。

例題 6

顧客表（USERSLIST）の関心事リスト列（INTEREST-LIST）を検索して，ファッションに興味を持っている顧客の顧客 ID（USERID）を検索します。

関心事リスト列（INTEREST-LIST）は配列型の列で，配列要素には顧客が興味を持っている分野についての情報が格納されています。

```
SELECT "USERID"
    FROM "USERSLIST",UNNEST("INTEREST-LIST") "CDT"("INTEREST")
        WHERE "CDT"."INTEREST" = 'fashion'
```

下線部分が表参照の指定です。

7.12　結合表

ここでは，結合表について説明します。

7.12.1　結合表の指定形式および規則

表の結合方法（直積，内結合，または外結合）を指定します。結合表は表参照中に指定します。

(1)　指定形式

```
結合表 ：：＝ {交差結合 | 修飾付き結合 | (結合表)}

  交差結合 ：：＝表参照 CROSS JOIN 表一次子

  修飾付き結合 ：：＝表参照 〔 {INNER | {LEFT | RIGHT | FULL} 〔OUTER〕 } 〕 JOIN 〔結合方式指定〕 表参照 結合指定
    結合指定 ：：＝ON 探索条件
```

(2)　指定形式の説明

●*交差結合*

```
交差結合 ：：＝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*の直積を求める場合に指定します。表参照については，「7.11　表参照」を参照してください。*表一次子*については，「7.11.1　表参照の指定形式」を参照してください。

なお，問合せ指定の選択式に*を指定した場合，検索結果の列は左側の*表参照*の列，右側の*表一次子*の列の順に並びます。

交差結合の例を次に示します。

(例)

■USERSLIST

顧客ID (USERID)	顧客名 (NAME)
U00555	Mike Johnson
U00358	Nancy White
U00212	Maria Gomez

■SALESLIST

顧客ID (USERID)	商品ｺｰﾄﾞ (PUR-CODE)	販売個数 (PUR-NUM)
U00555	P002	1
U00358	P001	3
U00358	P002	6
U00026	P101	25

実行する SELECT 文

```
SELECT * FROM "USERSLIST" CROSS JOIN "SALESLIST"
```

検索結果

USERSLIST		SALESLIST		
USERID	NAME	USERID	PUR-CODE	PUR-NUM
U00555	Mike Johnson	U00555	P002	1
U00555	Mike Johnson	U00358	P001	3
U00555	Mike Johnson	U00358	P002	6
U00555	Mike Johnson	U00026	P101	25
U00358	Nancy White	U00555	P002	1
U00358	Nancy White	U00358	P001	3
U00358	Nancy White	U00358	P002	6
U00358	Nancy White	U00026	P101	25
U00212	Maria Gomez	U00555	P002	1
U00212	Maria Gomez	U00358	P001	3
U00212	Maria Gomez	U00358	P002	6
U00212	Maria Gomez	U00026	P101	25

USERSLIST の各行に対して，SALESLIST のすべての行を組み合わせます。

●*修飾付き結合*

修飾付き結合 ：：＝*表参照* 〔 {INNER | {LEFT | RIGHT | FULL} 〔OUTER〕 } 〕 JOIN 〔*結合方式指定*〕 *表参照* *結合指定*
結合指定 ：：＝ON *探索条件*

内結合または外結合を行う場合に指定します。

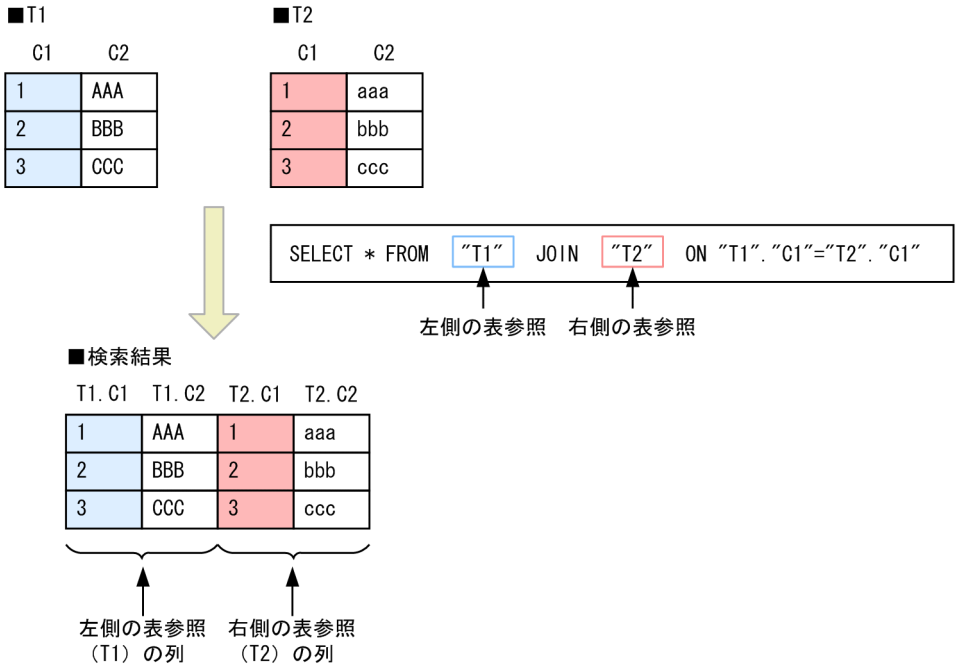
INNER JOIN を指定した結合を内結合といい，LEFT OUTER JOIN，RIGHT OUTER JOIN，またはFULL OUTER JOIN を指定した結合を外結合といいます。

表参照：

結合の対象とする表または結合表を指定します。表参照については，「7.11　表参照」を参照してください。

問合せ指定の選択式に*を指定した場合，検索結果の列は左側の表参照の列，右側の表参照の列の順に並びます。例を次に示します。

(例)



〔INNER〕 JOIN：

左側と右側に指定した表参照の直積のうち，*結合指定*に指定した*探索条件*が真となる行を結合表の結果とします。

INNER JOIN の例については，「7.12.2　INNER JOIN を指定した内結合」を参照してください。

LEFT 〔OUTER〕 JOIN：

次に示す行の和集合を結合表の結果とします。

- 左側と右側に指定した表参照の直積のうち，*結合指定*に指定した*探索条件*が真となる行（INNER JOIN を指定したときの結果と同じ）
- 左側と右側に指定した表参照の直積のうち，結合指定に指定した*探索条件*が 1 つも真とならない左側の表参照の各行に対して，右側の表参照の結果をナル値とした行

LEFT OUTER JOIN の例については，「7.12.3　LEFT OUTER JOIN を指定した外結合」を参照してください。

RIGHT 〔OUTER〕 JOIN：

次に示す行の和集合を結合表の結果とします。

- 左側と右側に指定した表参照の直積のうち，*結合指定*に指定した*探索条件*が真となる行（INNER JOIN を指定したときの結果と同じ）
- 左側と右側に指定した表参照の直積のうち，結合指定に指定した*探索条件*が 1 つも真とならない右側の表参照の各行に対して，左側の表参照の結果をナル値とした行

RIGHT OUTER JOIN の例については，「7.12.4　RIGHT OUTER JOIN を指定した外結合」を参照してください。

FULL 〔OUTER〕 JOIN：

次の行の和集合を結合表の結果とします。

- 左側と右側に指定した表参照の直積のうち，*結合指定*に指定した*探索条件*が真となる行（INNER JOIN を指定したときの結果と同じ）
- 左側と右側に指定した表参照の直積のうち，結合指定に指定した*探索条件*が 1 つも真とならない左側の表参照の各行に対して，右側の結果をナル値とした行
- 左側と右側に指定した表参照の直積のうち，結合指定に指定した*探索条件*が 1 つも真とならない右側の表参照の各行に対して，左側の結果をナル値とした行

FULL OUTER JOIN の例については，「7.12.5　FULL OUTER JOIN を指定した外結合」を参照してください。

結合方式指定：

両側に指定した表参照の結合方式を指定します。結合方式指定については，「7.13　結合方式指定」を参照してください。

なお，通常は*結合方式指定*を指定する必要はありません。*結合方式指定*を省略した場合，結合方式は HADB が自動的に決定します。

結合指定：

結合指定：：＝ON *探索条件*

2 つの表参照を結合するときの条件を指定します。

ON *探索条件*：

探索条件を指定します。探索条件については「7.19　探索条件」を参照してください。

探索条件中の列指定は，次に示すどちらかの条件を満たす必要があります。

- 結合させる 2 つの表参照に含まれる列
- 外への参照列

外への参照列については，「7.3.1　副問合せの指定形式および規則」の「(4)　規則」の「(a)　副問合せ共通の規則」を参照してください。

*探索条件*中の列名を表指定で修飾する場合，相関名を指定した表の列は，相関名で修飾する必要があります。

FULL OUTER JOIN を指定した結合表のON 探索条件に，副問合せは指定できません。

●(*結合表*)

表の結合順序を指定したい場合に，結合表を括弧で囲みます。

(3)　規則

1. LEFT OUTER JOIN を指定した場合，右側の表参照の結果は非ナル値制約なし（ナル値を許す）となります。
2. RIGHT OUTER JOIN を指定した場合，左側の表参照の結果は非ナル値制約なし（ナル値を許す）となります。
3. FULL OUTER JOIN を指定した場合，両側の表参照の結果は非ナル値制約なし（ナル値を許す）となります。

4. SQL 文中に指定できるFULL OUTER JOIN の数は，最大 63 個になります。結合対象の表参照がビュー表の場合は，CREATE VIEW 文で指定した問合せ式に基づいた内部導出表が適用されます。FULL OUTER JOIN の最大数の規則は，この内部導出表に対して適用されます。

5. FULL OUTER JOIN を指定した場合，導出表が作成されます。導出表の相関名は HADB サーバが自動的に次の形式で設定します。

```
##DRVTBL_xxxxxxxxxx
```

xxxxxxxxxx は，10 桁の整数です。

6. FULL OUTER JOIN の両側のどちらかの表参照に配列型の列を定義した表を指定する場合，その配列型の列は，そのFULL OUTER JOIN のON 探索条件だけに指定できます。

7. FULL OUTER JOIN の両側の表参照には，すべての列が配列型の列である表を指定できません。

8. 表の結合方式にハッシュジョインが選択され，かつハッシュジョインにハッシュフィルタが適用される場合は，適切な大きさのハッシュフィルタ領域が必要になります。ハッシュフィルタ領域の大きさは，サーバ定義またはクライアント定義のadb_sql_exe_hashflt_area_size オペランドで指定します。

9. HADB サーバは，INNER JOIN またはCROSS JOIN を等価なコンマ結合に変換して SQL 文を実行することがあります。コンマ結合については，「7.5.1　FROM 句の指定形式および規則」の「(2)　指定形式の説明」を参照してください。

(4)　例題

例題 1（INNER JOIN の例）

顧客表（USERSLIST）と販売履歴表（SALESLIST）から，商品コード（PUR-CODE）がP001 の商品を購入したことがある顧客の一覧（顧客 ID，名前）を，重複を除いて検索します。

```
SELECT DISTINCT "USERSLIST"."USERID","NAME"
    FROM "USERSLIST" INNER JOIN "SALESLIST"
            ON "USERSLIST"."USERID"="SALESLIST"."USERID"
        WHERE "SALESLIST"."PUR-CODE"='P001'
```

下線部分が結合表（内結合）の指定です。

■検索結果の例

USERID	NAME
U00212	Maria Gomez
U00358	Nancy White

■検索対象の表

USERSLIST

USERID	NAME
U00555	Mike Johnson
U00358	Nancy White
U00212	Maria Gomez
U00687	Taro Tanaka
U00869	Bob Clinton

SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2012-12-03
U00358	P001	1	2012-12-04
U00555	P002	5	2012-12-06
U00212	P003	10	2012-12-03
U00358	P003	2	2012-12-05
U00358	P002	6	2012-12-07
U00212	P002	12	2012-12-05
U00687	P002	8	2012-12-06
U00687	P003	5	2012-12-07
U00212	P001	6	2012-12-05
U00358	P001	9	2012-12-03
U00358	P002	3	2012-12-04

例題２（LEFT OUTER JOIN の例）

商品表（PRODUCTLIST）と販売履歴表（SALESLIST）から，2012 年 12 月の商品ごとの販売合計数を求めます。

```
SELECT "PRODUCTLIST"."PUR-NAME",SUM("SALESLIST"."PUR-NUM") AS "SUM"
    FROM "PRODUCTLIST" LEFT OUTER JOIN "SALESLIST"
       ON "PRODUCTLIST"."PUR-CODE"="SALESLIST"."PUR-CODE"
       AND "SALESLIST"."PUR-DATE" BETWEEN DATE'2012-12-01'
                                  AND DATE'2012-12-31'
    GROUP BY "PRODUCTLIST"."PUR-NAME"
```

下線部分が結合表（外結合）の指定です。

■検索結果の例

PUR-NAME	SUM
File	16
Highlighter	17
Paste	NULL
Pen	37
Scissors	NULL

販売数が0の商品は，販売合計数にナル値が格納されます。

■検索対象の表

PRODUCTLIST

PUR-CODE	PUR-NAME
P001	File
P002	Pen
P003	Highlighter
P004	Paste
P005	Scissors

SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2012-12-03
U00358	P001	1	2012-12-04
U00555	P002	5	2012-12-06
U00212	P003	10	2012-12-03
U00358	P003	2	2012-12-05
U00358	P002	6	2012-12-07
U00212	P002	12	2012-12-05
U00687	P002	8	2012-12-06
U00687	P003	5	2012-12-07
U00212	P001	6	2012-12-05
U00358	P001	9	2012-12-03
U00358	P002	3	2012-12-04

例題３（RIGHT OUTER JOIN の例）

商品表（PRODUCTLIST）と販売履歴表（SALESLIST）から，2012 年 12 月の商品ごとの販売合計数を求めます。

```
SELECT "PRODUCTLIST"."PUR-NAME",SUM("SALESLIST"."PUR-NUM") AS "SUM"
    FROM "SALESLIST" RIGHT OUTER JOIN "PRODUCTLIST"
       ON "SALESLIST"."PUR-CODE"="PRODUCTLIST"."PUR-CODE"
       AND "SALESLIST"."PUR-DATE" BETWEEN DATE'2012-12-01'
                                      AND DATE'2012-12-31'
    GROUP BY "PRODUCTLIST"."PUR-NAME"
```

下線部分が結合表（外結合）の指定です。

■検索結果の例

PUR-NAME	SUM
File	16
Highlighter	17
Paste	NULL
Pen	37
Scissors	NULL

販売数が0の商品は，販売合計数にナル値が格納されます。

■検索対象の表

PRODUCTLIST

PUR-CODE	PUR-NAME
P001	File
P002	Pen
P003	Highlighter
P004	Paste
P005	Scissors

SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2012-12-03
U00358	P001	1	2012-12-04
U00555	P002	5	2012-12-06
U00212	P003	10	2012-12-03
U00358	P003	2	2012-12-05
U00358	P002	6	2012-12-07
U00212	P002	12	2012-12-05
U00687	P002	8	2012-12-06
U00687	P003	5	2012-12-07
U00212	P001	6	2012-12-05
U00358	P001	9	2012-12-03
U00358	P002	3	2012-12-04

例題４（FULL OUTER JOIN の例）

顧客表（USERSLIST），商品表（PRODUCTLIST），および販売履歴表（SALESLIST）から，2012 年 12 月に商品を購入した顧客と販売商品の組み合わせ一覧（顧客名，商品名）を，重複を除いて求めます。

- 購入実績のない顧客については，商品名（PUR-NAME）にナル値を格納します。
- 販売個数が 0 の商品については，顧客名（NAME）にナル値を格納します。

```
SELECT DISTINCT "USERSLIST"."NAME","PRODUCTLIST"."PUR-NAME"
    FROM ("USERSLIST" LEFT OUTER JOIN "SALESLIST"
                        ON "USERSLIST"."USERID"="SALESLIST"."USERID"
                        AND "SALESLIST"."PUR-DATE" BETWEEN DATE'2012-12-01'
                                                       AND DATE'2012-12-31')
                    FULL OUTER JOIN "PRODUCTLIST"
                      ON "SALESLIST"."PUR-CODE"="PRODUCTLIST"."PUR-CODE"
```

下線部分が結合表（外結合）の指定です。

■検索結果の例

NAME	PUR-NAME	
Bob Clinton	NULL	← 購入実績のない顧客は，商品名にナル値が格納されます。
Maria Gomez	File	
Maria Gomez	Highlighter	
Maria Gomez	Pen	
Mike Johnson	Pen	
Nancy White	File	
Nancy White	Highlighter	
Nancy White	Pen	
Taro Tanaka	Highlighter	
Taro Tanaka	Pen	
NULL	Paste	← 販売個数が0の商品は，顧客名にナル値が格納されます。
NULL	Scissors	

■検索対象の表

USERSLIST

USERID	NAME
U00555	Mike Johnson
U00358	Nancy White
U00212	Maria Gomez
U00687	Taro Tanaka
U00869	Bob Clinton

PRODUCTLIST

PUR-CODE	PUR-NAME
P001	File
P002	Pen
P003	Highlighter
P004	Paste
P005	Scissors

SALESLIST

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P002	3	2012-12-03
U00358	P001	1	2012-12-04
U00555	P002	5	2012-12-06
U00212	P003	10	2012-12-03
U00358	P003	2	2012-12-05
U00358	P002	6	2012-12-07
U00212	P002	12	2012-12-05
U00687	P002	8	2012-12-06
U00687	P003	5	2012-12-07
U00212	P001	6	2012-12-05
U00358	P001	9	2012-12-03
U00358	P002	3	2012-12-04

例題５（結合指定の探索条件に副問合せを指定した例）

商品表（PRODUCTLIST）と販売履歴表（SALESLIST）から，最も多くの数量を購入した顧客の一覧を，商品ごとに求めます。1 個も購入されていない商品については，顧客 ID（USERID）および販売個数（PUR-NUM）にナル値を格納します。

```
SELECT "G"."PUR-NAME","S"."USERID","S"."PUR-NUM"
    FROM "PRODUCTLIST" "G" LEFT JOIN "SALESLIST" "S"
       ON "G"."PUR-CODE"="S"."PUR-CODE"
           AND "S"."PUR-NUM"=(
                              SELECT MAX("SMAX"."PUR-NUM")
                                FROM "SALESLIST" "SMAX"
                                  WHERE "S"."PUR-CODE"="SMAX"."PUR-CODE"
                             )
```

下線部分が結合表の指定です。

■検索結果の例

PUR-NAME	USERID	PUR-NUM
File	U00358	3
Pen	U00358	6
Highlighter	U00555	2
Paste	NULL	NULL
Scissors	U00212	2

← 販売個数が0の商品は，ナル値が格納されます。

■検索対象の表

PRODUCTLIST

PUR-CODE	PUR-NAME
P001	File
P002	Pen
P003	Highlighter
P004	Paste
P005	Scissors

SALESLIST

USERID	PUR-CODE	PUR-NUM
U00555	P001	1
U00555	P003	2
U00358	P001	3
U00358	P002	6
U00212	P005	2
U00614	P001	2

7.12.2 INNER JOIN を指定した内結合

INNER JOIN を指定した内結合の例を次に示します。

- 検索対象表

■USERSLIST

顧客ID (USERID)	顧客名 (NAME)
U00555	Mike Johnson
U00358	Nancy White
U00212	Maria Gomez

■SALESLIST

顧客ID (USERID)	商品ｺｰﾄﾞ (PUR-CODE)	販売個数 (PUR-NUM)
U00555	P002	1
U00358	P001	3
U00358	P002	6
U00026	P101	25

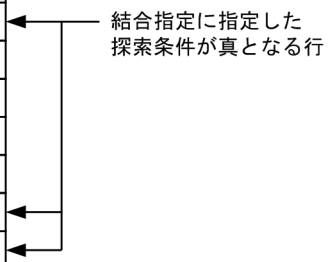
- 実行するSELECT 文

```
SELECT * FROM "USERSLIST" INNER JOIN "SALESLIST"
                          ON "USERSLIST"."USERID"="SALESLIST"."USERID"
```

両側に指定した表参照の直積（USERSLIST とSALESLIST の直積）のうち，結合指定に指定した探索条件（上記の下線部分）が真となる行が結合表の結果になります。

1.USERSLIST とSALESLIST の直積（すべての行の組み合わせ）

USERSLIST		SALESLIST		
USERID	NAME	USERID	PUR-CODE	PUR-NUM
U00555	Mike Johnson	U00555	P002	1
U00555	Mike Johnson	U00358	P001	3
U00555	Mike Johnson	U00358	P002	6
U00555	Mike Johnson	U00026	P101	25
U00358	Nancy White	U00555	P002	1
U00358	Nancy White	U00358	P001	3
U00358	Nancy White	U00358	P002	6
U00358	Nancy White	U00026	P101	25
U00212	Maria Gomez	U00555	P002	1
U00212	Maria Gomez	U00358	P001	3
U00212	Maria Gomez	U00358	P002	6
U00212	Maria Gomez	U00026	P101	25



USERSLIST の各行に対して，SALESLIST のすべての行を組み合わせます。

2. 検索結果

USERSLIST		SALESLIST		
USERID	NAME	USERID	PUR-CODE	PUR-NUM
U00555	Mike Johnson	U00555	P002	1
U00358	Nancy White	U00358	P001	3
U00358	Nancy White	U00358	P002	6

USERSLIST とSALESLIST の直積のうち，結合指定に指定した探索条件（"USERSLIST"."USERID"="SALESLIST"."USERID"）が真となる行が，結合表の結果になります。

この例の場合，商品の購入履歴がある顧客の一覧が取得されます。

7.12.3 LEFT OUTER JOIN を指定した外結合

LEFT OUTER JOIN を指定した外結合の例を次に示します。

- 検索対象表

■USERSLIST

顧客ID (USERID)	顧客名 (NAME)
U00555	Mike Johnson
U00358	Nancy White
U00212	Maria Gomez

■SALESLIST

顧客ID (USERID)	商品ｺｰﾄﾞ (PUR-CODE)	販売個数 (PUR-NUM)
U00555	P002	1
U00358	P001	3
U00358	P002	6
U00026	P101	25

- 実行するSELECT 文

```
SELECT * FROM "USERSLIST" LEFT OUTER JOIN "SALESLIST"
                          ON "USERSLIST"."USERID"="SALESLIST"."USERID"
```

次に示す行の和集合が結合表の結果になります。

- 両側に指定した表参照の直積（USERSLIST とSALESLIST の直積）のうち，結合指定に指定した探索条件（上記の下線部分）が真となる行
- 両側に指定した表参照の直積（USERSLIST とSALESLIST の直積）のうち，結合指定に指定した探索条件が 1 つも真とならない左側の表参照の各行に対して，右側の表参照の結果をナル値とした行

1. USERSLIST とSALESLIST の直積（すべての行の組み合わせ）

USERSLIST		SALESLIST		
USERID	NAME	USERID	PUR-CODE	PUR-NUM
U00555	Mike Johnson	U00555	P002	1
U00555	Mike Johnson	U00358	P001	3
U00555	Mike Johnson	U00358	P002	6
U00555	Mike Johnson	U00026	P101	25
U00358	Nancy White	U00555	P002	1
U00358	Nancy White	U00358	P001	3
U00358	Nancy White	U00358	P002	6
U00358	Nancy White	U00026	P101	25
U00212	Maria Gomez	U00555	P002	1
U00212	Maria Gomez	U00358	P001	3
U00212	Maria Gomez	U00358	P002	6
U00212	Maria Gomez	U00026	P101	25

結合指定に指定した探索条件が真となる行

結合指定に指定した探索条件が1つも真とならない行

USERSLIST の各行に対して，SALESLIST のすべての行を組み合わせます。

2. 検索結果

USERSLIST		SALESLIST		
USERID	NAME	USERID	PUR-CODE	PUR-NUM
U00555	Mike Johnson	U00555	P002	1
U00358	Nancy White	U00358	P001	3
U00358	Nancy White	U00358	P002	6
U00212	Maria Gomez	NULL	NULL	NULL

1. 探索条件が真となる行（INNER JOINを指定したときの結果と同じ行）

2. 探索条件が1つも真とならない行が取得され，SALESLISTの各列にナル値が格納されます。

この例の場合，次の一覧が取得されます。

1. 商品の購入履歴がある顧客の一覧（INNER JOIN と同じ）
2. 商品の購入履歴がない顧客の一覧（顧客 ID がU00212 の行）

7.12.4 RIGHT OUTER JOIN を指定した外結合

RIGHT OUTER JOIN を指定した外結合の例を次に示します。

- **検索対象表**

■USERSLIST

顧客ID (USERID)	顧客名 (NAME)
U00555	Mike Johnson
U00358	Nancy White
U00212	Maria Gomez

■SALESLIST

顧客ID (USERID)	商品ｺｰﾄﾞ (PUR-CODE)	販売個数 (PUR-NUM)
U00555	P002	1
U00358	P001	3
U00358	P002	6
U00026	P101	25

- 実行するSELECT 文

```
SELECT * FROM "SALESLIST" RIGHT OUTER JOIN "USERSLIST"
                         ON "USERSLIST"."USERID"="SALESLIST"."USERID"
```

次に示す行の和集合が結合表の結果になります。

- 両側に指定した表参照の直積（SALESLIST とUSERSLIST の直積）のうち，結合指定に指定した探索条件（上記の下線部分）が真となる行
- 両側に指定した表参照の直積（SALESLIST とUSERSLIST の直積）のうち，結合指定に指定した探索条件が 1 つも真とならない右側の表参照の各行に対して，左側の表参照の結果をナル値とした行

1.USERSLIST とSALESLIST の直積（すべての行の組み合わせ）

SALESLIST			USERSLIST		
USERID	PUR-CODE	PUR-NUM	USERID	NAME	
U00555	P002	1	U00555	Mike Johnson	← 結合指定に指定した探索条件が真となる行
U00358	P001	3	U00555	Mike Johnson	
U00358	P002	6	U00555	Mike Johnson	
U00026	P101	25	U00555	Mike Johnson	
U00555	P002	1	U00358	Nancy White	
U00358	P001	3	U00358	Nancy White	←
U00358	P002	6	U00358	Nancy White	←
U00026	P101	25	U00358	Nancy White	
U00555	P002	1	U00212	Maria Gomez	┐
U00358	P001	3	U00212	Maria Gomez	│ 結合指定に指定した探索条件が1つも真とならない行
U00358	P002	6	U00212	Maria Gomez	│
U00026	P101	25	U00212	Maria Gomez	┘

USERSLIST の各行に対して，SALESLIST のすべての行を組み合わせます。

2.検索結果

SALESLIST			USERSLIST		
USERID	PUR-CODE	PUR-NUM	USERID	NAME	
U00555	P002	1	U00555	Mike Johnson	┐ 1. 探索条件が真となる行（INNER JOINを指定したときの結果と同じ行）
U00358	P001	3	U00358	Nancy White	│
U00358	P002	6	U00358	Nancy White	┘
NULL	NULL	NULL	U00212	Maria Gomez	← 2. 探索条件が1つも真とならない行が取得され，SALESLISTの各列にナル値が格納されます。

この例の場合，次の一覧が取得されます。

1.商品の購入履歴がある顧客の一覧（INNER JOIN と同じ）

2. 商品の購入履歴がない顧客の一覧（顧客 ID がU00212 の行）

7.12.5 FULL OUTER JOIN を指定した外結合

FULL OUTER JOIN を指定した外結合の例を次に示します。

- 検索対象表

■USERSLIST

顧客ID (USERID)	顧客名 (NAME)
U00555	Mike Johnson
U00358	Nancy White
U00212	Maria Gomez

■SALESLIST

顧客ID (USERID)	商品ｺｰﾄﾞ (PUR-CODE)	販売個数 (PUR-NUM)
U00555	P002	1
U00358	P001	3
U00358	P002	6
U00026	P101	25

- 実行するSELECT 文

```
SELECT * FROM "SALESLIST" FULL OUTER JOIN "USERSLIST"
                          ON "USERSLIST"."USERID"="SALESLIST"."USERID"
```

次の行の和集合が結合表の結果になります。

- 両側に指定した表参照の直積（SALESLIST とUSERSLIST の直積）のうち，結合指定に指定した探索条件（上記の下線部分）が真となる行
- 両側に指定した表参照の直積（SALESLIST とUSERSLIST の直積）のうち，結合指定に指定した探索条件が 1 つも真とならない左側の表参照の各行に対して，右側の表参照の結果をナル値とした行
- 両側に指定した表参照の直積（SALESLIST とUSERSLIST の直積）のうち，結合指定に指定した探索条件が 1 つも真とならない右側の表参照の各行に対して，左側の表参照の結果をナル値とした行

1. USERSLIST とSALESLIST の直積（すべての行の組み合わせ）

SALESLIST			USERSLIST	
USERID	PUR-CODE	PUR-NUM	USERID	NAME
U00555	P002	1	U00555	Mike Johnson
U00358	P001	3	U00555	Mike Johnson
U00358	P002	6	U00555	Mike Johnson
U00026	P101	25	U00555	Mike Johnson
U00555	P002	1	U00358	Nancy White
U00358	P001	3	U00358	Nancy White
U00358	P002	6	U00358	Nancy White
U00026	P101	25	U00358	Nancy White
U00555	P002	1	U00212	Maria Gomez
U00358	P001	3	U00212	Maria Gomez
U00358	P002	6	U00212	Maria Gomez
U00026	P101	25	U00212	Maria Gomez

USERSLIST の各行に対して，SALESLIST のすべての行を組み合わせます。

2. 探索条件が真となる行を取得

SALESLIST			USERSLIST	
USERID	PUR-CODE	PUR-NUM	USERID	NAME
U00555	P002	1	U00555	Mike Johnson
U00358	P001	3	U00555	Mike Johnson
U00358	P002	6	U00555	Mike Johnson
U00026	P101	25	U00555	Mike Johnson
U00555	P002	1	U00358	Nancy White
U00358	P001	3	U00358	Nancy White
U00358	P002	6	U00358	Nancy White
U00026	P101	25	U00358	Nancy White
U00555	P002	1	U00212	Maria Gomez
U00358	P001	3	U00212	Maria Gomez
U00358	P002	6	U00212	Maria Gomez
U00026	P101	25	U00212	Maria Gomez

SALESLIST			USERSLIST	
USERID	PUR-CODE	PUR-NUM	USERID	NAME
U00555	P002	1	U00555	Mike Johnson
U00358	P001	3	U00358	Nancy White
U00358	P002	6	U00358	Nancy White

SALESLIST とUSERSLIST の直積のうち，結合指定に指定した探索条件が真となる行（INNER JOIN を指定したときの結果と同じ行）を取得します。

3. 探索条件が 1 つも真とならない左側の表参照の各行に対して，右側の表参照の結果をナル値とした行を取得

SALESLIST			USERSLIST	
USERID	PUR-CODE	PUR-NUM	USERID	NAME
U00555	P002	1	U00555	Mike Johnson
U00358	P001	3	U00555	Mike Johnson
U00358	P002	6	U00555	Mike Johnson
U00026	P101	25	U00555	Mike Johnson
U00555	P002	1	U00358	Nancy White
U00358	P001	3	U00358	Nancy White
U00358	P002	6	U00358	Nancy White
U00026	P101	25	U00358	Nancy White
U00555	P002	1	U00212	Maria Gomez
U00358	P001	3	U00212	Maria Gomez
U00358	P002	6	U00212	Maria Gomez
U00026	P101	25	U00212	Maria Gomez

SALESLIST			USERSLIST	
USERID	PUR-CODE	PUR-NUM	USERID	NAME
U00026	P101	25	NULL	NULL

SALESLIST とUSERSLIST の直積のうち，結合指定に指定した探索条件が 1 つも真とならない左側の表参照（SALESLIST）の各行に対して，右側の表参照（USERSLIST）の結果をナル値とした行を取得します。この例の場合，SALESLIST のUSERID 列の値がU00026 の行が該当します。

4. 探索条件が 1 つも真とならない右側の表参照の各行に対して，左側の表参照の結果をナル値とした行を取得

SALESLIST			USERSLIST	
USERID	PUR-CODE	PUR-NUM	USERID	NAME
U00555	P002	1	U00555	Mike Johnson
U00358	P001	3	U00555	Mike Johnson
U00358	P002	6	U00555	Mike Johnson
U00026	P101	25	U00555	Mike Johnson
U00555	P002	1	U00358	Nancy White
U00358	P001	3	U00358	Nancy White
U00358	P002	6	U00358	Nancy White
U00026	P101	25	U00358	Nancy White
U00555	P002	1	U00212	Maria Gomez
U00358	P001	3	U00212	Maria Gomez
U00358	P002	6	U00212	Maria Gomez
U00026	P101	25	U00212	Maria Gomez

SALESLIST			USERSLIST	
USERID	PUR-CODE	PUR-NUM	USERID	NAME
NULL	NULL	NULL	U00212	Maria Gomez

SALESLIST とUSERSLIST の直積のうち，結合指定に指定した探索条件が 1 つも真とならない右側の表参照（USERSLIST）の各行に対して，左側の表参照（SALESLIST）の結果をナル値とした行を取得します。この例の場合，USERSLIST のUSERID 列の値がU00212 の行が該当します。

5. 検索結果

SALESLIST			USERSLIST		
USERID	PUR-CODE	PUR-NUM	USERID	NAME	
U00555	P002	1	U00555	Mike Johnson	2. で取得した行
U00358	P001	3	U00358	Nancy White	2. で取得した行
U00358	P002	6	U00358	Nancy White	2. で取得した行
U00026	P101	25	NULL	NULL	← 3. で取得した行
NULL	NULL	NULL	U00212	Maria Gomez	← 4. で取得した行

この例の場合，次の一覧が取得されます。

1. 商品の購入履歴がある顧客の一覧（INNER JOIN と同じ）
2. 顧客リストにない顧客の販売履歴の一覧（顧客 ID がU00026 の行）
3. 商品の購入履歴がない顧客の一覧（顧客 ID がU00212 の行）

7.13　結合方式指定

ここでは，結合方式指定について説明します。

7.13.1　結合方式指定の指定形式および規則

結合表に指定した表参照の結合方式を指定します。結合方式については，マニュアル『HADB　AP 開発ガイド』の『表の結合方式』を参照してください。

通常は，結合方式指定を指定する必要はありません。結合方式指定を指定しない場合，HADB が自動的に結合方式を決定します。

(1)　指定形式

```
結合方式指定 : : =/*>> BY  {NEST | HASH}  〔( {LEFT | RIGHT}  FIRST)〕 <<*/
```

(2)　指定形式の説明

BY {NEST | HASH} :

NEST :

結合方式をネストループジョインとする場合に指定します。

HASH :

結合方式をハッシュジョインとする場合に指定します。

({LEFT | RIGHT} FIRST) :

LEFT FIRST :

結合表の左側に指定した表参照を外表として結合する場合に指定します。

RIGHT FIRST :

結合表の右側に指定した表参照を外表として結合する場合に指定します。

LEFT FIRST またはRIGHT FIRST を省略した場合，結合表に指定したどちらの表参照を外表にするかは HADB が自動的に決定します。

結合方式指定が適用されたかどうかは，アクセスパス情報で確認できます。確認方法については，マニュアル『HADB　AP 開発ガイド』の『ツリー表示に出力される情報』の『表の結合方式』を参照してください。

(3)　規則

1. HADB が実行できない結合方式が指定された場合，結合方式指定は無効になります。結合方式指定が無効になった場合，HADB が自動的に結合方式を決定します。

2.「/*>>」と「<<*/」で囲んだ文字列は注釈にはなりません。結合方式指定以外で指定するとエラーになります。

(4) 例題

例題 1

```
SELECT * FROM "T1" INNER JOIN /*>>BY NEST<<*/ "T2"
                    ON "T1"."C1"="T2"."C1"
```

下線部分が結合方式指定です。

上記のSELECT 文を実行した場合，表T1 と表T2 の結合方式がネストループジョインになります。外表，内表は HADB が自動的に決定します。

例題 2

```
SELECT * FROM "T1" INNER JOIN /*>>BY NEST (LEFT FIRST)<<*/ "T2"
                    ON "T1"."C1"="T2"."C1"
```

下線部分が結合方式指定です。

上記のSELECT 文を実行した場合，表T1 と表T2 の結合方式がネストループジョインになります。表T1 が外表，表T2 が内表になります。

例題 3

```
SELECT * FROM "T1" INNER JOIN /*>>BY NEST (RIGHT FIRST)<<*/ "T2"
                    ON "T1"."C1"="T2"."C1"
```

下線部分が結合方式指定です。

上記のSELECT 文を実行した場合，表T1 と表T2 の結合方式がネストループジョインになります。表T2 が外表，表T1 が内表になります。

例題 4

```
SELECT * FROM "T1" INNER JOIN /*>>BY HASH<<*/ "T2"
                    ON "T1"."C1"="T2"."C1"
```

下線部分が結合方式指定です。

上記のSELECT 文を実行した場合，表T1 と表T2 の結合方式がハッシュジョインになります。外表，内表は HADB が自動的に決定します。

例題 5

```
SELECT * FROM "T1" INNER JOIN /*>>BY HASH (LEFT FIRST)<<*/ "T2"
                    ON "T1"."C1"="T2"."C1"
```

下線部分が結合方式指定です。

上記のSELECT 文を実行した場合，表T1 と表T2 の結合方式がハッシュジョインになります。表T1 が外表，表T2 が内表になります。

例題６

```
SELECT * FROM "T1" INNER JOIN /*>>BY HASH (RIGHT FIRST)<<*/ "T2"
                    ON "T1"."C1"="T2"."C1"
```

下線部分が結合方式指定です。

上記のSELECT 文を実行した場合，表T1 と表T2 の結合方式がハッシュジョインになります。表T2 が外表，表T1 が内表になります。

7.14　インデクス指定

ここでは，インデクス指定について説明します。

7.14.1　インデクス指定の指定形式および規則

実表の検索時に使用するインデクスを指定します。または，インデクスの使用抑止を指定します。

インデクス指定は，B-tree インデクスおよびテキストインデクスが対象になります。レンジインデクスは対象外になります。

なお，実表の検索時に使用するインデクスは HADB が自動的に決定するため，インデクス指定は通常指定する必要はありません。インデクスの決定規則については，マニュアル『HADB　AP 開発ガイド』の『SQL 文の実行時に使用される B-tree インデクスおよびテキストインデクス』を参照してください。

(1)　指定形式

```
インデクス指定 : : = /*>>  {WITH INDEX (インデクス名) | WITHOUT INDEX}  <<*/
```

(2)　指定形式の説明

WITH INDEX (*インデクス名*)：

インデクス指定の直前に指定されている実表の検索時に使用するインデクスを指定します。インデクス名の指定規則については，「6.1.5　名前の修飾」の「(3)　インデクス名の指定形式」を参照してください。

WITHOUT INDEX：

インデクス指定の直前に指定されている実表の検索時に，インデクスを使用しない場合に指定します。この場合，実表の検索方式にテーブルスキャンが適用されます。テーブルスキャンについては，マニュアル『HADB　AP 開発ガイド』の『テーブルスキャンとは』を参照してください。

インデクス指定が適用されたかどうかは，アクセスパス情報で確認できます。確認方法については，マニュアル『HADB　AP 開発ガイド』の『ツリー表示に出力される情報』の『インデクス指定』を参照してください。

(3)　規則

1. ビュー表にはインデクス指定は指定できません。
2. 存在しないインデクス名を指定した場合，インデクス指定は無効になります。
3. 次の条件をすべて満たす場合，インデクス指定は無効になります。
   - ナル値除外指定を指定した B-tree インデクスを*インデクス名*に指定した

- B-tree インデクスの検索範囲にナル値が含まれる条件を指定した

4. *インデクス名*にテキストインデクスを指定しても，HADB がテキストインデクスを有効に使用できないと判断したときは，インデクス指定は無効になります。例えば，テキストインデクスで評価できる LIKE 述語が指定されていないケースなどが該当します。
5. インデクス指定が無効になった場合，検索時に使用されるインデクスは HADB が自動的に決定します。検索時に使用されるインデクスについては，マニュアル『HADB　AP 開発ガイド』の『AP の性能向上に関する設計』の『SQL 文の実行時に使用される B-tree インデクスおよびテキストインデクス』を参照してください。
6. 「/*>>」と「<<*/」で囲んだ文字列は注釈にはなりません。インデクス指定以外で指定するとエラーになります。

   （例）

```
SELECT * FROM "T1"  /*>> comment <<*/
```

   上記の下線部分は注釈になりません。そのため，上記の SQL 文は構文解析エラーになります。
7. 次のように指定した場合，/*と*/の間は注釈と見なされるため，インデクス指定にはなりません。

```
SELECT * FROM "T1" /* WITH INDEX ("IDX01") */
```

(4)　例題

インデクス指定の例を次に示します。

従業員表（EMPLOYEE）には，次のインデクスが定義されているとします。

- B-tree インデクス（BTREE_IDX）：インデクス構成列はSCODE
- テキストインデクス（TEXT_IDX）：インデクス構成列はADDRESS

例題 1

B-tree インデクスBTREE_IDX を使用して，従業員表（EMPLOYEE）を検索します。

```
SELECT "NAME" FROM "EMPLOYEE" /*>> WITH INDEX ("BTREE_IDX") <<*/
    WHERE "SCODE" = 'S003' AND "ADDRESS" LIKE '%TOKYO%'
```

下線部分がインデクス指定です。

例題 2

テキストインデクス（TEXT_IDX）を使用して，従業員表（EMPLOYEE）を検索します。

```
SELECT "NAME" FROM "EMPLOYEE" /*>> WITH INDEX ("TEXT_IDX") <<*/
    WHERE "SCODE" = 'S003' AND "ADDRESS" LIKE '%TOKYO%'
```

下線部分がインデクス指定です。

例題 3

インデクスを使用しないで，従業員表（EMPLOYEE）を検索します。

```
SELECT "NAME" FROM "EMPLOYEE" /*>> WITHOUT INDEX <<*/
    WHERE "SCODE" = 'S003' AND "ADDRESS" LIKE '%TOKYO%'
```

下線部分がインデクス指定です。

なお，SCODE 列にレンジインデクスが定義されている場合，上記のSELECT 文を実行した際，レンジインデクスだけが使用されます。検索時に使用されるレンジインデクスの条件については，マニュアル『HADB　AP 開発ガイド』の『SQL 文の実行時に使用されるレンジインデクス』を参照してください。

7.15　システム定義関数

ここでは，システム定義関数について説明します。

7.15.1　システム定義関数の指定形式および規則

HADB が提供する関数を**システム定義関数**といいます。

(1)　指定形式

システム定義関数：：＝ {*ADB_AUDITREAD関数* | *ADB_CSVREAD関数*}

(2)　指定形式の説明

ADB_AUDITREAD 関数：

ADB_AUDITREAD 関数とは，監査証跡ファイル中の監査証跡を，HADB サーバが検索できる表形式のデータ集合に変換する関数です。ADB_AUDITREAD 関数については，「7.15.2　ADB_AUDITREAD 関数」を参照してください。

ADB_CSVREAD 関数：

ADB_CSVREAD 関数とは，CSV ファイル中のデータを，HADB サーバが検索できる表形式のデータ集合に変換する関数です。ADB_CSVREAD 関数については，「7.15.3　ADB_CSVREAD 関数」を参照してください。

(3)　規則

1. システム定義関数に指定できる引数の数は，最大 1,000 個です。

7.15.2　ADB_AUDITREAD 関数

監査証跡ファイル中の監査証跡を，HADB サーバが検索できる表形式のデータ集合に変換します。

メモ

- 監査証跡機能の概要については，マニュアル『HADB　システム構築・運用ガイド』の『監査証跡機能』を参照してください。
- 監査証跡を検索するときの運用については，マニュアル『HADB　システム構築・運用ガイド』の『監査証跡機能の定期運用』を参照してください。

(1) 指定形式

```
ADB_AUDITREAD関数 ： ：＝
     〔MASTER.〕ADB_AUDITREAD(〔監査証跡ファイルのパス名指定〕)

        監査証跡ファイルのパス名指定 ： ：＝マルチ集合値式
```

(2) 指定形式の説明

監査証跡ファイルのパス名指定：

ADB_AUDITREAD 関数の入力情報となる監査証跡ファイルのパス名を，マルチ集合値式の形式で指定します。マルチ集合値式については，「7.16 マルチ集合値式」を参照してください。

指定規則を次に示します。

- マルチ集合値式の結果のデータ型は，文字データになるようにしてください。
- *監査証跡ファイルのパス名指定*に指定する監査証跡ファイルのパス名は，絶対パスで指定してください。
- *監査証跡ファイルのパス名指定*に指定する監査証跡ファイルのパス名は，存在するファイルを指定してください。

[マルチノード機能]

- マルチノード機能を使用している場合は，*監査証跡ファイルのパス名指定*は省略できません。

■*監査証跡ファイルのパス名指定*の例

例中の下線部分が，*監査証跡ファイルのパス名指定*です。

なお，ここでは，代表的な指定例を説明しています。監査証跡ファイルのパス名の指定規則については，「(4) 規則」の「(b) 監査証跡ファイルのパス名の指定規則」を参照してください。

(例 1)

```
ADB_AUDITREAD(MULTISET['/audit/adbaud-20170401-123000-159.aud','/audit/adbaud-20170415-123000-952.aud'])
```

この例の場合，2 つの監査証跡ファイルのパス名が指定されています。この 2 つの監査証跡ファイルがADB_AUDITREAD 関数の入力情報になります。

(例 2)

```
ADB_AUDITREAD(MULTISET['/audit/*.aud'])
```

この例の場合，特殊文字*を使った監査証跡ファイルのパス名が指定されています。この場合，/audit ディレクトリ下に格納されている監査証跡ファイル（ファイル拡張子がaud のファイル）が，ADB_AUDITREAD 関数の入力情報になります。ただし，現用の監査証跡ファイルは，ADB_AUDITREAD 関数の入力情報になりません。

(例 3)

```
ADB_AUDITREAD(MULTISET['/audit1/*.aud','/audit2/*.aud'])
```

この例の場合，/audit1 ディレクトリ下と/audit2 ディレクトリ下に格納されている監査証跡ファイルが，ADB_AUDITREAD 関数の入力情報になります。

（例 4）

```
ADB_AUDITREAD(MULTISET['/audit/adbaud-201707*.aud','/audit/adbaud-201708*.aud'])
```

この例の場合，/audit ディレクトリ下に格納されている監査証跡ファイルのうち，2017 年 7 月と 8 月に作成された監査証跡ファイルが，ADB_AUDITREAD 関数の入力情報になります。

重要

*監査証跡ファイルのパス名指定*に，OS のgzip コマンドで圧縮した監査証跡ファイルを指定することもできます。

（例）

```
ADB_AUDITREAD(MULTISET['/audit/*.gz'])
```

この例の場合，/audit ディレクトリ下に格納されている監査証跡ファイルを圧縮したファイル（ファイル拡張子がgz のファイル）が，ADB_AUDITREAD 関数の入力情報になります。

■*監査証跡ファイルのパス名指定*を省略した場合

*監査証跡ファイルのパス名指定*を省略した場合，監査証跡の出力先ディレクトリ（サーバ定義の adb_audit_log_path オペランドに指定したディレクトリ）下にある監査証跡ファイルが，ADB_AUDITREAD 関数の入力情報になります。ただし，次のファイルは，ADB_AUDITREAD 関数の入力情報になりません。

- 現用の監査証跡ファイル
- 監査証跡の出力先ディレクトリのサブディレクトリ下のファイル

（例）

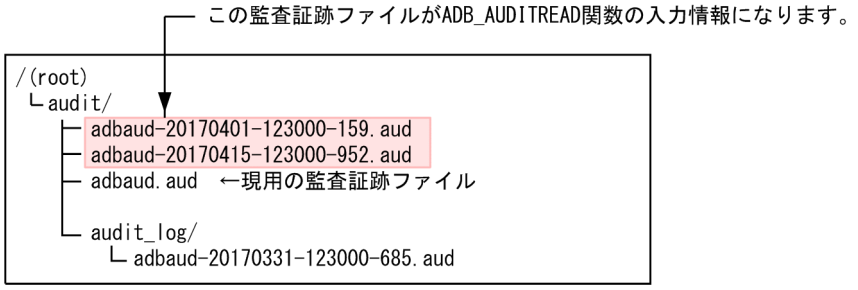
サーバ定義の指定

```
adb_audit_log_path = /audit
```

ADB_AUDITREAD 関数の指定

```
ADB_AUDITREAD()
```

サーバ定義とADB_AUDITREAD 関数が上記のように指定されている場合，ADB_AUDITREAD 関数の入力情報となる監査証跡ファイルは次のようになります。



メモ

*監査証跡ファイルのパス名指定*を省略した場合，「サーバ定義のadb_audit_log_path オペランドの指定値＋/*.aud」を，マルチ集合値式の*列挙によるマルチ集合値構成子*の形式で指定したと見なされます。なお，*は特殊文字の指定となります。

(3) 実行時に必要な権限

ADB_AUDITREAD 関数を実行する場合，監査参照権限が必要になります。

(4) 規則

(a) ADB_AUDITREAD 関数に関する規則

1. 監査証跡機能が有効なときに，ADB_AUDITREAD 関数を使用できます。
2. ADB_AUDITREAD 関数が実行されると，*監査証跡ファイルのパス名指定*に指定した監査証跡ファイル中の監査証跡を，表形式のデータ集合として返却します。ADB_AUDITREAD 関数によって返却される表形式のデータ集合の列名，列のデータ型，列に格納される情報については，マニュアル『HADB　システム構築・運用ガイド』の『監査証跡を検索するときの表関数導出表の列構成 』を参照してください。
3. 監査証跡のレコードが存在しない（ヘッダ情報だけがある）監査証跡ファイルを指定した場合，そのファイルに対する表関数導出表の結果は空集合になります。監査証跡のヘッダ情報も存在しない 0 バイトのファイルを指定した場合は，SQL 文がエラーになります。

   この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。

(b) 監査証跡ファイルのパス名の指定規則

1. *監査証跡ファイルのパス名指定*には，ADB_AUDITREAD 関数の入力情報とする監査証跡ファイルのパス名を指定します。監査証跡ファイルのパス名は絶対パスで指定してください。
2. 監査証跡ファイルのパス名のファイル名の部分に，次の特殊文字を指定できます。
   - *（アスタリスク）

     0 文字以上の任意の長さの文字列を意味します。
   - ?（疑問符）

     任意の 1 文字を意味します。

指定例 1：

```
ADB_AUDITREAD(MULTISET['/audit/*.aud'])
```

上記の例の場合，/audit ディレクトリ下の監査証跡ファイル（拡張子がaud のファイル）が ADB_AUDITREAD 関数の入力情報になります。

指定例 2：

```
ADB_AUDITREAD(MULTISET['/audit/adbaud-201704*.aud','/audit/adbaud-201705*.aud'])
```

上記の例の場合，次のような名称の監査証跡ファイルがADB_AUDITREAD 関数の入力情報になります。

- /audit/adbaud-20170401-123000-159.aud
- /audit/adbaud-20170415-123000-952.aud
- /audit/adbaud-20170501-123000-599.aud

重要

監査証跡ファイルのファイル名に*または?を指定した場合，その文字は特殊文字として扱われます。監査証跡ファイルのディレクトリ名に，*または?を指定した場合，その文字は通常文字として扱われます。

（例）

```
ADB_AUDITREAD(MULTISET['/audit*/adbaud-201706*.aud'])
```

上記の例の場合，ディレクトリ部分の指定は通常文字として扱われるため，ディレクトリ名は/audit*となります。ファイル部分の指定は特殊文字として扱われます。したがって，次の監査証跡ファイルなどが対象のファイルになります。

- /audit*/adbaud-20170601-123000-159.aud
- /audit*/adbaud-20170602-165522-656.aud

3. 監査証跡ファイルのパス名に特殊文字を指定した結果，入力情報となる監査証跡ファイルがない場合（*監査証跡ファイルのパス名指定*の結果が空集合の場合）は，SQL 文がエラーになります。
4. 監査証跡ファイルのパス名に特殊文字を指定した場合，次の監査証跡ファイルはADB_AUDITREAD 関数の入力情報になりません。
   - 現用の監査証跡ファイル
   - 監査証跡ファイルのパス名に指定したディレクトリのサブディレクトリ下の監査証跡ファイル
5. 監査証跡ファイルのパス名に特殊文字を指定した結果，入力情報となる監査証跡ファイル数が 65,536 以上になった場合，SQL 文がエラーになります。なお，現用の監査証跡ファイルは，特殊文字を指定したときの対象のファイルにはならないため，カウント対象外になります。
6. 次に示す以外のファイルを指定した場合，SQL 文がエラーになります。

- 監査証跡ファイル
- OS のgzip コマンドで圧縮した監査証跡ファイル

この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。

7. 監査証跡ファイルのパス名に，現用の監査証跡ファイルは指定できません。
8. 監査証跡ファイルのパス名の先頭または最後に空白がある場合，その空白は取り除かれて処理されます。

(例)

’△△△/audit/adbaud-20170420-123030-159.aud’ → ’/audit/adbaud-20170420-123030-159.aud’

’/audit/adbaud-20170420-123030-159.aud△△△’ → ’/audit/adbaud-20170420-123030-159.aud’

’△△△/audit/adbaud-20170420-123030-159.aud△△△’ → ’/audit/adbaud-20170420-123030-159.aud’

’△△△/audit/adbaud-20170420△-123030-159.aud△△△’ → ’/audit/adbaud-20170420△-123030-159.aud’

△：空白

> **重要**
>
> 監査証跡ファイルのパス名の先頭または最後に空白を指定しないでください。パス名の先頭または最後に空白を指定した場合，上記のように空白が取り除かれるため，意図しないパス名になるおそれがあります。

9. 監査証跡ファイルのパス名の長さの上限は 1,024 バイトです。1,025 バイト以上の監査証跡ファイルのパス名を指定すると，SQL 文がエラーになります。ただし，パス名の長さの上限チェックは，HADB サーバが次の処理を実行したあとに行われます。
   - 監査証跡ファイルのパス名の先頭または最後に空白がある場合，その空白を取り除く処理
   - 監査証跡ファイルのパス名に特殊文字が指定されている場合，入力情報となる監査証跡ファイルのパス名に置き換える処理
10. CREATE VIEW 文中にADB_AUDITREAD 関数を指定した場合，監査証跡ファイルのパス名に関する規則のチェックは，CREATE VIEW 文の実行時には実施されません。定義したビュー表を指定した SQL 文の実行時に，監査証跡ファイルのパス名に関する規則のチェックが実施され，違反している場合は SQL 文がエラーになります。

(5) 留意事項

1. 監査証跡ファイルの絶対パスに含まれる各ディレクトリに対して，HADB 管理者がアクセスできるように読み取り権限と実行権限を設定しておいてください。例えば，監査証跡ファイルが/adbmanager/audit ディレクトリ下に格納されている場合，/，/adbmanager，および/adbmanager/audit の各ディレクトリに対して，HADB 管理者がアクセスできるように読み取り権限と実行権限を設定しておく必要があります。また，監査証跡ファイルに対して，HADB 管理者がアクセスできるように読み取り権限を設定しておいてください。

2. ADB_AUDITREAD 関数を指定した SQL 文を実行した場合，HADB サーバは*監査証跡ファイルのパス名指定*に指定された監査証跡ファイルをオープンして，監査証跡を読み込みます。そのため，ADB_AUDITREAD 関数を指定した SQL 文の実行中は，*監査証跡ファイルのパス名指定*に指定した監査証跡ファイルを移動したり，削除したりしないでください。
3. 監査証跡ファイルのパス名に特殊文字を指定した場合，SQL 文の前処理時に，検索対象の監査証跡ファイルのパス名が抽出されます。SQL 文の前処理時に抽出された監査証跡ファイルが，SQL 文の実行時に存在しない場合，そのファイルは検索対象になりません（SQL 文はエラーになりません）。
4. ADB_AUDITREAD 関数の指定規則のうち，一部の規則は SQL 文の実行時にチェックされます（SQL 文の前処理時にはチェックされません）。SQL 文の実行時にチェックされる規則には，「**この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。**」の 1 文を記載しています。

(6) 例題

例題 1

2017 年 4 月 1 日～2017 年 4 月 30 日の間に，HADB サーバにアクセスした HADB ユーザの一覧を出力します。2017 年 4 月 1 日～2017 年 4 月 30 日に出力された監査証跡を格納している監査証跡ファイルを，/audit ディレクトリ下に保存しているとします。

```
SELECT DISTINCT "USER_NAME"
    FROM TABLE(ADB_AUDITREAD(MULTISET['/audit/*.aud'])) "DT"
       WHERE "EXEC_TIME" BETWEEN TIMESTAMP'2017/04/01 00:00:00.000000'
                             AND TIMESTAMP'2017/04/30 23:59:59.999999'
```

下線部分がADB_AUDITREAD 関数の指定です。

USER_NAME には HADB ユーザの認可識別子が格納されています。EXEC_TIME には，HADB ユーザが操作を行った時間が格納されています。

例題 2

2017 年 4 月 1 日～2017 年 4 月 30 日の間に，HADB サーバにアクセスした HADB ユーザの一覧を出力します。2017 年 4 月 1 日～2017 年 4 月 30 日に出力された監査証跡を格納している監査証跡ファイルを，サーバ定義のadb_audit_log_path オペランドに指定したディレクトリ下に保存しているとします。

```
SELECT DISTINCT "USER_NAME"
    FROM TABLE(ADB_AUDITREAD()) "DT"
       WHERE "EXEC_TIME" BETWEEN TIMESTAMP'2017/04/01 00:00:00.000000'
                             AND TIMESTAMP'2017/04/30 23:59:59.999999'
```

下線部分がADB_AUDITREAD 関数の指定です。

7.15.3 ADB_CSVREAD 関数

CSV ファイル中のデータを，HADB サーバが検索できる表形式に変換します。

メモ

- CSV ファイル中のデータの検索の概要については，マニュアル『HADB　システム構築・運用ガイド』の『CSV ファイル中のデータの検索』を参照してください。
- CSV ファイル中のデータを検索するときの運用については，マニュアル『HADB　システム構築・運用ガイド』の『CSV ファイル中のデータを検索するときの運用』を参照してください。

(1)　指定形式

```
ADB_CSVREAD関数 ： ：＝
    〔MASTER.〕 ADB_CSVREAD(CSVファイルのパス名指定,関数オプション指定)

      CSVファイルのパス名指定 ： ：＝マルチ集合値式

      関数オプション指定 ： ：='関数オプション〔;関数オプション〕…〔;〕'
        関数オプション ： ：＝ {圧縮形式オプション | 指定列オプション
                              | バイナリ文字列形式オプション | 囲み文字指定オプション
                              | 区切り文字指定オプション}
```

(2)　指定形式の説明

CSV ファイルのパス名指定：

ADB_CSVREAD 関数の入力情報となる CSV ファイルのパス名を，マルチ集合値式の形式で指定します。マルチ集合値式については，「7.16　マルチ集合値式」を参照してください。

指定規則を次に示します。

- マルチ集合値式の結果のデータ型は，文字データになるようにしてください。
- *CSV ファイルのパス名指定*に指定する CSV ファイルのパス名は，絶対パスで指定してください。

  この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。
- *CSV ファイルのパス名指定*に指定する CSV ファイルのパス名は，存在するファイルを指定してください。

  この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。

関数オプション指定：

次に示すADB_CSVREAD 関数のオプションを指定します。

- 圧縮形式オプション
- 指定列オプション
- バイナリ文字列形式オプション
- 囲み文字指定オプション
- 区切り文字指定オプション

各オプションについては，「(3)　圧縮形式オプション」以降で説明します。

指定規則を次に示します。

- *関数オプション指定*は，文字列定数の形式で指定します。文字列定数の記述形式については，「6.3.2 定数の記述形式」を参照してください。
- *関数オプション指定*全体をアポストロフィ（’）で囲んでください。
- 複数の関数オプションを指定する場合は，セミコロン（；）で区切って指定してください。
- 関数オプションの指定順序に決まりはありません。
- 同じ関数オプションを重複して指定できません。
- 関数オプション中に指定した半角英小文字は，半角英大文字として扱われます。ただし，囲み文字および区切り文字については，半角英小文字と半角英大文字を区別します。
- 分離符号は，オプションおよび特殊記号（「,」，「-」，「:」，「;」，「=」，「NL」，「CR」，「半角空白」，「全角空白」）の前後に指定できます。

(3)　圧縮形式オプション

圧縮形式オプションには，CSV ファイルの圧縮形式を指定します。圧縮形式オプションは省略できません。

(a)　指定形式

```
COMPRESSION_FORMAT= {GZIP | NONE}
```

(b)　指定形式の説明

GZIP：

CSV ファイルが GZIP 形式で圧縮されているときに指定します。

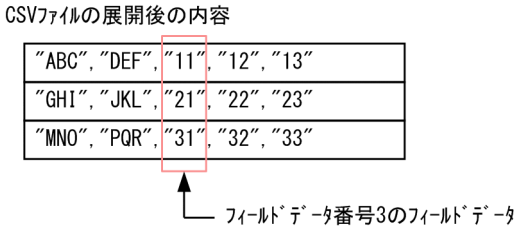
NONE：

CSV ファイルが圧縮されていないときに指定します。

(4)　指定列オプション

指定列オプションには，CSV ファイル中のフィールドデータのフィールドデータ番号を指定します。フィールドデータ番号とは，CSV ファイル中のフィールドデータの並び順を表す番号です。レコードの先頭のフィールドデータを 1 とし，順番に 2，3，…となります。

（例）



ここで指定したフィールドデータ番号に対応したフィールドデータが，ADB_CSVREAD 関数によって取り出されます。

(a) 指定形式

```
FIELD_NUM=フィールドデータ番号指定〔,フィールドデータ番号指定〕…
```

(b) 指定形式の説明

フィールドデータ番号指定：

取り出し対象とするフィールドデータのフィールドデータ番号を指定します。

複数のフィールドデータ番号を指定する場合は，コンマ（ , ）で区切って指定してください。また，フィールドデータ番号を「1-5」のように範囲指定することもできます。

（例）

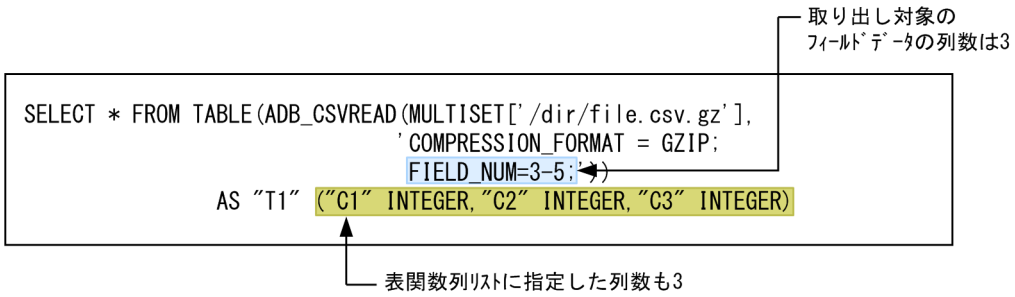
```
FIELD_NUM=3             ...1
FIELD_NUM=1,3,4,6       ...2
FIELD_NUM=1,3-5,8-10    ...3
```

1. フィールドデータ番号 3 のフィールドデータが取り出し対象になります。
2. フィールドデータ番号 1，3，4，6 のフィールドデータが取り出し対象になります。
3. フィールドデータ番号 1，3～5，8～10 のフィールドデータが取り出し対象になります。

(c) 規則

1. 取り出し対象のフィールドデータの列数と，表関数列リストに指定した列数を同じにする必要があります。

（例）



表関数列リストについては，「7.11.1　表参照の指定形式」の「(2)　指定形式の説明」を参照してください。

2. フィールドデータ番号には，1～30,000 の符号なし整数定数が指定できます。

3. フィールドデータ番号は重複して指定できません。

（例）エラーになる例

```
FIELD_NUM=1,2,2 ←2が重複しているためエラー
FIELD_NUM=1,1-3 ←1が重複しているためエラー
```

4. 対象となるフィールドデータの列数が，4,000 以下になるようにしてください。

（例）エラーになる例

```
FIELD_NUM=1-4001 ←対象となるフィールドデータの列数が4,001のためエラー
```

5. フィールドデータ番号に対応するフィールドデータがない場合，エラーになります。例えば，フィールドデータが 5 列のときに，次のような指定をした場合，エラーになります。

（例）エラーになる例

```
FIELD_NUM=6
FIELD_NUM=1-7
```

フィールドデータが 5 列しかないため，フィールドデータ番号 6 以降は指定できません。

なお，この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。

6. *指定列オプション*を省略した場合，「1～表関数列リストに指定した列数」のフィールドデータ番号が指定されたと仮定されます。フィールドデータ番号に対応するフィールドデータがない場合は，表関数導出表にはナル値が格納されます。

（例）

```
SELECT * FROM TABLE(ADB_CSVREAD(MULTISET['/dir/file.csv.gz'],
                                'COMPRESSION_FORMAT = GZIP;'))
               AS "T1" ("C1" INTEGER,"C2" INTEGER,"C3" INTEGER,
                        "C4" INTEGER,"C5" INTEGER)
```

下線部分が，表関数列リストの指定です。

CSVファイル(/dir/file.csv.gz)の展開後の内容

"11","12","13"
"21","22","23"
"31","32","33"

表関数導出表

C1列	C2列	C3列	C4列	C5列
11	12	13	ナル値	ナル値
21	22	23	ナル値	ナル値
31	32	33	ナル値	ナル値

CSV ファイル中のフィールドデータは 3 列です。一方，表関数列リストに指定した列数は 5 列です。そのため，表関数導出表のC4 列とC5 列にはナル値が格納されます。

(5) バイナリ文字列形式オプション

バイナリ文字列形式オプションには，CSV ファイル中のバイナリデータ（BINARY，VARBINARY）の形式を指定します。

(a) 指定形式

```
BINARY_STRING_FORMAT=フィールドデータ番号指定:バイナリ形式指定
                    〔,フィールドデータ番号指定:バイナリ形式指定〕…

  バイナリ形式指定：：＝ {HEX | BIN}
```

(b) 指定形式の説明

フィールドデータ番号指定：

CSV ファイル中のバイナリデータのフィールドデータ番号を指定します。フィールドデータ番号指定の指定規則については，「(4) 指定列オプション」を参照してください。

ここで指定するフィールドデータ番号は，指定列オプション（FIELD_NUM）で指定したフィールドデータ番号の中から指定する必要があります。

（例）

```
FIELD_NUM=1-5;BINARY_STRING_FORMAT=1:BIN,4-5:HEX;
```

指定列オプションを省略した場合は，表関数導出表の列数以下の整数を指定してください。

バイナリ形式指定：

バイナリデータの形式を指定します。

HEX：

バイナリデータが 16 進形式の場合に指定します。

BIN：

バイナリデータが 2 進形式の場合に指定します。

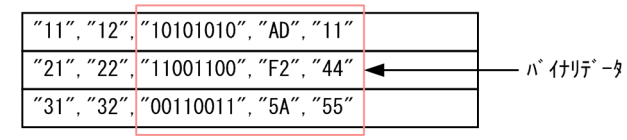
バイナリ文字列形式オプションの指定例を次に示します。

（例）

```
SELECT * FROM TABLE(ADB_CSVREAD(MULTISET['/dir/file.csv.gz'],
                                'COMPRESSION_FORMAT = GZIP;
                                 BINARY_STRING_FORMAT=3:BIN,4:HEX;'))
              AS "T1" ("C1" INTEGER,"C2" INTEGER,
                       "C3" BINARY(1),"C4" BINARY(1),"C5" BINARY(1))
```

下線部分がバイナリ文字列形式オプションの指定です。

CSVﾌｧｲﾙ(/dir/file.csv.gz)の展開後の内容



表関数導出表

C1列	C2列	C3列	C4列	C5列
11	12	0xAA	0xAD	0x11
21	22	0xCC	0xF2	0x44
31	32	0x33	0x5A	0x55

BINARY型の列

[説明]

- C3 列～C5 列に対応するフィールドデータはバイナリデータです。
- C3 列に対応するバイナリデータは 2 進形式のため，バイナリ形式指定にBIN を指定します。
- C4 列に対応するバイナリデータは 16 進形式のため，バイナリ形式指定にHEX を指定します。
- C5 列に対応するバイナリデータは 16 進形式のため，バイナリ形式指定を省略しています（省略値はHEX)。

(c) 規則

1. 表関数導出表にバイナリデータの列が指定されているときに，バイナリ文字列形式オプションを省略すると，バイナリ文字列形式オプションに次の指定がされていると仮定されます。
   - 表関数導出表のバイナリデータの列に対応するフィールドデータ番号指定が指定されている
   - バイナリ形式指定にHEX が指定されている
2. フィールドデータ番号に対応する表関数導出表の列のデータ型は，バイナリデータ（BINARY，VARBINARY）にしてください。

(6) 囲み文字指定オプション

囲み文字指定オプションには，CSV ファイル中のフィールドデータを囲んでいる囲み文字を指定します。

(a) 指定形式

```
ENCLOSING_CHAR= {囲み文字 | NONE}
```

(b) 指定形式の説明

囲み文字：

CSV ファイル中のフィールドデータを囲んでいる*囲み文字*を指定します。囲み文字には 1 バイトの文字を指定できます。

留意事項を次に示します。

- 次に示す文字などは，CSV ファイル中のフィールドデータの文字と重なる可能性があるため，囲み文字には適していません。

  符号（+，-），スラッシュ（/），コロン（：），ピリオド（．），|，¥，[,]，(,)，{,}，~

- 分離符号と同じ文字を*囲み文字*に指定しないでください。分離符号と同じ文字は囲み文字として見なされません（分離符号として扱われます）。そのため，囲み文字に分離符号と同じ文字を指定した場合，次に示す例のように意図しない結果となるおそれがあります。

  （例）囲み文字に分離符号である半角空白を指定した場合（△が半角空白を意味しています）

  ```
  '～;ENCLOSING_CHAR=△;'
  ```

  この例の場合，HADB は，囲み文字にセミコロン（；）が指定されたと認識します。

NONE：

CSV ファイル中のフィールドデータに，囲み文字を使用していない場合にNONE を指定します。

重要

フィールドデータ中に改行文字，または区切り文字と同じ文字がある場合は，NONE を指定しないでください。NONE を指定した場合，意図しない結果となるおそれがあります。

- 改行文字がある場合，改行文字までが 1 行のデータとして扱われます。
- 区切り文字と同じ文字がある場合，その文字が区切り文字として扱われます。

(c) 規則

1. *囲み文字指定オプション*を省略した場合，囲み文字として二重引用符（"）が仮定されます。
2. 次の文字は，囲み文字にはできません。
   - 空白，タブ，アスタリスク（*），改行（0x0A），復帰（0x0D）
   - *区切り文字指定オプション*に指定した区切り文字と同じ文字
3. 囲み文字にアポストロフィ（'）を指定する場合，アポストロフィを 2 個続けて指定してください。指定方法を次に示します。

   ```
   ENCLOSING_CHAR=''
   ```

(7) 区切り文字指定オプション

区切り文字指定オプションには，CSV ファイル中のフィールドデータを区切っている区切り文字を指定します。

(a) 指定形式

```
DELIMITER_CHAR= {区切り文字 | TAB | SP}
```

(b) 指定形式の説明

区切り文字：

CSV ファイル中のフィールドデータを区切っている*区切り文字*を指定します。区切り文字には 1 バイトの文字を指定できます。

留意事項を次に示します。

- 次に示す文字などは，CSV ファイル中のフィールドデータの文字と重なる可能性があるため，区切り文字には適していません。

  符号（+，-），スラッシュ（/），コロン（ : ），ピリオド（ . ），|，¥，[,]，(,)，{,}，~

- 分離符号と同じ文字を*区切り文字*に指定しないでください。分離符号と同じ文字は区切り文字として見なされません（分離符号として扱われます）。そのため，区切り文字に分離符号と同じ文字を指定した場合，次に示す例のように意図しない結果となるおそれがあります。

  （例）区切り文字に分離符号である半角空白を指定した場合（△が半角空白を意味しています）

  ```
  '～;DELIMITER_CHAR=△;'
  ```

  この例の場合，HADB は，区切り文字にセミコロン（ ; ）が指定されたと認識します。

TAB：

CSV ファイル中のフィールドデータを，タブで区切っている場合にTAB を指定します。

SP：

CSV ファイル中のフィールドデータを，空白で区切っている場合にSP を指定します。

(c) 規則

1. 区切り文字指定オプションを省略した場合，区切り文字としてコンマ（ , ）が仮定されます。
2. 次の文字は，区切り文字にはできません。
   - 英字（A～Z，a～z），数字（0～9），下線（_），二重引用符（"），空白，タブ，アスタリスク（*），改行（0x0A），復帰（0x0D）
   - 囲み文字指定オプションに指定した囲み文字と同じ文字
3. 区切り文字にアポストロフィ（ ' ）を指定する場合，アポストロフィを 2 個続けて指定してください。指定方法を次に示します。

   ```
   DELIMITER_CHAR=''
   ```

(8) 規則

(a) ADB_CSVREAD 関数に関する規則

CSV ファイルのパス名指定に指定したマルチ集合値式の結果が空集合の場合，表関数導出表の結果も空集合になります。

(b) CSV ファイルに関する規則

1. CSV ファイルは，次に示すどれかのファイルである必要があります。
   - OS の gzip コマンドで圧縮した GZIP 形式のファイル
   - adbexport コマンドでエクスポートした GZIP 形式の出力データファイル
   - 圧縮していない CSV ファイル

   この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。
2. CSV ファイルには，HADB 管理者に対する読み取り権限を付与してください。CSV ファイルの格納ディレクトリには，HADB 管理者に対する読み取り権限と実行権限を付与してください。

   この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。
3. CSV ファイルのパス名の先頭または最後に空白がある場合，その空白は取り除かれて処理されます。

   (例)

   '△△△/dir/file.csv.gz' → '/dir/file.csv.gz'

   '/dir/file.csv.gz△△△' → '/dir/file.csv.gz'

   '△△△/dir/file.csv.gz△△△' → '/dir/file.csv.gz'

   '△△△/dir/fi△le.csv.gz△△△' → '/dir/fi△le.csv.gz'

   △：空白

   > **重要**
   >
   > CSV ファイルのパス名の先頭または最後に空白を指定しないでください。パス名の先頭または最後に空白を指定した場合，上記のように空白が取り除かれるため，意図しないパス名になるおそれがあります。

4. CSV ファイルのパス名の長さは，パス名の前後の空白を除き 510 バイト以下である必要があります。

   この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。

(c) CSV ファイルの形式に関する規則

1. CSV ファイル中の 1 行が，表関数導出表の 1 行のデータになります。行の終わりには，X'0A'（LF），X'0D0A'（CRLF），またはX'00'の改行文字を指定してください。
2. フィールドデータとフィールドデータの間を，区切り文字で区切ってください。
3. 囲み文字で囲まれている文字列は，すべてフィールドデータとして扱われます。
4. CSV ファイルのデータは，環境変数ADBLANG に指定した文字コードで作成してください。
5. CSV ファイル中に，EOF 制御文字を指定しないでください。
6. 囲み文字を指定する場合は，区切り文字と囲み文字を連続して指定してください。区切り文字と囲み文字の間に空白がある場合，空白はフィールドデータとして扱われます。その結果，囲み文字の指定不正によるエラーになったり，囲み文字がフィールドデータとして扱われたりすることがあります。

   なお，この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。

7. 囲み文字と同じ文字がフィールドデータ中にある場合，その文字を 2 個連続で記述してください。
   （例）アポストロフィ（ ’ ）が囲み文字の場合
   ’AB’’CD’（フィールドデータ） → AB’CD（表関数導出表に格納されるデータ）
8. フィールドデータの先頭の文字が囲み文字と同じ場合（先頭に半角空白またはタブがある場合を除く），囲み文字を指定する必要があります。
   （例）アポストロフィ（ ’ ）が囲み文字の場合
   ’’’AB’（フィールドデータ） → ’AB（表関数導出表に格納されるデータ）
9. 区切り文字と同じ文字がフィールドデータ中にある場合，フィールドデータを囲み文字で囲んでください。囲み文字で囲んでない場合，フィールドデータ中の区切り文字と同じ文字は，区切り文字として扱われます。その結果，指定したフィールドが存在しないなどのエラーになります。
   （例）囲み文字が二重引用符（”）で，区切り文字がコンマ（ , ）の場合

```
1,”foo,bar”,3
```

   上記の場合，「1」，「foo,bar」，「3」の 3 列のフィールドデータと認識されます。

```
1,foo,bar,3
```

   上記の場合，「1」，「foo」，「bar」，「3」の 4 列のフィールドデータと認識されます。
   なお，この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。
10. フィールドデータの文字列と，表関数導出表に格納されるデータの例を次に示します。この例では，区切り文字にコンマ（ , ）を使用しています。

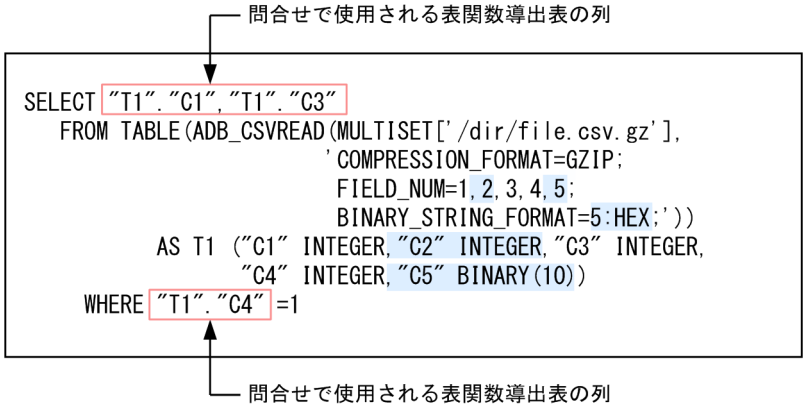
フィールドデータの文字列	表関数導出表に格納されるデータ	
	囲み文字指定オプションに二重引用符（”）を指定した場合	囲み文字指定オプションに NONE を指定した場合
ABC,DEF	• ABC • DEF	• ABC • DEF
”ABC””,”DEF”	• ABC” • DEF	• ”ABC”” • ”DEF”
”ABC,DEF”	• ABC,DEF	• ”ABC • DEF”
”ABC,DEF”	エラー	• ”ABC • DEF

11. CSV ファイル中のフィールドデータは，表関数導出表の各列のデータ型に対応したデータに変換されます。そのため，表関数導出表の列のデータ型とフィールドデータの記述形式に互換性がある必要があります。フィールドデータの記述形式規則の詳細については，「6.2.2　変換，代入，比較できるデータ型」の「(4)　表関数導出表の列への格納代入（ADB_CSVREAD 関数の場合）」を参照してください。
    なお，この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。

(9) 留意事項

1. ADB_CSVREAD 関数を指定した SQL 文を実行する場合，HADB サーバは CSV ファイルをオープンしてデータを読み込みます。そのため，SQL 文の実行中は，CSV ファイルを編集しないでください。
2. 表関数列リストに指定した列のうち，検索結果に影響を与えない列（問合せで使用されない列）については，フィールドデータを取り出しません。また，その列に関する指定を削除した SQL 文に自動的に変換されて実行されます。

   （例）



   上記の例の場合，C2 列とC5 列が検索結果に影響を与えない列です。そのため，C2 列とC5 列に関する指定を削除したSELECT 文に自動的に変換されて実行されます。青色の囲み部分が削除されてSELECT 文が実行されます。

   メモ

   - 削除対象の指定を次に示します。
     - ・表関数列リストに指定した列
     - ・指定列オプションのフィールドデータ番号の指定
     - ・バイナリ文字列形式オプションの指定
   - 列に関する指定が削除された場合，残った列のフィールドデータだけが取り出し対象となります。また，CSV ファイルに関する規則についても，残った列のフィールドデータだけが対象になります。
   - 表関数導出表のすべての列の指定が削除対象となった場合，指定列オプションに指定したフィールドデータ番号のうち，番号が最も小さいフィールドデータ番号の列の指定だけが削除されません。

3. ADB_CSVREAD 関数の指定規則のうち，一部の規則は SQL 文の実行時にチェックされます（SQL 文の前処理時にはチェックされません）。SQL 文の実行時にチェックされる規則には，**「この規則は，SQL 文の前処理時ではなく，SQL 文の実行時にチェックされます。」**の 1 文を記載しています。

(10) 例題

例題 1

GZIP 形式で圧縮した CSV ファイル（/dir/file.csv.gz）から，次のデータを取り出します。

- 顧客 ID（USERID）
- 顧客名（NAME）
- 年齢（AGE）

```
SELECT "USERID","NAME","AGE"
    FROM TABLE(ADB_CSVREAD(MULTISET ['/dir/file.csv.gz'],
                           'COMPRESSION FORMAT=GZIP;'))
         AS "USERSLIST" ("USERID" CHAR(5),
                         "NAME" VARCHAR(100),
                         "AGE" INTEGER,
                         "COUNTRY" VARCHAR(100),
                         "INFORMATION" VARBINARY(10))
```

下線部分がADB_CSVREAD 関数の指定です。

CSVファイル(/dir/file.csv.gz)の展開後の内容

"U0001","John","19","America","10010010"
"U0002","Mary","25","Canada","00010011"
"U0003","Taro","15","Japan","11000100"

検索結果

USERID	NAME	AGE
U0001	John	19
U0002	Mary	25
U0003	Taro	15

例題 2

GZIP 形式で圧縮した CSV ファイル（/dir/file.csv.gz）から，次のデータを取り出します。

- 顧客名（NAME）
- 出身国（COUNTRY）
- 各種フラグ情報（INFORMATION）

```
SELECT "NAME","COUNTRY",BIN("INFORMATION")
    FROM TABLE(ADB_CSVREAD(MULTISET ['/dir/file.csv.gz'],
                           'COMPRESSION FORMAT=GZIP;
                            FIELD_NUM=2,4,5;
                            BINARY_STRING_FORMAT=5:BIN;
                            ENCLOSING_CHAR=";
                            DELIMITER_CHAR=,;'))
         AS "USERSLIST" ("NAME" VARCHAR(100),
            "COUNTRY" VARCHAR(100),
            "INFORMATION" VARBINARY(10))
```

下線部分がADB_CSVREAD 関数の指定です。

CSVﾌｧｲﾙ(/dir/file.csv.gz)の展開後の内容

"U0001","John","19","America","10010010"
"U0002","Mary","25","Canada","00010011"
"U0003","Taro","15","Japan","11000100"

検索結果

NAME	COUNTRY	INFORMATION
John	America	10010010
Mary	Canada	00010011
Taro	Japan	11000100

例題 3

CSV ファイル（/dir/file.csv）から，次のデータを取り出します。

- 顧客 ID（USERID）
- 顧客名（NAME）
- 年齢（AGE）

```
SELECT "USERID","NAME","AGE"
    FROM TABLE(ADB_CSVREAD(MULTISET ['/dir/file.csv'],
                           'COMPRESSION FORMAT=NONE;'))
         AS "USERSLIST" ("USERID" CHAR(5),
                         "NAME" VARCHAR(100),
                         "AGE" INTEGER,
                         "COUNTRY" VARCHAR(100),
                         "INFORMATION" VARBINARY(10))
```

下線部分がADB_CSVREAD 関数の指定です。

CSVﾌｧｲﾙ(/dir/file.csv)の内容

"U0001","John","19","America","10010010"
"U0002","Mary","25","Canada","00010011"
"U0003","Taro","15","Japan","11000100"

検索結果

USERID	NAME	AGE
U0001	John	19
U0002	Mary	25
U0003	Taro	15

7.16　マルチ集合値式

ここでは，マルチ集合値式について説明します。

7.16.1　マルチ集合値式の指定形式および規則

マルチ集合値式は，複数の要素値を 1 つに集めたデータ集合を求める際に使用します。マルチ集合値式は，次の個所に指定できます。

- ADB_AUDITREAD 関数の*監査証跡ファイルのパス名指定*

  ADB_AUDITREAD 関数については，「7.15.2　ADB_AUDITREAD 関数」を参照してください。
- ADB_CSVREAD 関数の *CSV ファイルのパス名指定*

  ADB_CSVREAD 関数については，「7.15.3　ADB_CSVREAD 関数」を参照してください。

(1)　指定形式

マルチ集合値式：：＝｛*列挙によるマルチ集合値構成子*｜*問合せによるマルチ集合値構成子*｝

列挙によるマルチ集合値構成子：：＝MULTISET[*マルチ集合要素*〔,*マルチ集合要素*〕…]
問合せによるマルチ集合値構成子：：＝MULTISET *表副問合せ*

(2)　指定形式の説明

重要

ADB_AUDITREAD 関数中にマルチ集合値式を指定する場合は，*列挙によるマルチ集合値構成子*を指定してください。*問合せによるマルチ集合値構成子*は指定できません。

ADB_CSVREAD 関数中にマルチ集合値式を指定する場合は，次のように指定してください。

- ADB_CSVREAD 関数に CSV ファイル名を個々に指定する場合は，*列挙によるマルチ集合値構成子*を指定してください。
- ADB_CSVREAD 関数に指定する CSV ファイル名を表副問合せで求める場合は，*問合せによるマルチ集合値構成子*を指定してください。

●*列挙によるマルチ集合値構成子*

MULTISET[*マルチ集合要素*〔,*マルチ集合要素*〕…]：

■ADB_AUDITREAD 関数中に*列挙によるマルチ集合値構成子*を指定する場合

*マルチ集合要素*には，ADB_AUDITREAD 関数に指定する監査証跡ファイルのパス名を文字列定数の形式で指定します。文字列定数については，「6.3　定数」を参照してください。

指定例を次に示します。

```
MULTISET['/audit/adbaud-201707*.aud','/audit/adbaud-201708*.aud']
```

指定規則を次に示します。

- マルチ集合要素（監査証跡ファイルのパス名）は，最大 1,000 個指定できます。

■ADB_CSVREAD 関数中に*列挙によるマルチ集合値構成子*を指定する場合

*マルチ集合要素*には，ADB_CSVREAD 関数に指定する CSV ファイルのパス名を文字列定数の形式で指定します。文字列定数については，「6.3　定数」を参照してください。

指定例を次に示します。

```
MULTISET['/dir/file1.csv.gz','/dir/file2.csv.gz','/dir/file3.csv.gz']
```

上記の例では，3 つの CSV ファイルを指定しています。

指定規則を次に示します。

- マルチ集合要素（CSV ファイルのパス名）は，最大 1,000 個指定できます。

●*問合せによるマルチ集合値構成子*

MULTISET *表副問合せ*：

ADB_CSVREAD 関数に指定する CSV ファイルのパス名を表副問合せの形式で指定します。表副問合せについては，「7.3　副問合せ」を参照してください。

指定例を次に示します。

```
MULTISET (SELECT "FILE_NAME" FROM "FILELIST"
              WHERE "FILE_DATE" BETWEEN '2012/01/01' AND '2012/12/31')
```

上記の例では，ファイル管理表（FILELIST）中のFILE_DATE 列が，2012/01/01～2012/12/31 である CSV ファイル名（FILE_NAME）を指定しています。

指定規則を次に示します。

- 表副問合せの結果の列数は 1 にしてください。
- 表副問合せ中に，外への参照列を指定できません。

  （例）エラーとなる指定例

  下線部分が，外への参照列の指定です。

```
SELECT * FROM "T0"
  WHERE EXISTS (SELECT * FROM "T1",
                  TABLE(ADB_CSVREAD(MULTISET (SELECT "T2"."C1"
                                                FROM "T2"
                                                  WHERE "T2"."C2" = "T0"."C2"),
                                      'COMPRESSION_FORMAT=GZIP;'))
                  AS "TF1" ("TFC1" INTEGER,"TFC2" VARCHAR(32)))
```

(3) 例題

例題 1（ADB_AUDITREAD 関数中に監査証跡ファイルのパス名を指定する場合）

2017 年 4 月 1 日～2017 年 4 月 30 日の間に，HADB サーバにアクセスした HADB ユーザの一覧を出力します。2017 年 4 月 1 日～2017 年 4 月 30 日に出力された監査証跡を格納している監査証跡ファイルを，/audit ディレクトリ下に保存しているとします。

```
SELECT DISTINCT "USER_NAME"
    FROM TABLE(ADB_AUDITREAD(MULTISET['/audit/*.aud'])) "DT"
       WHERE "EXEC_TIME" BETWEEN TIMESTAMP'2017/04/01 00:00:00.000000'
                             AND TIMESTAMP'2017/04/30 23:59:59.999999'
```

下線部分がマルチ集合値式（*列挙によるマルチ集合値構成子*）の指定です。

例題 2（ADB_CSVREAD 関数中に CSV ファイルのパス名を指定する場合）

GZIP 形式で圧縮した CSV ファイル（/dir/file1.csv.gz，/dir/file2.csv.gz，/dir/file3.csv.gz）から，次のデータを取り出します。

- 顧客 ID（USERID）
- 顧客名（NAME）
- 年齢（AGE）

```
SELECT "USERID","NAME","AGE"
    FROM TABLE(ADB_CSVREAD(MULTISET ['/dir/file1.csv.gz','/dir/file2.csv.gz','/dir/file3.
csv.gz'],
                          'COMPRESSION_FORMAT=GZIP;'))
         AS "USERSLIST" ("USERID" CHAR(10),"NAME" VARCHAR(100),"AGE" INTEGER)
```

下線部分がマルチ集合値式（*列挙によるマルチ集合値構成子*）の指定です。

例題 3（ADB_CSVREAD 関数中に，表副問合せを使用して CSV ファイルのパス名を指定する場合）

2010 年に登録した顧客情報のデータを取り出します。データを取り出す際の条件は次のとおりとします。

- 顧客情報のデータを CSV 形式のファイルで保管している
- CSV ファイルを GZIP 形式で圧縮している
- CSV ファイルの一覧を，CSV ファイル管理表（FILELIST）で管理している
- CSV ファイル管理表には，CSV ファイルの絶対パス名（FILE_NAME）とファイルを登録した日（FILE_DATE）が格納されている

```
SELECT "USERID","NAME","AGE"
    FROM TABLE(ADB_CSVREAD(MULTISET (SELECT "FILE_NAME" FROM "FILELIST"
                                     WHERE "FILE_DATE" BETWEEN '2010/01/01'
                                                           AND '2010/12/31'),
                          'COMPRESSION_FORMAT=GZIP;'))
         AS "USERSLIST" ("USERID" CHAR(10),"NAME" VARCHAR(100),"AGE" INTEGER)
```

下線部分がマルチ集合値式（*問合せによるマルチ集合値構成子*）の指定です。

7.17　表値構成子

ここでは，表値構成子について説明します。

7.17.1　表値構成子の指定形式および規則

表値構成子には，導出表を構成する行（行値構成子の集合）を指定します。

(1)　指定形式

表値構成子：：＝VALUES *行値構成子*〔,*行値構成子*〕…

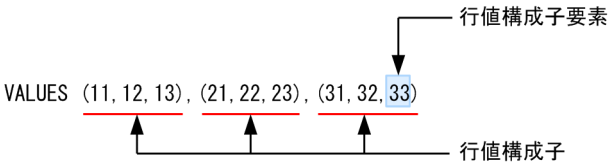
(2)　指定形式の説明

行値構成子：

行値構成子には，1 つ以上の行値構成子要素を指定します。各行値構成子要素の値が，導出表の 1 行の各列の値になります。行値構成子については，「7.18　行値構成子」を参照してください。

表値構成子の指定例を次に示します。

(例)



表値構成子中の行値構成子の行値構成子要素には，次のどれかを指定できます。

- 値指定

  値指定については，「7.22　値指定」を参照してください。

- スカラ関数CAST

  スカラ関数CAST については，「8.13.3　CAST」を参照してください。

  スカラ関数CAST を指定する場合，次の指定規則があります。

  - 変換対象データには，NULL または？パラメタだけを指定できます。

- スカラ関数CONVERT

  スカラ関数CONVERT については，「8.13.5　CONVERT」を参照してください。

  スカラ関数CONVERT を指定する場合，次の指定規則があります。

  - 変換対象データには，NULL または？パラメタだけを指定できます。
  - 書式指定は指定できません。

(3) 規則

1. 各行値構成子の行値構成子要素の数は同じにしてください。

   正しい指定例：VALUES (11,12,13),(21,22,23),(31,32,33)

   誤った指定例：VALUES (11,12,13),(21,22),(31,32,33,34)

2. 各行値構成子のi番目の行値構成子要素のデータ型は，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，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

   正しい指定例：VALUES (11,12,13),(21.1,22.2,23.3),(1.0E+1,1.0E+2,1.0E+3)

   誤った指定例：VALUES (11,12,13),('AB','CD',23)

   なお，次に示すデータは比較できません。

   - 日付データと日付を表す既定の入力表現
   - 時刻データと時刻を表す既定の入力表現
   - 時刻印データと時刻印を表す既定の入力表現

3. 表値構成子によって導出されるi番目の列の結果のデータ型とデータ長は，各行値構成子のi番目の行値構成子要素のデータ型によって決まります。詳細については，「7.21.2　値式の結果のデータ型」を参照してください。

4. 表値構成子中には行値構成子を，最大30,000個指定できます。

5. 1SQL文中に指定できる表値構成子と問合せ指定の合計数は，最大1,024個です。

6. 表値構成子中の行値構成子の行値構成子要素には，？パラメタを単独で指定できません。

(4) 例題

例題1

表値構成子を指定したSELECT文を実行します。

```
SELECT "C1","C2","C3" FROM (VALUES (11,12,13),
                                   (21,22,23)
                           ) AS "V1"("C1","C2","C3")
```

下線部分が表値構成子の指定です。

実行結果の例

C1	C2	C3
11	12	13
21	22	23

例題2

販売履歴表（SALESLIST）と，表値構成子によって導出された顧客表（USERSLIST）から，商品コード（PUR-CODE）がP001の商品を購入したことがある顧客の一覧（顧客ID，名前）を，重複を除いて検索します。

```
SELECT DISTINCT "USERSLIST"."USERID","NAME"
  FROM "SALESLIST"
         INNER JOIN
         (VALUES('U001','Maria'),('U002','Nancy')) AS "USERSLIST"("USERID","NAME")
            ON "USERSLIST"."USERID"="SALESLIST"."USERID"
        WHERE "SALESLIST"."PUR-CODE"='P001'
```

下線部分が表値構成子の指定です。

例題 3

顧客表（USERSLIST）に複数のデータを挿入します。

```
INSERT INTO "USERSLIST"("USERID","AGE")
    SELECT * FROM (VALUES('USER001',10),('USER002',20))
```

下線部分が表値構成子の指定です。

7.18　行値構成子

ここでは，行値構成子について説明します。

7.18.1　行値構成子の指定形式および規則

行値構成子には，順序付けられた値の並びから構成される行を指定します。

(1)　指定形式

行値構成子：：＝(*行値構成子要素*〔,*行値構成子要素*〕…)

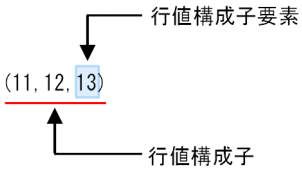
行値構成子要素：：＝*値式*

(2)　指定形式の説明

行値構成子要素：

行値構成子には，1 つ以上の行値構成子要素を値式の形式で指定します。値式については，「7.21　値式」を参照してください。

行値構成子の指定例を次に示します。



(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 4,000 個です。

(4)　例題

例題 1（表値構成子に行値構成子を指定した例）

販売履歴表（SALESLIST）を表値構成子から導出して検索します。

```
SELECT "USERID","PUR-CODE","PUR-NUM"
  FROM (VALUES('U001','P001',5)
              ,('U001','P002',3)
              ,('U002','P001',1)
       ) "SALESLIST"("USERID","PUR-CODE","PUR-NUM")
```

下線部分が行値構成子の指定です。

実行結果の例

■SALESLIST

USERID	PUR-CODE	PUR-NUM
U001	P001	5
U001	P002	3
U002	P001	1

例題２（比較述語に行値構成子を指定した例）

販売履歴表（SALESLIST）から，顧客 ID（USERID）がU001 かつ商品コード（PUR-CODE）がP001 の行を検索します。

```
SELECT "USERID","PUR-CODE","PUR-NUM"
  FROM "SALESLIST"
  WHERE ("USERID","PUR-CODE") = ('U001','P001')
```

下線部分が行値構成子の指定です。

上記のSELECT 文を実行した場合，検索結果は次のようになります。

■SALESLIST

USERID	PUR-CODE	PUR-NUM
U001	P001	5
U001	P002	3
U002	P001	1

■検索結果

USERID	PUR-CODE	PUR-NUM
U001	P001	5

例題３（IN 述語に行値構成子を指定した例）

販売履歴表（SALESLIST）から，次の条件と一致する行を検索します。

- 顧客 ID（USERID）がU001 かつ商品コード（PUR-CODE）がP001
- 顧客 ID（USERID）がU002 かつ商品コード（PUR-CODE）がP001

```
SELECT "USERID","PUR-CODE","PUR-NUM"
  FROM "SALESLIST"
  WHERE ("USERID","PUR-CODE") IN (('U001','P001'), ('U002','P001'))
```

下線部分が行値構成子の指定です。

上記のSELECT 文を実行した場合，検索結果は次のようになります。

■SALESLIST

USERID	PUR-CODE	PUR-NUM
U001	P001	5
U001	P002	3
U002	P001	1

■検索結果

USERID	PUR-CODE	PUR-NUM
U001	P001	5
U002	P001	1

7.19　探索条件

ここでは，探索条件について説明します。

7.19.1　探索条件の指定形式および規則

探索条件には，データの検索条件を指定します。指定された探索条件に従って論理演算が実行され，その結果が**真**のものだけが検索対象になります。探索条件は次の個所に指定できます。

- WHERE 句
- HAVING 句
- CASE 式
- 結合表のON 探索条件
- DELETE 文，PURGE CHUNK 文，UPDATE 文のWHERE 探索条件

(1)　指定形式

```
探索条件 ： ： ＝ ｛〔NOT〕　｛(探索条件)｜述語 ｜論理値指定｝
                ｜探索条件 OR　｛(探索条件)｜述語 ｜論理値指定｝
                ｜探索条件 AND　｛(探索条件)｜述語 ｜論理値指定｝ ｝

  論理値指定 ： ： ＝ ｛TRUE｜FALSE｝
```

(2)　指定形式の説明

NOT：

NOT を指定すると，探索条件を満たさない値が検索対象になります。例えば，「NOT ”USERID”=’U00358’」と指定した場合，U00358 以外のUSERID が検索対象になります。

探索条件：

複数の探索条件を指定する場合，探索条件をAND またはOR でつなぎます。AND とOR は混在して指定できます。AND とOR の意味を次に示します。

- *探索条件 1* AND *探索条件 2*

  探索条件 1 と*探索条件 2* の両方を満たす行が検索対象になります。

- *探索条件 1* OR *探索条件 2*

  探索条件 1 と*探索条件 2* のどちらかを満たす行が検索対象になります。

なお，探索条件中には，配列型の列を外への参照列として指定できません。

述語：

述語の詳細については，「7.20　述語」を参照してください。

論理値指定：

TRUE：*論理値指定*にTRUE を指定した場合，論理値指定の結果は真となります。

FALSE：*論理値指定*にFALSE を指定した場合，論理値指定の結果は偽となります。

探索条件の指定例を次に示します。

（例）

C1，C2，C3 は列名です。

- 比較述語を使用した指定例

```
"C1">=100
"C1"=?
"C2"=CURRENT_DATE
SUBSTR("C3",2,3)='150'
```

- IN 述語，BETWEEN 述語，LIKE 述語，またはNULL 述語を使用した指定例

```
"C1" IN (10,20)
"C1" BETWEEN 100 AND 200
"C3" LIKE 'M%'
"C3" IS NULL
```

- 複数の探索条件を指定する例

```
"C1"=100 AND "C2">=DATE'2011-09-06'
"C1" IN (10,20) AND "C2">=DATE'2011-09-06'
"C1"=10 OR "C1"=20
"C2">=DATE'2011-09-04' AND ("C1"=10 OR "C2"=20)
```

探索条件の評価の優先順位は，（ ）の指定，NOT，AND，OR の順になります。

(3) 規則

1. 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. 各論理演算をした場合の結果を次の図に示します。

図 7-3 論理演算をした場合の結果

（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

(4) 例題

探索条件の指定例を例題を使って説明します。

例題 1

販売履歴表（SALESLIST）から，2011/9/4 以降に商品コードP001 またはP003 の商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-04'
        AND ("PUR-CODE"='P001' OR "PUR-CODE"='P003')
```

下線部分がWHERE 句に指定した探索条件の指定です。

例題 2

販売履歴表（SALESLIST）から，2011/9/3 以降の商品コード（PUR-CODE）ごとの販売個数の合計値，平均値を求めます。

その際，販売個数の合計値が 20 個以下の商品コードだけを検索対象にします。

```
SELECT "PUR-CODE",SUM("PUR-NUM"),AVG("PUR-NUM")
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-03'
        GROUP BY "PUR-CODE"
        HAVING SUM("PUR-NUM")<=20
```

下線部分がHAVING 句に指定した探索条件の指定です。

例題 3

新商品表（PRODUCTLIST_NEW）に，商品表（PRODUCTLIST）の行を挿入します。行を挿入する際，商品価格（PRICE）を次に示すように新価格に変更します。

- 商品コード（PCODE）がP001 の場合：商品価格を 10%引きにする
- 商品コードがP002 の場合：商品価格を 20%引きにする
- 上記以外の場合：商品価格を 30%引きにする

```
INSERT INTO "PRODUCTLIST_NEW"("PCODE","PRICE")
    SELECT "PCODE",CASE WHEN "PCODE"='P001' THEN "PRICE"*0.9
                        WHEN "PCODE"='P002' THEN "PRICE"*0.8
                        ELSE "PRICE"*0.7
                   END
        FROM "PRODUCTLIST"
```

下線部分がCASE 式に指定した探索条件の指定です。

7.20　述語

次に示す述語を使用できます。ここでは，これらの述語の機能と指定形式について説明します。

- BETWEEN 述語
- EXISTS 述語
- IN 述語
- LIKE 述語
- LIKE_REGEX 述語
- NULL 述語
- 比較述語
- 限定述語

これらの述語は探索条件中に指定できます。

7.20.1　BETWEEN 述語

特定の範囲のデータを検索する際にBETWEEN 述語を使用します。

(1)　指定形式

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

(2)　指定形式の説明

値式 1：

BETWEEN 述語の評価対象列を指定します。値式の形式で指定します。値式については，「7.21　値式」を参照してください。

NOT：

NOT を指定すると，BETWEEN 以降に指定した条件を満たさない値が検索対象になります。

BETWEEN *値式 2* AND *値式 3*：

検索範囲の下限値と上限値を指定して，検索範囲を指定します。値式の形式で指定します。

値式 2 には検索範囲の下限値を指定し，*値式 3* には検索範囲の上限値を指定します。

(3)　述語の評価

次に示す条件を満たす行に対して，BETWEEN 述語は**真**になります。

値式 2≦*値式 1* AND *値式 1*≦*値式 3*

次に示すBETWEEN 述語は等価になります。

- *値式 1* NOT BETWEEN *値式 2* AND *値式 3*
- NOT(*値式 1* BETWEEN *値式 2* AND *値式 3*)

(4) 規則

1. ？パラメタだけの値式を*値式 1* に指定できません。
2. *値式 1*～*値式 3* に指定できるデータ型は，数データ，文字データ，または日時データです。
3. *値式 1*～*値式 3* の結果のデータ型が，比較可能なデータ型になるように*値式 1*～*値式 3* を指定してください。比較可能なデータ型については，「6.2.2 変換，代入，比較できるデータ型」の「(1) 比較できるデータ型」を参照してください。

   ただし，*値式 1* に日付，時刻，または時刻印を表す既定の入力表現を指定した場合，*値式 2* 以降に日時データを指定することはできません。既定の入力表現については，「6.3.3 既定の文字列表現」を参照してください。

(5) 例題

例題 1

販売履歴表（SALESLIST）から，2011/9/4～2011/9/5 に商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
  FROM "SALESLIST"
    WHERE "PUR-DATE" BETWEEN DATE'2011-09-04' AND DATE'2011-09-05'
```

下線部分がBETWEEN 述語の指定です。

例題 2

販売履歴表（SALESLIST）から，2011/9/4～2011/9/5 を除いた日に商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
  FROM "SALESLIST"
    WHERE "PUR-DATE" NOT BETWEEN DATE'2011-09-04' AND DATE'2011-09-05'
```

下線部分がBETWEEN 述語の指定です。

7.20.2 EXISTS 述語

表副問合せの結果が 0 行（空集合）でないかどうかを判定するときにEXISTS 述語を使用します。

(1) 指定形式

EXISTS述語 ：：＝EXISTS *表副問合せ*

(2) 指定形式の説明

表副問合せ：

表副問合せについては，「7.3 副問合せ」を参照してください。

(3) 述語の評価

表副問合せの結果が 1 行以上の場合，EXISTS 述語の結果が**真**になります。表副問合せの結果が 0 行（空集合）の場合，EXISTS 述語の結果が**偽**になります。EXISTS 述語の結果を次の表に示します。

表 7-3 EXISTS 述語の結果

項番	表副問合せの結果の行数	EXISTS 述語の結果
1	1 行以上	真
2	0 行（空集合）	偽

(4) 規則

表副問合せ中の選択リストに，*または*表指定*.*を指定した場合，表副問合せ中の表参照で指定した表の任意の 1 列を指定した意味になります。ただし，集合関数の指定は該当しません。

(5) 例題

例題

販売履歴表（SALESLIST）と商品一覧表（PRODUCTSLIST）から，販売実績のある商品の情報を検索します。

```
SELECT * FROM "PRODUCTSLIST"
    WHERE EXISTS(SELECT * FROM "SALESLIST"
            WHERE "SALESLIST"."PUR-CODE"="PRODUCTSLIST"."PUR-CODE")
```

下線部分がEXISTS 述語の指定です。

7.20.3 IN 述語

複数の条件値のうち，どれか 1 つの条件値を満たすデータを検索する際にIN 述語を使用します。

(1) 指定形式

```
IN述語 ： ： ＝ {値式 | 行値構成子}  〔IS〕  〔NOT〕 IN
                {( {値式 | 行値構成子}  〔, {値式 | 行値構成子} 〕 …) | 表副問合せ}
```

(2) 指定形式の説明

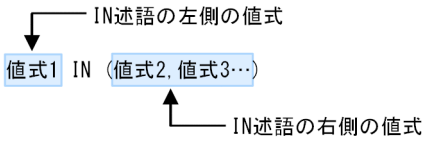
{値式 | 行値構成子}：

IN 述語の評価対象を，IN 述語の左側に値式または行値構成子の形式で指定します。

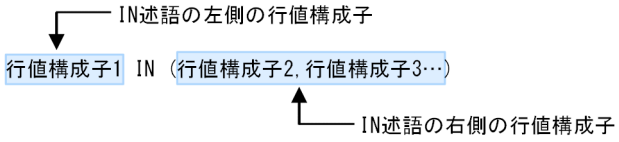
評価対象と比較する条件値を，IN 述語の右側に値式または行値構成子の形式で指定します。

IN 述語の左側（右側）の値式または行値構成子の意味を次に示します。

■値式の場合



■行値構成子の場合



- 値式の場合

  上の図の*値式 1* がIN 述語の左側の値式（IN 述語の評価対象），*値式 2* および*値式 3* がIN 述語の右側の値式（評価対象と比較する条件値）となります。値式については，「7.21　値式」を参照してください。

- 行値構成子の場合

  上の図の*行値構成子 1* がIN 述語の左側の行値構成子（IN 述語の評価対象），*行値構成子 2* および*行値構成子 3* がIN 述語の右側の行値構成子（評価対象と比較する条件値）となります。行値構成子については「7.18　行値構成子」を参照してください。

指定規則を次に示します。

- IN 述語の左右に行値構成子を指定する場合，行値構成子要素を 2 つ以上指定してください。行値構成子要素を 1 つだけ指定した場合，その行値構成子は括弧で囲まれた値式として扱われます。

  行値構成子として扱われる例

  ```
  ("C1","C2") IN ((1,'A'),(2,'B'))
  ```

  値式として扱われる例

  ```
  ("C1") IN ((1),(2))
  ```

  上記は「"C1" IN (1,2)」と同じです。

- IN 述語の左側に値式を指定する場合，IN 述語の右側には値式だけを指定するか，または表副問合せを指定してください。

- IN 述語の左側に行値構成子を指定する場合，IN 述語の右側には行値構成子だけを指定してください。
- IN 述語の右側に指定する値式または行値構成子は，30,000 個まで指定できます。

IS：

IS の指定は省略できます。指定のありなしに関係なく結果は同じになります。

NOT：

NOT を指定すると，IN 述語の右側に指定した条件値と一致しない値が検索対象になります。

NOT を指定しないと，IN 述語の右側に指定した条件値と一致する値が検索対象になります。

表副問合せ：

表副問合せを指定します。表副問合せについては，「7.3　副問合せ」を参照してください。

なお，IN 述語に表副問合せを指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。

(3)　述語の評価

IN 述語の結果が**真**となる条件を次の表に示します。

表 7-4　IN 述語の結果が真となる条件

NOT の指定	IN 述語の左側の指定値	IN 述語の右側の指定値	IN 述語の結果が真となる条件
指定しない	値式	値式	IN 述語の左側の指定値が，右側の指定値のどれか 1 つ以上と一致する場合
	行値構成子	行値構成子	
	値式	表副問合せ	IN 述語の左側の値式が，右側の表副問合せの結果行のどれか 1 つ以上と一致する場合
指定する	値式	値式	IN 述語の左側の指定値が，右側のすべての指定値と一致しない場合
	行値構成子	行値構成子	
	値式	表副問合せ	IN 述語の左側の値式が，右側の表副問合せのすべての結果行と一致しない場合

NOT を指定しないIN 述語の結果は，探索条件をOR 条件でつないだ探索条件の結果と同じになり，NOT を指定するIN 述語の結果は，探索条件をAND 条件でつないだ探索条件の結果と同じになります。IN 述語の指定と同じ結果になる探索条件を次の表に示します。

表 7-5　IN 述語の指定と同じ結果になる探索条件

IN 述語の指定	左記の指定と同じ結果になる探索条件
値式1 IN（*値式2*,*値式3*,…）※	（*値式1* = *値式2*）OR（*値式1* = *値式3*）OR …
行値構成子1 IN（*行値構成子2*,*行値構成子3*,…）	（*行値構成子1* = *行値構成子2*）OR（*行値構成子1* = *行値構成子3*）OR …
値式1 NOT IN（*値式2*,*値式3*,…）※	（*値式1* <> *値式2*）AND（*値式1* <> *値式3*）AND …
行値構成子1 NOT IN（*行値構成子2*,*行値構成子3*,…）	（*行値構成子1* <> *行値構成子2*）AND（*行値構成子1* <> *行値構成子3*）AND …

注※

IN 述語の右側に*表副問合せ*を指定する場合は，その表副問合せの各結果行が(*値式 2*,*値式 3*,…)になるものとして読み替えてください。

IN 述語の指定例と結果を次の表に示します。

表 7-6　IN 述語の指定例と結果

IN 述語の指定例	IN 述語の結果
1 IN (1,2,3)	真
1 IN (1,2,ナル値)	真
1 IN (2,3,4)	偽
1 IN (2,3,ナル値)	不定
(1,2) IN ((1,2),(2,3),(3,4))	真
(1,2) IN ((1,2),(2,3),(1,ナル値))	真
(1,2) IN ((2,3),(3,4),(4,5))	偽
(1,2) IN ((1,ナル値),(2,3),(3,4))	不定
1 NOT IN (2,3,4)	真
1 NOT IN (1,2,3)	偽
1 NOT IN (1,2,ナル値)	偽
1 NOT IN (2,3,ナル値)	不定
(1,2) NOT IN ((2,3),(3,4),(4,5))	真
(1,2) NOT IN ((1,2),(2,3),(3,4))	偽
(1,2) NOT IN ((1,2),(2,3),(1,ナル値))	偽
(1,2) NOT IN ((1,ナル値),(2,3),(3,4))	不定

(4) 規則

(a) IN 述語の左側に値式を指定する場合の規則

1. IN 述語の左側の値式に？パラメタを単独で指定できません。
2. IN 述語の左側の値式の結果のデータ型と，IN 述語の右側の値式の結果のデータ型が比較可能なデータ型になるように各値式を指定してください。比較可能なデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

   ただし，IN 述語の左側の値式に日付，時刻，または時刻印を表す既定の入力表現を指定した場合，IN 述語の右側の値式に日時データを指定することはできません。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
3. IN 述語の左側の値式の結果がナル値の場合，IN 述語の結果は不定となります。また，IN 述語の右側のすべての値式の結果がナル値の場合も，IN 述語の結果は不定となります。
4. IN 述語の左右の値式には，配列データを指定できません。
5. 表副問合せの結果が 0 行（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。
6. 表副問合せを指定した場合，表副問合せ中の選択式の数は 1 にしてください。
7. 表副問合せを指定したIN 述語は，HADB によって限定述語（=ANY 指定）に書き換えられて処理されます。
8. 表副問合せの選択式の値式には，配列データを指定できません。

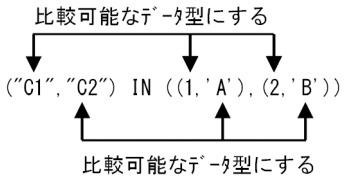
(b) IN 述語の左側に行値構成子を指定する場合の規則

1. IN 述語の左右の行値構成子の行値構成子要素の数はすべて同じにしてください。

   正しい指定例：(”C1”,”C2”) IN ((1,’A’),(2,’B’))

   誤った指定例：(”C1”,”C2”) IN ((1,’A’),(2,’B’,’C’))
2. IN 述語の左側の行値構成子中の各行値構成子要素の結果のデータ型と，IN 述語の右側の行値構成子中の各行値構成子要素の結果のデータ型が比較可能なデータ型になるようにしてください。比較可能なデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

   (例)



   ただし，IN 述語の左側の行値構成子の行値構成子要素に日付，時刻，または時刻印を表す既定の入力表現を指定した場合，IN 述語の右側の行値構成子の行値構成子要素に日時データを指定することはできません。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。

3. IN 述語の左側の行値構成子の行値構成子要素には，？パラメタを単独で指定できません。
4. IN 述語の右側の行値構成子の行値構成子要素に？パラメタを単独で指定した場合，その？パラメタのデータ型には，比較相手となるIN 述語の左側の行値構成子の行値構成子要素のデータ型が仮定されます。
5. IN 述語の右側の行値構成子の行値構成子要素は値指定だけを指定できます。
6. IN 述語の左側の行値構成子の行値構成子要素には，配列型のデータを指定できません。
7. IN 述語の左側の行値構成子の行値構成子要素中には，次の値式を指定できません。
   - 集合関数
   - スカラ副問合せ
   - 配列要素参照
8. 行値構成子を指定したIN 述語は，次のどちらかに指定できます。
   - WHERE 句
   - DELETE 文またはUPDATE 文のWHERE 探索条件
9. 行値構成子を指定したIN 述語は，CASE 式の探索条件中には指定できません。

(5) 例題

例題 1

販売履歴表（SALESLIST）から，商品コードP001 またはP003 の商品を 2011/9/5 以降に購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
       WHERE "PUR-CODE" IN ('P001','P003')
       AND "PUR-DATE">=DATE'2011-09-05'
```

下線部分がIN 述語の指定です。

例題 2

販売履歴表（SALESLIST）から，顧客 ID（USERID），商品コード（PUR-CODE），販売個数（PUR-NUM）を検索します。ただし，顧客 ID（USERID）がU00212 およびU00358 の顧客は，検索対象外とします。

```
SELECT "USERID","PUR-CODE","PUR-NUM"
    FROM "SALESLIST"
       WHERE "USERID" NOT IN ('U00212','U00358')
```

下線部分がIN 述語の指定です。

例題 3

顧客表（USERSLIST）と販売履歴表（SALESLIST）から，商品コード（PUR-CODE）がP001 の商品を購入した顧客の情報を検索します。

```
SELECT * FROM "USERSLIST"
    WHERE "USERID" IN (SELECT "USERID" FROM "SALESLIST"
                           WHERE "PUR-CODE"='P001')
```

下線部分がIN 述語の指定です。

例題 4

販売履歴表（SALESLIST）から，次の条件と一致する行を検索します。

- 顧客 ID（USERID）がU001 かつ商品コード（PUR-CODE）がP001
- 顧客 ID（USERID）がU002 かつ商品コード（PUR-CODE）がP001

```
SELECT "USERID","PUR-CODE","PUR-NUM"
  FROM "SALESLIST"
  WHERE ("USERID","PUR-CODE") IN (('U001','P001'), ('U002','P001'))
```

下線部分が行値構成子を指定したIN 述語の指定です。

7.20.4 LIKE 述語

特定の文字列が含まれているデータを検索する際にLIKE 述語を使用します。

(1) 指定形式

```
LIKE述語 ：：＝一致値 〔NOT〕 LIKE パターン文字列 〔ESCAPE エスケープ文字〕

  一致値 ：：＝値式
  パターン文字列 ：：＝値式
  エスケープ文字 ：：＝値式
```

(2) 指定形式の説明

一致値：

検索対象のデータを値式の形式で指定します。値式については，「7.21 値式」を参照してください。

*一致値*には，CHAR 型またはVARCHAR 型のデータを指定してください。

NOT：

NOT を指定すると，指定したパターン文字列と一致しない値が検索対象になります。

パターン文字列：

パターン文字列を値式の形式で指定します。値式については，「7.21 値式」を参照してください。

*パターン文字列*には，CHAR 型またはVARCHAR 型のデータを指定してください。

なお，パターン文字列には，_（下線）または%（パーセント）の特殊文字が指定できます。_ は任意の 1 文字を意味し，%は 0 文字以上の任意の文字列を意味します。これらの特殊文字を使うと，例えば次のような検索ができます。

- '電気'で始まる 5 文字の文字列：'電気_ _ _'
- 'OR'を含む文字列：'%OR%'

パターン文字列中に _ や%を指定すると，特殊文字と見なされ，通常文字としては扱われません。_ や%を通常文字として扱いたい場合は，エスケープ文字を指定する必要があります。

ESCAPE *エスケープ文字*：

エスケープ文字を値式の形式で指定します。値式については，「7.21　値式」を参照してください。

*エスケープ文字*には，1 バイトのCHAR 型またはVARCHAR 型のデータを指定してください。

エスケープ文字を指定すると，パターン文字列中の特殊文字（エスケープ文字の直後の特殊文字）を通常文字として扱うことができます。

（例）特殊文字%を指定している場合

```
LIKE 'ABC@%'                ...1
LIKE 'ABC@%' ESCAPE '@'     ...2
```

1 の場合は，%が特殊文字として扱われるため，'ABC@'で始まる文字列が検索対象になります。2 の場合は，%が通常文字として扱われるため，'ABC%'の文字列が検索対象になります。

パターン文字列中にエスケープ文字がある場合の扱いについては，「(4)　エスケープ文字を指定したときの扱い」を参照してください。

(3)　パターン文字列の指定例

パターン文字列の代表的な指定例を次の表に示します。

表 7-7　パターン文字列の代表的な指定例

項番	項目	パターン文字列の指定	意味	パターン文字列の指定例	検索対象となる文字列
1	前方一致	*xxx*%	文字列の先頭部分が「*xxx*」である	'ACT%'	ACT，ACTOR，ACTION など，ACT で始まる文字列
2	後方一致	%*xxx*	文字列の最後の部分が「*xxx*」である	'%ING'	ING，BEING，HAVING など，ING で終わる文字列
3	任意一致	%*xxx*%	文字列中の任意の部分に「*xxx*」を含む	'%日%'	日，日立，昨日，本日中など，日を含む文字列
4	完全一致	*xxx*	文字列が「*xxx*」と等しい	'EQUAL'	EQUAL
5	部分一致	_…_*xxx*_…_	文字列中の特定の部分が「*xxx*」と等しく，ほかの部分は任意の文字である _ 1 つが任意の 1 文字を意味している	'_I_'	BIT，HIT，KIT など，3 文字の文字列で，2 文字目が I の文字列
6				'_ _T_ _ _ _'	HITACHI など，7 文字の文字列で，3 文字目がT の文字列
7	そのほか	*xxx*%*yyy*	文字列の先頭部分が「*xxx*」で，最後の部分が「*yyy*」である	'O%N'	ON，OWN，ORIGIN など，O で始まりN で終わる文字列

項番	項目	パターン文字列の指定	意味	パターン文字列の指定例	検索対象となる文字列
8		%*xxx*%*yyy*%	文字列中の任意の部分に「*xxx*」を含み，その部分よりあとの任意の部分に「*yyy*」を含む	'%O%N%'	ON，ONE，DOWN，COUNT など，O を含み，それよりあとにN を含む文字列
9		*xxx*_…_*yyy*%	文字列の先頭部分が「*xxx*」で，そのあとに任意の文字列が続き，そのあとに「*yyy*」が続く _ 1 つが任意の 1 文字を意味している	'CO_ _ECT%'	CORRECT，CONNECTER，CONNECTION など，7 文字以上の文字列で，CO で始まり 5 文字目から 7 文字目がECT の文字列

注

- 「*xxx*」，「*yyy*」は，%，_ を含まない任意の文字列です。
- 空白も比較対象になるため，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。

(4) エスケープ文字を指定したときの扱い

パターン文字列中にエスケープ文字がある場合の扱いを次に示します。以下の例では@をエスケープ文字としています。

1. エスケープ文字の直後が特殊文字の場合，その特殊文字は通常文字として扱われます。

   (例 1)

   ```
   LIKE 'AB@%C%' ESCAPE '@'
   ```

   この場合，@の後ろの特殊文字は通常文字として扱われるため，AB%C，AB%CDE など，AB%C で始まる文字列が検索対象になります。

   (例 2)

   ```
   LIKE 'AB@_C%' ESCAPE '@'
   ```

   この場合，@の後ろの特殊文字は通常文字として扱われるため，AB_C，AB_CDE など，AB_C で始まる文字列が検索対象になります。

2. エスケープ文字の直後が通常文字の場合，エスケープ文字は読み飛ばされます。

   (例)

   ```
   LIKE 'ABC@D' ESCAPE '@' → LIKE 'ABCD'と等価
   ```

   この場合，@は読み飛ばされます。

3. エスケープ文字が 2 つ連続している場合，2 つ連続しているエスケープ文字は通常文字 1 文字として扱われます。

(例 1)

```
LIKE 'AB@@C' ESCAPE '@' → LIKE 'AB@C'と等価
```

この場合，@@は通常文字の@1 文字として扱われます。

(例 2)

```
LIKE 'AB@@@C' ESCAPE '@' → LIKE 'AB@C'と等価
```

この場合，最初の@@は通常文字の@1 文字として扱われます。3 文字目の@は，後ろの文字が通常文字のため，読み飛ばされます。

(例 3)

```
LIKE 'AB@@@@C' ESCAPE '@' → LIKE 'AB@@C'と等価
```

この場合，最初の@@は通常文字の@1 文字として扱われます。3～4 文字目の@も，通常文字の@1 文字として扱われます。

(例 4)

```
LIKE 'AB@@C@%D%' ESCAPE '@'
```

この場合，AB@C%D，AB@C%DE など，AB@C%D で始まる文字列が検索対象になります。

4. エスケープ文字の後ろに文字がない場合，そのエスケープ文字は読み飛ばされます。

(例)

```
LIKE 'ABC@' ESCAPE '@' → LIKE 'ABC'と等価
```

(5) 述語の評価

*一致値*と*パターン文字列*のパターンが一致した場合に真となります。そうでない場合は偽となります。

NOT を指定したときは，*一致値*と*パターン文字列*のパターンが一致しない場合に真となります。そうでない場合は偽となります。

一致値，または*パターン文字列*のどちらかの結果がナル値の場合，述語の結果は不定となります。

なお，*一致値*が長さ 0 バイトまたは長さ 0 文字のときは，次に示す場合に限りLIKE 述語の結果が真となります。

- パターン文字列が長さ 0 バイトまたは長さ 0 文字の場合
- パターン文字列に？パラメタを指定していて，その入力値が'%'の場合
- パターン文字列に定数で'%'を指定した場合

また，*パターン文字列*が長さ 0 バイトまたは長さ 0 文字のときは，*一致値*が長さ 0 バイトまたは長さ 0 文字の場合に限りLIKE 述語の結果が真となります。

(6) 規則

(a) 一致値に関する規則

1. *一致値*に指定する_ および%はすべて半角（最小バイト文字）で指定してください。
2. *一致値*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

(b) パターン文字列に関する規則

1. パターン文字列の長さには，_ および%を含みます。
2. *パターン文字列*中に%を指定しない場合，*一致値*のデータの長さとパターン文字列の長さが異なると，この述語は真になりません。
3. *パターン文字列*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型とデータ長は，次の表に示すとおりになります。

表 7-8　？パラメタに仮定されるデータ型とデータ長（パターン文字列に？パラメタを単独で指定した場合）

条件	？パラメタに仮定されるデータ型	？パラメタに仮定されるデータ長
エスケープ文字の指定がない場合	VARCHAR 型	一致値の結果のデータ長
エスケープ文字の指定がある場合	VARCHAR 型	• 一致値の結果のデータ長が 32,000 バイト以下の場合 MIN（*一致値の結果のデータ長*×2，32,000） • 一致値の結果のデータ長が 32,001 バイト以上の場合 MIN（*一致値の結果のデータ長*×2，64,000）

(c) エスケープ文字に関する規則

1. *エスケープ文字*が長さ 0 バイトまたは長さ 0 文字の場合，エスケープ文字の指定がされていないと扱われます。

（例）

```
LIKE 'ABC' ESCAPE '' → LIKE 'ABC'と等価
LIKE 'ABC' ESCAPE ?  → ?パラメタでNULLを指定した場合，LIKE 'ABC'と等価
```

2. *エスケープ文字*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(1)になります。また，このときのデータの実長は 1 バイトである必要があります。
3. パターン文字列中のエスケープ文字の判定は，1 バイトごとでなく 1 文字ごとに行われます。
4. エスケープ文字に指定できる文字コードの範囲を次の表に示します。

表 7-9　エスケープ文字に指定できる文字コードの範囲

環境変数 ADBLANG の指定値	エスケープ文字に指定できる文字コードの範囲
UTF8（Unicode（UTF-8）の場合）	0x00～0x7F※
SJIS（Shift-JIS の場合）	0x00～0xFF

注※

Shift-JIS の¥（0x5C），および ~（0x7E）は，UTF-8 でマルチバイト文字と扱われた場合は範囲対象外となります。

(7)　例題

例題 1

名前が M で始まる顧客の，顧客 ID（USERID），名前（NAME）を検索します。

```
SELECT "USERID","NAME"
    FROM "USERSLIST"
        WHERE "NAME" LIKE 'M%'
```

下線部分がLIKE 述語の指定です。

例題 2

名前が M で始まらない女性の顧客の，顧客 ID（USERID），名前（NAME）を検索します。

```
SELECT "USERID","NAME"
    FROM "USERSLIST"
        WHERE "NAME" NOT LIKE 'M%'
        AND SEX='F'
```

下線部分がLIKE 述語の指定です。

例題 3

パターン表（T_PATTERN）のパターン列（PATTERN）中の文字列で始まる商品を，売上表（T_SALES）の商品列（GOODS）から検索します。

```
SELECT "A"."GOODS" FROM "T_SALES" AS "A","T_PATTERN" AS "B"
    WHERE "A"."GOODS" LIKE "B"."PATTERN" + '%'
```

下線部分がLIKE 述語の指定です。

売上表（T_SALES）

商品ID(ID)	商品(GOODS)	売り上げ(SALES)
01	NOTE01	10000
02	BOOK02	15000
03	PENCIL03	20000
04	ERA04	5000
05	PEN05	50000
06	NOTE02	800

パターン表（T_PATTERN）

商品ID(ID)	パターン(PATTERN)
01	NOTE
02	PEN

検索結果

GOODS
NOTE01
PENCIL03
PEN05
NOTE02

例題４

売上表（T_SALES）から，次の条件に該当する商品名（GOODS）と売り上げ金額（SALES）を検索します。

- 商品名に _（下線）が含まれている商品
- 支店コード（BRANCH_CODE）がA001 の支店で扱っている商品

_（下線）は特殊文字のため，エスケープ文字に@を指定して，_を通常文字として扱うようにします。

```
SELECT ”GOODS”,”SALES” FROM ”T_SALES”
    WHERE ”GOODS” LIKE ’%@_%’ ESCAPE ’@’
    AND ”BRANCH_CODE”=’A001’
```

売上表（T_SALES）

商品ID(ID)	商品(GOODS)	売り上げ(SALES)	支店コード(BRANCH_CODE)
01	NOTE_01	10000	A001
02	BOOK_01	15000	A003
03	PEN	20000	A001
04	NOTE_02	5000	A002
05	PENCIL	50000	A003
06	BOOK_02	800	A001

検索結果

GOODS	SALES
NOTE_01	10000
BOOK_02	800

7.20.5　LIKE_REGEX 述語

正規表現を使用してデータを検索する際にLIKE_REGEX 述語を使用します。

(1)　指定形式

```
LIKE_REGEX述語 ： ：＝一致値　〔NOT〕 LIKE_REGEX
                          正規表現文字列　〔FLAG　{I｜IGNORECASE}〕
  一致値 ： ：＝値式
  正規表現文字列 ： ：＝文字列定数
```

(2)　指定形式の説明

一致値：

検索対象のデータを値式の形式で指定します。値式については，「7.21　値式」を参照してください。

一致値には，CHAR 型またはVARCHAR 型のデータを指定してください。

NOT：

NOT を指定した場合，指定した正規表現文字列によって表現される文字列の要素を含まない文字列が検索対象になります。

正規表現文字列：

正規表現を文字列定数の形式で指定します。文字列定数については，「6.3　定数」を参照してください。

正規表現文字列の長さは，1,024 バイト以下にしてください。

正規表現文字列は次の形式で指定します。

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　繰り返し因子*}

繰り返し因子 ： ：＝*左波括弧　下限値*　〔,〔*上限値*〕〕*右波括弧*

正規一次子 ： ：＝ {*文字指定子*｜*文字クラス*｜．｜＾｜＄｜*正規文字集合*｜(*正規表現*)}

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 ： ：＝ { [*文字列挙…*] ｜ [＾*文字列挙…*] }

文字列挙 ： ：＝ {*文字指定子*｜*文字指定子* － *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 ： ：＝ [:*正規文字集合識別子*:]

正規表現の規則を次の表に示します。

表 7-10　正規表現の規則

項番	正規表現		意味
1	文字指定子		長さ 1（文字数）の文字列を意味します。
2	.（ピリオド）		長さ 1（文字数）の任意の文字を意味します。
3	^（サーカムフレックス）		一致値の先頭を意味します。一致値に改行が含まれている場合，改行後の先頭は対象になりません。 正規文字集合を意味する角括弧の中に ^ を指定した場合は，列挙された文字を除く任意の文字を意味します。
4	$（ドル記号）		一致値の末尾を意味します。一致値に改行が含まれている場合，改行前の行末は対象になりません。
5	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。
6	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。
7	正規一次子?		直前の正規一次子の，0 回または 1 回の繰り返しを意味します。
8	正規表現 垂直棒 正規表現		垂直棒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。
9	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*}：直前の正規一次子の *n* 回繰り返しを意味します。 {*n*,*m*}：直前の正規一次子の *n* 回以上 *m* 回以下の繰り返しを意味します。 {*n*,}：直前の正規一次子の *n* 回以上の繰り返しを意味します。
10	上限値		最小値 0，最大値 256 の整数を意味します。
11	下限値		最小値 0，最大値 256 の整数を意味します。
12	[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。
13	[^ 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。
14	文字指定子 1 - 文字指定子 2		文字指定子 1 の文字から文字指定子 2 の文字までの，任意の文字（文字コードによる範囲）を意味します。 半角英大文字と半角英小文字を区別しない検索を指定した場合（FLAG I またはFLAG IGNORECASE を指定した場合），指定した文字コードの範囲，指定した文字コードの範囲にある半角英小文字を半角英大文字に変換したもの，および指定した文字コードの範囲にある半角英大文字を半角英小文字に変換したものを意味します。
15	正規文字集合識別子	alpha	任意の半角英大文字（¥，@，#を除く），または半角英小文字を意味します。[a-zA-Z]と同じ意味です。
16		upper	任意の半角英大文字（¥，@，#を除く）を意味します。[A-Z]と同じ意味です。
17		lower	任意の半角英小文字を意味します。[a-z]と同じ意味です。
18		digit	任意の数字を意味します。[0-9]と同じ意味です。
19		alnum	任意の半角英大文字（¥，@，#を除く），半角英小文字，または数字を意味します。[a-zA-Z0-9]と同じ意味です。

項番	正規表現		意味
20		space	半角スペース，タブ，キャリッジリターン，ラインフィード，垂直タブ，または改ページ文字を意味します。
21		blank	半角スペースまたはタブを意味します。
22		cntrl	制御文字を意味します。0x00～0x1f のどれかか，または 0x7f を意味します。
23		graph	0x21～0x7e のどれかを意味します。
24		print	0x20～0x7e のどれかを意味します。
25		punct	1 バイト 0x7e 以下の記号文字を意味します。[!-/¥:-@¥[-`¥{-~]と同じ意味です。
26		xdigit	16 進文字を意味します。[a-fA-F0-9]と同じ意味です。
27	文字クラス	¥d	任意の数字を意味します。[0-9]と同じ意味です。
28		¥D	数字以外を意味します。[^0-9]と同じ意味です。
29		¥w	任意の半角英大文字（¥，@，#を除く），半角英小文字，数字，および下線文字（_）を意味します。[a-zA-Z0-9_]と同じ意味です。
30		¥W	「半角英大文字（¥，@，#を除く），半角英小文字，数字，および下線文字（_）」以外を意味します。[^a-zA-Z0-9_]と同じ意味です。
31		¥s	半角スペース，タブ，キャリッジリターン，ラインフィード，垂直タブ，または改ページ文字を意味します。
32		¥S	「半角スペース，タブ，キャリッジリターン，ラインフィード，垂直タブ，および改ページ文字」以外を意味します。
33		¥A	一致値の先頭を意味します。
34		¥Z	一致値の末尾を意味します。

FLAG {I | IGNORECASE} ：

半角英大文字（¥，@，#を除く）と半角英小文字を区別しない検索をする場合に指定します。

I またはIGNORECASE のどちらを指定しても，同じ処理が実行されます。

なお，HADB サーバで使用している文字コードが Shift-JIS の場合，このオプションは指定できません。

(3) 正規表現の指定例

正規表現の代表的な指定例を次の表に示します。

表 7-11 正規表現の代表的な指定例

項番	項目	正規表現の指定	意味	正規表現の指定例	検索対象となる文字列
1	前方一致	^*nnn*	文字列の先頭部分が「*nnn*」である	^ACT	ACT，ACTOR，ACTION など，ACT で始まる文字列

項番	項目	正規表現の指定	意味	正規表現の指定例	検索対象となる文字列
2	後方一致	*nnn*$	文字列の最後の部分が「*nnn*」である	ING$	ING，BEING，HAVING など，ING で終わる文字列
3	任意一致	*nnn*	文字列中の任意の部分に「*nnn*」を含む	Sun	Sun，Sunday，Sundays など，Sun を含む文字列
4	完全一致	^*nnn*$	文字列が「*nnn*」と同じ	^EQUAL$	EQUAL
5	部分一致	.*nnn*.	文字列中の特定の部分が「*nnn*」と同じであり，その前後の部分に任意の文字がある	.I.	BIT，HIT，KIT など，3 文字で 2 文字目がI である文字列
6	1 回以上の繰り返し	*mmm*[0-9]+ または *mmm*[[:digit:]]+	文字列中の任意の部分に「*mmm*」があり，「*mmm*」の後ろに任意の数字がある	KFAA[0-9]+ または KFAA[[:digit:]]+	KFAA123，KFAA11104-E，KFAA11901-E など，KFAA で始まり，その後ろが任意の数字の文字列
7	幾つかの文字の選択	^*mmm*.*(n|o) または ^*mmm*.*[no]	文字列の先頭部分が「*mmm*」であり，*i* 文字目が n または o である（*i* は任意の数字）。	^KFAA.*(W|E) または ^KFAA.*[WE]	KFAA20008-W や，KFAA11901-E など，KFAA で始まり，その後ろにW または E の文字を含む文字列
8	*n* 回の繰り返し	*mmm*{*n*}	文字列の先頭部分が「*mmm*」であり，最後の文字を *n* 回繰り返す	123{3}	12333

(4) 述語の評価

指定した一致値の値が，正規表現文字列の表現する文字列の集合の要素を含む場合に真となります。そうでない場合は偽となります。ただし，正規表現文字列の長さが 0 の場合，一致値がナル値以外のときに真となります。

NOT を指定したときは，指定した一致値の値が，正規表現文字列の表現する文字列の要素を含まない場合に真となります。そうでない場合は偽となります。ただし，正規表現文字列の長さが 0 の場合，一致値がナル値以外のときに偽となります。

一致値がナル値の場合，述語は不定となります。

(5) 規則

(a) 一致値に関する規則

1. 一致値に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

(b) エスケープ文字に関する規則

1. 正規表現文字列中に¥がある場合，その¥はエスケープ文字として扱われます。
2. エスケープ文字の直後が特殊文字の場合，その特殊文字は通常文字と見なされます。特殊文字を次に示します。
   - . （ピリオド）
   - * （アスタリスク）
   - + （正符号）
   - ？ （疑問符）
   - | （垂直棒）
   - (（左括弧）
   -) （右括弧）
   - { （左波括弧）
   - } （右波括弧）
   - [（左角括弧）
   -] （右角括弧）
   - ¥ （円記号）
   - - （負符号）※
   - : （コロン）※
   - ^ （サーカムフレックス）
   - $ （ドル記号）

   注※　文字列挙の中に指定した場合に限り特殊文字として扱われます。
3. エスケープ文字の直後が通常文字の場合，そのエスケープ文字は読み飛ばされます。
4. エスケープ文字の後ろに文字がない場合，そのエスケープ文字は読み飛ばされます。
5. エスケープ文字が 2 つ連続している場合，2 つ連続しているエスケープ文字は通常文字 1 文字と見なされます。

(6) 性能面の考慮点

テキストインデクスを定義している場合，正規表現中のリテラル文字を使用して，テキストインデクスによるページの絞り込みが行われます。そのため，LIKE 述語やスカラ関数CONTAINS と同様に，正規表現中のリテラル文字が「a」または「0」などの単純で短い文字の場合は，テキストインデクスによるページの絞り込みの効果が少なくなります。また，パターンを表すメタ文字（丸括弧，角括弧，量指定子）は，組み合わせのパターン数が増加すると，テキストインデクスによるページの絞り込みに時間が掛かるため，検索時にテキストインデクスが使用されません。

そのため，パターンを表すメタ文字を使用しないと，テキストインデクスによるページの絞り込みの効果を上げることができます。例えば，HADB と HiRDB の 2 つの条件で検索する場合，「H(A|iR)DB」と指定するよりも，「HADB | HiRDB」と指定した方が，テキストインデクスによるページの絞り込みが効きます。

また，繰り返し因子でも，「(abc){1,3}」と指定するよりも，「abc | abcabc | abcabcabc 」と指定した方が，テキストインデクスによるページの絞り込みが効きます。

(7) 例題

例題

表T_MSG のMSG 列中のデータで，文字列がKFAA30 で始まり，そのあとに数字が 3 つ続き，-E で終わる行を検索します。

```
SELECT * FROM "T_MSG"
    WHERE "MSG" LIKE REGEX 'KFAA30[0-9]{3}-E'
```

下線部分がLIKE_REGEX 述語の指定です。

上記のLIKE_REGEX 述語の指定では，例えば，「KFAA30101-E」の文字列が対象になります。

7.20.6 NULL 述語

ナル値を検索する際にNULL 述語を使用します。ナル値については，「6.7 ナル値」を参照してください。

(1) 指定形式

```
NULL述語 : :=値式 IS 〔NOT〕 NULL
```

(2) 指定形式の説明

値式：

NULL 述語の評価対象列を指定します。値式の形式で指定します。値式については，「7.21 値式」を参照してください。

？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

NOT：

NOT を指定すると，ナル値でない行が検索対象になります。

(3) 述語の評価

指定した値式の結果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。

(4) 例題

例題 1

顧客表（USERSLIST）の名前（NAME）がナル値である顧客 ID（USERID）を検索します。

```
SELECT "USERID"
    FROM "USERSLIST"
        WHERE "NAME" IS NULL
```

下線部分がNULL 述語の指定です。

例題 2

顧客表（USERSLIST）の名前（NAME）がナル値ではない顧客 ID（USERID）を検索します。

```
SELECT "USERID"
    FROM "USERSLIST"
        WHERE "NAME" IS NOT NULL
```

下線部分がNULL 述語の指定です。

7.20.7 比較述語

探索条件に比較述語を指定できます。比較述語の例を次に示します。

（例）

販売履歴表（SALESLIST）から，2011/9/6 以降に商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT "USERID","PUR-CODE","PUR-DATE"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2011-09-06'
```

説明

- 下線部分が比較述語となります。
- >= を**比較演算子**といいます。
- 比較演算子の左右にある項を**比較演算項**といいます。この例では，PUR-DATE（列名）と DATE'2011-09-06'（定数）が比較演算項になります。

(1) 指定形式

```
比較述語 : :=比較演算項1 比較演算子 比較演算項2

  比較演算項 : := {値式 | 行値構成子}
  比較演算子 : := {= | <> | != | ^= | < | <= | > | >=}
```

(2) 指定形式の説明

比較演算項 1，*比較演算項 2*：

比較演算項を値式または行値構成子の形式で指定します。値式については，「7.21 値式」を参照してください。行値構成子については，「7.18 行値構成子」を参照してください。

比較演算項に行値構成子を指定する場合の指定規則を次に示します。

- 比較演算項に行値構成子を指定する場合，行値構成子要素を 2 つ以上指定してください。行値構成子要素を 1 つだけ指定した場合，その行値構成子は括弧で囲まれた値式として扱われます。

  行値構成子として扱われる例

  ```
  ("C1","C2") = (1,2)
  ```

  値式として扱われる例

  ```
  ("C1") = (1)
  ```

  上記は「"C1" = 1」と同じです。

- 比較演算項に行値構成子を指定する場合，比較演算子の左右両方の比較演算項に行値構成子を指定してください。

比較演算子：

比較演算子には，=，<>，!=，^=，<，<=，>，>=のどれかを指定します。各指定値の意味を次の表に示します。

表 7-12 比較演算子の意味

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

（凡例）

X，*Y*：比較演算項

重要

比較演算項に行値構成子を指定する場合，比較演算子には上記の表の項番 1～2 の=，<>，!=，^=を指定できます。項番 3～6 の比較演算子は指定できません。

(3) 述語の評価

■比較演算項に値式を指定する場合

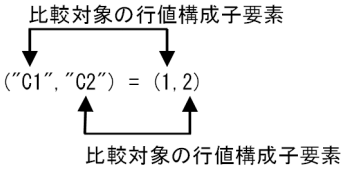
比較演算子の左右の比較演算項が，比較条件を満たす場合に**真**となります。

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。

■比較演算項に行値構成子を指定する場合

比較演算子の左右の行値構成子中の各行値構成子要素の値を比較して判定します。

（例）



比較演算子が「=」の場合

比較対象の行値構成子要素の値が，すべて等しい場合は**真**となります。

比較対象の行値構成子要素の値が，1 つ以上等しくない場合は**偽**となります。

行値構成子要素にナル値が 1 つ以上あり，かつ比較対象の行値構成子要素の値（ただし，ナル値とナル値の比較対象の行値構成子要素の値を除く）がすべて等しい場合は不定となります。

（例）

比較述語の指定例	比較述語の結果
(1,2,3) = (1,2,3)	真
(1,2,3) = (1,2,4)	偽
(1,2,3) = (1,ナル値,4)	偽
(1,2,3) = (1,2,ナル値)	不定
(1,2,ナル値) = (1,2,ナル値)	不定

比較演算子が「<>」，「!=」，「^=」の場合

比較対象の行値構成子要素の値が，1 つ以上等しくない場合は**真**となります。

比較対象の行値構成子要素の値が，すべて等しい場合は**偽**となります。

行値構成子要素にナル値が 1 つ以上あり，かつ比較対象の行値構成子要素の値（ただし，ナル値とナル値の比較対象の行値構成子要素の値を除く）がすべて等しい場合は不定となります。

（例）

比較述語の指定例	比較述語の結果
(1,2,3) <> (1,2,4)	真
(1,2,3) <> (1,ナル値,4)	真
(1,2,3) <> (1,2,3)	偽

比較述語の指定例	比較述語の結果
(1,2,3) <> (1,2,ナル値)	不定
(1,2,ナル値) <> (1,2,ナル値)	不定

(4) 規則

(a) 比較演算項に値式を指定する場合の規則

1. *比較演算項 1* と*比較演算項 2* の結果のデータ型が，比較可能なデータ型になるようにしてください。比較可能なデータについては，「6.2.2 変換，代入，比較できるデータ型」の「(1) 比較できるデータ型」を参照してください。
2. 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。
3. 比較演算項の値式にバイナリデータを指定した場合，比較演算子には「=」，「<>」，「!=」，または「^=」だけが指定できます。
4. 比較演算項の値式には配列データを指定できません。
5. 比較演算子の両側に？パラメタだけの比較演算項を指定できません。
   - 指定できないケース：?=?
   - 指定できるケース：”C1”=?，?=10
6. *比較演算項 2* には，スカラ関数CONTAINS を指定できません。
7. *比較演算項 1* にスカラ関数CONTAINS を指定した場合，比較演算子には「>」を指定してください。また，*比較演算項 2* には0 を指定してください。

(b) 比較演算項に行値構成子を指定する場合の規則

1. 比較演算子の左右に指定する行値構成子の行値構成子要素の数は同じにしてください。

   正しい指定例：(”C1”,”C2”,”C3”) = (1,2,3)

   誤った指定例：(”C1”,”C2”,”C3”) = (1,2,3,4)
2. 比較演算子の左右に指定する行値構成子の各行値構成子要素の組み合わせのデータ型は，比較可能なデータ型になるようにしてください。比較可能なデータについては，「6.2.2 変換，代入，比較できるデータ型」の「(1) 比較できるデータ型」を参照してください。
3. 比較演算子の左右に指定する行値構成子の各行値構成子要素に，両方の行値構成子要素が？パラメタとなる組み合わせは指定できません。

   正しい指定例：(”C1”,?,”C3”) = (1,2,?)

   誤った指定例：(”C1”,?,”C3”) = (1,?,3)
4. 行値構成子要素に？パラメタを指定する場合，その？パラメタのデータ型には，比較相手となる行値構成子要素のデータ型が仮定されます。

5. 比較演算子の左右に指定する行値構成子のうち，行値構成子要素が値指定だけとなる行値構成子を 1 つ以上指定してください。

   正しい指定例：(”C1”,”C2”,”C3”) = (1,2,3)

   誤った指定例：(”C1”,”C2”,”C3”) = (”C4”,”C5”,”C6”)

6. 比較演算子の左右に指定する行値構成子の行値構成子要素には，配列型のデータを指定できません。

7. 比較演算子の左右に指定する行値構成子の行値構成子要素中には，次の値式を指定できません。

   - 集合関数
   - スカラ副問合せ
   - 配列要素参照

8. 行値構成子を指定した比較述語は，次のどちらかに指定できます。

   - WHERE 句
   - DELETE 文またはUPDATE 文のWHERE 探索条件

9. 行値構成子を指定した比較述語は，CASE 式の探索条件中には指定できません。

(5) 例題

例題 1

販売履歴表（SALESLIST）から，2011/9/4 以降に商品コードP001 またはP003 の商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE），購入日（PUR-DATE）を検索します。

```
SELECT ”USERID”,”PUR-CODE”,”PUR-DATE”
    FROM ”SALESLIST”
        WHERE ”PUR-DATE”>=DATE’2011-09-04’
        AND (”PUR-CODE”=’P001’ OR ”PUR-CODE”=’P003’)
```

下線部分が比較述語の指定です。

比較演算項には値式を指定しています。

例題 2

販売履歴表（SALESLIST）から，顧客 ID（USERID）がU001 かつ商品コード（PUR-CODE）がP001 の行を検索します。

```
SELECT ”USERID”,”PUR-CODE”,”PUR-NUM”
  FROM ”SALESLIST”
  WHERE (”USERID”,”PUR-CODE”) = (’U001’,’P001’)
```

下線部分が比較述語の指定です。

比較演算項には行値構成子を指定しています。

7.20.8　限定述語

表副問合せの結果と値式の結果を比較して判定します。

(1)　指定形式

限定述語 ： ：＝*値式* {=｜<>｜!=｜^=｜<｜<=｜>｜>=}　{ {ANY｜SOME}　｜ALL} *表副問合せ*

(2)　指定形式の説明

値式：

限定述語の評価対象列を指定します。値式の形式で指定します。値式については，「7.21　値式」を参照してください。

ANY またはSOME：

表副問合せの結果の任意の行のうち，*値式*との比較条件を満たしている行が 1 つでもある場合は，限定述語の結果が真になります。

なお，ANY またはSOME のどちらを指定しても結果は同じになります。

ALL：

次に示すどちらかの条件を満たしている場合に，限定述語の結果が真になります。

- 表副問合せの結果のすべての行が，*値式*との比較条件を満たしている場合
- 表副問合せの結果が 0 行（空集合）の場合

表副問合せ：

表副問合せについては，「7.3　副問合せ」を参照してください。

(3)　述語の評価

(a)　ANY または SOME を指定した場合

- 表副問合せの結果の任意の行のうち，*値式*との比較条件を満たしている行が 1 つでもある場合は，限定述語の結果が真になります。
- 次に示すどちらかの条件を満たしている場合は，限定述語の結果が偽になります。
  - 表副問合せの結果のすべての行が，*値式*との比較条件を満たしていない場合
  - 表副問合せの結果が 0 行（空集合）の場合
- 上記以外の場合は不定になります。

ANY またはSOME を指定した限定述語の結果を次の表に示します。

表 7-13　ANY または SOME を指定した場合の限定述語の結果

項番	表副問合せの各行に対する比較結果		限定述語の結果
1	真の行あり		真
2	真の行なし	不定あり	不定
3		不定なし	偽
4	0 行（空集合）		偽

(b)　ALL を指定した場合

- 次に示すどちらかの条件を満たしている場合は，限定述語の結果が真になります。
  - 表副問合せの結果のすべての行が，*値式*との比較条件を満たしている場合
  - 表副問合せの結果が 0 行（空集合）の場合
- 表副問合せの結果の任意の行のうち，*値式*との比較条件を満たさない行が 1 つでもある場合は，限定述語の結果が偽になります。
- 上記以外の場合は不定になります。

ALL を指定した限定述語の結果を次の表に示します。

表 7-14　ALL を指定した場合の限定述語の結果

項番	表副問合せの各行に対する比較結果		限定述語の結果
1	偽の行あり		偽
2	偽の行なし	不定あり	不定
3		不定なし	真
4	0 行（空集合）		真

(4)　規則

1. 表副問合せの結果の列数は 1 になるようにしてください。
2. 値式にバイナリデータを指定した場合，比較演算子には=，<>，!=，および^=だけが指定できます。
3. 値式には配列データを指定できません。
4. 表副問合せの選択式の値式には配列データを指定できません。

(5)　留意事項

1. 限定述語を指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してくださ

い。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。

2. 限定述語（=ANY 指定または=SOME 指定）を指定した場合，HADB によって表副問合せの結果の重複排除を行います。

(6)　例題

例題

顧客表（USERSLIST）と販売履歴表（SALESLIST）から，商品コード（PUR-CODE）がP001 の商品を購入した顧客の情報を検索します。

```
SELECT * FROM "USERSLIST"
    WHERE "USERID"=ANY(SELECT "USERID" FROM "SALESLIST"
                           WHERE "PUR-CODE"='P001')
```

下線部分が限定述語の指定です。

7.21　値式

ここでは，値式について説明します。

7.21.1　値式の指定形式および規則

SQL 文中で指定する値は，列名，定数，集合関数，スカラ関数，ウィンドウ関数，CASE 式，四則演算（＋，－，＊，／），連結演算（＋，||）などを使って指定できます。この指定を**値式**といいます。値式の例を次に示します。

（例）

- 列名だけを指定する　"C1"
- 定数だけを指定する　'HADB', 100, DATE'2011-09-06'
- 列名と四則演算を使って指定する　"C1"+10
- 列名と連結演算を使って指定する　"C1"||"C2"
- 集合関数と四則演算を使って指定する　MAX("C1")/2

(1)　指定形式

```
値式：：＝｛数値式｜文字値式｜日時値式｜バイナリ値式｝


  数値式：：＝｛値式一次子｜四則演算｝
  文字値式：：＝｛値式一次子｜連結演算｝
  日時値式：：＝｛値式一次子｜日時演算｝
  バイナリ値式：：＝｛値式一次子｜連結演算｝

    値式一次子：：＝｛(値式)｜列指定｜値指定｜集合関数
                    ｜スカラ関数｜ウィンドウ関数｜CASE式
                    ｜ラベル付き間隔｜スカラ副問合せ
                    ｜配列要素参照｝
```

(2)　指定形式の説明

四則演算：

四則演算の詳細については，「7.26　四則演算」を参照してください。

連結演算：

連結演算の詳細については，「7.27　連結演算」を参照してください。

日時演算：

日時演算の詳細については，「7.28　日時演算」を参照してください。

列指定：

列指定の詳細については，「6.1.5　名前の修飾」の「(5)　列指定の指定形式」を参照してください。

値指定：

値指定の詳細については，「7.22　値指定」を参照してください。

集合関数：

集合関数の詳細については，「7.23　集合関数」を参照してください。

スカラ関数：

スカラ関数の詳細については，「8.　スカラ関数」を参照してください。

ウィンドウ関数：

ウィンドウ関数については，「7.24　ウィンドウ関数」を参照してください。

CASE 式：

CASE 式の詳細については，「7.30　CASE 式」を参照してください。

ラベル付き間隔：

ラベル付き間隔の詳細については，「7.29　ラベル付き間隔」を参照してください。

スカラ副問合せ：

スカラ副問合せの詳細については，「7.3　副問合せ」を参照してください。

配列要素参照：

配列要素参照の詳細については，「7.31　配列要素参照」を参照してください。

(3)　規則

1. 値式中には，スカラ演算と集合関数を合計 500 個まで指定できます。値式中に指定した列が，ビュー表の列，導出表の列，または問合せ名の列の場合，その基となる値式を展開したあとの値式の総数が 10,000 個以内となるようにしてください。

   なお，ビュー表，導出表，または問合せ名の指定内容によっては，値式が付加されることがあります。値式が付加される条件については，「7.32.6　内部導出表にスカラ関数 CONVERT を付加する条件」を参照してください。

   スカラ演算とは，値式中で指定できる演算のうち，次に示すものの総称です。

   - 四則演算
   - 連結演算
   - 日時演算
   - スカラ関数
   - ウィンドウ関数
   - CASE 式
   - 配列要素参照

2. 次に示すスカラ演算を入れ子にして指定する場合，入れ子の上限は 15 回になります。値式中に指定した列がビュー表の列，または導出表の列の場合，その基となる値式を展開したあとに対象となるスカラ演算の入れ子の数が 15 回以内となるようにしてください。

   なお，ビュー表または導出表の指定内容によっては，値式が付加されることがあります。値式が付加される条件については，「7.32.6　内部導出表にスカラ関数 CONVERT を付加する条件」を参照してください。

   また，スカラ演算を混在させて入れ子にした場合も，合計の入れ子の上限が 15 回になります。

   - スカラ関数
   - ウィンドウ関数
   - CASE 式

   入れ子の数え方の例を次に示します。

   例 1：スカラ関数SUBSTR を 15 回入れ子にして指定

```
SUBSTR(SUBSTR(SUBSTR(SUBSTR(SUBSTR(
SUBSTR(SUBSTR(SUBSTR(SUBSTR(SUBSTR(
SUBSTR(SUBSTR(SUBSTR(SUBSTR(SUBSTR(
SUBSTR("C1",1),1),1),1),1),1),1),1),1),1),1),1),1),1),1)
```

   例 2：CASE 式中にCASE 式を繰り返し，2 回入れ子にして指定

```
CASE WHEN
    CASE WHEN
                CASE WHEN "C1">100
                    THEN  "C1"-100
                    ELSE "C1"
                END >100
            THEN "C1"-100
            ELSE "C1"
        END >100
    THEN "C1"-100
    ELSE "C1"
END
```

   例 3：CASE 式とスカラ関数SUBSTR を混在させて指定（この場合，入れ子は 2 回）

```
SUBSTR(CASE WHEN "C1">100 THEN SUBSTR("C2",1,10)
                          ELSE SUBSTR("C2",1,5)
       END,1,5)
```

3. 演算途中でオーバフローが発生した場合，SQL がエラーになります。
4. スカラ演算の評価順序の優先順位を次に示します。
   - 括弧内
   - *または/
   - ＋，-，または||
5. 定数と等価な値式となる条件を次の表に示します。ただし，値式の結果のデータ型およびデータ長は，定数のデータ型およびデータ長ではなく，各値式から導き出されたデータ型およびデータ長となります。

表 7-15　定数と等価な値式となる条件

項番	値式の種類		定数と等価な値式となる条件
1	四則演算		第 1 演算項，および第 2 演算項に定数を指定した場合
2	連結演算		
3	日時演算		次に示すすべての条件を満たした場合 • 第 1 演算項に定数を指定した場合 • 第 2 演算項がラベル付き間隔で，ラベル付き間隔に指定した*値式一次子*に定数を指定した場合 • 乗除算する*値式一次子*に定数を指定した場合（ラベル付き間隔に乗除算を指定した場合に限る）
4	スカラ関数	ABS	対象データに定数を指定した場合
5		ACOS	
6		ARRAY_MAX_CARDINALITY	常に定数として扱います。
7		ASCII	対象データに定数を指定した場合（ただし，次の場合を除きます） • 対象データに実長 0 バイトまたは実長 0 文字のデータを指定した場合
8		ASIN	対象データに定数を指定した場合
9		ATAN	
10		ATAN2	対象データ 1 および対象データ 2 に定数を指定した場合
11		BIN	対象データに定数を指定した場合
12		BITAND	対象データ 1 および対象データ 2 に定数を指定した場合
13		BITLSHIFT	対象データおよびシフトビット数に定数を指定した場合
14		BITNOT	対象データに定数を指定した場合
15		BITOR	対象データ 1 および対象データ 2 に定数を指定した場合
16		BITRSHIFT	対象データおよびシフトビット数に定数を指定した場合
17		BITXOR	対象データ 1 および対象データ 2 に定数を指定した場合
18		CAST	変換対象データに定数を指定した場合（ただし，次の場合を除きます） • 実長が 0 バイトの文字列定数に対して，文字データ以外への変換を指定した場合
19		CEIL	対象データに定数を指定した場合
20		CHR	対象データに定数を指定した場合（ただし，次の場合を除きます） • 対象データに負の整数値を指定した場合
21		COALESCE	引数に指定した対象データが 1 つであり，かつ定数を指定した場合
22		CONCAT	対象データ 1 および対象データ 2 に定数を指定した場合
23		CONVERT	変換対象データに定数を指定した場合（ただし，次の場合を除きます）

項番	値式の種類		定数と等価な値式となる条件
			• 実長が 0 バイトの文字列定数に対して，文字データ以外への変換を指定した場合 • 書式指定を指定した場合
24		COS	対象データに定数を指定した場合
25		COSH	
26		DATEDIFF	開始日時および終了日時に定数を指定した場合
27		DAYOFWEEK	対象データに定数を指定した場合
28		DAYOFYEAR	
29		DEGREES	角度に定数を指定した場合
30		EXP	指数に定数を指定した場合
31		EXTRACT	抽出元データに定数を指定した場合
32		FLOOR	対象データに定数を指定した場合
33		GETAGE	生年月日および基準日に定数を指定した場合
34		HEX	対象データに定数を指定した場合
35		LASTDAY	日時データに定数を指定した場合
36		LEFT	抽出元の文字データおよび抽出文字数に定数を指定した場合（ただし，次の場合を除きます） • 抽出文字数に負の値を指定した場合
37		LENGTH	対象データに定数を指定した場合
38		LENGTHB	
39		LN	
40		LOG	底および対象データに定数を指定した場合
41		LOWER	変換対象の文字データに定数を指定した場合
42		LPAD	対象データ，文字数，および埋め込み文字列に定数を指定した場合（ただし，次の場合を除きます） • 文字数に負の値を指定した場合
43		LTRIM	対象データおよび削除文字に定数を指定した場合
44		MOD	被除数および除数に定数を指定した場合
45		PI	常に定数として扱います。
46		POWER	対象データおよび指数に定数を指定した場合
47		RADIANS	角度に定数を指定した場合
48		REPLACE	次のどちらかの条件を満たす場合

項番	値式の種類		定数と等価な値式となる条件
			• 対象データ，置換対象文字列，および置換後の文字列に定数を指定した場合 • 対象データおよび置換対象文字列に定数を指定し，置換後の文字列を省略した場合
49		RIGHT	抽出元の文字データおよび抽出文字数に定数を指定した場合（ただし，次の場合を除きます） • 抽出文字数に負の値を指定した場合
50		ROUND	• 数学関数のROUND の場合 対象データおよび桁数に定数を指定した場合 • 日時関数のROUND の場合 日時データに定数を指定した場合
51		RPAD	対象データ，文字数，および埋め込み文字列に定数を指定した場合（ただし，次の場合を除きます） • 文字数に負の値を指定した場合
52		RTRIM	対象データおよび削除文字に定数を指定した場合
53		SIGN	対象データに定数を指定した場合
54		SIN	
55		SINH	
56		SQRT	
57		SUBSTR	抽出元の文字データ，開始位置，および抽出文字数に定数を指定した場合（ただし，次の場合を除きます） • 抽出文字数に負の値を指定した場合
58		SUBSTRB	抽出元のバイナリデータ，開始位置，および抽出バイト数に定数を指定した場合（ただし，次の場合を除きます） • 抽出バイト数に負の値を指定した場合
59		TAN	対象データに定数を指定した場合
60		TANH	
61		TIMESTAMPADD	加算値および対象データに定数を指定した場合
62		TIMESTAMPDIFF	開始日時および終了日時に定数を指定した場合
63		TRANSLATE	対象データ，置換対象文字，および置換後の文字に定数を指定した場合
64		TRIM	対象データおよび削除文字に定数を指定した場合
65		TRUNC	• 数学関数のTRUNC の場合 対象データおよび桁数に定数を指定した場合 • 日時関数のTRUNC の場合 日時データに定数を指定した場合
66		UPPER	変換対象の文字データに定数を指定した場合

7.21.2　値式の結果のデータ型

次の個所に指定した値式のデータ型によって結果のデータ型が決まります。

- CASE 式
- 集合演算の結果導出される列
- 表値構成子によって導出される列
- スカラ関数COALESCE，DECODE，GREATEST，LEAST，LTDECODE，NULLIF，BITAND，BITOR，およびBITXOR

（例 1）CASE 式の場合

```
CASE
   WHEN 探索条件 THEN 値式1
   WHEN 探索条件 THEN 値式2
   ELSE 値式3
END
```

値式 1～値式 3 のデータ型によって，CASE 式の結果のデータ型が決まります。

（例 2）スカラ関数GREATEST の場合

```
GREATEST（値式1，値式2，値式3）
```

値式 1～値式 3 のデータ型によって，スカラ関数GREATEST の結果のデータ型が決まります。

ここでは，結果のデータ型の決定規則について説明します。

(1)　値式のデータ型が文字データの場合

- CASE 式，スカラ関数COALESCE，DECODE，GREATEST，LEAST，LTDECODE，NULLIF の場合
  - 結果のデータ型はVARCHAR 型になります。
  - 結果のデータ長は，データ長がいちばん長い値式のデータ長になります。
- 集合演算の結果導出される列，および表値構成子によって導出される列の場合
  - すべての値式のデータ型がCHAR 型であり，かつすべてのデータ長が同じ場合は，結果のデータ型はCHAR 型になります。それ以外の場合は，結果のデータ型はVARCHAR 型になります。
  - 結果のデータ長は，データ長がいちばん長い値式のデータ長になります。

(2)　値式のデータ型が数データの場合

値式のデータ型が数データの場合，結果のデータ型は次の表に示すとおりに決まります。

表 7-16　値式のデータ型と結果のデータ型の関係（値式のデータ型が数データの場合）

項番	値式 N のデータ型	値式 N+1 のデータ型	結果のデータ型
1	SMALLINT	SMALLINT	SMALLINT
2		INTEGER	INTEGER

項番	値式 N のデータ型	値式 N+1 のデータ型	結果のデータ型
3		DECIMAL, NUMERIC	DECIMAL
4		DOUBLE PRECISION, FLOAT	DOUBLE PRECISION
5	INTEGER	SMALLINT	INTEGER
6		INTEGER	
7		DECIMAL, NUMERIC	DECIMAL
8		DOUBLE PRECISION, FLOAT	DOUBLE PRECISION
9	DECIMAL, NUMERIC	SMALLINT	DECIMAL
10		INTEGER	
11		DECIMAL, NUMERIC	
12		DOUBLE PRECISION, FLOAT	DOUBLE PRECISION
13	DOUBLE PRECISION, FLOAT	SMALLINT	DOUBLE PRECISION
14		INTEGER	
15		DECIMAL, NUMERIC	
16		DOUBLE PRECISION, FLOAT	

（凡例）N：1 以上の整数

■結果のデータ型がDECIMAL 型の場合

精度と位取りは次のように決まります。*値式 1* がDECIMAL(*p1*,*s1*)，*値式 2* がDECIMAL(*p2*,*s2*)，*値式 N* がDECIMAL(*pN*,*sN*)とします。

- 精度＝ MIN（38，$Pmax + Smax$）
- 位取り＝ MIN（$Smax$，$38-Pmax$）

$Pmax$ ＝ MAX（$p1-s1$，$p2-s2$，…，$pN-sN$）

$Smax$ ＝ MAX（$s1$，$s2$，…，sN）

値式のデータ型がINTEGER 型の場合は，DECIMAL(20,0)として計算してください。値式のデータ型がSMALLINT 型の場合は，DECIMAL(10,0)として計算してください。

なお，結果の数データが，ここで求めた精度，位取りに収まらない場合，小数部が切り捨てられます。例を次に示します。

例 1

表T1 のC1 列，C2 列のデータ型がそれぞれDECIMAL(37,0)，DECIMAL(10,2)で，値がそれぞれナル値，12345678.12 の場合，次のSELECT 文を実行します。

```
SELECT COALESCE(”C1”,”C2”) FROM ”T1”
```

＜検索結果＞

```
12345678.1
```

この場合，スカラ関数COALESCE の実行結果のデータ型がDECIMAL(38,1)になり，小数部の下位の桁が切り捨てられます。

例 2

行値構成子要素に1.1234567890123456789 および10 を指定した表値構成子から導出された導出表DT を含む次のSELECT 文を実行したとします。この場合，定数1.1234567890123456789 が DECIMAL(20,19)，定数10 がINTEGER として扱われます。また，INTEGER はDECIMAL(20,0)として扱われるため，表値構成子から導出される列C1 の結果のデータ型はDECIMAL(38,18)になります。その結果，小数部の下位の桁が切り捨てられます。

```
SELECT ”C1” FROM (VALUES(1.1234567890123456789),(10)) ”DT”(”C1”)
```

＜検索結果＞

```
1.123456789012345678
10.000000000000000000
```

また，次のように定数10 を明示的に 10 進数定数の表記とし，DECIMAL 型とすることで小数部の下位の桁が切り捨てられることを防ぐことができます。この場合，定数1.1234567890123456789 が DECIMAL(20,19)，定数10.0 がDECIMAL(3,1)として扱われるため，表値構成子から導出される列C1 の結果のデータ型がDECIMAL(21,19)になります。

```
SELECT ”C1” FROM (VALUES(1.1234567890123456789),(10.0)) ”DT”(”C1”)
```

＜検索結果＞

```
1.1234567890123456789
10.0000000000000000000
```

(3) 値式のデータ型が日時データの場合

値式のデータ型が日時データの場合，結果のデータ型は次の表に示すとおりに決まります。

表 7-17 値式のデータ型と結果のデータ型の関係（値式のデータ型が日時データの場合）

項番	値式 N のデータ型	値式 N+1 のデータ型	結果のデータ型
1	DATE	DATE	DATE
2		TIMESTAMP	TIMESTAMP※
3	TIME	TIME	TIME

項番	値式 N のデータ型	値式 N+1 のデータ型	結果のデータ型
4	TIMESTAMP	DATE	TIMESTAMP※
5		TIMESTAMP	TIMESTAMP

（凡例）*N*：1 以上の整数

注※

DATE 型のデータをTIMESTAMP 型のデータに変換します。その際，時刻部分に 00:00:00 を補います。

■小数秒精度の指定がある場合

値式 1～*値式 N* の結果に小数秒精度が含まれる場合，結果の小数秒精度は次のように決まります。*値式 1* の小数秒精度を*p1*，*値式 2* の小数秒精度を*p2*，*値式 N* の小数秒精度を*pN* とします。

```
MAX（p1，p2，…，pN）
```

小数秒精度によって，結果のデータ長が決まります。

(4) 値式のデータ型がバイナリデータの場合

値式のデータ型がバイナリデータの場合，結果のデータ型は次のとおりに決まります。

- CASE 式，およびスカラ関数COALESCE，NULLIF の場合
  - 結果のデータ型はVARBINARY 型になります。
  - 結果のデータ長は，データ長がいちばん長い値式のデータ長になります。
- 集合演算の結果導出される列，表値構成子によって導出される列，およびスカラ関数BITAND，BITOR，BITXOR の場合
  - 結果のデータ型は，すべての値式のデータ型がBINARY 型であり，かつすべてのデータ長が同じ場合は，BINARY 型になります。それ以外の場合は，VARBINARY 型になります。
  - 結果のデータ長は，データ長がいちばん長い値式のデータ長になります。

7.22　値指定

ここでは，値指定について説明します。

7.22.1　値指定の指定形式

SQL 文中の値指定ができる個所には，次に示す指定ができます。

- 定数
- ？パラメタ
- 日時情報取得関数のCURRENT_DATE
- 日時情報取得関数のCURRENT_TIME
- 日時情報取得関数のCURRENT_TIMESTAMP
- ユーザ情報取得関数のCURRENT_USER

(1)　指定形式

```
値指定 : := {定数 | ?パラメタ | CURRENT_DATE | CURRENT_TIME
            | CURRENT_TIMESTAMP | CURRENT_USER}
```

(2)　指定形式の説明

定数：

定数の詳細については，「6.3　定数」を参照してください。

値指定の結果のデータ型は，指定した定数のデータ型になります。

?パラメタ：

？パラメタの詳細については，「6.6　変数（？パラメタ）」を参照してください。

値指定の結果のデータ型は，？パラメタを指定した個所によって変わります。

CURRENT_DATE：

CURRENT_DATE の詳細については，「6.4.1　CURRENT_DATE」を参照してください。

値指定の結果のデータ型はDATE 型になります。

CURRENT_TIME：

CURRENT_TIME の詳細については，「6.4.2　CURRENT_TIME」を参照してください。

値指定の結果のデータ型はTIME 型になります。

CURRENT_TIMESTAMP：

CURRENT_TIMESTAMP の詳細については，「6.4.3　CURRENT_TIMESTAMP」を参照してください。

値指定の結果のデータ型はTIMESTAMP 型になります。

CURRENT_USER：

CURRENT_USER の詳細については，「6.5.1　CURRENT_USER」を参照してください。

値指定の結果のデータ型はVARCHAR 型になります。

(3)　例題

値指定の指定例を例題を使って説明します。

例題 1

販売履歴表（SALESLIST）に次に示すデータ（行）を挿入します。

- 顧客 ID（USERID）：U00358
- 商品コード（PUR-CODE）：P003
- 販売個数（PUR-NUM）：5
- 購入日（PUR-DATE）：2011-09-08

```
INSERT INTO "SALESLIST"
    ("USERID","PUR-CODE","PUR-NUM","PUR-DATE")
    VALUES('U00358','P003',5,DATE'2011-09-08')
```

下線部分が値指定の個所です。

例題 2

販売履歴表（SALESLIST）から，本日商品を購入した顧客の，顧客 ID（USERID），商品コード（PUR-CODE）を検索します。

```
SELECT "USERID","PUR-CODE"
    FROM "SALESLIST"
        WHERE "PUR-DATE"=CURRENT_DATE
```

下線部分が値指定の個所です。

7.23　集合関数

集合関数を使用すると，複数の行の集計値を算出できます。集合関数の一覧を次の表に示します。

表 7-18　集合関数の一覧

項番	集合関数の一覧		説明
1	COUNT(*)		行数（件数）を求めます。
2	一般集合関数	AVG	平均値を求めます。
3		COUNT	行数（件数）を求めます。
4		MAX	最大値を求めます。
5		MIN	最小値を求めます。
6		SUM	合計値を求めます。
7		STDDEV_POP	母集団標準偏差を求めます。
8		STDDEV_SAMP	標本標準偏差を求めます。
9		VAR_POP	母集団分散を求めます。
10		VAR_SAMP	標本分散を求めます。
11	逆分布関数	MEDIAN	順序付けされた一連の値の中央値を求めます。線形補間された値となることがあります。
12		PERCENTILE_CONT	順序付けされた一連の値のパーセンタイル（百分位数）を求めます。線形補間された値となることがあります。
13		PERCENTILE_DISC	順序付けされた一連の値のパーセンタイル（百分位数）を求めます。一連の値の中から結果を返します。
14	LISTAGG 集合関数	LISTAGG	順序付けされた一連の値を連結し，値と値の間に区切り文字列を挿入した文字列を求めます。
15	ARRAY_AGG 集合関数	ARRAY_AGG	対象データに指定した値式の結果（値式によって集計された値）を，先頭から順に配列要素とする配列データを返します。

7.23.1　COUNT(*)

行数（件数）を求めます。

(1)　指定形式

集合関数COUNT(*)：：＝COUNT(*)

(2) 規則

1. 実行結果のデータ型はINTEGER になります。
2. 入力行数が 0 の場合，実行結果は0 になります。

(3) 例題

例題

社員表（EMPLIST）から，部門コードS001 の男性社員の人数を求めます。

```
SELECT COUNT(*) AS "COUNT"
    FROM "EMPLIST"
        WHERE "SCODE"='S001' AND "SEX"='M'
```

実行結果の例

COUNT
15

7.23.2 AVG

平均値を求めます。

(1) 指定形式

一般集合関数AVG：：＝ {AVG(〔ALL〕 *値式*)｜AVG(DISTINCT *値式*)}

(2) 指定形式の説明

AVG(〔ALL〕 *値式*)：

値式の結果の平均値を求めます。値式については，「7.21 値式」を参照してください。

ALL は省略できます。指定のありなしに関係なく結果は同じになります。

AVG(DISTINCT *値式*)：

値式の結果の平均値を求めます。値式については，「7.21 値式」を参照してください。

なお，同じ値がある場合，その値は 1 つだけカウント対象にします。例えば，値式の結果の値が2，3，2，4 の場合，実行結果は（2 ＋ 3 ＋ 4）/3 ＝ 3 になります。

(3) 規則

1. ナル値は集計対象に含まれません。
2. 平均値を算出する際，有効数字以下は切り捨てられます。
3. 次に示す場合，実行結果はナル値になります。

- 入力行数が 0 の場合
- 集計対象の値がすべてナル値の場合

4. 値式に指定できるデータ型と一般集合関数AVG の実行結果のデータ型を次の表に示します。

表 7-19　値式に指定できるデータ型と一般集合関数 AVG の実行結果のデータ型

項番	値式に指定できるデータ型	一般集合関数 AVG の実行結果のデータ型
1	INTEGER	INTEGER
2	SMALLINT	
3	DECIMAL(*m*,*n*)	DECIMAL(38,38-*m*＋*n*)
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	DOUBLE PRECISION
6	FLOAT	

5. 集合関数の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となります。

(4)　例題

例題

社員表（EMPLIST）から，社員の平均年齢（AGE）を部門（SCODE）ごとに求めます。

```
SELECT ”SCODE”,AVG(”AGE”) AS ”AVG-AGE”
    FROM ”EMPLIST”
        GROUP BY ”SCODE”
```

実行結果の例

SCODE	AVG-AGE
S001	35
S002	29
S003	33

7.23.3　COUNT

行数（件数）を求めます。

(1)　指定形式

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

(2) 指定形式の説明

COUNT(〔ALL〕 *値式*)：

値式を指定します。値式については，「7.21 値式」を参照してください。

ALL は省略できます。指定のありなしに関係なく結果は同じになります。

COUNT(DISTINCT *値式*)：

値式を指定します。値式については，「7.21 値式」を参照してください。値が同じ行は重複してカウントされません。

なお，DISTINCT を指定した場合，値式には配列データを指定できません。

上記 2 つの指定形式の実行結果の違いを例を使って説明します。GROUP BY 句を指定しない場合と，GROUP BY 句を指定する場合に分けて説明します。

(a) 例 1 GROUP BY 句を指定しない場合

USERID	NAME	SEX	SCODE
U00555	Taro Tanaka	M	S001
U00358	Nancy White	F	S003
U00799	Taro Tanaka	M	S001
U00212	Maria Gomez	F	S001
U00687	Taro Tanaka	M	S002
U00869	NULL	M	S003

上記の社員表（EMPLIST）に対して，次に示すSELECT 文を実行します。

```
SELECT COUNT(”NAME”) AS ”COUNT-ALL”,
       COUNT(DISTINCT ”NAME”) AS ”COUNT-DISTINCT”
    FROM ”EMPLIST”
```

実行結果は次のようになります。

COUNT-ALL	COUNT-DISTINCT
5	3

説明

- COUNT(NAME)の場合，同じ名前（Taro Tanaka）についても重複してカウントします。また，ナル値の行はカウントしないため，実行結果は5 になります。
- COUNT(DISTINCT NAME)の場合，同じ名前（Taro Tanaka）については重複してカウントしません。また，ナル値の行はカウントしないため，実行結果は3 になります。

(b) 例 2 GROUP BY 句を指定する場合

例 1 に示す社員表（EMPLIST）に対して，次に示すSELECT 文を実行します。

```
SELECT "SCODE",COUNT("NAME") AS "COUNT-ALL",
       COUNT(DISTINCT "NAME") AS "COUNT-DISTINCT"
    FROM "EMPLIST"
    GROUP BY "SCODE"
```

実行結果は次のようになります。

SCODE	COUNT-ALL	COUNT-DISTINCT	
S001	3	2	...1
S002	1	1	...2
S003	1	1	...3

説明

1. COUNT(NAME)の場合，同じ名前（Taro Tanaka）についても重複してカウントするため，実行結果は3になります。COUNT(DISTINCT NAME)の場合，同じ名前（Taro Tanaka）については重複してカウントしないため，実行結果は2 になります。
2. 値が重複する行がないため，両方とも実行結果は1 になります。
3. 値が重複する行がないため，また，ナル値の行はカウントしないため，両方とも実行結果は1 になります。

(3) 規則

1. 値式には，バイナリデータを指定できません。
2. ナル値は集計対象に含まれません。
3. 値式に指定できるデータ型と一般集合関数COUNT の実行結果のデータ型を次の表に示します。

表 7-20 値式に指定できるデータ型と一般集合関数 COUNT の実行結果のデータ型の関係

項番	値式に指定できるデータ型	一般集合関数 COUNT の実行結果のデータ型
1	INTEGER	INTEGER
2	SMALLINT	
3	DECIMAL(*m*,*n*)	
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	
6	FLOAT	
7	CHARACTER(*n*)	
8	VARCHAR(*n*)	
9	DATE	
10	TIME(*p*)	
11	TIMESTAMP(*p*)	

項番	値式に指定できるデータ型	一般集合関数 COUNT の実行結果のデータ型
12	ARRAY	

4. 次に示す場合，実行結果は0 になります。
   - 入力行数が 0 の場合
   - 集計対象の値がすべてナル値の場合
5. 集合関数の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となります。

(4) 例題

例題

販売履歴表（SALESLIST）から，2014/1/1 以降に商品を購入した人数を求めます。同じ人が複数回購入している場合は，全部で 1 人とカウントします。

```
SELECT COUNT(DISTINCT "USERID") AS "COUNT"
    FROM "SALESLIST"
        WHERE "PUR-DATE">=DATE'2014-01-01'
```

実行結果の例

COUNT
150

7.23.4 MAX

最大値を求めます。

(1) 指定形式

```
一般集合関数MAX：：＝ {MAX(〔ALL〕 値式)|MAX(DISTINCT 値式)}
```

注　どちらの形式で指定しても結果は同じになります。

(2) 指定形式の説明

MAX(〔ALL〕 *値式*)：

値式の結果の最大値を求めます。値式については，「7.21　値式」を参照してください。

ALL は省略できます。指定のありなしに関係なく結果は同じになります。

MAX(DISTINCT *値式*)：

値式の結果の最大値を求めます。値式については，「7.21　値式」を参照してください。

(3) 規則

1. ナル値は集計対象に含まれません。
2. 次に示す場合，実行結果はナル値になります。
   - 入力行数が 0 の場合
   - 集計対象の値がすべてナル値の場合
3. 値式に指定できるデータ型と一般集合関数MAX の実行結果のデータ型を次の表に示します。

表 7-21　値式に指定できるデータ型と一般集合関数 MAX の実行結果のデータ型の関係

項番	値式に指定できるデータ型	一般集合関数 MAX の実行結果のデータ型
1	INTEGER	INTEGER
2	SMALLINT	SMALLINT
3	DECIMAL(*m*,*n*)	DECIMAL(*m*,*n*)
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	DOUBLE PRECISION
6	FLOAT	
7	CHARACTER(*n*)	CHARACTER(*n*)
8	VARCHAR(*n*)	VARCHAR(*n*)
9	DATE	DATE
10	TIME(*p*)	TIME(*p*)
11	TIMESTAMP(*p*)	TIMESTAMP(*p*)

(4) 例題

例題 1

社員表（EMPLIST）から，最年長の男性社員の年齢（AGE）を求めます。

```
SELECT MAX(”AGE”) AS ”MAX-AGE”
    FROM ”EMPLIST”
       WHERE ”SEX”='M'
```

実行結果の例

MAX-AGE
63

例題 2

社員表（EMPLIST）から，最年長の社員の年齢（AGE）を，部門（SCODE）ごとに求めます。

```
SELECT ”SCODE”,MAX(”AGE”) AS ”MAX-AGE”
    FROM ”EMPLIST”
        GROUP BY ”SCODE”
```

実行結果の例

SCODE	MAX-AGE
S001	55
S002	63
S003	43

7.23.5 MIN

最小値を求めます。

(1) 指定形式

```
一般集合関数MIN：：＝ {MIN(〔ALL〕 値式)|MIN(DISTINCT 値式)}
```

注 どちらの形式で指定しても結果は同じになります。

(2) 指定形式の説明

MIN(〔ALL〕 *値式*)：

値式の結果の最小値を求めます。値式については，「7.21 値式」を参照してください。

ALL は省略できます。指定のありなしに関係なく結果は同じになります。

MIN(DISTINCT *値式*)：

値式の結果の最小値を求めます。値式については，「7.21 値式」を参照してください。

(3) 規則

1. ナル値は集計対象に含まれません。
2. 次に示す場合，実行結果はナル値になります。
   - 入力行数が 0 の場合
   - 集計対象の値がすべてナル値の場合
3. 値式に指定できるデータ型と一般集合関数MIN の実行結果のデータ型を次の表に示します。

表 7-22 値式に指定できるデータ型と一般集合関数 MIN の実行結果のデータ型の関係

項番	値式に指定できるデータ型	一般集合関数 MIN の実行結果のデータ型
1	INTEGER	INTEGER

項番	値式に指定できるデータ型	一般集合関数 MIN の実行結果のデータ型
2	SMALLINT	SMALLINT
3	DECIMAL(*m*,*n*)	DECIMAL(*m*,*n*)
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	DOUBLE PRECISION
6	FLOAT	
7	CHARACTER(*n*)	CHARACTER(*n*)
8	VARCHAR(*n*)	VARCHAR(*n*)
9	DATE	DATE
10	TIME(*p*)	TIME(*p*)
11	TIMESTAMP(*p*)	TIMESTAMP(*p*)

(4) 例題

例題 1

社員表（EMPLIST）から，最年少の女性社員の年齢（AGE）を求めます。

```
SELECT MIN(”AGE”) AS ”MIN-AGE”
    FROM ”EMPLIST”
       WHERE ”SEX”='F'
```

実行結果の例

MIN-AGE
22

例題 2

社員表（EMPLIST）から，最年少の社員の年齢（AGE）を，部門（SCODE）ごとに求めます。

```
SELECT ”SCODE”,MIN(”AGE”) AS ”MIN-AGE”
    FROM ”EMPLIST”
        GROUP BY ”SCODE”
```

実行結果の例

SCODE	MIN-AGE
S001	28
S002	22
S003	27

例題 3

社員表（EMPLIST）から，最年長の社員の年齢（AGE）と，最年少の社員の年齢差が 20 以下の部門（SCODE）の，最年長の社員の年齢と最年少の社員の年齢を求めます。

```
SELECT ”SCODE”,MAX(”AGE”) AS ”MAX-AGE”,MIN(”AGE”) AS ”MIN-AGE”
    FROM ”EMPLIST”
        GROUP BY ”SCODE”
        HAVING MAX(”AGE”)-MIN(”AGE”)<=20
```

実行結果の例

SCODE	MAX-AGE	MIN-AGE
S003	43	27

7.23.6 SUM

合計値を求めます。

(1) 指定形式

一般集合関数SUM：：＝ {SUM(〔ALL〕 *値式*)｜SUM(DISTINCT *値式*)}

(2) 指定形式の説明

SUM(〔ALL〕 *値式*)：

値式の結果の合計値を求めます。値式については，「7.21 値式」を参照してください。

ALL は省略できます。指定のありなしに関係なく結果は同じになります。

SUM(DISTINCT *値式*)：

値式の結果の合計値を求めます。値式については，「7.21 値式」を参照してください。

なお，同じ値がある場合，その値は 1 つだけカウント対象にします。例えば，値式の結果の値が2，3，2，5 の場合，実行結果は 2 ＋ 3 ＋ 5 ＝ 10 になります。

(3) 規則

1. ナル値は集計対象に含まれません。
2. 次に示す場合，実行結果はナル値になります。
   - 入力行数が 0 の場合
   - 集計対象の値がすべてナル値の場合
3. 値式に指定できるデータ型と一般集合関数SUM の実行結果のデータ型を次の表に示します。

表 7-23 値式に指定できるデータ型と一般集合関数 SUM の実行結果のデータ型

項番	値式に指定できるデータ型	一般集合関数 SUM の実行結果のデータ型
1	INTEGER	INTEGER
2	SMALLINT	

項番	値式に指定できるデータ型	一般集合関数 SUM の実行結果のデータ型
3	DECIMAL(*m*,*n*)	DECIMAL(38,*n*)
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	DOUBLE PRECISION
6	FLOAT	

4. 集合関数の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となります。

(4) 例題

例題

給与表（SALARYLIST）から，社員の給与額（SAL）の合計を，部門（SCODE）ごとに求めます。

```
SELECT "SCODE",SUM("SAL") AS "SUM-SAL"
    FROM "SALARYLIST"
        GROUP BY "SCODE"
```

実行結果の例

SCODE	SUM-SAL
S001	1500988
S002	1742557
S003	1665424

7.23.7 STDDEV_POP

母集団標準偏差を求めます。

(1) 指定形式

一般集合関数STDDEV_POP : : =STDDEV_POP(*値式*)

(2) 指定形式の説明

値式：

母集団標準偏差を求める際の入力値を値式の形式で指定します。値式については，「7.21 値式」を参照してください。

(3) 規則

1. ナル値は集計対象に含まれません。
2. 次に示す場合，実行結果はナル値になります。

- 入力行数が 0 の場合
- 集計対象の値がすべてナル値の場合

3. 一般集合関数STDDEV_POP の実行結果は，一般集合関数VAR_POP の平方根と等しくなります。
4. 値式に指定できるデータ型と一般集合関数STDDEV_POP の実行結果のデータ型を次の表に示します。

表 7-24　値式に指定できるデータ型と一般集合関数 STDDEV_POP の実行結果のデータ型

項番	値式に指定できるデータ型	一般集合関数 STDDEV_POP の実行結果のデータ型
1	INTEGER	DOUBLE PRECISION
2	SMALLINT	
3	DECIMAL(*m*,*n*)	
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	
6	FLOAT	

(4)　例題

例題

給与表（SALARYLIST）から，社員の給料（SALARY）の母集団標準偏差を求めます。

```
SELECT STDDEV_POP(”SALARY”) AS ”STDDEV_POP”
    FROM ”SALARYLIST”
```

実行結果の例

```
STDDEV_POP
2.7704873217540629E4
```

7.23.8　STDDEV_SAMP

標本標準偏差を求めます。

(1)　指定形式

```
一般集合関数STDDEV_SAMP ： ：＝STDDEV_SAMP(値式)
```

(2)　指定形式の説明

値式：

標本標準偏差を求める際の入力値を値式の形式で指定します。値式については，「7.21　値式」を参照してください。

(3) 規則

1. ナル値は集計対象に含まれません。
2. 次に示す場合，実行結果はナル値になります。
   - 入力行数が 0 または 1 の場合
   - 集計対象の値がすべてナル値の場合
3. 一般集合関数STDDEV_SAMP の実行結果は，一般集合関数VAR_SAMP の平方根と等しくなります。
4. 値式に指定できるデータ型と一般集合関数STDDEV_SAMP の実行結果のデータ型を次の表に示します。

表 7-25　値式に指定できるデータ型と一般集合関数 STDDEV_SAMP の実行結果のデータ型

項番	値式に指定できるデータ型	一般集合関数 STDDEV_SAMP の実行結果のデータ型
1	INTEGER	DOUBLE PRECISION
2	SMALLINT	
3	DECIMAL(*m*,*n*)	
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	
6	FLOAT	

(4) 例題

例題

給与表（SALARYLIST）から，社員の給料（SALARY）の標本標準偏差を求めます。

```
SELECT STDDEV_SAMP("SALARY") AS "STDDEV_SAMP"
    FROM "SALARYLIST"
```

実行結果の例

```
STDDEV_SAMP
2.9203500551208657E4
```

7.23.9 VAR_POP

母集団分散を求めます。

(1) 指定形式

```
一般集合関数VAR_POP : : =VAR_POP(値式)
```

(2) 指定形式の説明

値式：

母集団分散を求める際の入力値を値式の形式で指定します。値式については，「7.21　値式」を参照してください。

(3) 規則

1. ナル値は集計対象に含まれません。
2. 次に示す場合，実行結果はナル値になります。
   - 入力行数が 0 の場合
   - 集計対象の値がすべてナル値の場合
3. 入力行数を *N*，入力値の合計値を *S1*，入力値を 2 乗した値の合計値を *S2* とした場合，一般集合関数 VAR_POP の計算結果は次のようになります。

$$(S2-S1\times S1\div N)\div N$$

4. 値式に指定できるデータ型と一般集合関数VAR_POP の実行結果のデータ型を次の表に示します。

表 7-26　値式に指定できるデータ型と一般集合関数 VAR_POP の実行結果のデータ型

項番	値式に指定できるデータ型	一般集合関数 VAR_POP の実行結果のデータ型
1	INTEGER	DOUBLE PRECISION
2	SMALLINT	
3	DECIMAL(*m*,*n*)	
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	
6	FLOAT	

(4) 例題

例題

給与表（SALARYLIST）から，社員の給料（SALARY）の母集団分散を職級（POSITION）別に求めます。

```
SELECT ”POSITION”,VAR_POP(”SALARY”) AS ”VAR_POP”
    FROM ”SALARYLIST”
    GROUP BY ”POSITION”
    ORDER BY ”POSITION”
```

実行結果の例

POSITION	VAR_POP
Chief	1.3250000000000000E7
Director	5.6250000000000000E7
Manager	4.2187500000000000E7

7.23.10　VAR_SAMP

標本分散を求めます。

(1)　指定形式

一般集合関数VAR_SAMP：：＝VAR_SAMP(*値式*)

(2)　指定形式の説明

値式：

標本分散を求める際の入力値を値式の形式で指定します。値式については，「7.21　値式」を参照してください。

(3)　規則

1. ナル値は集計対象に含まれません。
2. 次に示す場合，実行結果はナル値になります。
   - 入力行数が 0 または 1 の場合
   - 集計対象の値がすべてナル値の場合
3. 入力行数を*N*，入力値の合計値を*S1*，入力値を 2 乗した値の合計値を*S2* とした場合，一般集合関数 VAR_SAMP の計算結果は次のようになります。

   (*S2*-*S1*×*S1*÷*N*) ÷ (*N*-1)

4. 値式に指定できるデータ型と一般集合関数VAR_SAMP の実行結果のデータ型を次の表に示します。

表 7-27　値式に指定できるデータ型と一般集合関数 VAR_SAMP の実行結果のデータ型

項番	値式に指定できるデータ型	一般集合関数 VAR_SAMP の実行結果のデータ型
1	INTEGER	DOUBLE PRECISION
2	SMALLINT	
3	DECIMAL(*m*,*n*)	
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	

項番	値式に指定できるデータ型	一般集合関数 VAR_SAMP の実行結果のデータ型
6	FLOAT	

(4) 例題

例題

給与表（SALARYLIST）から，社員の給料（SALARY）の標本分散を職級（POSITION）別に求めます。

```
SELECT "POSITION",VAR_SAMP("SALARY") AS "VAR_SAMP"
    FROM "SALARYLIST"
    GROUP BY "POSITION"
    ORDER BY "POSITION"
```

実行結果の例

POSITION	VAR_SAMP
Chief	1.7666666666666668E7
Director	1.1250000000000000E8
Manager	5.6250000000000000E7

7.23.11 MEDIAN

順序付けされた一連の値の中央値を求めます。線形補間された値となることがあります。

メモ

MEDIAN は，PERCENTILE_CONT の引数（パーセンタイルの指定）に中央値（0.5）を指定したときと同じ結果が得られる逆分布関数です。

MEDIAN の被集約引数をARG1 とした場合，MEDIAN は次のPERCENTILE_CONT と等価になります。

```
PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY ARG1 ASC)
```

(1) 指定形式

```
逆分布関数MEDIAN：：＝MEDIAN(値式)
```

(2) 指定形式の説明

値式：

中央値を求める際の入力値を値式の形式で指定します。値式については，「7.21　値式」を参照してください。

(3) 規則

1. ナル値は集計対象に含まれません。
2. 入力行数が 0 の場合，実行結果はナル値になります。
3. 値式に指定できるデータ型と逆分布関数MEDIAN の実行結果のデータ型を次の表に示します。

表 7-28　値式に指定できるデータ型と逆分布関数 MEDIAN の実行結果のデータ型

項番	値式に指定できるデータ型	逆分布関数 MEDIAN の実行結果のデータ型
1	INTEGER	DOUBLE PRECISION
2	SMALLINT	
3	DECIMAL(*m*,*n*)	
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	
6	FLOAT	

4. MEDIAN は，順序付けされた一連の値に対して線形補間することによって計算されます。入力行数を*N*とした場合，最初に行番号*RN* ＝ {1 ＋ 0.5×（*N*−1)} が計算されます。続いて，行番号*CRN* ＝ CEIL(*RN*)および*FRN* ＝ FLOOR(*RN*)の行の値間の線形補間によって，MEDIAN の実行結果が算出されます。計算結果は次のようになります。
   - *CRN* ＝*FRN* ＝*RN* の場合：行*RN* の値
   - それ以外の場合：(*CRN*−*RN*）×（行*FRN* の値）＋（*RN*−*FRN*）×（行*CRN* の値）

(4) 例題

例題 1

給与表（SALARYLIST）から，社員の給料（SALARY）の中央値（50 パーセンタイル）を職級（POSITION）別に求めます。

```
SELECT "POSITION",MEDIAN("SALARY") AS "MEDIAN"
    FROM "SALARYLIST"
    GROUP BY "POSITION"
    ORDER BY "POSITION"
```

実行結果の例

POSITION	MEDIAN
Chief	6.9000000000000000E4
Director	1.4250000000000000E5
Manager	1.0500000000000000E5

例題 2

給与表（SALARYLIST）から，社員の給料（SALARY）の中央値（50 パーセンタイル）を求めます。

```
SELECT MEDIAN(”SALARY”) AS ”MEDIAN”
    FROM ”SALARYLIST”
```

```
        MEDIAN
9.7500000000000000E4
```

7.23.12　PERCENTILE_CONT

順序付けされた一連の値のパーセンタイル（百分位数）を求めます。線形補間された値となることがあります。

(1)　指定形式

```
逆分布関数PERCENTILE_CONT：：＝PERCENTILE_CONT(値指定) WITHINグループ指定

  WITHINグループ指定：：＝WITHIN GROUP(ORDER BY ソート指定リスト)
```

(2)　指定形式の説明

値指定：

求めるパーセンタイルの値を値指定の形式で指定します。値指定については，「7.22　値指定」を参照してください。

指定規則を次に示します。

- 0～1 の値（INTEGER 型，SMALLINT 型，DECIMAL 型，またはNUMERIC 型のデータ）を指定してください。
- ナル値を指定した場合，実行結果はナル値になります。
- ？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はDECIMAL(3,2)になります。

WITHIN グループ指定：

```
WITHINグループ指定：：＝WITHIN GROUP(ORDER BY ソート指定リスト)
```

WITHIN グループ指定には，パーセンタイルを求めるデータとデータの並び順を指定します。ソート指定リストのソートキーにパーセンタイルを求めるデータを指定し，順序付け指定にデータの並び順（昇順，降順）を指定します。ソート指定リストについては，「7.25　ソート指定リスト」を参照してください。

指定規則を次に示します。

- WITHIN グループ指定のソート指定リストには，ナル値ソート順指定は指定できません。
- WITHIN グループ指定のソート指定リストには，ソート指定を 2 つ以上指定できません。

(3)　規則

1. ナル値は集計対象に含まれません。

2. 入力行数が 0 の場合，実行結果はナル値になります。

3. ソート指定リストのソートキーに指定できるデータ型と逆分布関数PERCENTILE_CONT の実行結果のデータ型を次の表に示します。

表 7-29　ソート指定リストのソートキーに指定できるデータ型と逆分布関数 PERCENTILE_CONT の実行結果のデータ型

項番	ソート指定リストのソートキーに指定できるデータ型	逆分布関数 PERCENTILE_CONT の実行結果のデータ型
1	INTEGER	DOUBLE PRECISION
2	SMALLINT	
3	DECIMAL(*m*,*n*)	
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	
6	FLOAT	

4. PERCENTILE_CONT は，順序付けされた一連の値に対して線形補間することによって計算されます。入力行数を*N*，指定した引数の値を*P* とした場合，最初に行番号$RN = \{1 + P \times (N-1)\}$が計算されます。続いて，行番号$CRN = \mathrm{CEIL}(RN)$および$FRN = \mathrm{FLOOR}(RN)$の行の値間の線形補間によって，PERCENTILE_CONT の実行結果が算出されます。計算結果は次のようになります。

- $CRN = FRN = RN$ の場合：行*RN* の値
- それ以外の場合：(*CRN*−*RN*）×（行*FRN* の値）＋（*RN*−*FRN*）×（行*CRN* の値）

(4)　例題

例題 1

給与表（SALARYLIST）から，社員の給料（SALARY）の中央値（50 パーセンタイル）を職級（POSITION）別に求めます。

```
SELECT "POSITION",
        PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY "SALARY") AS "PERCENTILE_CONT"
    FROM "SALARYLIST"
    GROUP BY "POSITION"
    ORDER BY "POSITION"
```

POSITION	PERCENTILE_CONT
Chief	6.9000000000000000E4
Director	1.4250000000000000E5
Manager	1.0500000000000000E5

例題 2

給与表（SALARYLIST）から，社員の給料（SALARY）の中央値（50 パーセンタイル）を求めます。

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY "SALARY") AS "PERCENTILE_CONT"
    FROM "SALARYLIST"
```

```
PERCENTILE_CONT
9.7500000000000000E4
```

7.23.13 PERCENTILE_DISC

順序付けされた一連の値のパーセンタイル（百分位数）を求めます。一連の値の中から結果を返します。

(1) 指定形式

```
逆分布関数PERCENTILE_DISC：：＝PERCENTILE_DISC(値指定) WITHINグループ指定

  WITHINグループ指定：：＝WITHIN GROUP(ORDER BY ソート指定リスト)
```

(2) 指定形式の説明

値指定：

求めるパーセンタイルの値を値指定の形式で指定します。値指定については，「7.22 値指定」を参照してください。

指定規則を次に示します。

- 0～1 の値（INTEGER 型，SMALLINT 型，DECIMAL 型，またはNUMERIC 型のデータ）を指定してください。
- ナル値を指定した場合，実行結果はナル値になります。
- ？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はDECIMAL(3,2)になります。

WITHIN グループ指定：

```
WITHINグループ指定：：＝WITHIN GROUP(ORDER BY ソート指定リスト)
```

WITHIN グループ指定には，パーセンタイルを求めるデータとデータの並び順を指定します。*ソート指定リスト*のソートキーにパーセンタイルを求めるデータを指定し，順序付け指定にデータの並び順（昇順，降順）を指定します。ソート指定リストについては，「7.25 ソート指定リスト」を参照してください。

指定規則を次に示します。

- WITHIN グループ指定のソート指定リストには，ナル値ソート順指定は指定できません。
- WITHIN グループ指定のソート指定リストには，ソート指定を 2 つ以上指定できません。

(3) 規則

1. ナル値は集計対象に含まれません。

2. 入力行数が 0 の場合，実行結果はナル値になります。

3. ソート指定リストのソートキーに指定できるデータ型と逆分布関数PERCENTILE_DISC の実行結果のデータ型を次の表に示します。

表 7-30　ソート指定リストのソートキーに指定できるデータ型と逆分布関数 PERCENTILE_DISC の実行結果のデータ型

項番	ソート指定リストのソートキーに指定できるデータ型	逆分布関数 PERCENTILE_DISC の実行結果のデータ型
1	INTEGER	INTEGER
2	SMALLINT	SMALLINT
3	DECIMAL(*m*,*n*)	DECIMAL(*m*,*n*)
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	DOUBLE PRECISION
6	FLOAT	
7	CHAR(*n*)	CHAR(*n*)
8	VARCHAR(*n*)	VARCHAR(*n*)
9	DATE	DATE
10	TIME(*p*)	TIME(*p*)
11	TIMESTAMP(*p*)	TIMESTAMP(*p*)

4. PERCENTILE_DISC は，順序付けされた一連の値の中から結果を返します。引数に値P を指定した場合，ソート指定リストで指定された値式の値を並べ替えて，その値の中から同じソート指定リストに従う CUME_DIST の値がP 以上となる最小の値を返します。

(4)　例題

例題

給与表（SALARYLIST）から，社員の給料（SALARY）の中央値（50 パーセンタイル）を職級（POSITION）別に求めます。

PERCENTILE_CONT とPERCENTILE_DISC を使用して，それぞれの中央値を求めます。

```
SELECT "POSITION",
        PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY "SALARY" ASC) AS "PERCENTILE_CONT",
        PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY "SALARY" ASC) AS "PERCENTILE_DISC"
    FROM "SALARYLIST"
    GROUP BY "POSITION"
    ORDER BY "POSITION"
```

POSITION	PERCENTILE_CONT	PERCENTILE_DISC
Chief	6.9000000000000000E4	68000
Director	1.4250000000000000E5	135000
Manager	1.0500000000000000E5	100000

上記のように，PERCENTILE_CONT とPERCENTILE_DISC の結果が異なることがあります。これは，PERCENTILE_CONT は，線形補間された結果を返していますが，PERCENTILE_DISC は，集計対象の一連の値だけを使用して結果を返しているためです。

7.23.14　LISTAGG

順序付けされた一連の値を連結し，値と値の間に区切り文字列を挿入した文字列を求めます。

> **メモ**
>
> 「7.23.14　LISTAGG」の説明では，LISTAGG 集合関数をLISTAGG と表記します。

(1)　指定形式

```
LISTAGG集合関数 : :=LISTAGG ( 〔 {ALL | DISTINCT} 〕 値式
                              〔,LISTAGG区切り文字列〕
                              〔,LISTAGG結果の最大長〕
                              〔ON OVERFLOW 結果溢れ動作〕
                            ) WITHINグループ指定

結果溢れ動作 : := {ERROR | TRUNCATE 〔切り捨て末尾文字列〕 WITHOUT COUNT}

WITHINグループ指定 : :=WITHIN GROUP(ORDER BY ソート指定リスト)
```

(2)　指定形式の説明

● 〔{ALL | DISTINCT}〕 *値式*

{ALL | DISTINCT}：

値式から導出された値に重複した値がある場合，重複した値を排除するかどうかを指定します。

ALL：重複した値がある場合でも，重複した値を排除しません。

DISTINCT：重複した値がある場合，重複した値を排除して 1 つの値にします。

ALL およびDISTINCT の指定を省略した場合，ALL が仮定されます。

値式：

LISTAGG の集計対象を求める値式を指定します。値式については，「7.21　値式」を参照してください。

値式には，数データ（INTEGER 型，SMALLINT 型，DECIMAL 型，NUMERIC 型，DOUBLE PRECISION 型，FLOAT 型），または文字データ（CHAR 型，VARCHAR 型）を指定してください。数データおよび文字データについては，「6.2.1　データ型の種類」を参照してください。

なお，値式から導出された値を，**LISTAGG の集計値**といいます。

> **メモ**
>
> 指定した値式が数データの場合，値式から導出された数データの値は文字データの値に変換されます。その際のデータ変換規則は，スカラ関数CONVERT によって数データを文字データに変換する際の規則（数値書式の指定なしの場合）に従います。スカラ関数CONVERT による数データを文字データに変換する際の規則については，「8.13.5 CONVERT」の「(5)　規則」の「(c)　文字データに変換する場合の規則」を参照してください。

●*LISTAGG 区切り文字列*

LISTAGG の集計値と集計値の間に挿入する区切り文字列を，文字列定数の形式で指定します。文字列定数については，「6.3　定数」を参照してください。

*LISTAGG 区切り文字列*の指定と，LISTAGG の結果の例を次に示します。例で使用されているBob, Mike, Nancy, Stephanie, Tom は，LISTAGG の集計値です。

（例）

- *LISTAGG 区切り文字列*に'|'を指定した場合

  LISTAGG の結果は次のようになります。

  ```
  Bob|Mike|Nancy|Stephanie|Tom
  ```

- *LISTAGG 区切り文字列*に','を指定した場合

  LISTAGG の結果は次のようになります。

  ```
  Bob,Mike,Nancy,Stephanie,Tom
  ```

- *LISTAGG 区切り文字列*に':::'を指定した場合

  LISTAGG の結果は次のようになります。

  ```
  Bob:::Mike:::Nancy:::Stephanie:::Tom
  ```

指定規則を次に示します。

- *LISTAGG 区切り文字列*の指定を省略した場合，LISTAGG 区切り文字列には実長0 バイトの文字データが仮定されます。*LISTAGG 区切り文字列*の指定を省略した場合の，LISTAGG の結果の例を次に示します。

  （例）

  ```
  BobMikeNancyStephanieTom
  ```

- LISTAGG 区切り文字列の長さは，*LISTAGG 結果の最大長*の指定値（バイト）以下にしてください。

●*LISTAGG 結果の最大長*

LISTAGG の結果の最大長（VARCHAR 型の定義長）をバイト数で指定します。

（例）

```
SELECT LISTAGG("C1",'|',20) WITHIN GROUP (ORDER BY "C1") FROM "T1"
```

下線部分が，*LISTAGG 結果の最大長*の指定です。この場合，LISTAGG 結果の最大長が20 バイト（VARCHAR(20)）になります。

指定規則を次に示します。

- *LISTAGG 結果の最大長*には，3～32,000 の符号なし整数定数を指定してください。
- *LISTAGG 結果の最大長*の指定を省略した場合，LISTAGG 結果の最大長には1,024 バイト（VARCHAR(1024)）が仮定されます。

●ON OVERFLOW *結果溢れ動作*

結果溢れ動作：：＝ {ERROR | TRUNCATE〔*切り捨て末尾文字列*〕 WITHOUT COUNT}

結果溢れ動作には，**連結データ**のデータ長が，LISTAGG 結果の最大長を超えた場合の動作を指定します。

メモ

連結データとは，LISTAGG によって集計されて連結された文字データ（LISTAGG 区切り文字列を含む）のことです。

ERROR：

連結データのデータ長がLISTAGG 結果の最大長を超えた場合，SQL 文をエラーにします。SQL 文がエラーになる例を次に示します。

（例）

- 連結データ：'Bob|Mike|Nancy|Stephanie|Tom'　←28 バイト
- *LISTAGG 結果の最大長*の指定：15 バイト

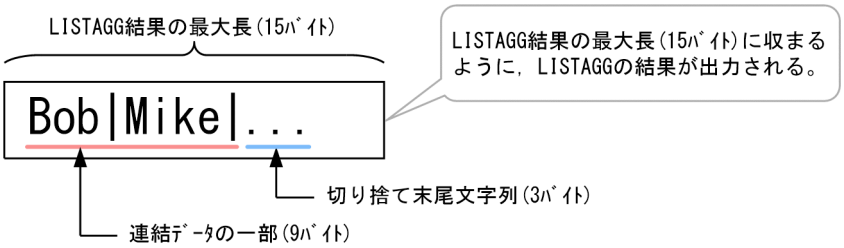
上記の場合，連結データのデータ長（28 バイト）が，LISTAGG 結果の最大長（15 バイト）を超えるため，SQL 文がエラーになります。

TRUNCATE〔*切り捨て末尾文字列*〕 WITHOUT COUNT：

連結データのデータ長がLISTAGG 結果の最大長を超えた場合でも，SQL 文をエラーにしません。連結データのデータ長が，LISTAGG 結果の最大長を超えた場合，LISTAGG の結果には，連結データの一部と切り捨て末尾文字列が出力されます。例を次に示します。

（例）

- 連結データ：'Bob|Mike|Nancy|Stephanie|Tom'　←28 バイト
- *LISTAGG 結果の最大長*の指定：15 バイト
- LISTAGG の結果：Bob|Mike|...　←12 バイト



［説明］

LISTAGG 結果の最大長（15 バイト）に収まるようにLISTAGG の結果が出力されます。

なお，切り捨て末尾文字列は，'...'から任意の文字列に変更できます。変更する場合は，*切り捨て末尾文字列*を指定してください。

切り捨て末尾文字列：

連結データのデータ長がLISTAGG 結果の最大長を超えた場合に，LISTAGG の結果の末尾に出力する文字列を指定します。*切り捨て末尾文字列*は文字列定数の形式で指定します。文字列定数については，「6.3　定数」を参照してください。

*切り捨て末尾文字列*の指定を省略した場合，'...'（ピリオド 3 つの実長 3 バイトの文字データ）が切り捨て末尾文字列に仮定されます。

切り捨て末尾文字列の長さは，*LISTAGG 結果の最大長*の指定値（バイト）以下にしてください。

> **メモ**
>
> このオプションの指定を，**LISTAGG 結果溢れ動作指定**といいます。

●*WITHIN グループ指定*

WITHINグループ指定：：＝WITHIN GROUP(ORDER BY *ソート指定リスト*)

WITHIN グループ指定には，LISTAGG の集計値の連結順序（昇順，降順）を指定します。ソート指定リストについては，「7.25　ソート指定リスト」を参照してください。

指定規則を次に示します。

- WITHIN グループ指定のソート指定リストには，ナル値ソート順指定は指定できません。
- WITHIN グループ指定のソート指定リストには，ソート指定を 2 つ以上指定できません。

(3)　規則

1. 問合せ指定中にLISTAGG を 64 個まで指定できます。
2. LISTAGG の引数に指定した値式のデータ型によって，LISTAGG の結果のデータ型が次の表のとおりに決まります。

表 7-31　LISTAGG の引数に指定した値式のデータ型とLISTAGG の結果のデータ型の関係

値式のデータ型		LISTAGG の結果のデータ型
数データ	INTEGER	• *LISTAGG 結果の最大長*（指定値を *t* とする）を指定した場合 VARCHAR(*t*) • *LISTAGG 結果の最大長*の指定を省略した場合 VARCHAR(1024)
	SMALLINT	
	DECIMAL	
	NUMERIC	
	DOUBLE PRECISION	
	FLOAT	

値式のデータ型		LISTAGG の結果のデータ型
文字データ	CHAR	
	VARCHAR	

3. LISTAGG の引数の値式から導出された値（LISTAGG の集計値）に対して次の処理を順に実行した結果が，LISTAGG の入力行になります。
   - 集計値にナル値がある場合，ナル値を排除する。
   - DISTINCT が指定されている場合，重複している集計値を排除して 1 つの値にする。
   - *WITHIN グループ指定*の*ソート指定*に従って，集計値のソート処理を行う。
4. LISTAGG の入力行数が 0 の場合，実行結果はナル値になります。
5. LISTAGG の実行結果を次に示します。

   例の説明では，連結データが'Bob|Mike|Nancy|Stephanie|Tom'（28 バイト）であるとします。

   ■連結データのデータ長≦LISTAGG 結果の最大長の場合

   LISTAGG の結果には，連結データがすべて出力されます。

   （例）

```
Bob|Mike|Nancy|Stephanie|Tom
```

   ■連結データのデータ長＞ LISTAGG 結果の最大長の場合

   - *結果溢れ動作*にERROR（省略値）を指定した場合

     実行した SQL 文がエラーになります。
   - *結果溢れ動作*にTRUNCATE を指定した場合

     ＜実行結果 1 ＞

     LISTAGG の結果には，連結データの一部と切り捨て末尾文字列が出力されます。

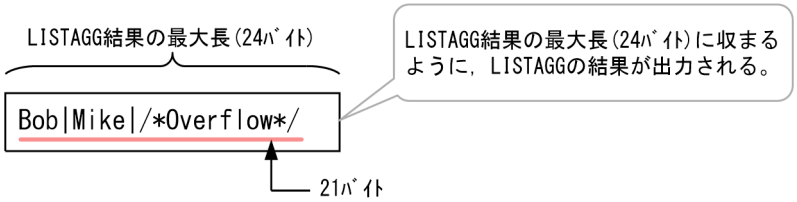
     （例）

     SQL 文の例

```
SELECT LISTAGG("C1",'|',24 ON OVERFLOW TRUNCATE '/*Overflow*/' WITHOUT COUNT)
        WITHIN GROUP (ORDER BY "C1") FROM "T1"
```

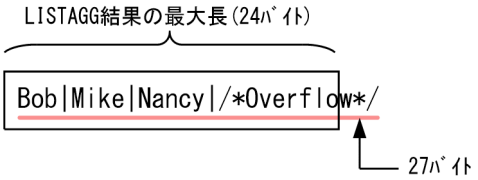
     上記の SQL 文では，*LISTAGG 結果の最大長*に24 バイトを指定し，*切り捨て末尾文字列*に'/*Overflow*/'を指定しています。

     LISTAGG の結果



     [説明]

- 連結データの一部と切り捨て末尾文字列が，LISTAGG 結果の最大長（24 バイト）に収まるように出力されます。
- 出力される連結データの一部の最後の文字列は，LISTAGG 区切り文字列（'|'）になります。
- 次のように出力する集計値が 1 つ多い場合，データ長が 27 バイトになるため，LISTAGG 結果の最大長（24 バイト）を超えてしまいます。そのため，この形式では出力されません。



＜実行結果 2＞

*LISTAGG 結果の最大長*の指定値によっては，LISTAGG の結果に集計値が 1 つも出力されないことがあります。

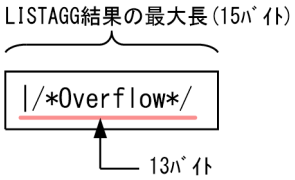
（例）

SQL 文の例

```
SELECT LISTAGG("C1",'|',15 ON OVERFLOW TRUNCATE '/*Overflow*/' WITHOUT COUNT)
         WITHIN GROUP (ORDER BY "C1") FROM "T1"
```

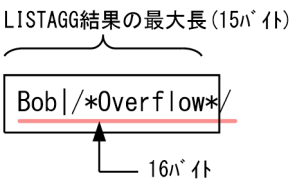
上記の SQL 文では，*LISTAGG 結果の最大長*に15 バイトを指定し，*切り捨て末尾文字列*に'/*Overflow*/'を指定しています。

LISTAGG の結果



[説明]

- *LISTAGG 結果の最大長*の指定値が小さ過ぎる場合，上記のようにLISTAGG 区切り文字列（'|'）と切り捨て末尾文字列（'/*Overflow*/'）しか出力されません。
- 次のように集計値を 1 つ出力した場合，データ長が 16 バイトになるため，LISTAGG 結果の最大長（15 バイト）を超えてしまいます。そのため，この形式では出力されません。



＜実行結果 3＞

*LISTAGG 結果の最大長＜LISTAGG 区切り文字列のデータ長＋切り捨て末尾文字列のデータ長*の場合，LISTAGG の結果には'...'だけが出力されます。

（例）

SQL 文の例

```
SELECT LISTAGG(”C1”,’|’,12 ON OVERFLOW TRUNCATE ’/*Overflow*/’ WITHOUT COUNT)
         WITHIN GROUP(ORDER BY ”C1”) FROM ”T1”
```

上記の SQL 文では，*LISTAGG 結果の最大長*に12 バイトを指定し，*切り捨て末尾文字列*に’/*Overflow*/’を指定しています。

LISTAGG の結果

```
...
```

上記の例の場合，LISTAGG 区切り文字列（’|’）が 1 バイトで，切り捨て末尾文字列（’/*Overflow*/’）が 12 バイトです。よって，*LISTAGG 区切り文字列のデータ長＋切り捨て末尾文字列のデータ長*は 13 バイトになります。LISTAGG 結果の最大長（12 バイト）を超えるため，LISTAGG の結果には’...’だけが出力されます。

(4) 例題

次に示す売上表（SALES）を検索対象とする SQL 文の実行例を説明します。

顧客ID(CID)	商品ID(PID)	単価(PRICE)
C0001	P0013	1000
C0001	P0014	1500
C0001	P0013	1000
C0002	P0008	2000
C0002	P0010	3000
C0002	P0013	1000
C0002	P0016	4000
C0003	P0013	1000
C0003	P0010	3000
C0004	P0013	1000

例題 1

顧客ID（CID）ごとに，購入した商品の商品ID（PID）を求めます。LISTAGG の結果（PID_LIST）には，次の条件で商品ID を出力します。

- 単価（PRICE）順に商品ID（PID）を連結する。
- 各商品ID を’|’で区切る。

```
SELECT ”CID”,
       LISTAGG(”PID”,’|’) WITHIN GROUP (ORDER BY ”PRICE” ASC) AS ”PID_LIST”
  FROM ”SALES”
    GROUP BY ”CID”
```

上記のSELECT 文では，次の指定（上記の下線部分）をしています。

- *LISTAGG 区切り文字列*に’|’を指定しています。
- *WITHIN グループ指定のソート指定リスト*に，商品ID を単価（PRICE）順に連結する指定をしています。

実行結果の例

CID	PID_LIST
C0001	P0013|P0013|P0014
C0002	P0013|P0008|P0010|P0016
C0003	P0013|P0010
C0004	P0013

顧客IDがC0001のお客様が
購入された商品の商品IDが,
商品の単価順に連結される。
各商品IDは'|'で区切られる。

例題２（LISTAGG の集計値から重複した値を排除する場合）

顧客ID（CID）ごとに，購入した商品の商品ID（PID）を求めます。LISTAGG の結果（PID_LIST）には，次の条件で商品ID を出力します。

- 単価（PRICE）順に商品ID（PID）を連結する。
- 各商品ID を'|'で区切る。
- 商品ID の重複を排除する。※

注※　例題１と差異がある条件です。

```
SELECT ”CID”,
       LISTAGG(DISTINCT ”PID”,'|') WITHIN GROUP (ORDER BY ”PRICE” ASC) AS ”PID_LIST”
  FROM ”SALES”
    GROUP BY ”CID”
```

上記のSELECT 文では，LISTAGG の集計値から重複した値を排除するDISTINCT が指定（上記の下線部分）されています。

実行結果の例

CID	PID_LIST
C0001	P0013|P0014
C0002	P0013|P0008|P0010|P0016
C0003	P0013|P0010
C0004	P0013

例題1の実行結果では,
P0013が重複しているが,
DISTINCTを指定しているため,
P0013の重複が排除される。

例題３（切り捨て末尾文字列に'...'を出力する場合）

顧客ID（CID）ごとに，購入した商品の商品ID（PID）を求めます。LISTAGG の結果（PID_LIST）には，次の条件で商品ID を出力します。

- 単価（PRICE）順に商品ID（PID）を連結する。
- 各商品ID を'|'で区切る。
- LISTAGG 結果の最大長を 20 バイト（VARCHAR(20)）とする。※
- 連結データのデータ長が，LISTAGG 結果の最大長を超えた場合，切り捨て末尾文字列（'...'）を出力する。※

注※　例題１と差異がある条件です。

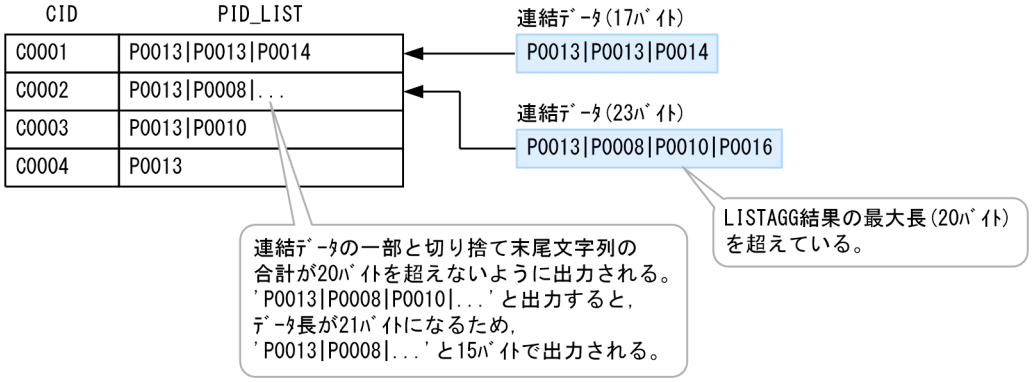
```
SELECT ”CID”,
       LISTAGG(”PID”,'|',20 ON OVERFLOW TRUNCATE WITHOUT COUNT)
         WITHIN GROUP (ORDER BY ”PRICE” ASC) AS ”PID_LIST”
```

```
  FROM "SALES"
    GROUP BY "CID"
```

上記のSELECT 文では，次の指定（上記の下線部分）をしています。

- *LISTAGG 結果の最大長*に20 バイトを指定しています。
- *結果溢れ動作*には，連結データのデータ長が，LISTAGG 結果の最大長を超えた場合，切り捨て末尾文字列に'...'（省略値）を出力する指定をしています。

実行結果の例



[説明]

- CID（顧客ID）が'C0001'の行のPID_LIST（LISTAGG の結果）には，連結データのデータ長（17 バイト）が，LISTAGG 結果の最大長（20 バイト）以下のため，連結データがすべて出力されます。
- CID（顧客ID）が'C0002'の行のPID_LIST（LISTAGG の結果）には，連結データのデータ長（23 バイト）が，LISTAGG 結果の最大長（20 バイト）を超えるため，連結データの一部と，末尾に切り捨て末尾文字列（'...'）が出力されます。

例題４（切り捨て末尾文字列に任意の文字列を出力する場合）

顧客ID（CID）ごとに，購入した商品の商品ID（PID）を求めます。LISTAGG の結果（PID_LIST）には，次の条件で商品ID を出力します。

- 単価（PRICE）順に商品ID（PID）を連結する。
- 各商品ID を'|'で区切る。
- LISTAGG 結果の最大長を 20 バイト（VARCHAR(20)）とする。
- 連結データのデータ長が，LISTAGG 結果の最大長を超えた場合，切り捨て末尾文字列（'/*Overflow*/'）を出力する。※

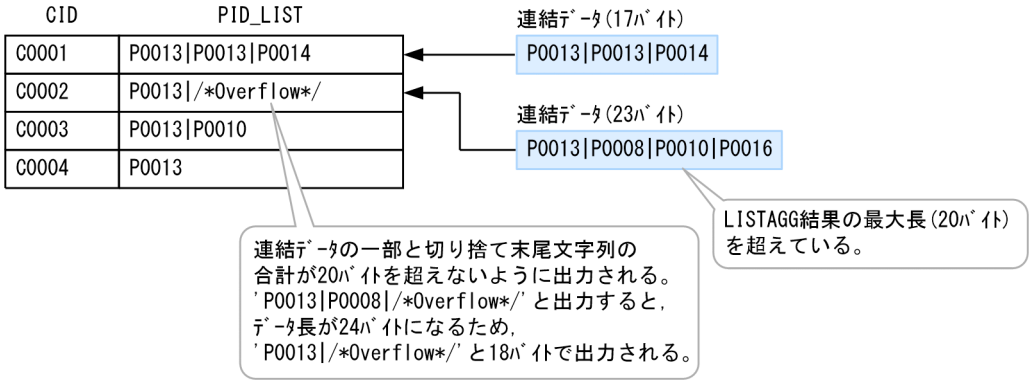
注※　例題３と差異がある条件です。

```
SELECT "CID",
       LISTAGG("PID",'|',20 ON OVERFLOW TRUNCATE '/*Overflow*/' WITHOUT COUNT)
         WITHIN GROUP (ORDER BY "PRICE" ASC) AS "PID_LIST"
  FROM "SALES"
    GROUP BY "CID"
```

上記のSELECT 文では，次の指定（上記の下線部分）をしています。

- *LISTAGG 結果の最大長*に20 バイトを指定しています。
- *結果溢れ動作*には，連結データのデータ長が，LISTAGG 結果の最大長を超えた場合，切り捨て末尾文字列に'/*Overflow*/'を出力する指定をしています。

実行結果の例



[説明]

- CID（顧客ID）が'C0001'の行のPID_LIST（LISTAGG の結果）には，連結データのデータ長（17 バイト）が，LISTAGG 結果の最大長（20 バイト）以下のため，連結データがすべて出力されます。
- CID（顧客ID）が'C0002'の行のPID_LIST（LISTAGG の結果）には，連結データのデータ長（23 バイト）が，LISTAGG 結果の最大長（20 バイト）を超えるため，連結データの一部と，末尾に切り捨て末尾文字列（'/*Overflow*/'）が出力されます。

例題 5（連結データのデータ長が LISTAGG 結果の最大長を超えたときは SQL 文をエラーにする場合）

顧客ID（CID）ごとに，購入した商品の商品ID（PID）を求めます。LISTAGG の結果（PID_LIST）には，次の条件で商品ID を出力します。

- 単価（PRICE）順に商品ID（PID）を連結する。
- 各商品ID を'|'で区切る。
- LISTAGG 結果の最大長を 20 バイト（VARCHAR(20)）とする。
- 連結データのデータ長が，LISTAGG 結果の最大長を超えた場合，SQL 文をエラーにする。※

注※　例題 3 および例題 4 と差異がある条件です。

```
SELECT "CID",
       LISTAGG("PID",'|',20 ON OVERFLOW ERROR)
         WITHIN GROUP (ORDER BY "PRICE" ASC) AS "PID_LIST"
  FROM "SALES"
    GROUP BY "CID"
```

上記のSELECT 文では，次の指定（上記の下線部分）をしています。

- *LISTAGG 結果の最大長*に20 バイトを指定しています。
- *結果溢れ動作*には，連結データのデータ長が，LISTAGG 結果の最大長を超えた場合，SQL 文をエラーとする指定をしています。

実行結果の例

連結データが'P0013|P0008|P0010|P0016'（23 バイト）となる行があり，連結データのデータ長が LISTAGG 結果の最大長を超えるため，実行した SQL 文はエラーになります。

(5) 留意事項

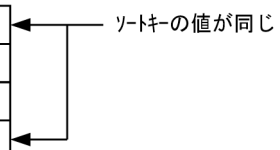
1. *WITHIN グループ指定*に指定したソートキーに同じ値がある場合，同じ SQL 文を実行してもLISTAGG の集計値の連結順序が異なることがあります。例を次に示します。

（例）

検索対象の表

表T1

C1列	C2列
Mike	2
Nancy	4
Tom	1
Stephanie	2
Flora	3



実行する SQL 文

```
SELECT LISTAGG("C1",'|') WITHIN GROUP(ORDER BY "C2") AS "Name" FROM "T1"
```

実行結果の例 1

```
Tom|Mike|Stephanie|Flora|Nancy
```

実行結果の例 2

```
Tom|Stephanie|Mike|Flora|Nancy
```

Mike とStephanie は，ソートキーの値が同じであるため，同じ SQL 文を実行しても上記の例のように実行結果（LISTAGG の集計値の連結順序）が異なることがあります。

2. *LISTAGG 結果の最大長*の指定を省略した場合，LISTAGG 結果の最大長には1,024 バイトが仮定されます。連結データのデータ長が 1,024 バイトよりも著しく小さいと，LISTAGG の処理の際にリソースを必要以上に使用してしまい，その結果，作業表の行長の最大値を超えてしまうなどのエラーが発生するおそれがあります。このようなエラーが発生した場合，*LISTAGG 結果の最大長*には，次に示す計算式から求めた値を指定することを推奨します。

（*data_len*＋*LISTAGG区切り文字列の長さ*）×*LISTAGGの集計値の数*

data_len：LISTAGG の引数に指定した値式のデータ型に従って次の値を代入してください。

表 7-32　data_len に代入する値

値式のデータ型	*data_len* に代入する値	例
CHAR(*n*)	*n*	なし
VARCHAR	LISTAGG の集計値の実長の最大値を代入します。	LISTAGG の集計値が，'AB'，'ABC'，'ABCD'の場合，*data_len* には4 を代入します。

値式のデータ型	*data_len* に代入する値	例
INTEGER またはSMALLINT	LISTAGG の集計値の数データを文字データに変換した結果※の文字列の長さの最大値を代入します。	INTEGER 型またはSMALLINT 型のデータを文字データに変換した結果が，'123'，'1234'，'-1234' の場合，*data_len* には5 を代入します。
DECIMAL またはNUMERIC		DECIMAL 型またはNUMERIC 型のデータを文字データに変換した結果が，'1.20'，'12.34'，'-12.34' の場合，*data_len* には6 を代入します。
DOUBLE PRECISION またはFLOAT		DOUBLE PRECISION 型またはFLOAT 型のデータを文字データに変換した結果が，'1.23E45'，'-1.23E45'，'1.23E-45' の場合，*data_len* には8 を代入します。

注※

数データを文字データに変換する際の規則は，スカラ関数CONVERT によって数データを文字データに変換する際の規則（数値書式の指定なしの場合）に従います。スカラ関数CONVERT による数データを文字データに変換する際の規則については，「8.13.5　CONVERT」の「(5)　規則」の「(c) 文字データに変換する場合の規則」を参照してください。

7.23.15　ARRAY_AGG

対象データに指定した値式の結果（値式によって集計された値）を，先頭から順に配列要素とする配列データを返します。

> **メモ**
>
> 「7.23.15　ARRAY_AGG」の説明では，ARRAY_AGG 集合関数をARRAY_AGG と表記します。

(1)　指定形式

ARRAY_AGG集合関数 ：：＝ARRAY_AGG(*対象データ*　〔ORDER BY *ソート指定リスト*〕)

対象データ ：：＝*値式*

(2)　指定形式の説明

対象データ：

対象データには，集計対象となる値式を指定します。

指定規則を次に示します。

- 値式には，配列データを指定できません。
- 値式には，？パラメタを単独で指定できません。

ORDER BY *ソート指定リスト*：

ソート指定リストには，集計対象の値のソート順を指定します。この指定によって，配列要素の値の順序を昇順または降順に並べることができます。

この指定を省略した場合，同じ SQL 文を実行しても，ARRAY_AGG の実行結果の値の順序が変わることがあります（配列要素の値の順序が変わることがあります）。

ソート指定リストについては，「7.25　ソート指定リスト」を参照してください。

指定規則を次に示します。

- ARRAY_AGG のソート指定リストには，ナル値ソート順指定は指定できません。
- ARRAY_AGG のソート指定リストには，ソート指定を 2 つ以上指定できません。

(3)　規則

1. ARRAY_AGG の実行結果のデータ型は，配列型になります。要素データ型は，対象データに指定した値式のデータ型と同じになります。また，最大要素数は 30,000 になります。
2. 問合せ指定中にARRAY_AGG を 256 個まで指定できます。
3. ソート指定リストを省略した場合，対象データに指定した値式から導出された集計対象の値が，ARRAY_AGG の入力行になります。入力行になる値の並び順は保証されません。
4. ソート指定リストを指定した場合，対象データに指定した値式から導出された集計対象の値に対して，ソート指定に従ってソート処理を行った結果が，ARRAY_AGG の入力行になります。

   なお，集計対象の値にナル値がある場合，ソート指定リストの順序付け指定によってナル値の扱いが次のように異なります。

   - 順序付け指定にASC を指定した場合，または順序付け指定を省略した場合

     ナル値を末尾に並べ替えます。
   - 順序付け指定にDESC を指定した場合

     ナル値を先頭に並べ替えます。
5. ARRAY_AGG の入力行を求める際，集計対象に含まれるナル値は排除されません。
6. ARRAY_AGG の入力行数が 0 の場合，ARRAY_AGG の実行結果はナル値になります。
7. ARRAY_AGG の実行結果は，入力行に含まれる値を先頭から順に配列要素とする配列データになります。
8. ARRAY_AGG の入力行数が 30,000 を超えると，SQL 文がエラーになります。

(4)　例題

次に示す売上表（SALES）を検索対象とする SQL 文の実行例を説明します。

顧客ID(CID)	商品ID(PID)	単価(PRICE)
C0001	P0008	2000
C0001	P0010	3000
C0001	P0013	1000
C0001	P0016	4000
C0002	P0013	1000
C0002	P0014	2000
C0002	ナル値	1500
C0003	P0013	1000
C0003	P0010	3000
C0004	P0013	1000

例題

売上表（SALES）を検索して，商品ID（PID）を配列要素とする配列データを求めます。求める配列データの条件を次に示します。

- 配列要素の値を単価（PRICE）の昇順に並べる。
- 顧客ID（CID）ごとに配列データを求める。

```
SELECT "CID",ARRAY_AGG("PID" ORDER BY "PRICE") AS "PID_LIST"
  FROM "SALES"
    GROUP BY "CID"
```

実行結果の例

CID	PID_LIST
C0001	{P0013, P0008, P0010, P0016}
C0002	{P0013, ナル値, P0014}
C0003	{P0013, P0010}
C0004	{P0013}

7.23.16　集合関数共通の規則と留意事項

(1)　用語の説明

1. DISTINCT を指定した一般集合関数を **DISTINCT 集合関数**といいます。また，ALL を指定した一般集合関数を **ALL 集合関数**といいます。
2. 次の値式を集合関数の**被集約引数**といいます。
   - 逆分布関数PERCENTILE_CONT およびPERCENTILE_DISC の場合は，WITHIN グループ指定中のソートキーに指定した値式

     (例)

     ```
     SELECT PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY "C1") FROM "T1"
     ```

     下線部分が被集約引数になります。

- ARRAY_AGG集合関数の場合は，引数に指定した値式，およびソート指定リスト中のソートキーに指定した値式

  （例）

  ```
  SELECT ”C1”,ARRAY_AGG(”C2” ORDER BY ”C3”) AS ”C2_LIST” FROM ”T1” GROUP BY ”C1”
  ```

  下線部分が被集約引数になります。

- LISTAGG集合関数の場合は，引数に指定した値式，およびWITHINグループ指定中のソートキーに指定した値式

  （例）

  ```
  SELECT LISTAGG(”C1”,’|’) WITHIN GROUP(ORDER BY ”C2”) AS ”C1_LIST” FROM ”T1”
  ```

  下線部分が被集約引数になります。

- 上記以外の集合関数の場合は，集合関数の引数に指定した値式

  （例）

  ```
  SELECT ”C1”,SUM(”C2”) FROM ”T1” GROUP BY ”C1”
  ```

  下線部分が被集約引数になります。

3. 被集約引数中に含まれる列指定を被集約列指定といいます。

   （例）

   ```
   SELECT ”C1”,SUM(”C2”+1) FROM ”T1” GROUP BY ”C1”
   ```

   下線部分が被集約列指定になります。

4. 被集約列指定が参照する表参照を含むFROM句を直接含む問合せ指定を，その被集約列指定の**修飾問合せ**といいます。

   （例）

   ```
   SELECT ”C1”,SUM(”T1”.”C2”) FROM ”T1” GROUP BY ”C1”
   ```

   下線部分（全体）が修飾問合せになります。

外への参照がある場合の修飾問合せの例を説明します。

（例）

```
SELECT ”C1” FROM ”T1” GROUP BY ”C1” HAVING EXISTS   ...[1]
    (SELECT * FROM ”T2” WHERE MAX(”T1”.”C2”)>”T2”.”C1”)
```

[説明]

- 集合関数MAX(”T1”.”C2”)の被集約列指定は”T1”.”C2”になります。
- ”T1”.”C2”が参照する表参照は”T1”になります。
- ”T1”をFROM句に直接含む問合せ指定は，[1]の部分になります。
- この問合せの修飾問合せは，[1]の部分の問合せ指定になります。

(2) 共通の規則

1. 集合関数は，その集合関数の修飾問合せに直接含まれる選択式，HAVING 句，またはORDER BY 句に指定できます。ただし，ORDER BY 句に集合関数を指定する場合は制限事項があります。制限事項については，「7.25.2 ORDER BY 句にソート指定リストを指定した場合の規則」の「(2) ソートキーに値式を指定した場合の規則」を参照してください。

2. 被集約引数に指定した値式が単独の列指定ではない場合，同一問合せ指定中に逆分布関数を複数指定できません。

   （例）エラーになる SQL 文の例

```
SELECT PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY "C1"+"C2"),
       PERCENTILE_DISC(0.25) WITHIN GROUP (ORDER BY "C1"+"C2")
    FROM "T1"
```

   被集約引数に指定した値式が単独の列指定ではないため，逆分布関数を複数指定できません。

3. 同一問合せ指定中に逆分布関数を複数指定する場合，被集約引数に指定する列指定は，同じ列を参照する必要があります。

   （例）エラーになる SQL 文の例

```
SELECT PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY "C1"),
       PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY "C2")
    FROM "T1"
```

   逆分布関数の列指定は，同じ列を参照する必要があります。

   メモ

   逆分布関数を複数指定できる例を次に示します。

   （例）

```
SELECT PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY "C1"),
       PERCENTILE_DISC(0.25) WITHIN GROUP (ORDER BY "GC1")
    FROM "T1"
    GROUP BY "C1" AS "GC1"
```

   逆分布関数の列指定は，同じグループ化列を参照しているため，エラーにはなりません。

4. 同一問合せ指定中に逆分布関数を複数指定する場合，WITHIN グループ指定のソート指定に指定する順序付け指定は，すべて同じにしてください。

   （例）エラーになる SQL 文の例

```
SELECT PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY "C1" ASC),
       PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY "C1" DESC)
    FROM "T1"
```

5. 集合関数の被集約引数に指定する値式には，？パラメタを指定できません。

   （例）エラーになる SQL 文の例

```
SELECT MAX(CASE WHEN "C1">? THEN "C1" ELSE "C1"*100 END) FROM "T1"
```

下線部分に？パラメタを指定できません。

6. 集合関数中に，集合関数，副問合せ，ウィンドウ関数，およびスカラ関数 RANDOMROW は指定できません。

   （例）エラーになる SQL 文の例

```
SELECT SUM(CASE WHEN MAX("C1")>10000 THEN "C1" END) FROM "T1"
```

   下線部分に集合関数は指定できません。

7. 被集約引数が異なるDISTINCT 集合関数（ウィンドウ関数中に指定したDISTINCT 集合関数は含みません）は，同一問合せ指定中に 64 個まで指定できます。

8. 集合関数の被集約引数に指定する値式に単独の列指定以外の指定をした場合，その値式中に含まれる列指定には外への参照列を指定できません。

   （例）エラーになる SQL 文の例

```
SELECT SUM("C1") FROM "T1"
   HAVING EXISTS(SELECT * FROM "T2" WHERE AVG("T1"."C2"*1.05)>"C2")
```

   下線部分に外への参照列は指定できません。

9. FROM 句，WHERE 句，またはGROUP BY 句のうち，最後に指定した句の結果が集合関数の入力となります。GROUP BY 句の指定がある場合は，グループごとの結果が集合関数の入力となります。

10. 集合関数をウィンドウ関数として使用した場合，現在行のウィンドウ枠に含まれる行集合を集合関数の入力とします。

11. 次の被集約引数には，外への参照列を指定できません。

    - 逆分布関数の被集約引数
    - ARRAY_AGG 集合関数の被集約引数
    - LISTAGG 集合関数の被集約引数

12. 次のソート指定リストには，ソート指定を 2 つ以上指定できません。

    - 逆分布関数に指定するソート指定リスト
    - ARRAY_AGG 集合関数に指定するソート指定リスト
    - LISTAGG 集合関数に指定するソート指定リスト

(3) 共通の留意事項

1. DISTINCT 集合関数，逆分布関数，ARRAY_AGG 集合関数，またはLISTAGG 集合関数を指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。

2. DISTINCT 集合関数，ARRAY_AGG 集合関数，またはLISTAGG 集合関数を指定した場合，導出表が作成されることがあります。導出表の相関名は，HADB が次の名称規則に従って決定します。

```
##DRVTBL_xxxxxxxxxx
```

xxxxxxxxxx は，10 桁の整数です。

3. GROUP BY 句またはHAVING 句の指定がある場合，入力行数が 0 のグループの実行結果は出力されません。

7.24　ウィンドウ関数

ウィンドウ関数を使用すると，表式の結果から導出された行に対して集計範囲を設定し，集計範囲の行の集計値を求めることができます。

ウィンドウ関数の一覧を次の表に示します。

表 7-33　ウィンドウ関数の一覧

項番	ウィンドウ関数	説明
1	RANK	順序付けされた行の集合内の行のランクを求めます。ランクの値は連続した整数値にならないことがあります。
2	DENSE_RANK	順序付けされた行の集合内の行のランクを求めます。ランクの値は連続した整数値となります。
3	CUME_DIST	順序付けされた行の集合内での，行の相対位置を求めます。行*R* のCUME_DIST は，ウィンドウ（区画）内で行*R* より前方にあるか，または行*R* と同じソートキーの値を持つ行の数を，行*R* のウィンドウ（区画）内の行数で割った値になります。
4	ROW_NUMBER	行の集合内の各行に一意の番号を割り当てます。
5	集合関数	ウィンドウ枠に対する集合関数の値を求めます。

7.24.1　ウィンドウ関数の指定形式

(1)　指定形式

```
ウィンドウ関数：：＝ {RANK()
                    |DENSE_RANK()
                    |CUME_DIST()
                    |ROW_NUMBER()
                    |集合関数} OVER(ウィンドウ指定)


  ウィンドウ指定：：＝〔ウィンドウ分割句〕 〔ウィンドウ順序句〕 〔ウィンドウ枠句〕
    ウィンドウ分割句：：＝PARTITION BY 値式 〔,値式〕…
    ウィンドウ順序句：：＝ORDER BY ソート指定リスト
    ウィンドウ枠句：：＝ {ROWS|RANGE}  {開始指定ウィンドウ枠|範囲指定ウィンドウ枠}

  開始指定ウィンドウ枠：：＝ {UNBOUNDED PRECEDING
                             |ウィンドウ枠値指定 PRECEDING
                             |CURRENT ROW}
  範囲指定ウィンドウ枠：：＝BETWEEN 開始指定ウィンドウ枠境界
                                   AND 終了指定ウィンドウ枠境界
    開始指定ウィンドウ枠境界：：＝ウィンドウ枠境界
    終了指定ウィンドウ枠境界：：＝ウィンドウ枠境界
    ウィンドウ枠境界：：＝ {UNBOUNDED PRECEDING
                           |ウィンドウ枠値指定 PRECEDING
                           |CURRENT ROW
                           |ウィンドウ枠値指定 FOLLOWING
```

```
                    | UNBOUNDED FOLLOWING}
    ウィンドウ枠値指定 : : = {符号なし値指定 | ラベル付き間隔}
```

(2) 指定形式の説明

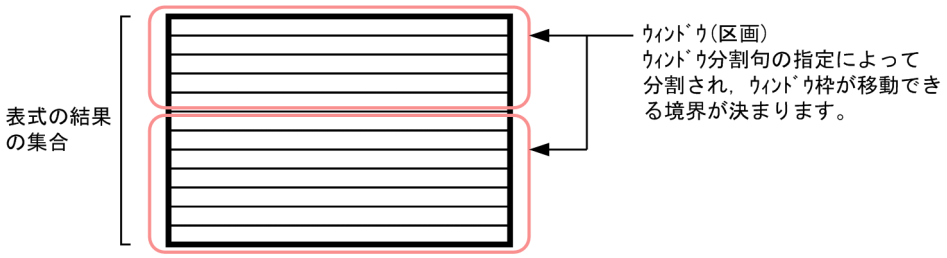
(a) ウィンドウ分割句

```
ウィンドウ分割句 : : =PARTITION BY 値式 〔,値式〕…
```

*値式*の結果を用いて，表式の結果を区画分けします。*ウィンドウ分割句*を省略した場合，表式のすべての結果が 1 つのウィンドウ（区画）になります。

ウィンドウ分割句の機能概要を次の図に示します。

図 7-4 ウィンドウ分割句の機能概要



指定規則を次に示します。

- ウィンドウ分割句には，列指定を含む値式を指定してください。
- ウィンドウ分割句には，最大 16 個の値式を指定できます。
- ウィンドウ分割句に単独の列指定を指定する場合，同じ列は指定できません。
- ウィンドウ分割句の値式には，バイナリデータおよび配列データを指定できません。

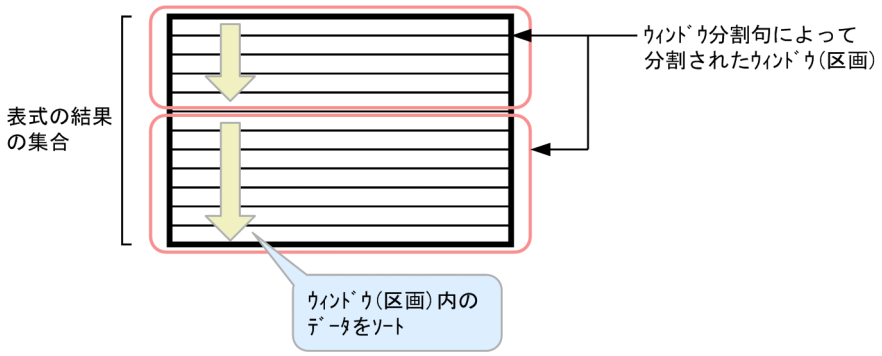
(b) ウィンドウ順序句

```
ウィンドウ順序句 : : =ORDER BY ソート指定リスト
```

ウィンドウ（区画）内のデータを順序付け（ソート）する場合に指定します。ソート指定リストの指定形式，および指定規則については，「7.25 ソート指定リスト」を参照してください。

ウィンドウ順序句の機能概要を次の図に示します。

図 7-5　ウィンドウ順序句の機能概要



指定規則を次に示します。

- ソート指定リストのソートキーに指定できるデータ型を次の表に示します。

表 7-34　ウィンドウ順序句のソート指定リストのソートキーに指定できるデータ型

ウィンドウ枠句の指定		ウィンドウ枠値の指定	ソートキーのデータ型				
			数データ	文字データ	日時データ	バイナリデータ	配列データ
指定あり	ROWS	－	○	○	○	×	×
	RANGE	指定あり	○	×	○	×	×
		指定なし	○	○	○	×	×
指定なし		－	○	○	○	×	×

(凡例)

○：指定できます。

×：指定できません。

－：該当しません。

- ソート指定リストのソートキーに？パラメタを単独で指定した場合，？パラメタのデータ型にはINTEGER型が仮定されます。
- DISTINCT 集合関数，または逆分布関数をウィンドウ関数として使用する場合，ウィンドウ順序句は指定できません。
- RANK，DENSE_RANK，またはCUME_DIST を指定した場合は，ウィンドウ指定中にウィンドウ順序句を指定してください。
- ウィンドウ枠句にRANGE を指定し，ウィンドウ枠境界にウィンドウ枠値指定を指定する場合，ウィンドウ順序句のソート指定リストには，ソート指定を複数指定できません。

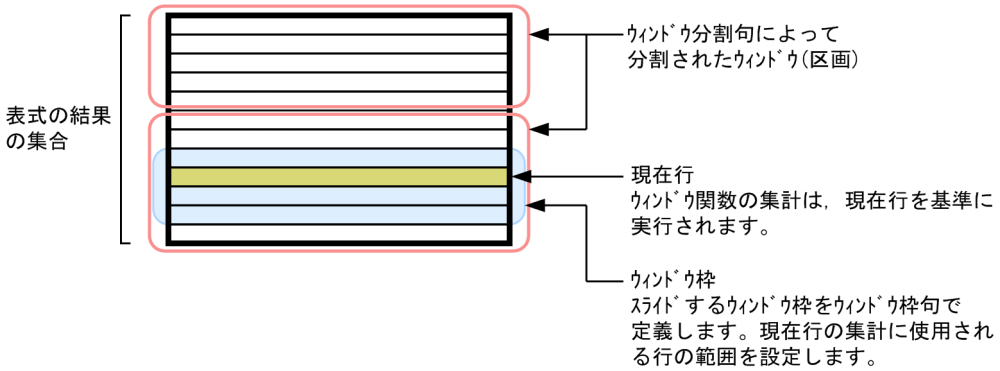
(c)　ウィンドウ枠句

ウィンドウ枠句：：＝ {ROWS | RANGE}　{*開始指定ウィンドウ枠* | *範囲指定ウィンドウ枠*}

ウィンドウ関数の集計範囲をウィンドウ枠として指定します。

ウィンドウ枠句の機能概要を次の図に示します。

図 7-6　ウィンドウ枠句の機能概要



ウィンドウ枠句にROWS を指定した場合は，物理的な行単位のウィンドウ枠になります。RANGE を指定した場合は，論理的な値のオフセット（日時などの論理間隔）としてのウィンドウ枠になります。

*ウィンドウ枠句*の指定を省略した場合のウィンドウ枠の範囲を次に示します。

- **ウィンドウ順序句の指定がある場合**

  ウィンドウ枠の範囲は，次の範囲指定ウィンドウ枠を指定したときと等価になります。

  ```
  RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ```

  ウィンドウ（区画）の先頭行から現在行までがウィンドウ関数の集計範囲になります。ただし，RANGE が仮定されているため，現在行のソートキーの値と同じソートキーの値を持つ後方の行も集計対象になります。

- **ウィンドウ順序句の指定がない場合**

  現在行が含まれるウィンドウ（区画）がウィンドウ関数の集計範囲になります。

指定規則を次に示します。

- ウィンドウ枠句を指定する場合，ウィンドウ関数として使用できる集合関数は，COUNT(*)または一般集合関数（DISTINCT 集合関数を除く）となります。
- 次のウィンドウ（区画）全体を表すウィンドウ枠句以外のウィンドウ枠句を指定する場合は，ウィンドウ順序句の指定が必要です。
  - ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
  - RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
- RANK，DENSE_RANK，CUME_DIST，またはROW_NUMBER を指定した場合は，ウィンドウ指定中にウィンドウ枠句を指定できません。
- ウィンドウ枠句に*開始指定ウィンドウ枠*を指定する場合は，次の範囲指定ウィンドウ枠と等価になります。

```
BETWEEN 開始指定ウィンドウ枠 AND CURRENT ROW
```

- ウィンドウ枠句に範囲指定ウィンドウ枠を指定した場合，*開始指定ウィンドウ枠境界*をウィンドウ枠の前方の境界，*終了指定ウィンドウ枠境界*をウィンドウ枠の後方の境界に設定します。
- ウィンドウ枠境界は次のようになります。
  - UNBOUNDED PRECEDING を指定した場合

    ウィンドウ（区画）内の先頭行からウィンドウ枠が開始します。UNBOUNDED PRECEDING は，*開始指定ウィンドウ枠境界*に指定できます。
  - UNBOUNDED FOLLOWING を指定した場合

    ウィンドウ（区画）内の最終行でウィンドウ枠が終了します。UNBOUNDED FOLLOWING は，*終了指定ウィンドウ枠境界*に指定できます。
  - CURRENT ROW を指定した場合

    ・ROWS を指定した場合：

    *開始指定ウィンドウ枠境界*に指定した場合は，現在行からウィンドウ枠が開始します。*終了指定ウィンドウ枠境界*に指定した場合は，現在行でウィンドウ枠が終了します。

    ・RANGE を指定した場合：

    *開始指定ウィンドウ枠境界*に指定した場合は，現在行のソートキーの値と同じソートキーの値を持つ最初の行からウィンドウ枠が開始します。*終了指定ウィンドウ枠境界*に指定した場合は，現在行のソートキーの値と同じソートキーの値を持つ最後の行でウィンドウ枠が終了します。
  - 「*ウィンドウ枠値指定* PRECEDING」または「*ウィンドウ枠値指定* FOLLOWING」を指定した場合

    ・ROWS を指定した場合：

    *ウィンドウ枠値指定*は，現在行からの物理的な行単位のオフセットになります。*符号なし値指定*のデータ型は，INTEGER 型が指定できます。ラベル付き間隔は指定できません。

    ・RANGE を指定した場合：

    *ウィンドウ枠値指定*は，現在行のソートキーの値からの論理的なオフセットになります。ウィンドウ順序句に指定したソートキーのデータ型と，指定できる符号なし値指定またはラベル付き間隔を次の表に示します。

表 7-35　ウィンドウ順序句に指定したソートキーのデータ型と，指定できる符号なし値指定またはラベル付き間隔（RANGE を指定した場合）

ウィンドウ順序句に指定したソートキーのデータ型	指定できる符号なし値指定またはラベル付き間隔
数データ	数データの符号なし値指定
DATE	ラベル付き間隔（YEARS，MONTHS，DAYS）
TIME	ラベル付き間隔（HOURS，MINUTES，SECONDS，MILLISECONDS，MICROSECONDS，NANOSECONDS，PICOSECONDS）
TIMESTAMP	ラベル付き間隔（YEARS，MONTHS，DAYS，HOURS，MINUTES，SECONDS，MILLISECONDS，MICROSECONDS，NANOSECONDS，PICOSECONDS）

- 範囲指定ウィンドウ枠に指定するウィンドウ枠境界には，指定できない組み合わせがあります。
  - *開始指定ウィンドウ枠境界*にUNBOUNDED FOLLOWING を指定できません。
  - *終了指定ウィンドウ枠境界*にUNBOUNDED PRECEDING を指定できません。

  組み合わせの指定可否を次の表に示します。

表 7-36　組み合わせの指定可否

開始指定ウィンドウ枠境界の指定	終了指定ウィンドウ枠境界の指定			
	UNBOUNDED FOLLOWING	CURRENT ROW	ウィンドウ枠値指定 PRECEDING	ウィンドウ枠値指定 FOLLOWING
UNBOUNDED PRECEDING	○	○	○	○
CURRENT ROW	○	○	×	○
ウィンドウ枠値指定 PRECEDING	○	○	○	○
ウィンドウ枠値指定 FOLLOWING	○	×	×	○

  （凡例）

  ○：指定できます。

  ×：指定できません。

- 範囲指定ウィンドウ枠に指定するウィンドウ枠境界に，*ウィンドウ枠値指定*としてラベル付き間隔を指定する場合は，*開始指定ウィンドウ枠境界*と*終了指定ウィンドウ枠境界*のラベル付き間隔修飾子は同じにしてください。ラベル付き間隔修飾子については，「7.29.1　ラベル付き間隔の指定形式および規則」を参照してください。

  （例）

```
BETWEEN 2 DAYS PRECEDING AND 1 DAYS PRECEDING
```

- *ウィンドウ枠値指定*としてラベル付き間隔を指定する場合は，ラベル付き間隔の*値式一次子*には値指定だけが指定できます。
- *ウィンドウ枠値指定*としてラベル付き間隔を指定する場合は，次の範囲の値が指定できます。

  YEARS：0～9,998

  MONTHS：0～119,987

  DAYS：0～3,652,058

  HOURS：0～87,649,415

  MINUTES：0～5,258,964,959

  SECONDS：0～315,537,897,599

  MILLISECONDS：0～315,537,897,599,999

  MICROSECONDS：0～315,537,897,599,999,999

  NANOSECONDS：0～9,223,372,036,854,775,807

PICOSECONDS：0～9,223,372,036,854,775,807

- *ウィンドウ枠値指定*が負の値，またはナル値の場合はエラーになります。
- *ウィンドウ枠値指定*に？パラメタを指定した場合に仮定されるデータ型を次の表に示します。

表 7-37　ウィンドウ枠値指定に？パラメタを指定した場合に仮定されるデータ型

ウィンドウ枠の指定	ウィンドウ順序句のソートキーのデータ型	ウィンドウ枠値指定に仮定されるデータ型
RANGE	SMALLINT	SMALLINT
	INTEGER	INTEGER
	DECIMAL，NUMERIC	DECIMAL
	DOUBLE PRECISION，FLOAT	DOUBLE PRECISION
	DATE	－（ラベル付き間隔だけが指定できます）
	TIME	
	TIMESTAMP	
ROWS	－	INTEGER

（凡例）

－：該当しません。

7.24.2　ウィンドウ（区画）の設定規則

ウィンドウ（区画）内の構成行，およびウィンドウ（区画）内の行の順序について説明します。

(1)　ウィンドウ（区画）内の構成行について

ウィンドウ分割句に指定した値式の結果の値に基づいて，ウィンドウ指定の適用対象表の行の集合をウィンドウ（区画）に分割します。次の規則に従ってウィンドウ（区画）の構成行が決まります。

- ウィンドウ分割句の値式の結果がナル値でない場合

  値式の結果が同じ値の行が，同じウィンドウ（区画）に属します。同じ値と判定する際の比較規則については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。
- ウィンドウ分割句の値式の結果がナル値の場合

  値式の結果がナル値の行が，同じウィンドウ（区画）に属します。

(2)　ウィンドウ（区画）内の行の順序について

ウィンドウ（区画）内の行の順序は，次のようになります。

- ウィンドウ順序句を指定している場合

ウィンドウ順序句のソート指定リストの指定に従います。ソート指定リストについては，「7.25　ソート指定リスト」を参照してください。

- ウィンドウ順序句を指定していない場合

  順序性の規則は保証されません。ウィンドウ（区画）内の行の順序は，実際に行が取り出された順序によって決まります。

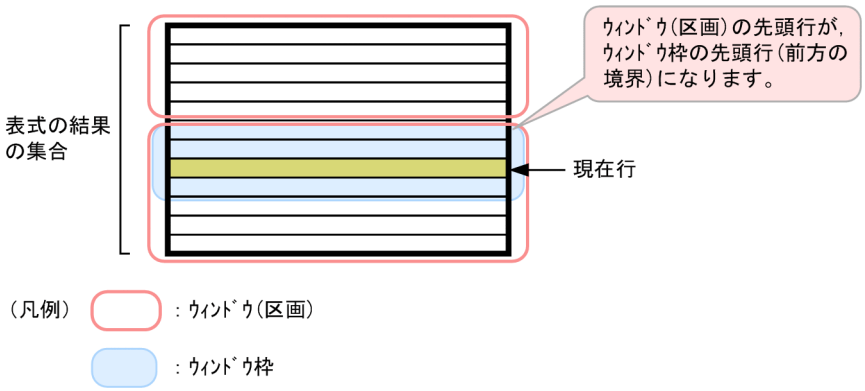
7.24.3　ウィンドウ枠の設定規則（ウィンドウ枠句に RANGE を指定した場合）

ウィンドウ枠句にRANGE を指定した場合，ウィンドウ枠の前方の境界と後方の境界は，次のように決まります。

(1)　ウィンドウ枠の前方の境界

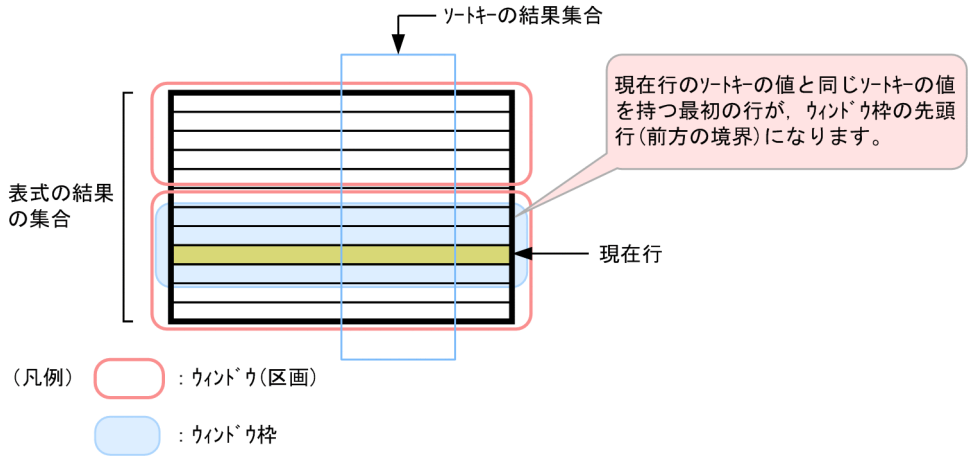
(a)　開始指定ウィンドウ枠境界に UNBOUNDED PRECEDING を指定した場合

ウィンドウ（区画）の先頭行が，ウィンドウ枠の先頭行（前方の境界）になります。



(b)　開始指定ウィンドウ枠境界に CURRENT ROW を指定した場合

現在行のソートキーの値と同じソートキーの値を持つ最初の行が，ウィンドウ枠の先頭行（前方の境界）になります。



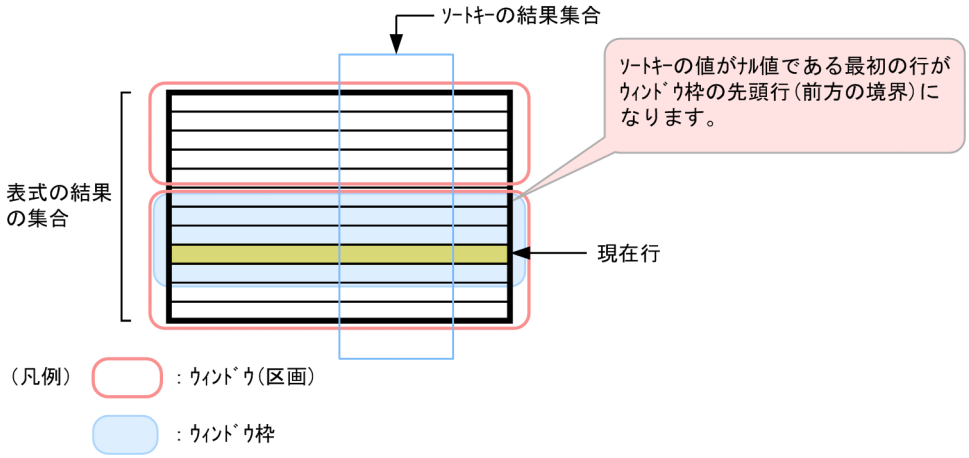
(c)　開始指定ウィンドウ枠境界に「ウィンドウ枠値指定 PRECEDING」または「ウィンドウ枠値指定 FOLLOWING」を指定した場合

ウィンドウ順序句に指定したソートキーの値によって，ウィンドウ枠の前方の境界が決まります。

■現在行のソートキーの値がナル値の場合

「*ウィンドウ枠値指定* PRECEDING」および「*ウィンドウ枠値指定* FOLLOWING」のどちらを開始指定ウィンドウ枠境界に指定しても，ウィンドウ枠の先頭行（前方の境界）は同じになります。

ソートキーの値がナル値である最初の行がウィンドウ枠の先頭行（前方の境界）になります。



■現在行のソートキーの値がナル値でない場合

ウィンドウ枠の前方の境界は，次のように決まります。

- **ウィンドウ枠境界に「*ウィンドウ枠値指定* PRECEDING」を指定した場合**

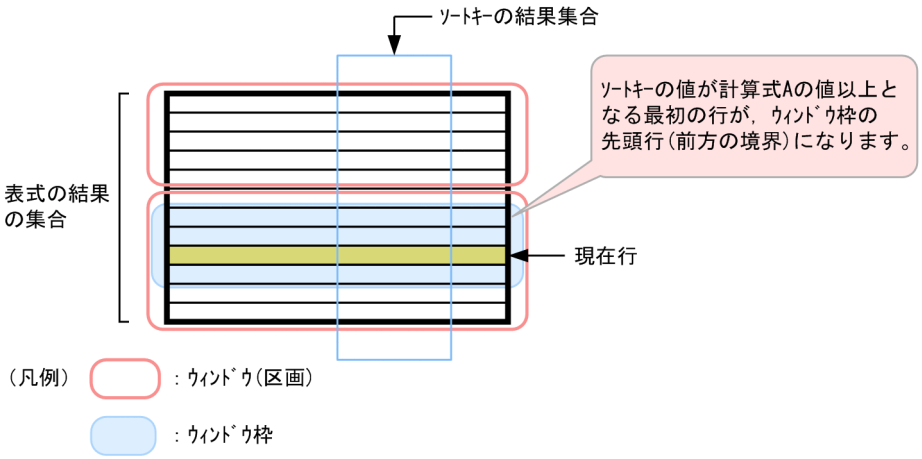
  ソート指定の順序付け指定がASC（昇順）の場合は，ソートキーの値が計算式 A の値以上となる最初の行が，ウィンドウ枠の先頭行（前方の境界）になります。

  ソート指定の順序付け指定がDESC（降順）の場合は，ソートキーの値が計算式 A の値以下となる最初の行が，ウィンドウ枠の先頭行（前方の境界）になります。

  ＜計算式 A ＞

  ・ソート指定の順序付け指定がASC（昇順）の場合：*現在行のソートキーの値－ウィンドウ枠値指定*

・ソート指定の順序付け指定がDESC（降順）の場合：*現在行のソートキーの値＋ウィンドウ枠値指定*



注　ｿｰﾄ指定の順序付け指定がASC（昇順）の場合の例です。

計算式Ａの値が，結果のデータ型で表現できない値になった場合は，結果のデータ型で表現できる最大または最小の値と仮定して，ウィンドウ枠が決定されます。

- **ウィンドウ枠境界に「*ウィンドウ枠値指定* FOLLOWING」を指定した場合**

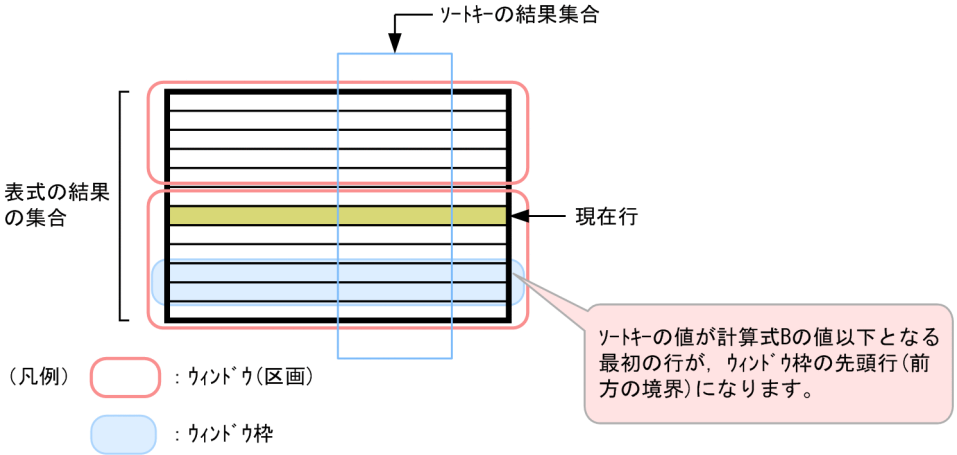
  ソート指定の順序付け指定がASC（昇順）の場合は，ソートキーの値が計算式Ｂの値以上となる最初の行が，ウィンドウ枠の先頭行（前方の境界）になります。

  ソート指定の順序付け指定がDESC（降順）の場合は，ソートキーの値が計算式Ｂの値以下となる最初の行が，ウィンドウ枠の先頭行（前方の境界）になります。

  ＜計算式Ｂ＞

  ・ソート指定の順序付け指定がASC（昇順）の場合：*現在行のソートキーの値＋ウィンドウ枠値指定*

  ・ソート指定の順序付け指定がDESC（降順）の場合：*現在行のソートキーの値－ウィンドウ枠値指定*



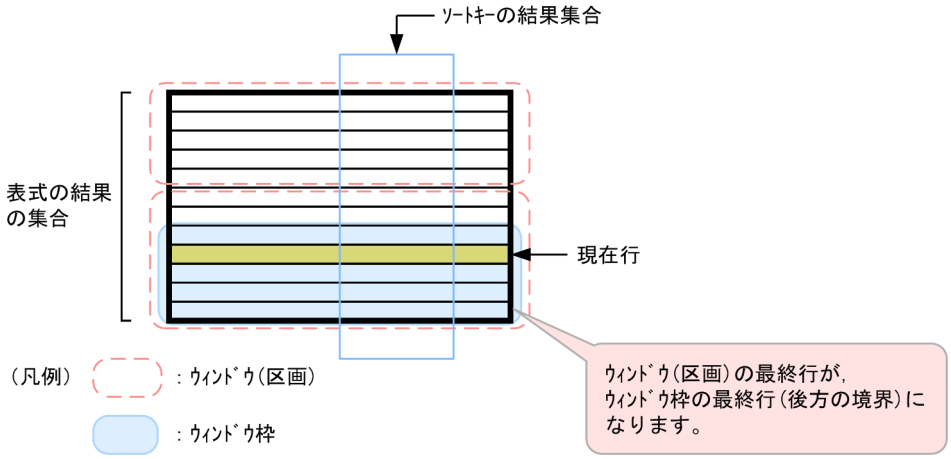
注　ｿｰﾄ指定の順序付け指定がDESC（降順）の場合の例です。

計算式Ｂの値が，結果のデータ型で表現できない値になった場合は，結果のデータ型で表現できる最大または最小の値と仮定して，ウィンドウ枠が決定されます。

(2)　ウィンドウ枠の後方の境界

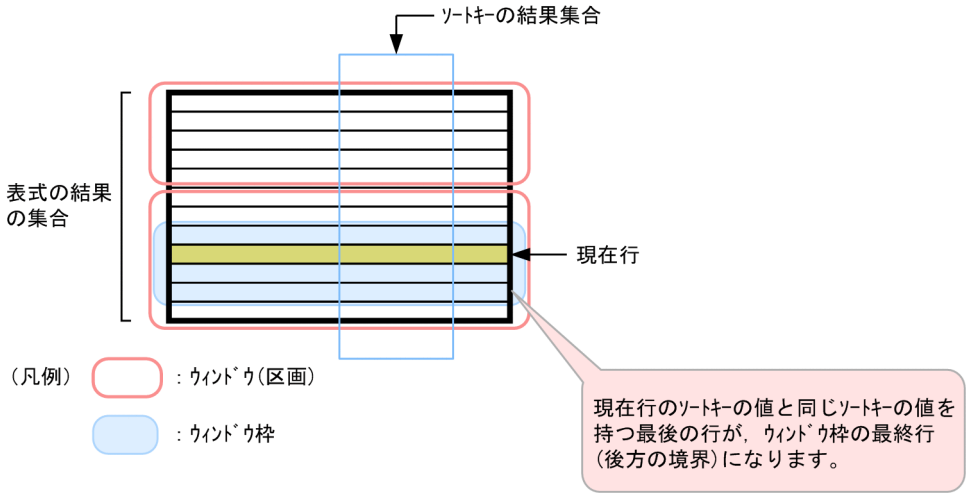
(a)　終了指定ウィンドウ枠境界に UNBOUNDED FOLLOWING を指定した場合

ウィンドウ（区画）の最終行が，ウィンドウ枠の最終行（後方の境界）になります。



(b)　終了指定ウィンドウ枠境界に CURRENT ROW を指定した場合

現在行のソートキーの値と同じソートキーの値を持つ最後の行が，ウィンドウ枠の最終行（後方の境界）になります。



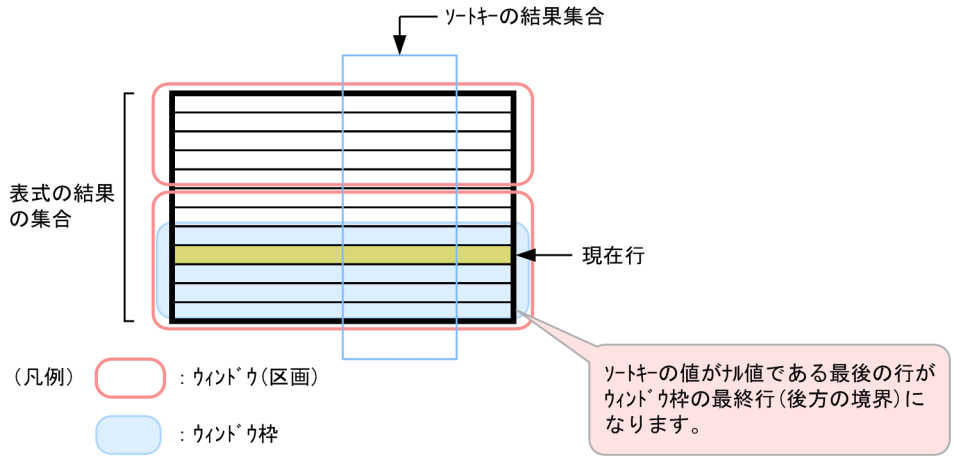
(c)　終了指定ウィンドウ枠境界に「ウィンドウ枠値指定 PRECEDING」または「ウィンドウ枠値指定 FOLLOWING」を指定した場合

ウィンドウ順序句に指定したソートキーの値によって，ウィンドウ枠の後方の境界が決まります。

■現在行のソートキーの値がナル値の場合

「*ウィンドウ枠値指定* PRECEDING」および「*ウィンドウ枠値指定* FOLLOWING」のどちらを終了指定ウィンドウ枠境界に指定しても，ウィンドウ枠の最終行（後方の境界）は同じになります。

ソートキーの値がナル値である最後の行がウィンドウ枠の最終行（後方の境界）になります。



■現在行のソートキーの値がナル値でない場合

ウィンドウ枠の前方の境界は，次のように決まります。

- **ウィンドウ枠境界に「*ウィンドウ枠値指定* PRECEDING」を指定した場合**

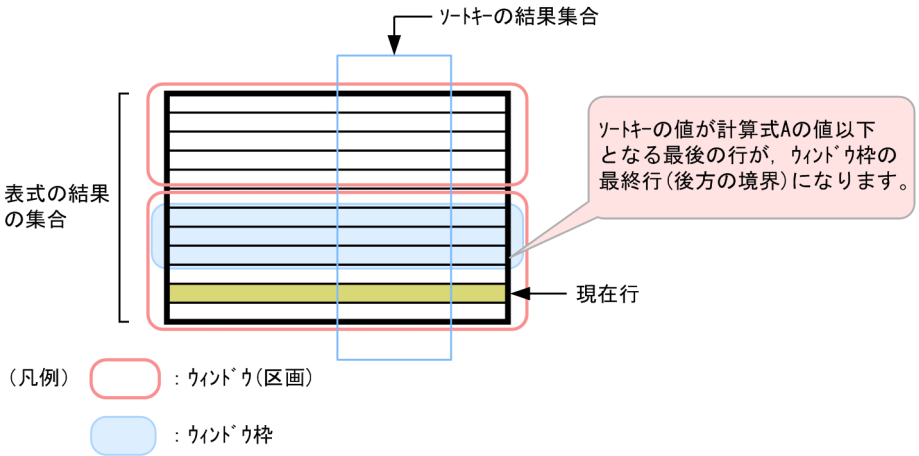
  ソート指定の順序付け指定がASC（昇順）の場合は，ソートキーの値が計算式 A の値以下となる最後の行が，ウィンドウ枠の最終行（後方の境界）になります。

  ソート指定の順序付け指定がDESC（降順）の場合は，ソートキーの値が計算式 A の値以上となる最後の行が，ウィンドウ枠の最終行（後方の境界）になります。

  ＜計算式 A ＞

  ・ソート指定の順序付け指定がASC（昇順）の場合：*現在行のソートキーの値－ウィンドウ枠値指定*

  ・ソート指定の順序付け指定がDESC（降順）の場合：*現在行のソートキーの値＋ウィンドウ枠値指定*



注　ｿｰﾄ指定の順序付け指定がASC(昇順)の場合の例です。

計算式 A の値が，結果のデータ型で表現できない値になった場合は，結果のデータ型で表現できる最大または最小の値と仮定して，ウィンドウ枠が決定されます。

- **ウィンドウ枠境界に「*ウィンドウ枠値指定* FOLLOWING」を指定した場合**

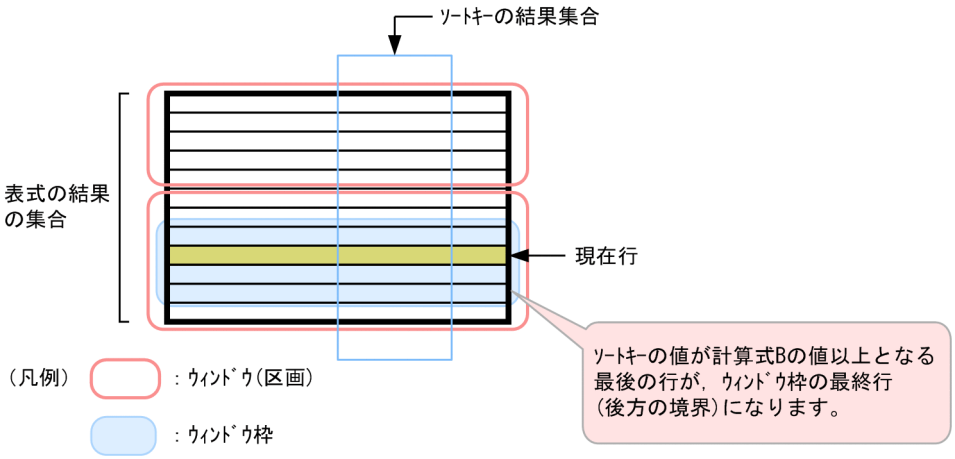
  ソート指定の順序付け指定がASC（昇順）の場合は，ソートキーの値が計算式 B の値以下となる最後の行が，ウィンドウ枠の最終行（後方の境界）になります。

ソート指定の順序付け指定がDESC（降順）の場合は，ソートキーの値が計算式Ｂの値以上となる最後の行が，ウィンドウ枠の最終行（後方の境界）になります。

＜計算式Ｂ＞

・ソート指定の順序付け指定がASC（昇順）の場合：*現在行のソートキーの値＋ウィンドウ枠値指定*

・ソート指定の順序付け指定がDESC（降順）の場合：*現在行のソートキーの値－ウィンドウ枠値指定*



注　ｿｰﾄ指定の順序付け指定がDESC(降順)の場合の例です。

計算式Ｂの値が，結果のデータ型で表現できない値になった場合は，結果のデータ型で表現できる最大または最小の値と仮定して，ウィンドウ枠が決定されます。

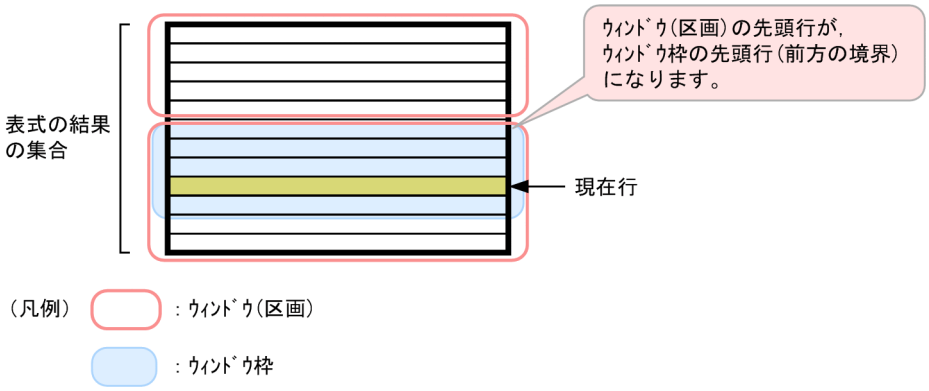
7.24.4　ウィンドウ枠の設定規則（ウィンドウ枠句に ROWS を指定した場合）

ウィンドウ枠句にROWS を指定した場合，ウィンドウ枠の前方の境界と後方の境界は，次のように決まります。

(1)　ウィンドウ枠の前方の境界

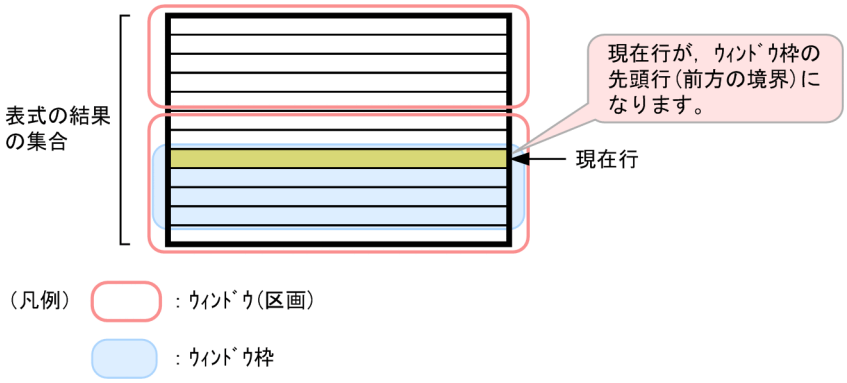
(a)　開始指定ウィンドウ枠境界に UNBOUNDED PRECEDING を指定した場合

ウィンドウ（区画）の先頭行が，ウィンドウ枠の先頭行（前方の境界）になります。



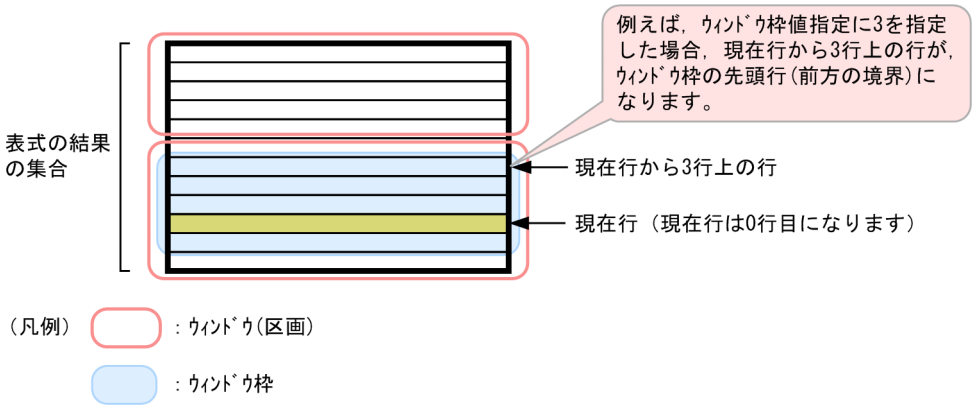
(b)　開始指定ウィンドウ枠境界に CURRENT ROW を指定した場合

現在行が，ウィンドウ枠の先頭行（前方の境界）になります。



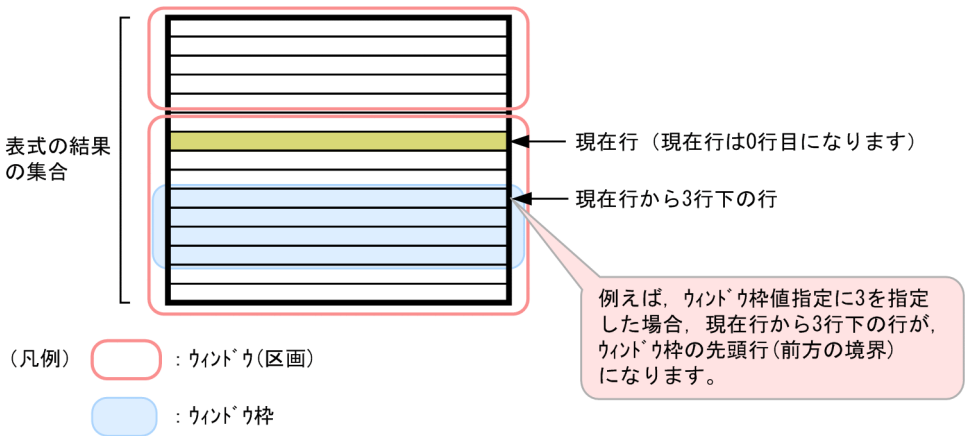
(c)　開始指定ウィンドウ枠境界に「ウィンドウ枠値指定 PRECEDING」を指定した場合

現在行から数えて，*ウィンドウ枠値指定*の行数分の前方行が，ウィンドウ枠の先頭行（前方の境界）になります。



(d)　開始指定ウィンドウ枠境界に「ウィンドウ枠値指定 FOLLOWING」を指定した場合

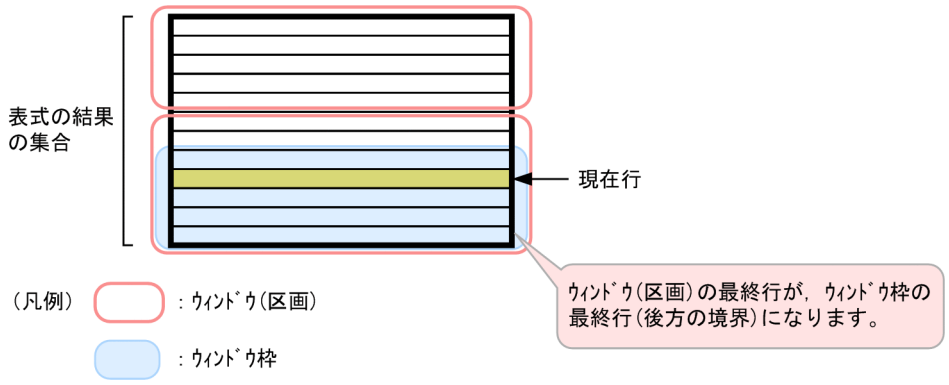
現在行から数えて，*ウィンドウ枠値指定*の行数分の後方行が，ウィンドウ枠の先頭行（前方の境界）になります。



(2)　ウィンドウ枠の後方の境界

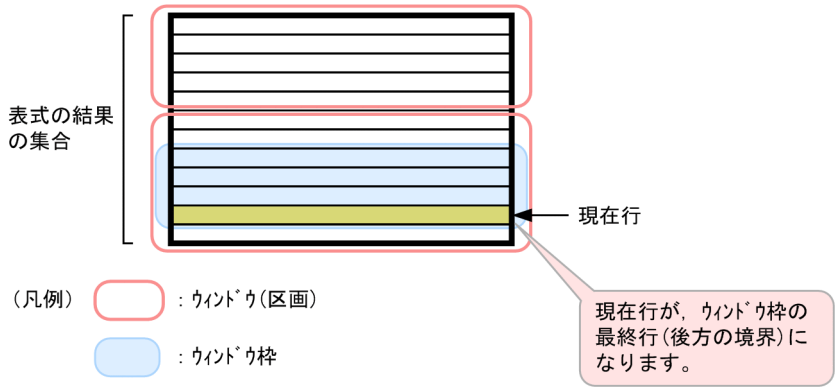
(a)　終了指定ウィンドウ枠境界に UNBOUNDED FOLLOWING を指定した場合

ウィンドウ（区画）の最終行が，ウィンドウ枠の最終行（後方の境界）になります。



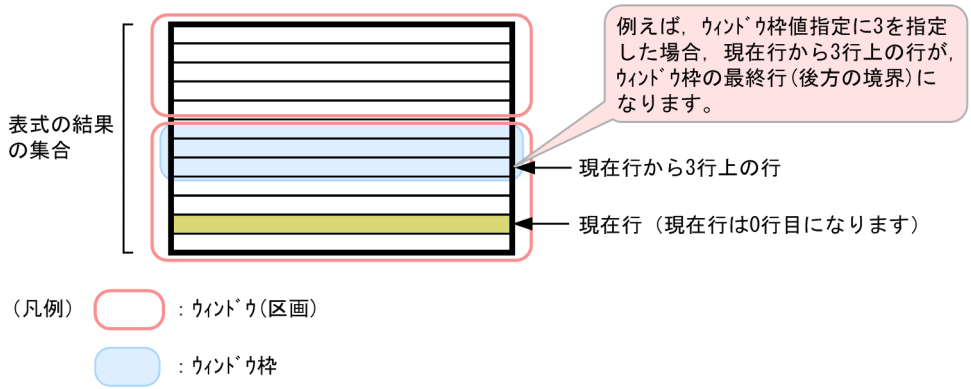
(b)　終了指定ウィンドウ枠境界に CURRENT ROW を指定した場合

現在行が，ウィンドウ枠の最終行（後方の境界）になります。



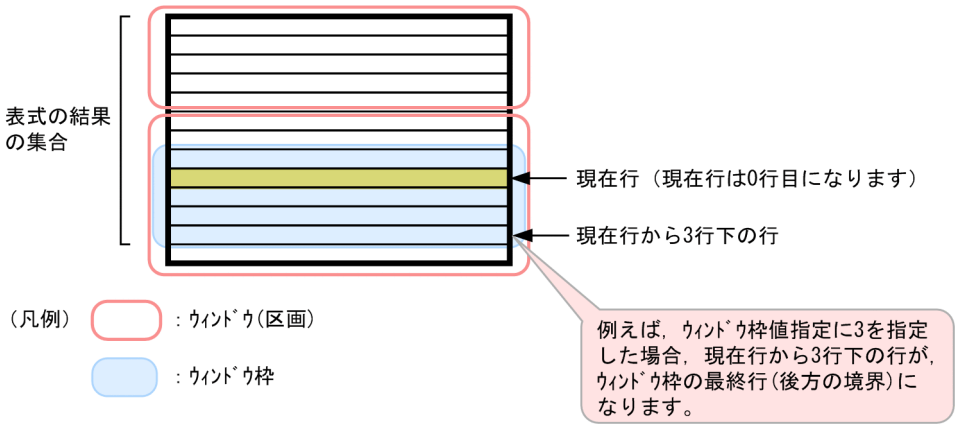
(c)　終了指定ウィンドウ枠境界に「ウィンドウ枠値指定 PRECEDING」を指定した場合

現在行から数えて，*ウィンドウ枠値指定*の行数分の前方行が，ウィンドウ枠の最終行（後方の境界）になります。



(d)　終了指定ウィンドウ枠境界に「ウィンドウ枠値指定 FOLLOWING」を指定した場合

現在行から数えて，*ウィンドウ枠値指定*の行数分の後方行が，ウィンドウ枠の最終行（後方の境界）になります。



7.24.5　ウィンドウ関数の規則および留意事項

1. ウィンドウ関数は選択式またはORDER BY 句に指定できます。ただし，ORDER BY 句にウィンドウ関数を指定した場合，ORDER BY 句のソートキーに指定した値式と同じ値式を選択式にも指定する必要があります。
2. ウィンドウ関数は 1 つの問合せ指定中に 8 個まで指定できます。
3. ウィンドウ関数中にウィンドウ関数，副問合せ，スカラ関数RANDOMROW，ARRAY_AGG 集合関数，およびLISTAGG 集合関数は指定できません。
4. ウィンドウ関数のRANK，DENSE_RANK，またはROW_NUMBER の実行結果のデータ型はINTEGER 型になります。CUME_DIST の実行結果のデータ型はDOUBLE PRECISION 型になります。ウィンドウ関数として使用する集合関数の実行結果のデータ型については，「7.23　集合関数」の各集合関数の説明を参照してください。
5. ウィンドウ関数は表式の結果（FROM 句およびWHERE 句の結果）から導出される行の集合に対して適用されます。表式の結果の行がない場合は，ウィンドウ関数は実行されません。
6. ROW とウィンドウ関数は同時に指定できません。
7. GROUP BY 句，HAVING 句，または集合関数を指定した場合，ウィンドウ関数，およびウィンドウ指定に含まれる集合関数の被集約引数でない位置に指定した列指定には，グループ化列を指定する必要があります。

   (例)

```
SELECT COUNT("C1") OVER(PARTITION BY SUM("C2")
                        ORDER BY "C1" RANGE UNBOUNDED PRECEDING)
    FROM "T1"
    GROUP BY "C1"
```

上記の SQL 文の場合，GROUP BY 句に指定されているC1 列がグループ化列になります。COUNT(”C1”)および ORDER BY ”C1”に指定したC1 は，集合関数の被集約引数ではないため，グループ化列を指定する必要があります。一方，SUM(”C2”)に指定したC2 は，集合関数の被集約引数のため，グループ化列を指定する必要はありません。

8. ウィンドウ関数を指定した場合，作業表が作成されることがあります。作業表が作成される作業表用 DB エリアの容量が正しく見積もられていない場合，性能低下の原因となることがあります。作業表用 DB エリアの容量見積もりについては，マニュアル『HADB　システム構築・運用ガイド』を参照してください。作業表の詳細については，マニュアル『HADB　AP 開発ガイド』の『作業表が作成される SQL を実行する際の考慮点』を参照してください。

7.24.6　ウィンドウ関数の使用例

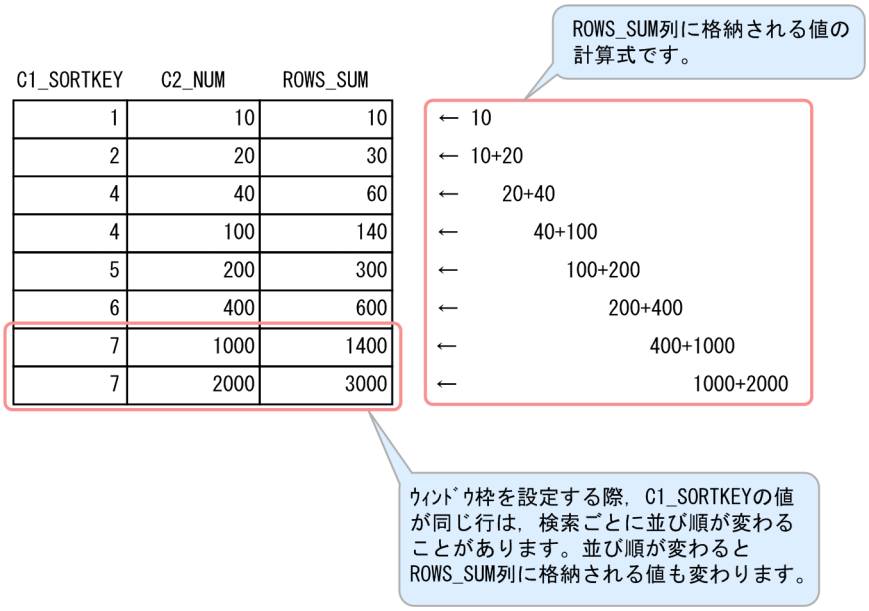
(1)　ウィンドウ枠句に ROWS または RANGE を指定する例

ウィンドウ枠句のROWS とRANGE の指定の違いについて，次の移動累計を求める SQL 文を例にして説明します。

(a)　ROWS を指定した例

```
SELECT ”C1_SORTKEY”, ”C2_NUM”,
        SUM(”C2_NUM”) OVER(ORDER BY ”C1_SORTKEY”
                        ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) AS ”ROWS_SUM”
    FROM ”T1”
    ORDER BY ”C1_SORTKEY”, ”C2_NUM”
```

実行結果の例



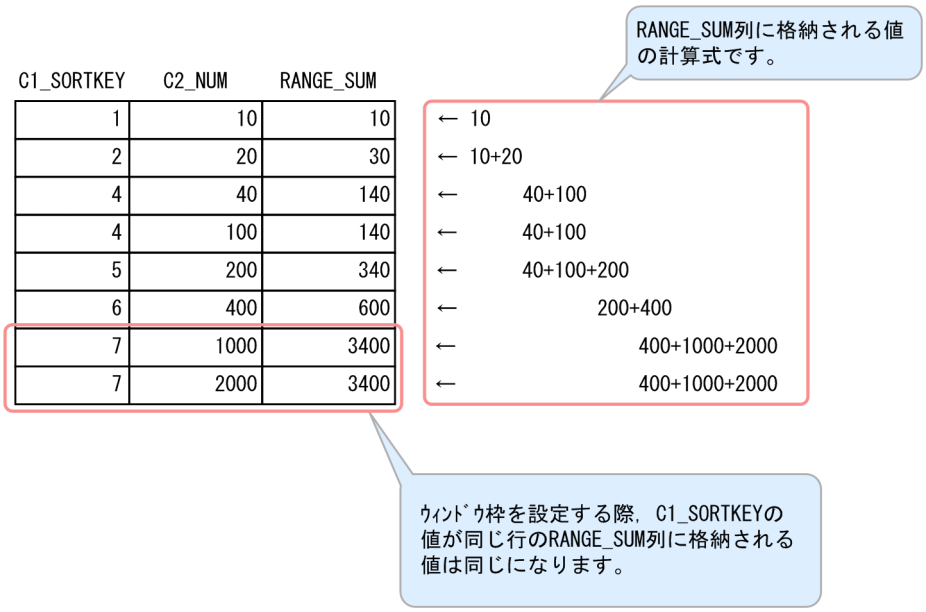
[説明]

- 上記の例の場合，C1_SORTKEY の値が昇順に並んだ状態でウィンドウ枠が設定されます。設定されるウィンドウ枠は，現在行の 1 行前から現在行までをウィンドウ関数の集計範囲としています。
- ROWS_SUM 列には，集計範囲の行のC2_NUM 列の値を合計した値が格納されます。

(b) RANGE を指定した例

```
SELECT "C1_SORTKEY","C2_NUM",
        SUM("C2_NUM") OVER(ORDER BY "C1_SORTKEY"
                        RANGE BETWEEN 1 PRECEDING AND CURRENT ROW) AS "RANGE_SUM"
    FROM "T1"
    ORDER BY "C1_SORTKEY","C2_NUM"
```

実行結果の例



[説明]

- 上記の例の場合，C1_SORTKEY の値が昇順に並んだ状態でウィンドウ枠が設定されます。設定されるウィンドウ枠は，C1_SORTKEY の値が現在行の値より 1 小さい行から，現在行と同じ値を持つ行までをウィンドウ関数の集計範囲としています。
- RANGE_SUM 列には，集計範囲の行のC2_NUM 列の値を合計した値が格納されます。

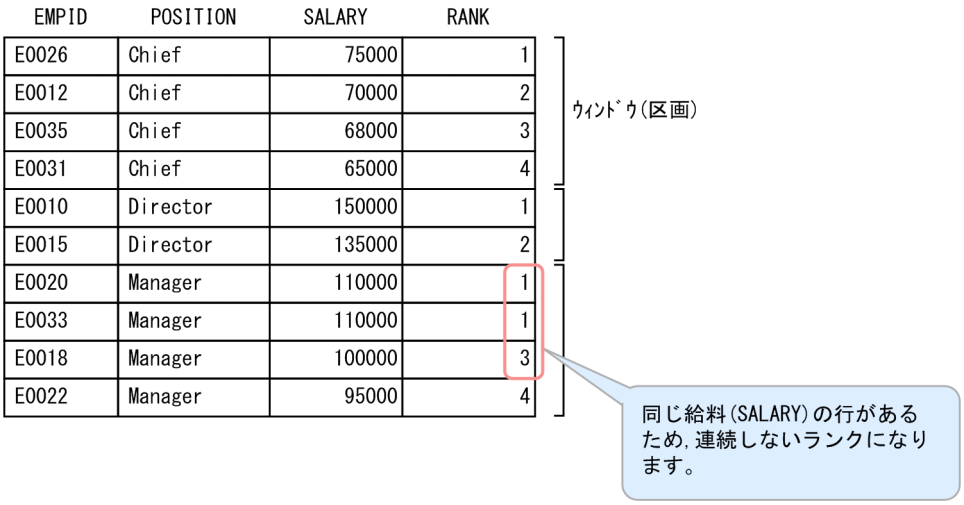
(2) RANK の指定例

給与表（SALARYLIST）から，社員の職級（POSITION）ごとに，社員の給料（SALARY）のランクを求めます。

```
SELECT "EMPID","POSITION","SALARY",
        RANK() OVER(PARTITION BY "POSITION" ORDER BY "SALARY" DESC) AS "RANK"
    FROM "SALARYLIST"
    ORDER BY "POSITION","SALARY" DESC,"EMPID"
```

実行結果の例

EMPID	POSITION	SALARY	RANK
E0026	Chief	75000	1
E0012	Chief	70000	2
E0035	Chief	68000	3
E0031	Chief	65000	4
E0010	Director	150000	1
E0015	Director	135000	2
E0020	Manager	110000	1
E0033	Manager	110000	1
E0018	Manager	100000	3
E0022	Manager	95000	4



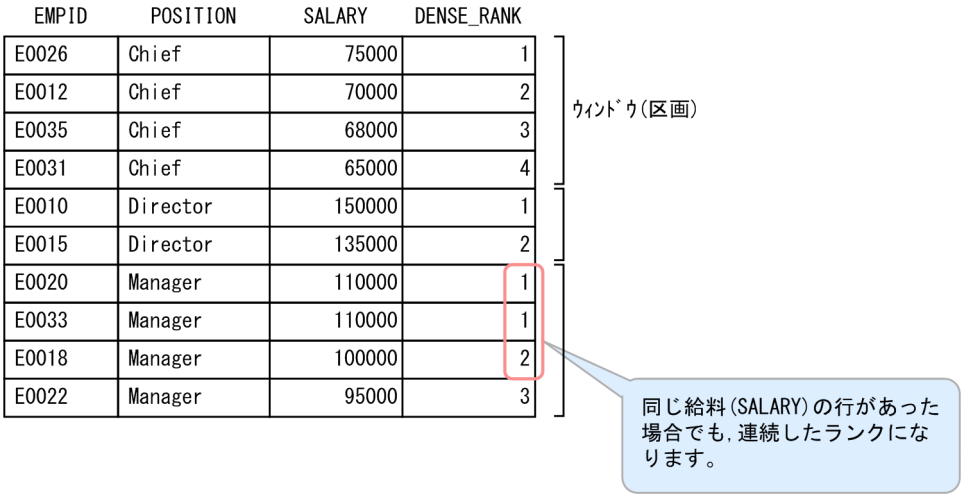
(3) DENSE_RANK の指定例

給与表（SALARYLIST）から，社員の職級（POSITION）ごとに，社員の給料（SALARY）のランクを求めます。

```
SELECT ”EMPID”,”POSITION”,”SALARY”,
        DENSE_RANK() OVER(PARTITION BY ”POSITION” ORDER BY ”SALARY” DESC) AS ”DENSE_RANK”
    FROM ”SALARYLIST”
    ORDER BY ”POSITION”,”SALARY” DESC,”EMPID”
```

実行結果の例

EMPID	POSITION	SALARY	DENSE_RANK
E0026	Chief	75000	1
E0012	Chief	70000	2
E0035	Chief	68000	3
E0031	Chief	65000	4
E0010	Director	150000	1
E0015	Director	135000	2
E0020	Manager	110000	1
E0033	Manager	110000	1
E0018	Manager	100000	2
E0022	Manager	95000	3



(4) CUME_DIST の指定例

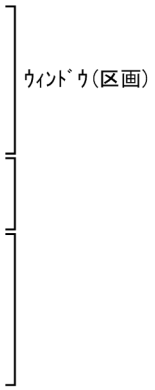
給与表（SALARYLIST）から，社員の職級（POSITION）ごとに，社員の給料（SALARY）の相対位置を求めます。

```
SELECT ”EMPID”,”POSITION”,”SALARY”,
        CUME_DIST() OVER(PARTITION BY ”POSITION” ORDER BY ”SALARY” DESC) AS ”CUME_DIST”
```

```
    FROM "SALARYLIST"
    ORDER BY "POSITION","SALARY" DESC,"EMPID"
```

実行結果の例

EMPID	POSITION	SALARY	CUME_DIST
E0026	Chief	75000	2.5000000000000000E-1
E0012	Chief	70000	5.0000000000000000E-1
E0035	Chief	68000	7.5000000000000000E-1
E0031	Chief	65000	1.0000000000000000E0
E0010	Director	150000	5.0000000000000000E-1
E0015	Director	135000	1.0000000000000000E0
E0020	Manager	110000	5.0000000000000000E-1
E0033	Manager	110000	5.0000000000000000E-1
E0018	Manager	100000	7.5000000000000000E-1
E0022	Manager	95000	1.0000000000000000E0



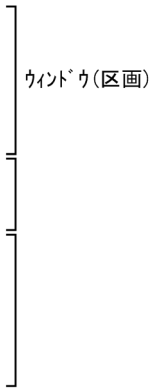
(5) ROW_NUMBER の指定例

給与表（SALARYLIST）から，社員の職級（POSITION）ごとに，社員の給料（SALARY）に対する降順の行番号を求めます。

```
SELECT "EMPID","POSITION","SALARY",
        ROW_NUMBER() OVER(PARTITION BY "POSITION" ORDER BY "SALARY" DESC) AS "ROW_NUMBER"
    FROM "SALARYLIST"
    ORDER BY "POSITION","SALARY" DESC,"EMPID"
```

実行結果の例

EMPID	POSITION	SALARY	ROW_NUMBER
E0026	Chief	75000	1
E0012	Chief	70000	2
E0035	Chief	68000	3
E0031	Chief	65000	4
E0010	Director	150000	1
E0015	Director	135000	2
E0020	Manager	110000	1
E0033	Manager	110000	2
E0018	Manager	100000	3
E0022	Manager	95000	4



(6) PERCENTILE_CONT の指定例

給与表（SALARYLIST）から，社員の職級（POSITION）ごとに，社員の給料（SALARY）の中央値（50 パーセンタイル）を求めます。

```
SELECT "EMPID","POSITION","SALARY",
        PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY "SALARY")
```

```
        OVER(PARTITION BY "POSITION") AS "PERCENTILE_CONT"
FROM "SALARYLIST"
ORDER BY "POSITION","SALARY","EMPID"
```

実行結果の例

EMPID	POSITION	SALARY	PERCENTILE_CONT
E0031	Chief	65000	6.9000000000000000E4
E0035	Chief	68000	6.9000000000000000E4
E0012	Chief	70000	6.9000000000000000E4
E0026	Chief	75000	6.9000000000000000E4
E0015	Director	135000	1.4250000000000000E5
E0010	Director	150000	1.4250000000000000E5
E0022	Manager	95000	1.0500000000000000E5
E0018	Manager	100000	1.0500000000000000E5
E0020	Manager	110000	1.0500000000000000E5
E0033	Manager	110000	1.0500000000000000E5

ｳｨﾝﾄﾞｳ(区画)

7.25　ソート指定リスト

ここでは，ソート指定リストについて説明します。ソート指定リストは，次の個所に指定します。

- ORDER BY 句
  「4.4.1　SELECT 文の指定形式および規則」を参照してください。
- WITHIN グループ指定
  「7.23.12　PERCENTILE_CONT」，「7.23.13　PERCENTILE_DISC」，および「7.23.14　LISTAGG」を参照してください。
- ウィンドウ順序句
  「7.24.1　ウィンドウ関数の指定形式」を参照してください。
- ARRAY_AGG 集合関数
  「7.23.15　ARRAY_AGG」を参照してください。

7.25.1　ソート指定リストの指定形式

ソート指定リストでデータのソート順を指定します。

(1)　指定形式

```
ソート指定リスト：：＝ソート指定〔,ソート指定〕…


  ソート指定：：＝ソートキー　〔順序付け指定〕　〔ナル値ソート順指定〕
    ソートキー：：＝｛値式｜ソート項目指定番号｝
    順序付け指定：：＝｛ASC｜DESC｝
    ナル値ソート順指定：：＝NULLS　｛FIRST｜LAST｝
```

(2)　指定形式の説明

●*ソート指定*

```
ソート指定：：＝ソートキー　〔順序付け指定〕　〔ナル値ソート順指定〕
```

ソート指定には，ソートキー，順序付け指定，およびナル値ソート順指定を指定します。

指定規則を次に示します。

- ORDER BY 句またはウィンドウ順序句に指定したソート指定リストには，ソート指定を最大 64 個指定できます。
- WITHIN グループ指定またはARRAY_AGG 集合関数に指定したソート指定リストには，ソート指定を 1 つだけ指定できます（2 つ以上指定できません）。

●*ソートキー*

ソートキー：：＝｛*値式*｜*ソート項目指定番号*｝

ソートキーを値式またはソート項目指定番号のどちらかの形式で指定します。

重要

- ソート項目指定番号は，ORDER BY 句のソートキーにだけ指定できます。
- ソートキーに整数定数を指定した場合は，ソート項目指定番号と見なされます。整数定数以外の定数を指定した場合は，値式と見なされます。

値式：

ソートキーを値式の形式で指定します。値式については，「7.21　値式」を参照してください。

なお，複数のソートキーを指定した場合，最初に指定したソートキーが，並び替えの優先順位がいちばん高くなります。例えば，ORDER BY ”C1”,”C2”,”C3”と指定した場合，ソートキーの優先順位はC1，C2，C3 の順になります。

また，値式には配列データを指定できません。

ソート項目指定番号：

ソートキーとする列の番号を指定します。例えば，2 を指定した場合，問合せ式本体によって導出される表の 2 番目の列が，ソートキーになります。

（例）

```
SELECT ”C1”,”C2” FROM ”T1”
    ORDER BY 2 ASC
```

上記のSELECT 文を実行した場合，C2 列がソートキーになります。

指定規則を次に示します。

- ソート項目指定番号には整数定数を指定します。
- ソート項目指定番号の指定範囲は，「1～問合せ式本体によって導出される表の列数」となります。

  （例）

  ```
  SELECT ”C1”,SUM(”C2”),AVG(”C2”) FROM ”T1”
      GROUP BY ”C1” ORDER BY 3 ASC
  ```

  上記のSELECT 文を実行する場合，ORDER BY 句に指定できるソート項目指定番号は1～3 になります。
- 2 つ以上のソート項目指定番号を指定した場合，最初に指定したソート項目指定番号が，並び替えの優先順位がいちばん高くなります。例えば，ORDER BY 2,3,1 と指定した場合，優先順位はソート項目指定番号 2 の列，ソート項目指定番号 3 の列，ソート項目指定番号 1 の列の順になります。
- ［*表指定*.］ROW に対応するソート項目指定番号は指定できません。

  （例）エラーになる SQL 文の例

  ```
  SELECT ”C1”,ROW FROM ”T1” ORDER BY 2
  ```

重要

次の個所に指定したソート指定リストのソートキーに整数定数を指定した場合，ソート項目指定番号ではなく値式と見なされます。

- WITHIN グループ指定
- ウィンドウ順序句
- ARRAY_AGG 集合関数

●*順序付け指定*

順序付け指定：：＝ {ASC | DESC}

ソート結果を昇順に並べるか，または降順に並べるかを指定します。次のどちらかを指定します。

ASC：ソート結果を昇順に並べる場合に指定します。

DESC：ソート結果を降順に並べる場合に指定します。

順序付け指定を省略した場合は，ASC が仮定されます。

●*ナル値ソート順指定*

ナル値ソート順指定：：＝NULLS {FIRST | LAST}

ソート実行時のナル値の並びを指定します。次のどちらかを指定します。

NULLS FIRST：ナル値を先頭に並べます。

NULLS LAST：ナル値を末尾に並べます。

ナル値ソート順指定を省略した場合は，ナル値の並びは次のようになります。

- 順序付け指定にASC を指定した場合，または順序付け指定を省略した場合は，ナル値を末尾に並べます。NULLS LAST と同じです。
- 順序付け指定にDESC を指定した場合は，ナル値を先頭に並べます。NULLS FIRST と同じです。

重要

WITHIN グループ指定またはARRAY_AGG 集合関数に指定したソート指定リストには，ナル値ソート順指定を指定できません。

7.25.2 ORDER BY 句にソート指定リストを指定した場合の規則

(1) 共通の規則

1. ソートキーには，値式とソート項目指定番号の両方を混在して指定できます。

（例）

```
SELECT ”C1”, AVG(”C2”) FROM ”T1”
    GROUP BY ”C1”
    ORDER BY ”C1” ASC, 2 ASC
```

2. ソートキーを重複して指定した場合，先に指定した順序付け指定，およびナル値ソート順指定が優先されます。

   （例）

```
SELECT ”C1”,”C2” FROM ”T1”
    ORDER BY ”C1” ASC NULLS FIRST,”C1” DESC NULLS LAST
```

   上記の場合，先に指定した下線部分の指定が優先されます。

3. 選択リストに同じ列を 2 つ以上指定した場合，その列をソートキーに指定できません。エラーになる SQL 文の例を次に示します。

   （例）エラーになる SQL 文の例

```
SELECT ”C1”,”C2”,”C1” FROM ”T1” ORDER BY ”C1”
```

4. ORDER BY 句に指定した導出列名が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。例えば，次に示す 3 つの SQL 文は同じ検索結果になります。

```
SELECT ”T1”.”C1” DR1,”T1”.”C2” DR2 FROM ”T1” ORDER BY DR1
SELECT ”T1”.”C1” DR1,”T1”.”C2” DR2 FROM ”T1” ORDER BY ”T1”.”C1”
SELECT ”T1”.”C1” DR1,”T1”.”C2” DR2 FROM ”T1” ORDER BY 1
```

(2) ソートキーに値式を指定した場合の規則

1. ソートキーに？パラメタを単独で指定できません。

2. ソートキーに値式（列指定だけの値式を除く）を指定した場合，導出列名（単純な列指定から成る導出列を除く）をソートキーに指定することはできません。

   ■エラーになる SQL 文の例

```
SELECT ”C1”,SUM(”C2”) AS ”SUMC2” FROM ”T1”
    GROUP BY ”C1”
    ORDER BY ”SUMC2”+1
```

   また，集合演算を指定した場合，集合演算によって導出される導出列名をソートキーに指定することはできません（単純な列指定から成る導出列であっても指定できません）。

   ■エラーになる SQL 文の例

```
SELECT ”C1”,”C2” FROM ”T1” UNION ALL SELECT ”C1”+”C2”,”C3” FROM ”T1”
    ORDER BY ”C1”+”C2”
```

3. 集合演算を指定したSELECT 文のソートキーに値式を指定した場合，同じ値式を最初の問合せ指定の選択式に指定する必要があります。

   ■正しい SQL 文の例

   （例 1）

```
SELECT "C1"+"C2","C3" FROM "T1" UNION SELECT "C1","C2" FROM "T2"
    ORDER BY "C1"+"C2"
SELECT "C1"+"C2","C2" FROM "T1" UNION SELECT "C1","C2" FROM "T2"
    ORDER BY "C2"
```

上記の例の場合，最初の問合せ指定の選択式に，ソートキーで指定した値式と同じ値式が指定されているため，SELECT 文を実行できます。

（例 2）

```
SELECT "C1"+"C2" AS "C1","C2" FROM "T1" UNION SELECT "C1","C2" FROM "T2"
    ORDER BY "C1"
```

上記の例のようにソートキーに指定した値式が列指定の場合は，AS 句を指定することでSELECT 文を実行できます。

■エラーになる SQL 文の例

```
SELECT "C1","C2" FROM "T1" UNION SELECT "C1"+"C2","C2" FROM "T2"
    ORDER BY "C1"+"C2"
```

上記の例の場合，最初の問合せ指定の選択式に，ソートキーで指定した値式と同じ値式が指定されていないため，エラーになります。

4. DISTINCT 指定ありのSELECT 文のソートキーに値式を指定した場合，同じ値式を選択式に指定する必要があります。

■正しい SQL 文の例

（例 1）

```
SELECT DISTINCT "C1"+"C2","C2" FROM "T1" ORDER BY "C1"+"C2"
SELECT DISTINCT "C1"+"C2","C2" FROM "T1" ORDER BY "C2"
```

上記の例の場合，ソートキーで指定した値式と同じ値式が選択式に指定されているため，SELECT 文を実行できます。

（例 2）

```
SELECT DISTINCT "C1"+"C2" AS "C1","C2" FROM "T1" ORDER BY "C1"
```

上記の例のようにソートキーに指定した値式が列指定の場合は，AS 句を指定することでSELECT 文を実行できます。

■エラーになる SQL 文の例

```
SELECT DISTINCT "C1","C2" FROM "T1" ORDER BY "C1"+"C2"
```

上記の例の場合，ソートキーで指定した値式と同じ値式が選択式に指定されていないため，エラーになります。

5. ソートキーにウィンドウ関数を指定する場合，ソートキーに指定した値式と同じ値式を選択式に指定する必要があります。

■正しい SQL 文の例

```
SELECT SUM("C1") OVER()/100 FROM "T1" ORDER BY SUM("C1") OVER()/100
```

上記の例の場合，ソートキーに指定した値式が選択式にも指定されているため，SQL 文を実行できます。

■エラーになる SQL 文の例

```
SELECT SUM(”C1”) OVER() FROM ”T1” ORDER BY SUM(”C1”) OVER()/100
```

上記の例の場合，ソートキーで指定した値式と同じ値式が選択式に指定されていないため，エラーになります。

6. 次に示す場合は，ソートキーの値式に副問合せまたは？パラメタを指定できません。

- 集合演算を指定している場合
- DISTINCT 指定ありのSELECT 文の場合

■エラーになる SQL 文の例

```
SELECT ”C1”+?,”C2” FROM ”T1” UNION SELECT ”C1”,”C2” FROM ”T2”
    ORDER BY ”C1”+?
```

7. 最も外側の問合せ指定に指定された表参照の表名を，ORDER BY 句中の副問合せから参照することはできません。

■エラーになる SQL 文の例

```
SELECT * FROM ”T1”
    ORDER BY ”C1”,(SELECT ”C1” FROM ”T2” WHERE ”C2”=”T1”.”C2”)+”C2”
```

8. ソートキーに集合関数を指定した場合，次のどちらかの条件を満たす必要があります。

(1)集合関数の修飾問合せに指定している選択式にグループ化列を指定する

(2)ソートキーに指定した値式に含まれる列指定を，グループ化列か，または被集約引数に指定する

■正しい SQL 文の例

```
SELECT ”C1” FROM ”T1” GROUP BY ”C1” ORDER BY AVG(”C2”)
```

上記の例は，(1)の条件を満たしています。

```
SELECT ”C2” FROM ”T1” GROUP BY ”C1”,”C2” ORDER BY SUM(”C1”),”C2”
```

上記の例は，(2)の条件を満たしています。

■エラーになる SQL 文の例

```
SELECT ”C1” FROM ”T1” ORDER BY AVG(”C2”)
```

9. ソートキーと選択式に同じ値式を指定した場合，選択式に指定した値式を使ってソート処理が行われます。ソートキーに指定した値式はソート処理で使われません。

(例)

```
SELECT ”C1”,”C2” FROM ”T1” ORDER BY ”C1”
SELECT ”C1”+”C2”,”C2” FROM ”T1” ORDER BY ”C1”+”C2”
```

上記の例の場合，選択式に指定した値式を使ってソート処理が行われます。

10. ソートキーと選択式に異なる値式を指定した場合，ソートキーに指定した値式を使ってソート処理が行われます。ただし，このとき，ソートキーに指定した値式を選択式に内部的に追加し，ソート処理が行われます。

    （例）

```
SELECT "C1" FROM "T1" ORDER BY "C2"
```

    上記の例の場合，C2 列の値でソート処理が行われて，C1 列の値がソート順に返されます。このとき，ソートキーに指定した値式（C2）を選択式に内部的に追加し，ソート処理が行われます。そのため，問合せ指定中で制限されている値式の規則は，内部的に追加したソートキーの値式にも適用されます。

    ■エラーになる SQL 文の例

```
SELECT MEDIAN("C1"*0.5) FROM "T1"
    ORDER BY PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY "C1"*0.5)
```

    上記の例の場合，下線部分のソートキーは選択式に内部的に追加されますが，逆分布関数の制限（被集約引数に単独の列指定ではない値式を指定した逆分布関数を，同一問合せ指定中に複数指定できない）によってエラーとなります。

7.25.3 ORDER BY 句以外にソート指定リストを指定した場合の規則

- WITHIN グループ指定またはARRAY_AGG 集合関数にソート指定リストを指定した場合の規則
  1. ソートキーに指定した値式中に集合関数は指定できません。
- ウィンドウ順序句にソート指定リストを指定した場合の規則
  1. ソートキーに指定した値式中に集合関数は指定できません。
  2. ソートキーを重複して指定した場合，先に指定した順序付け指定，およびナル値ソート順指定が優先されます。

7.25.4 例題

(1) ORDER BY 句にソート指定リストを指定した例

例題 1（ソートキーに 1 つの列を指定する場合）

顧客表（USERSLIST）の全データを顧客 ID（USERID）順に並べます。

```
SELECT "USERID","NAME","SEX"
    FROM "USERSLIST"
    ORDER BY "USERID" ASC
```

下線部分がソート指定リストの指定です。

例題２（ソートキーに複数の列を指定する場合）

販売履歴表（SALESLIST）の全データを購入日（PUR-DATE）順に並べます。購入日が同じ場合は，顧客ID（USERID）順に並べます。

```
SELECT ”USERID”,”PUR-CODE”,”PUR-NUM”,”PUR-DATE”
    FROM ”SALESLIST”
    ORDER BY ”PUR-DATE” ASC,”USERID” ASC
```

下線部分がソート指定リストの指定です。

例題３（ソートキーに値式を指定する場合）

販売履歴コード（HIS-CODE）の先頭３文字目から８文字分のデータを抽出し，販売履歴表（SALESLIST）の全データを，抽出したデータの順に並べます。

```
SELECT * FROM ”SALESLIST”
    ORDER BY SUBSTR(”HIS-CODE”,3,8) ASC
```

下線部分がソート指定リストの指定です。

例題４（ソートキーにソート項目指定番号を指定する場合）

販売履歴表（SALESLIST）から，商品コード（PUR-CODE）ごとの販売個数（PUR-NUM）の合計を求めます。検索結果は販売個数の合計順に並べます。

```
SELECT ”PUR-CODE”,SUM(”PUR-NUM”)
    FROM ”SALESLIST”
    GROUP BY ”PUR-CODE”
    ORDER BY 2 ASC
```

下線部分がソート指定リストの指定です。

例題５（ナル値ソート順指定を指定する場合）

販売履歴表（SALESLIST）の全データを購入日（PUR-DATE）順に並べます。PUR-DATE 列がナル値の行は先頭にソートします。

```
SELECT ”USERID”,”PUR-CODE”,”PUR-NUM”,”PUR-DATE”
    FROM ”SALESLIST”
    ORDER BY ”PUR-DATE” ASC NULLS FIRST
```

下線部分がソート指定リストの指定です。

(2)　WITHIN グループ指定にソート指定リストを指定した例

例題

給与表（SALARYLIST）から，社員の給料（SALARY）の中央値（50 パーセンタイル）を求めます。

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY ”SALARY”) ”PERCENTILE_CONT”
    FROM ”SALARYLIST”
```

下線部分がソート指定リストの指定です。

(3) ウィンドウ順序句にソート指定リストを指定した例

例題

給与表（SALARYLIST）から，社員の職級（POSITION）ごとに，社員の給料（SALARY）のランクを求めます。

```
SELECT ”EMPID”,”POSITION”,”SALARY”,
        RANK() OVER(PARTITION BY ”POSITION” ORDER BY ”SALARY” DESC) ”RANK”
    FROM ”SALARYLIST”
    ORDER BY ”POSITION”,”SALARY” DESC,”EMPID”
```

下線部分がウィンドウ順序句のソート指定リストの指定です。

(4) ARRAY_AGG 集合関数にソート指定リストを指定した例

例題

売上表（SALES）を検索して，商品ID（PID）を配列要素とする配列データを求めます。求める配列データの条件を次に示します。

- 配列要素の値を単価（PRICE）の昇順に並べる。
- 顧客ID（CID）ごとに配列データを求める。

```
SELECT ”CID”,ARRAY_AGG(”PID” ORDER BY ”PRICE”) AS ”PID_LIST”
  FROM ”SALES”
    GROUP BY ”CID”
```

下線部分がARRAY_AGG 集合関数のソート指定リストの指定です。

7.26　四則演算

ここでは，四則演算の種類と規則について説明します。

7.26.1　四則演算の指定形式および規則

値式中に四則演算を指定できます。

(1)　指定形式

四則演算：：＝｛*項*｜*数値式*＋*項*｜*数値式*－*項*｝

項：：＝｛*値式一次子*｜*数値式*＊*値式一次子*｜*数値式*／*値式一次子*｝

(2)　指定形式の説明

数値式：

*数値式*については，「7.21.1　値式の指定形式および規則」を参照してください。

値式一次子：

*値式一次子*については，「7.21.1　値式の指定形式および規則」を参照してください。

(3)　四則演算の種類

四則演算の種類を次の表に示します。

表 7-38　四則演算の種類

項番	四則演算	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。
3	＊	乗算	第 1 演算項に第 2 演算項を掛けます。
4	／	除算	第 1 演算項を第 2 項演算項で割ります。

例えば，3 ＋ 1 という演算の場合，3 が第 1 演算項，1 が第 2 演算項になります。

(4)　規則

1. 四則演算は，数データ（INTEGER 型，SMALLINT 型，DECIMAL 型，NUMERIC 型，DOUBLE PRECISION 型，またはFLOAT 型のデータ）に対して指定できます。

2. 四則演算は，最大 500 個の演算子（＋，－，＊，／）を使った演算ができます。ただし，演算項に指定した値式中に指定した列が，ビュー表の列，導出表の列，または問合せ名の列の場合，その基となる値式を展開したあとの値式の総数が 10,000 個以内となるようにしてください。
3. 四則演算（＋，－，＊，／）の両側に，？パラメタだけの値式を指定できません。
4. 四則演算に？パラメタを指定した場合，その？パラメタのデータ型には，演算対象となる相手側のデータ型が仮定されます。
5. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
6. 演算項がナル値の場合は，結果もナル値になります。

7.26.2 四則演算の結果のデータ型

四則演算の結果のデータ型は，第 1 演算項と第 2 演算項のデータ型によって決まります。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 7-39 演算項のデータ型と演算結果のデータ型の関係

項番	第 1 演算項のデータ型	第 2 演算項のデータ型	演算結果のデータ型
1	SMALLINT	SMALLINT	INTEGER
2		INTEGER	
3		DECIMAL, NUMERIC	DECIMAL
4		DOUBLE PRECISION, FLOAT	DOUBLE PRECISION
5	INTEGER	SMALLINT	INTEGER
6		INTEGER	
7		DECIMAL, NUMERIC	DECIMAL
8		DOUBLE PRECISION, FLOAT	DOUBLE PRECISION
9	DECIMAL, NUMERIC	SMALLINT	DECIMAL
10		INTEGER	
11		DECIMAL, NUMERIC	
12		DOUBLE PRECISION, FLOAT	DOUBLE PRECISION
13	DOUBLE PRECISION, FLOAT	SMALLINT	DOUBLE PRECISION

項番	第 1 演算項のデータ型	第 2 演算項のデータ型	演算結果のデータ型
14		INTEGER	
15		DECIMAL, NUMERIC	
16		DOUBLE PRECISION, FLOAT	

演算結果のデータ型がDECIMAL 型の場合，精度と位取りは次のようになります。

第 1 演算項がDECIMAL(*p1*,*s1*)またはNUMERIC(*p1*,*s1*)，第 2 演算項がDECIMAL(*p2*,*s2*)または NUMERIC(*p2*,*s2*)，演算結果がDECIMAL(*p*,*s*)とします。

- 加算および減算の場合

  $p = 1 + \mathrm{MAX}\ (p1 - s1,\ p2 - s2) + s$

  $s = \mathrm{MAX}\ (s1,\ s2)$

  p の計算結果が 39 以上の場合，$p = 38$ になります。
- 乗算の場合

  $p = p1 + p2$

  $s = s1 + s2$

  p の計算結果が 39 以上の場合，$p = 38$，$s = \mathrm{MAX}\ (s1,\ s2)$ になります。
- 除算の場合

  $p = 38$

  $s = \mathrm{MAX}\ \{0^{※},\ 38 - (p1 - s1 + s2)\}$

  注※

  サーバ定義またはクライアント定義のadb_sql_prep_dec_div_rs_prior オペランドの指定値がFRACTIONAL_PART の場合は，*s1* になります。

なお，データ型がINTEGER 型の場合は，DECIMAL(20,0)として計算してください。SMALLINT 型の場合は，DECIMAL(10,0)として計算してください。

7.26.3　除算結果のデータ型が DECIMAL 型の場合の留意事項

除算結果のデータ型がDECIMAL 型の場合，除算結果の位取りは，サーバ定義またはクライアント定義の adb_sql_prep_dec_div_rs_prior オペランドの指定値によって決まります。

(1)　実表を検索する場合

次に示す内容の実表T1 を検索する場合，adb_sql_prep_dec_div_rs_prior オペランドの指定値によって実行結果が変わります。

表T1

C1列 DECIMAL(38,4)	C2列 DECIMAL(10,5)
30.5256	0.05223

実行するSELECT 文

```
SELECT ”C1”/”C2” AS ”除算結果” FROM ”T1”
```

- adb_sql_prep_dec_div_rs_prior オペランドにINTEGRAL_PART（省略値）を指定した場合

  ”C1”/”C2”の除算結果のデータ型は，整数部を優先してDECIMAL(38,0)になります。

  除算結果

  ```
  584.
  ```

- adb_sql_prep_dec_div_rs_prior オペランドにFRACTIONAL_PART を指定した場合

  ”C1”/”C2”の除算結果のデータ型は，第 1 演算項の位取りを優先してDECIMAL(38,4)になります。

  除算結果

  ```
  584.4457
  ```

(2) ビュー表を検索する場合

CREATE VIEW 文の問合せ式本体に指定された除算（四則演算）の結果のデータ型がDECIMAL 型の場合，その精度と位取りは，サーバ定義またはクライアント定義のadb_sql_prep_dec_div_rs_prior オペランドの指定値（ビュー表を検索したときの指定値）によって決まります。

（例）

表T1 の内容は次のとおりとします。

表T1

C1列 DECIMAL(38,1)	C2列 DECIMAL(2,1)
10.1	0.5

ビュー表の定義

```
CREATE VIEW ”VT1”(”VC1”) AS SELECT ”C1”/”C2” FROM ”T1”
```

- adb_sql_prep_dec_div_rs_prior オペランドにINTEGRAL_PART を指定して，次のSELECT 文を実行した場合

  ```
  SELECT ”VC1” FROM ”VT1”
  ```

  検索結果

  ```
  20
  ```

このとき，ビュー表VT1 のVC1 列のデータ型はDECIMAL(38,0)になります。

- adb_sql_prep_dec_div_rs_prior オペランドにFRACTIONAL_PART を指定して，次のSELECT 文を実行した場合

```
SELECT ”VC1” FROM ”VT1”
```

検索結果

```
20.2
```

このとき，ビュー表VT1 のVC1 列のデータ型はDECIMAL(38,1)になります。

メモ

SQL_COLUMNS 表に格納されているビュー表の列情報には，adb_sql_prep_dec_div_rs_prior オペランドの指定値（ビュー表を定義したときの指定値）によって決定されたデータ型が格納されます。そのため，ビュー表の定義時とビュー表の検索時で，adb_sql_prep_dec_div_rs_prior オペランドの指定値が異なると，次の 2 つのデータ型が不一致になることがあります。

- ビュー表を等価変換した内部導出表の導出列のデータ型
- SQL_COLUMNS 表に格納されているビュー表の列のデータ型

なお，除算の第 1 演算項および第 2 演算項が定数の場合，除算結果の精度と位取りは，ビュー表を定義したときのadb_sql_prep_dec_div_rs_prior オペランドの指定値によって決定されます。

7.27　連結演算

連結演算を使用して 2 つの文字データ，または 2 つのバイナリデータを連結できます。ここでは，連結演算の種類と規則について説明します。

7.27.1　連結演算の指定形式および規則

値式中に連結演算を指定できます。

(1)　指定形式

```
連結演算 ： ： ＝ {値式一次子
                 | 文字値式 ＋値式一次子
                 | 文字値式 | |値式一次子
                 | バイナリ値式 ＋値式一次子
                 | バイナリ値式 | |値式一次子}
```

(2)　指定形式の説明

値式一次子：

*値式一次子*については，「7.21.1　値式の指定形式および規則」を参照してください。

文字値式：

*文字値式*については，「7.21.1　値式の指定形式および規則」を参照してください。

バイナリ値式：

*バイナリ値式*については，「7.21.1　値式の指定形式および規則」を参照してください。

(3)　連結演算の種類

連結演算の種類を次の表に示します。

表 7-40　連結演算の種類

項番	連結演算	機能
1	＋	第 1 演算項と第 2 演算項を連結します。
2	||	

例えば，'ABC'＋'DEF'という演算の場合，'ABC'が第 1 演算項，'DEF'が第 2 演算項になります。

(4)　規則

1. 第 1 演算項および第 2 演算項には，文字データまたはバイナリデータを指定してください。

2. 第 1 演算項および第 2 演算項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 7-41　連結演算の第 1 演算項および第 2 演算項に指定できるデータ型の組み合わせ

第 1 演算項のデータ型	第 2 演算項のデータ型			
	CHAR	VARCHAR	BINARY	VARBINARY
CHAR	○	○	×	×
VARCHAR	○	○	×	×
BINARY	×	×	○	○
VARBINARY	×	×	○	○

（凡例）

○：指定できます。

×：指定できません。

3. 連結演算は，最大 500 個の演算子（+，||）を使った演算ができます。ただし，演算項に指定した値式中に指定した列が，ビュー表の列，導出表の列，または問合せ名の列の場合，その基となる値式を展開したあとの値式の総数が 10,000 個以内となるようにしてください。

4. 第 1 演算項または第 2 演算項には，？パラメタを単独で指定できません。

5. 連結演算の結果の値は，非ナル値制約なし（ナル値を許す）となります。

6. 第 1 演算項または第 2 演算項のどちらかがナル値の場合，連結演算の結果はナル値となります。

7. 連結演算の結果，最大長が 32,000 バイトを超える文字データまたはバイナリデータを連結することはできません。

8. CHAR のデータの末尾に半角空白が格納されている場合，その半角空白も連結の対象になります。

（例）

C1 列がCHAR(5)で値が'ABC△△'，C2 列がVARCHAR(10)で値が'XYZ'の場合，次のように連結されます。

"C1"＋"C2" → 'ABC△△XYZ'

"C2"＋"C1" → 'XYZABC△△'

（凡例）△：半角空白

(5)　例題

例題 1（文字データを連結する場合）

表T1 のC2 列の文字データとC3 列の文字データを連結し，連結結果が'efg03v03'の行を検索します。

```
SELECT * FROM "T1"
    WHERE "C2"||"C3"='efg03v03'
```

表T1

C1列 CHAR	C2列 VARCHAR	C3列 VARCHAR
A10101	abc010587	rs3354
A15014	efg03	v03
A31399	hijk99842688	wxyz22725

検索結果

A15014	efg03	v03

例題 2（バイナリデータを連結する場合）

表T1 のC2 列のバイナリデータとC3 列のバイナリデータを連結し，連結結果がX'ABC1230000DEF456'の行を検索します。

```
SELECT * FROM "T1"
    WHERE "C2"||"C3"=X'ABC1230000DEF456'
```

表T1

C1列 CHAR(6)	C2列 BINARY(5)	C3列 VARBINARY(10)
A10101	X'0000000000'	X'0000'
A15014	X'ABC1230000'	X'DEF456'
A31399	X'1111111111'	X'DEF4'

検索結果

A15014	X'ABC1230000'	X'DEF456'

7.27.2 連結演算の結果のデータ型

連結演算の結果のデータ型は，第 1 演算項と第 2 演算項のデータ型によって決まります。

(1) 演算項のデータ型が文字データの場合

演算項のデータ型と演算結果のデータ型の関係（演算項のデータ型が文字データの場合）を次の表に示します。

表 7-42 演算項のデータ型と演算結果のデータ型の関係（演算項のデータ型が文字データの場合）

項番	第 1 演算項のデータ型とデータ長	第 2 演算項のデータ型とデータ長	演算結果のデータ型とデータ長
1	CHAR(*m*)	CHAR(*n*)	CHAR(*m* + *n*)
2		VARCHAR(*n*) 実データ長：*L2*	VARCHAR(*m* + *n*) 実データ長：*m* + *L2*
3	VARCHAR(*m*) 実データ長：*L1*	CHAR(*n*)	VARCHAR(*m* + *n*) 実データ長：*L1* + *n*
4		VARCHAR(*n*) 実データ長：*L2*	VARCHAR(*m* + *n*) 実データ長：*L1* + *L2*

（凡例）

m：第 1 演算項のデータの最大長

n：第 2 演算項のデータの最大長

L1：第 1 演算項のデータの実データ長

L2：第 2 演算項のデータの実データ長

(2) 演算項のデータ型がバイナリデータの場合

演算項のデータ型と演算結果のデータ型の関係（演算項のデータ型がバイナリデータの場合）を次の表に示します。

表 7-43 演算項のデータ型と演算結果のデータ型の関係（演算項のデータ型がバイナリデータの場合）

項番	第 1 演算項のデータ型とデータ長	第 2 演算項のデータ型とデータ長	演算結果のデータ型とデータ長
1	BINARY(*m*)	BINARY(*n*)	BINARY(*m*＋*n*)
2		VARBINARY(*n*) 実データ長：*L2*	VARBINARY(*m*＋*n*) 実データ長：*m*＋*L2*
3	VARBINARY(*m*) 実データ長：*L1*	BINARY(*n*)	VARBINARY(*m*＋*n*) 実データ長：*L1*＋*n*
4		VARBINARY(*n*) 実データ長：*L2*	VARBINARY(*m*＋*n*) 実データ長：*L1*＋*L2*

（凡例）

m：第 1 演算項のデータの最大長

n：第 2 演算項のデータの最大長

L1：第 1 演算項のデータの実データ長

L2：第 2 演算項のデータの実データ長

7.28　日時演算

ここでは，日時演算の種類と規則について説明します。

7.28.1　日時演算の指定形式および規則

値式中に日時演算を指定して，日時データの演算を使用した検索ができます。

(1)　指定形式

```
日時演算：：＝｛値式一次子
　　　　　　｜日時値式＋ラベル付き間隔　〔｛*｜/｝値式一次子〕
　　　　　　｜日時値式-ラベル付き間隔　〔｛*｜/｝値式一次子〕｝
```

(2)　指定形式の説明

値式一次子：

*値式一次子*については，「7.21.1　値式の指定形式および規則」を参照してください。

日時値式：

*日時値式*については，「7.21.1　値式の指定形式および規則」を参照してください。

ラベル付き間隔：

ラベル付き間隔については，「7.29　ラベル付き間隔」を参照してください。

(3)　日時演算ができるデータ型

DATE 型，TIME 型，およびTIMESTAMP 型のデータに対して日時演算を実行できます。

なお，日付を表す既定の入力表現，時刻を表す既定の入力表現，または時刻印を表す既定の入力表現の形式に従っている文字列定数（CHAR，VARCHAR）に対しても日時演算を実行できます。文字列定数が指定された場合，その文字列定数を日時データに変換して日時演算が行われます。

日付を表す既定の入力表現，時刻を表す既定の入力表現，および時刻印を表す既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。

(4)　規則

(a)　共通の規則

1. DATE 型の演算を実行した場合は，演算結果のデータ型もDATE 型になります。
2. TIME 型の演算を実行した場合は，演算結果のデータ型もTIME 型になります。
3. TIMESTAMP 型の演算を実行した場合は，演算結果のデータ型もTIMESTAMP 型になります。

4. 最大 500 個の演算子（＋または-）を使った日時演算ができます。ただし，演算項に指定した値式中に指定した列が，ビュー表の列，導出表の列，または問合せ名の列の場合，その基となる値式を展開したあとの値式の総数が 10,000 個以内となるようにしてください。
5. 演算子（＋または-）の左側に，？パラメタだけの値式を指定できません。
6. 演算結果の値は，非ナル値制約なし（ナル値を許す）となります。
7. 演算項がナル値の場合は，演算結果もナル値になります。
8. 日時演算をする場合は，上記の規則以外に「7.21　値式」の規則も確認してください。

(b)　DATE 型の日時演算を実行する場合の規則

1. 演算結果の範囲は，0001 年 01 月 01 日～9999 年 12 月 31 日の間にする必要があります。
2. 年または月を繰り越して演算されます。例を次に示します。

   （例 1）

   ```
   DATE'2012-12-31'+2 DAY → DATE'2013-01-02'
   ```

   （例 2）

   ```
   DATE'2013-01-01'-1 DAY → DATE'2012-12-31'
   ```

3. 年，月の演算結果が存在しない日付（小の月の 31 日，2 月 30 日，およびうるう年以外の年の 2 月 29 日）の場合，その月の最終日に修正されます。例を次に示します。

   （例）

   ```
   DATE'2013-03-31'+1 MONTH → DATE'2013-04-30'
   ```

   このように，演算結果が存在しない日付の場合，その月の最終日に自動的に修正されます。そのため，ある日付に任意の月数を加算し，その結果の日付から同じ月数を引いた場合，必ずしも元の日付に戻りません。例を次に示します。

   （例）

   ```
   DATE'2013-03-31'+1 MONTH → DATE'2013-04-30'
   DATE'2013-04-30'-1 MONTH → DATE'2013-03-30'
   ```

(c)　TIME 型の日時演算を実行する場合の規則

1. 演算結果の範囲は，0 時 0 分 0.000000000000 秒～23 時 59 分 59.999999999999 秒の間にする必要があります。
2. 小数秒精度が異なる演算の場合，精度が高い方に合わせます（精度が低い方に 0 を補います）。例えば，演算対象のTIME 型のデータの小数秒精度が 0 の場合に，ラベル付き間隔にMILLISECONDS を指定したときは，TIME 型のデータの小数秒精度を 3 に拡張して演算します。

(d)　TIMESTAMP 型の日時演算を実行する場合の規則

1. 演算結果の範囲は，0001 年 01 月 01 日 0 時 0 分 0.000000000000 秒～9999 年 12 月 31 日 23 時 59 分 59.999999999999 秒の間にする必要があります。
2. 年，月，日の部分の計算方法は，「(b)　DATE 型の日時演算を実行する場合の規則」に従います。
3. 小数秒精度が異なる演算の場合，精度が高い方に合わせます（精度が低い方に 0 を補います）。例えば，演算対象のTIMESTAMP 型のデータの小数秒精度が 0 の場合に，ラベル付き間隔にMILLISECONDS を指定したときは，TIMESTAMP 型のデータの小数秒精度を 3 に拡張して演算します。
4. 日を繰り越して演算されます。例を次に示します。

   （例 1）

```
TIMESTAMP'2014-02-01 23:59:59'+1 SECOND → TIMESTAMP'2014-02-02 00:00:00'
```

   （例 2）

```
TIMESTAMP'2014-02-02 00:00:00'-1 SECOND → TIMESTAMP'2014-02-01 23:59:59'
```

   （例 3）

```
TIMESTAMP'2013-12-31 23:05:06'+2 HOUR → TIMESTAMP'2014-01-01 01:05:06'
```

(e)　ラベル付き間隔に乗除算を指定した場合の規則

1. ラベル付き間隔に乗除算を指定した場合，次のラベル付き間隔と等価になります。
   - *値式 1 ラベル付き間隔修飾子* * *値式 2* → （*値式 1*値式 2*）*ラベル付き間隔修飾子*
   - *値式 1 ラベル付き間隔修飾子* / *値式 2* → （*値式 1/値式 2*）*ラベル付き間隔修飾子*

   （例）

```
C1 DAYS * C2 → (C1*C2) DAYS
(C1+C2) MINUTES / (C3+C4) → ((C1+C2)/(C3+C4)) MINUTES
```

2. ラベル付き間隔の乗除算に指定する*値式一次子*には，整数（SMALLINT 型またはINTEGER 型）を指定してください。
3. ラベル付き間隔の乗除算の*値式一次子*に？パラメタを単独で指定した場合，？パラメタのデータ型にINTEGER 型が仮定されます。
4. ラベル付き間隔の乗除算の*値式一次子*の結果がナル値の場合，ラベル付き間隔の結果はナル値になります。

(5)　例題

例題

社員表（EMPLIST）を検索して，入社日（ENT-DAY）から 2 年以上経過した社員 ID（USERID）および社員名（NAME）を検索します。

```
SELECT "USERID","NAME" FROM "EMPLIST"
    WHERE "ENT-DAY" <= CURRENT_DATE -2 YEARS
```

下線部分が日時演算の指定で，「2 YEARS」がラベル付き間隔の指定です。

7.29　ラベル付き間隔

ここでは，ラベル付き間隔について説明します。

7.29.1　ラベル付き間隔の指定形式および規則

ラベル付き間隔は，日時演算をする場合に使用し，数値とそれに続く間隔キーワード（YEAR，MONTH，DAY，HOUR，MINUTE，SECOND，MILLISECOND，MICROSECOND，NANOSECOND，PICOSECOND）で特定の時間間隔を表します。ラベル付き間隔は，日時データに対する加減算の第 2 演算項，およびウィンドウ枠句に指定できます。

(1)　指定形式

```
ラベル付き間隔：：＝値式一次子　ラベル付き間隔修飾子

  ラベル付き間隔修飾子：：＝ {YEAR〔S〕 |MONTH〔S〕 |DAY〔S〕
                            |HOUR〔S〕 |MINUTE〔S〕 |SECOND〔S〕
                            |MILLISECOND〔S〕 |MICROSECOND〔S〕
                            |NANOSECOND〔S〕 |PICOSECOND〔S〕}
```

(2)　指定形式の説明

値式一次子：

*値式一次子*には，SMALLINT 型，INTEGER 型のデータを指定してください。値式一次子については，「7.21.1　値式の指定形式および規則」の「(1)　指定形式」を参照してください。

ラベル付き間隔修飾子：

```
ラベル付き間隔修飾子：：＝ {YEAR〔S〕 |MONTH〔S〕 |DAY〔S〕
                          |HOUR〔S〕 |MINUTE〔S〕 |SECOND〔S〕
                          |MILLISECOND〔S〕 |MICROSECOND〔S〕
                          |NANOSECOND〔S〕 |PICOSECOND〔S〕}
```

次のどれかを指定します。

YEAR〔S〕：

年単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-9,998～9,998 になります。※

MONTH〔S〕：

月単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-119,987～119,987 になります。※

DAY〔S〕：

日単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-3,652,058～3,652,058 になります。※

HOUR〔S〕：

時単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-87,649,415～87,649,415 になります。※

MINUTE〔S〕：

分単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-5,258,964,959～5,258,964,959 になります。※

SECOND〔S〕：

秒単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-315,537,897,599～315,537,897,599 になります。※

MILLISECOND〔S〕：

ミリ秒単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-315,537,897,599,999～315,537,897,599,999 になります。※

MICROSECOND〔S〕：

マイクロ秒単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-315,537,897,599,999,999～315,537,897,599,999,999 になります。※

NANOSECOND〔S〕：

ナノ秒単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-9,223,372,036,854,775,807～9,223,372,036,854,775,807 になります。※

PICOSECOND〔S〕：

ピコ秒単位の間隔を表します。

*値式一次子*に指定できる数データの値の範囲は，-9,223,372,036,854,775,807～9,223,372,036,854,775,807 になります。※

注※

日時演算でラベル付き間隔に乗算を指定した場合は，*値式一次子*の積の値の範囲になります。例えば，”C1” DAYS *”C2”と指定した場合，(C1*C2)で指定できる範囲が，-3,652,058～3,652,058 になります。

日時演算でラベル付き間隔に乗除算を指定した場合の規則については，「7.28.1　日時演算の指定形式および規則」の「(4)　規則」の「(e)　ラベル付き間隔に乗除算を指定した場合の規則」を参照してください。

(3)　規則

1. 日時演算を行うデータ型によって，指定できるラベル付き間隔修飾子が次の表に示すように異なります。

表 7-44　指定できるラベル付き間隔修飾子

ラベル付き間隔修飾子	日時演算を行うデータ型		
	DATE	TIME	TIMESTAMP
YEAR	○	×	○
MONTH	○	×	○
DAY	○	×	○
HOUR	×	○	○
MINUTE	×	○	○
SECOND	×	○	○
MILLISECOND	×	○	○
MICROSECOND	×	○	○
NANOSECOND	×	○	○
PICOSECOND	×	○	○

（凡例）

○：指定できます。

×：指定できません。

2. *値式一次子*に？パラメタを単独で指定した場合，？パラメタのデータ型にはINTEGER が仮定されます。

3. ラベル付き間隔の結果の値は，非ナル値制約なし（ナル値を許す）となります。

4. *値式一次子*の結果がナル値の場合，ラベル付き間隔の結果はナル値になります。

5. YEARS，MONTHS，DAYS，HOURS，MINUTES，SECONDS，MILLISECONDS，MICROSECONDS，NANOSECONDS，PICOSECONDS の末尾のS の指定は省略できます。指定例を次に示します。

（例）1 年を指定する場合

1 YEAR または1 YEARS

6. ウィンドウ関数のウィンドウ枠値指定に指定するラベル付き間隔は，*値式一次子*に値指定だけが指定できます。

7. ラベル付き間隔修飾子を指定した場合に仮定される小数秒精度を次の表に示します。

表 7-45　仮定される小数秒精度

指定したラベル付き間隔修飾子	仮定される小数秒精度
MILLISECOND	3
MICROSECOND	6
NANOSECOND	9
PICOSECOND	12

7.30　CASE 式

ここでは，CASE 式について説明します。

7.30.1　CASE 式の指定形式および規則

指定した探索条件に該当する場合，指定した値式の結果を返します。

(1)　指定形式

```
CASE式 : : =
      CASE
          {WHEN 探索条件 THEN  {値式 | NULL} } …
          〔ELSE  {値式 | NULL} 〕
      END
```

(2)　指定形式の説明

WHEN *探索条件*：

探索条件を指定します。探索条件については「7.19　探索条件」を参照してください。指定した探索条件に該当した場合，THEN 以降に指定した値が結果として返ります。

なお，1 つのCASE 式中にWHEN を 255 個まで指定できます。

THEN {*値式* | NULL} ：

指定した探索条件に該当する場合に，結果として返す値を値式の形式で指定します。ナル値を返したい場合は，NULL を指定します。

ELSE {*値式* | NULL} ：

WHEN で指定したすべての探索条件に該当しなかった場合に，結果として返す値を値式の形式で指定します。ナル値を返したい場合は，NULL を指定します。

ELSE の指定を省略した場合，ELSE にはNULL が指定されたと仮定されます。

(3)　規則

1. CASE 式中にWHEN を複数指定し，2 つ以上の探索条件が真となった場合，探索条件が真となった最初のWHEN の結果がCASE 式の結果となります。
2. THEN およびELSE に指定する値式の結果のデータ型は，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，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

   ただし，次のデータ型は比較できません。

   - 日付データと文字データ（日付を表す既定の入力表現）
   - 時刻データと文字データ（時刻を表す既定の入力表現）

- 時刻印データと文字データ（時刻印を表す既定の入力表現）

既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。

3. CASE 式の結果のデータ型およびデータ長は，THEN およびELSE に指定した各値式の結果のデータ型およびデータ長によって決まります。詳細については，「7.21.2　値式の結果のデータ型」を参照してください。
4. THEN またはELSE に，少なくとも 1 つは値式を指定してください。次のようにすべての値式にNULL を指定することはできません。

   （例）

```
CASE
    WHEN "C1"=100 THEN NULL
    ELSE NULL
END
```

5. CASE，THEN，およびELSE に指定する値式に，？パラメタを単独で指定できません。
6. THEN およびELSE の値式には，配列データを指定できません。
7. 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。

(4)　例題

例題 1

次に示すように表T1 の検索を行います。

- C1 列が200 の場合：C2 列に 20 を加算した値を検索結果とする
- C1 列が100 の場合：C2 列に 10 を加算した値を検索結果とする
- C1 列が100 および200 以外の値の場合：C2 列に 5 を加算した値を検索結果とする

```
SELECT "C1","C2",CASE WHEN "C1"=200 THEN "C2"+20
                      WHEN "C1"=100 THEN "C2"+10
                      ELSE "C2"+5
                 END AS "CASE"
    FROM "T1"
```

C1	C2	CASE
200	60	80
100	40	50
80	30	35

CASE式の結果

例題 2

社員表（EMPLIST）を検索して次に示す値を求めます。

- 部署コード（SCODE）ごとに，所属する男性と女性の人数を求める

```
SELECT "SCODE",SUM(CASE WHEN "SEX"='M' THEN 1 ELSE 0 END) AS "Men",
        SUM(CASE WHEN "SEX"='F' THEN 1 ELSE 0 END) AS "Women"
```

```
    FROM "EMPLIST"
    GROUP BY "SCODE"
```

SCODE	Men	Women
S001	12	5
S002	21	18
S003	19	33

例題 3

新商品表（PRODUCTLIST_NEW）に，商品表（PRODUCTLIST）の行を挿入します。行を挿入する際，商品価格（PRICE）を次に示すように新価格に変更します。

- 商品コード（PCODE）がP001 の場合：商品価格を 10%引きにする
- 商品コードがP002 の場合：商品価格を 20%引きにする
- 上記以外の場合：商品価格を 30%引きにする

```
INSERT INTO "PRODUCTLIST_NEW"("PCODE","PRICE")
    SELECT "PCODE",CASE WHEN "PCODE"='P001' THEN "PRICE"*0.9
                        WHEN "PCODE"='P002' THEN "PRICE"*0.8
                        ELSE "PRICE"*0.7
                   END
    FROM "PRODUCTLIST"
```

商品表（PRODUCTLIST）

PCODE	PRICE
P001	100
P002	200
P003	300

新商品表（PRODUCTLIST_NEW）

PCODE	PRICE
P001	90
P002	160
P003	210

7.31　配列要素参照

ここでは，配列要素参照について説明します。

7.31.1　配列要素参照の指定形式および規則

配列要素参照は，配列データ中の配列要素を参照する際に使用します。参照する配列要素を添え字で指定します。

(1)　指定形式

```
配列要素参照：：＝配列値式[添え字]

  配列値式：：＝値式一次子
  添え字：：＝｛符号なし整数定数｜ANY〔(識別番号)〕｝
  識別番号：：＝符号なし整数定数
```

(2)　指定形式の説明

配列値式：

配列値式には，配列データを値式一次子の形式で指定します。値式一次子については，「7.21　値式」を参照してください。

指定規則を次に示します。

- 配列値式には，配列データを指定してください。
- 配列値式には，？パラメタを指定できません。

添え字：

添え字を次のどちらかの形式で指定します。

- 符号なし整数定数
- ANY〔(*識別番号*)〕

符号なし整数定数：

配列値式に指定した配列データの要素番号を指定します。

(例)

```
"C1"[2]
```

下線部分が添え字で，符号なし整数定数を指定しています。この場合，C1 列（配列型の列）の要素番号が 2 である配列要素を参照します。添え字に符号なし整数定数を指定した場合の例については，「(4)　例題」を参照してください。

指定規則を次に示します。

- 添え字には，1～30,000 の符号なし整数定数を指定します。

- 添え字には，配列値式に指定した配列データの最大要素数よりも大きい値を指定できません。

ANY〔(*識別番号*)〕：

ANY を指定した配列要素参照は，配列値式に指定した配列データ中の任意の配列要素を参照する場合に指定します。

指定規則を次に示します。

- ANY を指定した配列要素参照は，WHERE 句の述語中にだけ指定できます。
- ANY を指定した配列要素参照を指定した条件式が評価対象にできる表は，1 つの表だけです。

識別番号：

複数の配列要素参照を 1 つの組と見なして述語の評価を行いたい場合，各配列要素参照に同じ識別番号を指定します。識別番号を指定した場合の例については，「(4)　例題」を参照してください。

指定規則を次に示します。

- 識別番号には，1～255 の符号なし整数定数を指定します。
- 識別番号を省略した配列要素参照に対しては，WHERE 句中に指定したほかの配列要素参照の識別番号と重複しない識別番号を HADB が割り当てます。

  (例)

  ”C1”[ANY]=1 AND ”C2”[ANY]=2 と指定した場合，"C1”[ANY(1)]=1 AND ”C2”[ANY(2)]=2 となります。

  ”C1”[ANY]=”C2”[ANY]と指定した場合，”C1”[ANY(1)]=”C2”[ANY(2)]となります。

(3)　規則

■共通規則

1. 配列値式がナル値の場合，配列要素参照の結果はナル値になります。
2. 配列値式が空の配列データの場合，配列要素参照の結果はナル値になります。
3. 配列値式の結果の要素データ型が，配列要素参照の結果のデータ型になります。
4. 配列要素参照は，行値構成子の行値構成子要素の値式には指定できません。

■添え字に符号なし整数定数を指定した場合の規則

1. 配列要素参照の結果は，配列値式に指定した配列データ中の，添え字に指定した値と同じ要素番号を持つ配列要素の値になります。
2. 配列値式に指定した配列データの配列要素数よりも大きい値を添え字に指定した場合，配列要素参照の結果はナル値になります。

■添え字に ANY を指定した場合（ANY(識別番号)の場合も含む）の規則

1. ANY を指定した配列要素参照は，WHERE 句に指定する述語のうち，次の個所に指定できます。
   - NULL 述語の値式
   - IN 述語の左側の値式

ただし，IN 述語の右側に表副問合せを指定した場合，IN 述語の左側の値式には，ANY を指定した配列要素参照を指定できません。

- BETWEEN 述語の左側の値式 1
- 比較述語の左側または右側の値式
- LIKE 述語の一致値
- LIKE_REGEX 述語の一致値

2. ANY を指定した配列要素参照は，CASE 式の探索条件中には指定できません。
3. ANY を指定した配列要素参照は，次の値式には指定できません。
   - CASE 式のTHEN に指定する値式
   - CASE 式のELSE に指定する値式
   - スカラ関数DECODE の返却値または既定返却値に指定する値式
   - スカラ関数LTDECODE の返却値または既定返却値に指定する値式
4. 識別番号が同じかどうかは，WHERE 句ごとに判定されます。

   （例）

```
SELECT ”T1”.”C2” FROM ”T1” WHERE ”T1”.”C1”[ANY(1)] = 1
UNION ALL
SELECT ”T2”.”C2” FROM ”T2” WHERE ”T2”.”C1”[ANY(1)] = 2
```

   上記のSELECT 文には，識別番号 1 が 2 つ指定されていますが，識別番号が同じかどうかは，WHERE 句ごとに判定されます。そのため，上記の場合は，識別番号が同じであるとは見なされません（SELECT 文はエラーになりません）。

5. ANY を指定した配列要素参照を含む述語には，異なる表の列を指定できません。

   （例）エラーになる SQL 文の例

```
SELECT * FROM ”T1”,”T2” WHERE ”T1”.”C1”[ANY(1)] = ”T2”.”C1”
```

```
SELECT * FROM ”T1”,”T2” WHERE ”T1”.”C1”[ANY] = ”T2”.”C1”
```

   上記のSELECT 文は，ANY を指定した配列要素参照を下線部分の述語に指定しています。異なる表（表T1 と表T2）の列を指定しているため，上記のSELECT 文はエラーになります。

6. 同じ識別番号を指定した配列要素参照の列指定には，異なる表の列を指定できません。

   （例）エラーになる SQL 文の例

```
SELECT * FROM ”T1”,”T2”
  WHERE ”T1”.”C1”[ANY(1)] = 1 AND ”T2”.”C1”[ANY(1)] = 2
```

   下線部分が，配列要素参照の列指定です。両方の識別番号には同じ 1 を指定していますが，列指定には異なる表（表T1 と表T2）の列を指定しているため，上記のSELECT 文はエラーになります。

7. ANY を指定した配列要素参照を含む述語には，スカラ副問合せを指定できません。

   （例）エラーになる SQL 文の例

```
SELECT "T1"."C1","T1"."C2" FROM "T1"
  WHERE "T1"."C1"[ANY] = (SELECT "T2"."C1" FROM "T2" WHERE "T2"."C2" = 10)
```

```
SELECT "T1"."C1","T1"."C2" FROM "T1"
  WHERE "T1"."C1"[ANY(1)] = (SELECT "T2"."C1" FROM "T2" WHERE "T2"."C2" = "T1"."C2")
```

下線部分が，ANY を指定した配列要素参照を含む述語です。この述語中にスカラ副問合せを指定しているため，上記のSELECT 文はエラーになります。

なお，ANY を指定した配列要素参照を含む述語にスカラ副問合せを指定したい場合は，集まり導出表を指定した SQL 文に書き換えることを検討してください。上記の例を，集まり導出表を指定したSELECT 文に書き換えた例を次に示します。

（例）書き換えの例

```
SELECT "T1"."C1","T1"."C2" FROM "T1",UNNEST("T1"."C1") "DT"
  WHERE "DT"."C1" = (SELECT "T2"."C1" FROM "T2" WHERE "T2"."C2" = 10)
```

```
SELECT "T1"."C1","T1"."C2" FROM "T1",UNNEST("T1"."C1") "DT"
  WHERE "DT"."C1" = (SELECT "T2"."C1" FROM "T2" WHERE "T2"."C2" = "T1"."C2")
```

下線部分が，集まり導出表の指定です。

8. SQL 文中に指定できる配列要素参照の識別番号の種類の数は，255 が上限です。

   なお，SQL 文中にビュー表を指定している場合は，そのビュー表が導出表に等価変換されたあとに，識別番号の種類の数のチェックが行われます。

9. 問合せ指定のWHERE 句に指定した配列要素参照の識別番号は，それぞれ 1 から連番になるようにHADB が自動的に採番し直します。

10. 導出問合せ中にANY を指定した配列要素参照を含む内部導出表を SQL 文中に指定し，その内部導出表が展開される場合，内部導出表の導出問合せ中の配列要素参照の識別番号は，内部導出表を指定した問合せ指定に指定した配列要素参照の識別番号と重複しないように HADB が自動的に採番し直します。

11. ANY を指定した配列要素参照を含む探索条件の結果は，次のように求められます。

■複数の述語に同じ識別番号の配列要素参照を指定している場合

同じ識別番号の配列要素参照を指定した複数の述語を含む探索条件に対して結果が求められます。探索条件の結果は次のように求められます。

- 探索条件の結果を求める際，探索条件に含まれる同じ識別番号を指定した複数の配列要素参照を，同じ添え字の配列要素ごとに 1 組と見なします。そして，すべての添え字（複数の配列要素参照の配列値式の中で最も多い配列要素数分）に対してそれぞれ評価が行われます。その際，添え字が配列要素参照の配列値式の配列要素数を超える場合，超えた添え字に対応する値にナル値が補完されて評価が行われます。探索条件の結果は，次の表の内容に従って決定されます。

項番	条件	探索条件の結果
1	探索条件の結果が真となる組が少なくとも 1 つ存在する場合	真

項番	条件	探索条件の結果
2	項番 1 の条件を満たさない場合で，かつ探索条件の結果が不定となる組が少なくとも 1 つ存在する場合	不定※
3	項番 1 と項番 2 の条件を満たさない場合	偽

注※　NULL 述語の場合，項番 2 の条件は除外されます。したがって，項番 1 の条件を満たさない場合，探索条件の結果は偽になります。

ANY を指定した配列要素参照を含む探索条件の結果は，配列要素の組ごとに評価が行われる探索条件をOR 条件でつないだ探索条件の結果と同じになります。例を次に示します。

(例)

C1 列は，配列要素数が 2 の配列型の列とします。C2 列は，配列要素数が 3 の配列型の列とします。

同じ識別番号の配列要素参照を指定した複数の述語を含む探索条件	左記の指定と同じ結果になる探索条件
”C1”[ANY(1)] = 1 AND ”C2”[ANY(1)] = 2	(”C1”[1] = 1 AND ”C2”[1] = 2) OR (”C1”[2] = 1 AND ”C2”[2] = 2) OR (”C1”[3] = 1 AND ”C2”[3] = 2) 添え字が配列値式の配列要素数を超える”C1”[3]には，ナル値が補完されて評価が行われます。

■複数の述語に同じ識別番号の配列要素参照を指定していない場合

各述語に対して結果が求められます。述語の結果は次のように求められます。

- 述語の結果を求める際，述語中の 1 か所だけに指定した配列要素参照は，それぞれの配列要素に対して評価されます。述語の結果は，次の表の内容に従って決定されます。

項番	条件	述語の結果
1	述語の結果が真となる配列要素が少なくとも 1 つ存在する場合	真
2	項番 1 の条件を満たさない場合で，かつ述語の結果が不定となる配列要素が少なくとも 1 つ存在する場合	不定※
3	項番 1 と項番 2 の条件を満たさない場合	偽

注※　NULL 述語の場合，項番 2 の条件は除外されます。したがって，項番 1 の条件を満たさない場合，述語の結果は偽になります。

ANY を指定した配列要素参照を含む述語の結果は，配列要素ごとに評価が行われる述語をOR 条件でつないだ探索条件の結果と同じになります。例を次に示します。

(例)

C1 列は，配列要素数が 3 の配列型の列とします。

ANY を指定した配列要素参照を含む述語	左記の指定と同じ結果になる探索条件
”C1”[ANY] = 1	”C1”[1] = 1 OR ”C1”[2] = 1 OR ”C1”[3] = 1

- 述語の結果を求める際，同じ識別番号を指定した複数の配列要素参照を，同じ添え字の配列要素ごとに 1 組と見なします。そして，すべての添え字（複数の配列要素参照の配列値式の中で最も多い配列要素数分）に対してそれぞれ評価が行われます。その際，添え字が配列要素参照の配列値式の配列要素数を超える場合，超えた添え字に対応する値にナル値を補完して評価が行われます。述語の結果は，次の表の内容に従って決定されます。

項番	条件	述語の結果
1	述語の結果が真となる組が少なくとも 1 つ存在する場合	真
2	項番 1 の条件を満たさない場合で，かつ述語の結果が不定となる組が少なくとも 1 つ存在する場合	不定※
3	項番 1 と項番 2 の条件を満たさない場合	偽

注※　NULL 述語の場合，項番 2 の条件は除外されます。したがって，項番 1 の条件を満たさない場合，述語の結果は偽になります。

ANY を指定した配列要素参照を含む述語の結果は，配列要素の組ごとに評価が行われる述語をOR 条件でつないだ探索条件の結果と同じになります。例を次に示します。

(例)

C1 列は，配列要素数が 2 の配列型の列とします。C2 列は，配列要素数が 3 の配列型の列とします。

ANY を指定した配列要素参照を含む述語	左記の指定と同じ結果になる探索条件
"C1"[ANY(1)] = "C2"[ANY(1)]	"C1"[1] = "C2"[1] OR "C1"[2] = "C2"[2] OR "C1"[3] = "C2"[3] 添え字が配列値式の配列要素数を超える"C1"[3]には，ナル値が補完されて評価が行われます。

- 述語の結果を求める際，異なる識別番号を指定した複数の配列要素参照は，添え字の配列要素ごとに 1 組と見なし，添え字のすべての組み合わせに対してそれぞれ評価が行われます。述語の結果は，次の表の内容に従って決定されます。

項番	条件	述語の結果
1	述語の結果が真となる組が少なくとも 1 つ存在する場合	真
2	項番 1 の条件を満たさない場合で，かつ述語の結果が不定となる組が少なくとも 1 つ存在する場合	不定※
3	項番 1 と項番 2 の条件を満たさない場合	偽

注※　NULL 述語の場合，項番 2 の条件は除外されます。したがって，項番 1 の条件を満たさない場合，述語の結果は偽になります。

ANY を指定した配列要素参照を含む述語の結果は，配列要素の組ごとに評価が行われる述語をOR 条件でつないだ探索条件の結果と同じになります。例を次に示します。

(例)

C1 列は，配列要素数が 2 の配列型の列とします。C2 列は，配列要素数が 3 の配列型の列とします。

ANY を指定した配列要素参照を含む述語	左記の指定と同じ結果になる探索条件
"C1"[ANY(1)] = "C2"[ANY(2)]	"C1"[1] = "C2"[1] OR "C1"[1] = "C2"[2] OR "C1"[1] = "C2"[3] OR "C1"[2] = "C2"[1] OR "C1"[2] = "C2"[2] OR "C1"[2] = "C2"[3]

(4) 例題

■添え字に符号なし整数定数を指定する例

次に示す表T1 を検索対象とする SQL 文の実行例を説明します。

C1	C2
A	{1, 2, 3, 4}
B	{0, 1}
C	{2, 3, 4}
D	{ナル値, ナル値}

C2 列は配列型の列で，{ } 内の値は各配列要素の値を示しています。

C2 列の要素データ型は，INTEGER 型とします。

例題 1

表T1 のC2 列の 2 番目の配列要素の値を求めます。

```
SELECT "C2"[2] AS "AER1" FROM "T1"
```

下線部分が，配列要素参照の指定です。2 番目の配列要素の値を求めるため，添え字に 2 を指定しています。

実行結果の例

AER1
2
1
3
ナル値

C2 列の要素データ型がINTEGER 型のため，実行結果のデータ型はINTEGER 型になります。

例題 2

表T1 のC2 列の 2 番目の配列要素の値が 1 である行のC1 列の値を求めます。

```
SELECT "C1" FROM "T1" WHERE "C2"[2]=1
```

下線部分が，配列要素参照の指定です。2 番目の配列要素の値を検索するため，添え字に 2 を指定しています。

実行結果の例

C1
B

■添え字に ANY を指定する例

次に示すREPORT 表を検索対象とする SQL 文の実行例を説明します。

名前（NAME）	受検科目(SUBJECT)	点数(SCORE)
USER1	{math,chemistry,biology}	{75,50,80}
USER2	{math,chemistry,physics}	{50,85,95}
USER3	{math,chemistry,biology}	{90,90,50}
USER4	{math,chemistry,physics}	{85,95,75}
USER5	{math,chemistry,physics}	{80,55,80}

SUBJECT 列およびSCORE 列は配列型の列で，｛　｝内の値は各配列要素の値を示しています。

例題 1

物理を受験している人の名前を求めます。

```
SELECT "NAME" FROM "REPORT"
    WHERE "SUBJECT"[ANY] = 'physics'
```

下線部分が，配列要素参照の指定です。SUBJECT 列の配列要素の値に物理（physics）があるかどうかを検索するために，添え字にANY を指定しています。

実行結果の例

NAME
USER2
USER4
USER5

例題 2

数学の点数が 80 点以上で，かつ物理の点数が 80 点以上の人の名前を求めます。

```
SELECT "NAME" FROM "REPORT"
    WHERE "SUBJECT"[ANY(1)] = 'math' AND "SCORE"[ANY(1)] >= 80          ...1
      AND "SUBJECT"[ANY(2)] = 'physics' AND "SCORE"[ANY(2)] >= 80       ...2
```

下線部分が，配列要素参照の指定です。

1. 数学の点数が 80 点以上である条件を指定しています。SUBJECT 列の配列要素の値に数学（math）があり，SCORE 列の数学の点数を示す配列要素の値が 80 以上である条件を指定しています。
2. 物理の点数が 80 点以上である条件を指定しています。SUBJECT 列の配列要素の値に物理（physics）があり，SCORE 列の物理の点数を示す配列要素の値が 80 以上である条件を指定しています。

重要

同じ識別番号を指定した配列要素参照は，1 つの組と見なされて述語の評価が行われます。上記の 1.の配列要素参照には，識別番号に1 を指定しているため，1 つの組と見なさ

れて述語の評価が行われます。上記の 2.の配列要素参照には，識別番号に2 を指定しているため，1 つの組と見なされて述語の評価が行われます。

実行結果の例

NAME
USER5

7.32　内部導出表

ここでは，内部導出表の適用例，および導出表の展開規則について説明します。

7.32.1　内部導出表の適用例

ビュー表を使用した問合せを実行する場合，ビュー表を指定したFROM 句に，CREATE VIEW 文の指定に基づいた内部的な導出表が適用されます。この導出表を**内部導出表**といいます。

同様に，WITH リスト要素に指定した問合せ名を使用した問合せを実行する場合も，内部導出表が適用されます。

内部導出表の適用例を次に示します。

(1)　例 1（ビュー表を使用した問合せを実行する場合）

ビュー表を使用した問合せを実行する場合，ビュー表を指定したFROM 句に内部導出表が適用されます。内部導出表の適用例を次に示します。

ビュー表の定義例：

```
CREATE VIEW ”V1” (”VC1”,”VC2”)
    AS SELECT * FROM ”T1” WHERE ”C1”>100
```

実行する SELECT 文の例：

```
SELECT * FROM ”V1”
```

上記のSELECT 文を実行した場合，次の内部導出表が適用されます。

内部導出表の適用例：

```
SELECT * FROM (SELECT * FROM ”T1” WHERE ”C1”>100) AS ”V1” (”VC1”,”VC2”)
```

下線部分が内部導出表です。

(2)　例 2（WITH 句に指定した問合せ名を使用した問合せを実行する場合）

WITH 句に指定した問合せ名を使用した問合せを実行する場合，問合せ名を指定したFROM 句に内部導出表が適用されます。内部導出表の適用例を次に示します。

実行する SELECT 文の例：

```
WITH ”Q1”(”QC1”,”QC2”) AS (SELECT * FROM ”T1” WHERE ”C1”>100)
SELECT * FROM ”Q1”
```

上記の場合，Q1 が問合せ名，下線部分がWITH 句に指定した問合せ式本体になります。上記のSELECT 文を実行した場合，次の内部導出表が適用されます。

内部導出表の適用例：

```
SELECT * FROM (SELECT * FROM "T1" WHERE "C1">100) AS "Q1" ("QC1","QC2")
```

下線部分が内部導出表です。

7.32.2 導出問合せおよび導出問合せ名

導出表を導出する問合せ式本体を**導出問合せ**といいます。また，導出問合せによって導出される表の名前を，**導出問合せ名**といいます。導出表の導出問合せ名は相関名となります。

導出問合せと導出問合せ名の例を次に示します。

ビュー定義の場合：

```
CREATE VIEW "V1" AS SELECT * FROM "T1" WHERE "C1">100
```

導出問合せ：下線部分

導出問合せ名："V1"

導出表の場合：

```
SELECT "C1","C2"*1.05
   FROM (SELECT "C1","C2" FROM "T1" GROUP BY "C1","C2") "X"
```

導出問合せ：下線部分

導出問合せ名："X"

WITH 句を指定した問合せの場合：

```
WITH "Q1" AS (SELECT "C1","C2" FROM "T1" GROUP BY "C1","C2")
SELECT "C1","C2"*1.05 FROM "Q1"
```

導出問合せ：下線部分

導出問合せ名："Q1"

7.32.3 導出表の展開規則

導出表の導出問合せを実行する際，導出表を含まない等価な形式となるように，外側の問合せ式本体に展開されます。これを**導出表の展開**といいます。導出表の展開例を次に示します。

導出表を使用した問合せの例：

```
SELECT "PN1","PR2"*1.05 AS "TXPRICE"
   FROM (SELECT "PNAME","PRICE","PLACE" FROM "STOCK"
         WHERE "PRICE">10000)
      AS "X"("PN1","PR2","PL3")
         WHERE "PL3" IN('Alaska','Arizona')
```

導出表の展開例：

```
SELECT "PNAME" AS "PN1","PRICE"*1.05 AS "TXPRICE"
   FROM "STOCK"
      WHERE "PRICE">10000 AND "PLACE" IN('Alaska','Arizona')
```

導出表の展開規則を次に示します。

1. 次に示す条件をすべて満たす場合は，内部導出表は展開されません。
   - 1SQL 文中に同じビュー表が複数指定されている場合
   - そのビュー表を定義した際に，問合せ式本体中に次に示すどれかの指定がある場合
     - SELECT DISTINCT
     - 集合演算
     - 副問合せ
     - コンマ結合
     - 結合表
     - ビュー表
     - 問合せ名
     - 集合関数
     - ウィンドウ関数
     - GROUP BY 句
     - HAVING 句
     - 表関数導出表
     - アーカイブマルチチャンク表
     - WHERE 句

   同様に，1SQL 文中に同じ問合せ名が複数指定されていて，その問合せ名のWITH リスト要素に指定した問合せ式本体に上記のどれかの指定がある場合は，内部導出表は展開されません。
2. WITH リスト要素に再帰的問合せを指定し，その再帰的問合せの再帰的メンバ中にビュー表または問合せ名を指定している場合，そのビュー表または問合せ名に対応する内部的な導出表は展開されません。
3. WITH リスト要素に再帰的問合せが含まれる場合，そのWITH リスト要素の問合せ名に対応する内部的な導出表は展開されません。
4. 次の条件 1 または条件 2 のどちらかを満たすWITH リスト要素の問合せ名に対応する内部導出表は展開されません。また，CREATE VIEW 文の問合せ式にWITH 句を指定し，かつ問合せ式中から次のどちらかの条件を満たすWITH リスト要素の問合せ名を参照している場合，CREATE VIEW 文で定義したビュー表から導出される導出表は展開されません。

   ＜条件 1 ＞

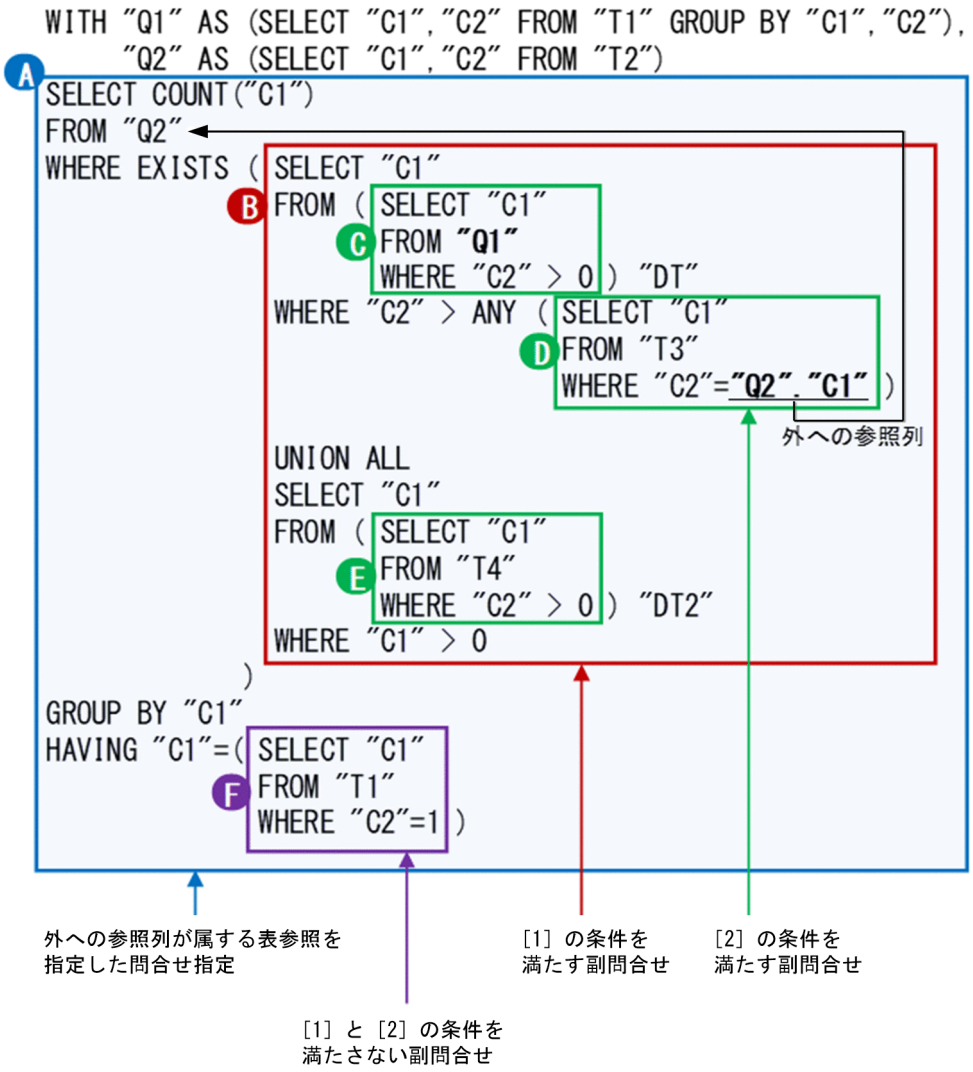
   次の条件をすべて満たす場合

- WITH リスト要素を複数指定している
- WITH リスト要素に指定した問合せ式本体に次のどれかを指定している
  - SELECT DISTINCT
  - 集合演算
  - 副問合せ
  - コンマ結合
  - 結合表
  - ビュー表
  - 問合せ名
  - 集合関数
  - ウィンドウ関数
  - GROUP BY 句
  - HAVING 句
  - 表関数導出表
  - アーカイブマルチチャンク表
- 次のどれかの条件を満たしている
  - WITH リスト要素に対応する問合せ名を，SQL 文中に 2 か所以上指定している
  - WITH リスト要素に対応する問合せ名を，SQL 文中に 1 か所指定していて，かつ次のどちらかの条件を満たす副問合せにその問合せ名を指定している

    [1] 外への参照列が属する表参照を指定した問合せ指定に指定された副問合せのうち，その外への参照列を指定した副問合せを含んでいる副問合せである

    [2] [1] の副問合せ中に含まれる副問合せである

    (例) 問合せ名Q1 が上記の [2] の条件を満たす例



上記の例の場合，[1] の条件を満たす副問合せは，問合せ B になります。また，[2] の条件を満たす副問合せは，問合せ C～問合せ E になります。一方，[1] と [2] の両方の条件を満たさない副問合せは，問合せ F になります。問合せ名Q1 は，問合せ C に指定されているため，[2] の条件を満たします。

＜条件 2 ＞

次の条件をすべて満たす場合

- WITH リスト要素を 1 つ指定している
- WITH リスト要素に指定した問合せ式本体中に，同じビュー表を複数指定している
  または，WITH 句以外にWITH リスト要素に対応する問合せ名を複数指定している
- WITH リスト要素に対応する問合せ名を，次のどちらかの副問合せに指定している
  [1] 外への参照列が属する表参照を指定した問合せ指定に指定された副問合せのうち，その外への参照列を指定した副問合せを含んでいる副問合せである
  [2] [1] の副問合せ中に含まれる副問合せである

5. 次に示す導出表の展開は行われません。

- CREATE VIEW 文の問合せ式に指定したディクショナリ表またはシステム表から等価変換された内部導出表
- 表値構成子によって導出された導出表
- FULL OUTER JOIN の指定によって等価変換された導出表
- アーカイブマルチチャンク表から等価変換された導出表

  アーカイブマルチチャンク表から等価変換される導出表については，マニュアル『HADB　AP 開発ガイド』の『アーカイブマルチチャンク表を検索する SQL 文の等価変換』を参照してください。
- OR 条件の指定によって等価変換された導出表

  OR 条件の指定によって等価変換された導出表については，マニュアル『HADB　AP 開発ガイド』の『OR 条件に関する等価変換（集合演算 UNION ALL を指定した導出表への等価変換)』を参照してください。

7.32.4　導出表の展開が行われないケース

次の条件を 1 つでも満たす場合，導出表の展開は行われません。

1. 導出問合せの最も外側の問合せ指定に，SELECT DISTINCT を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合
   - GROUP BY 句，HAVING 句，または集合関数を指定している場合

     (例)

     ```
     SELECT SUM("C1") FROM (SELECT DISTINCT * FROM "T1") AS "V1"
         GROUP BY "C2"
     ```
   - 表の結合（結合表を含む）を指定している場合

     (例)

     ```
     SELECT * FROM (SELECT DISTINCT * FROM "T1") AS "V1","T1"
         WHERE "V1"."C1"="T1"."C1"
     ```
   - 列指定を含む値式から導出される導出列を，選択式に単独の列指定で指定していない場合

     (例)

     ```
     SELECT "VC1" FROM (SELECT DISTINCT "C1","C2" FROM "T1") AS "V1"("VC1","VC2")
     ```

     導出列VC2 が選択式に指定されていないため，導出表V1 の展開は行われません。

     ```
     SELECT "VC1"*1.05,"VC2" FROM (SELECT DISTINCT * FROM "T1") AS "V1"("VC1","VC2")
     ```

     導出列VC1 が，選択式に単独の列指定として指定されていないため，導出表V1 の展開は行われません。
   - 選択式にスカラ関数RANDOM，RANDOM_NORMAL，RANDOMROW，スカラ副問合せ，またはウィンドウ関数を含む値式を指定している場合

     (例)

```
SELECT ”VC1”,”VC2”,RANDOM()
    FROM (SELECT DISTINCT ”C1”,”C2” FROM ”T1”) AS ”V1”(”VC1”,”VC2”)
```

- スカラ関数RANDOM，RANDOM_NORMAL，RANDOMROW，または集合関数のどれかを含み，かつ列指定を含まない値式から導出される導出列を，選択式に単独の列指定で指定していない場合

  （例）

```
SELECT ”VC1” FROM (SELECT DISTINCT ”C1”,RANDOM() FROM ”T1”) AS ”V1”(”VC1”,”VC2”)
```

- 選択式に指定していない値式をソートキーに指定している場合

  （例）

```
SELECT * FROM (SELECT DISTINCT * FROM ”T1”) AS ”V1”
    ORDER BY (”C2”+1)
```

2. 導出問合せの最も外側の問合せ指定に，GROUP BY 句を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合

- DISTINCT 集合関数，逆分布関数，LISTAGG 集合関数，またはARRAY_AGG 集合関数を指定している場合

  （例）

```
SELECT ”C1”,SUM(DISTINCT ”C2”)
    FROM (SELECT ”C1”,”C2” FROM ”T1” GROUP BY ”C1”,”C2”) AS ”V1”
    GROUP BY ”C1”,”C2”
```

- 表の結合（結合表を含む）を指定している場合

  （例）

```
SELECT * FROM (SELECT ”C1”,”C2” FROM ”T1” GROUP BY ”C1”,”C2”)
    AS ”V1”,”T1”
```

- 述語中に行値構成子を指定している場合

  （例）

```
SELECT * FROM (SELECT ”C1”,”C2” FROM ”T1” GROUP BY ”C1”,”C2”)
    AS ”V1” WHERE (”C1”,”C2”) = (10,20)
```

3. 導出問合せの最も外側の問合せ指定に，GROUP BY 句を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中にGROUP BY 句，HAVING 句，または集合関数があり，かつ次のどちらかの場合

- 導出問合せを操作する問合せ指定に指定したグループ化列数と，導出問合せに指定したグループ化列数が異なる場合

  （例）

```
SELECT ”C1”,”C2” FROM (SELECT ”C1”,”C2” FROM ”T1” GROUP BY ”C1”,”C2”,”C3”) AS ”V1”
    GROUP BY ”C1”,”C2”
```

- 導出問合せのグループ化列の参照列から導出される導出列を，導出問合せを操作する問合せ指定のグループ化列に単独で指定していない場合

  （例）

```
SELECT "C1","C2"+1 FROM (SELECT "C1","C2" FROM "T1" GROUP BY "C1","C2") AS "V1"
    GROUP BY "C1","C2"+1
```

4. 導出問合せの最も外側の問合せ指定に，値式を含むGROUP BY 句を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合

   - 列指定を含む値式から導出された導出問合せ中のグループ化列を外への参照を行う列として指定している場合

     (例)

```
SELECT * FROM (SELECT "G1" FROM "T1" GROUP BY C1+1 "G1") AS "V1"
    WHERE EXISTS (SELECT * FROM "T2" WHERE "T2"."C1" = "V1"."G1")
```

5. 導出問合せの最も外側の問合せ指定の選択式に，列指定を含む値式を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合

   - 列指定を含む値式から導出された導出問合せ名の列を，グループ化列かつ外への参照を行う列として選択式またはHAVING 句中に指定している場合

     (例)

```
SELECT "DC1" FROM (SELECT "C1"+1 AS "DC1" FROM "T1") AS "V1"
    GROUP BY "DC1"
    HAVING EXISTS (SELECT * FROM "T2" WHERE "T2"."C1" = "V1"."DC1")
```

   - 逆分布関数を複数指定している場合

     (例)

```
SELECT MEDIAN("C1")
    FROM (SELECT ABS("C1") AS "C1","C2" FROM "T1") AS "V1"
    HAVING MEDIAN("C1")>100
```

   - 列指定を含む値式から導出された導出問合せ名の列を，外への参照を行う列として集合関数の引数に指定している場合

     (例)

```
SELECT "C1" FROM (SELECT SUBSTR("C1",5) AS "C1","C2" FROM "T1") AS "V1"
    GROUP BY "C1"
    HAVING EXISTS(SELECT * FROM "T1" WHERE MAX("V1"."C1")="C1")
```

   - 配列要素参照を含む値式から導出された導出問合せ名の列を，行値構成子の行値構成子要素に指定している場合

     (例)

```
SELECT "C1" FROM (SELECT "C1"[1] AS "C1","C2"[2] AS "C2" FROM "T1") AS "V1"
    WHERE ("C1","C2") = (1,2)
```

   - 配列型の列指定を含む値式（ただし，単独で指定した配列型の列指定，およびスカラ関数ARRAY_MAX_CARDINALITY は除く）から導出された導出問合せ名の列を外への参照列として指定している場合

     (例)

```
SELECT ”C1” FROM (SELECT ”C1”[1] AS ”C1”, ”C2”[2] AS ”C2” FROM ”T1”) AS ”V1”
    WHERE EXISTS (SELECT * FROM ”T2” WHERE ”T2”.”C1” = ”V1”.”C2”)
```

6. 導出問合せの最も外側の問合せ指定の選択式に，列指定を含まない値式を指定した導出問合せ名を，結合表でのナル値を補う側の表参照に指定している場合

   （例）

```
SELECT * FROM ”T1” LEFT OUTER JOIN
                  (SELECT SUBSTR(’ABC’,2) AS ”C1”,”C2” FROM ”T2”) AS ”V1”
                  ON ”T1”.”C1”=”V1”.”C1”
```

7. 導出問合せの最も外側の問合せ指定の選択式に，列指定を含まない値式を指定した導出問合せ名をFROM句に指定し，そのFROM句を直接含む問合せ指定中に次の指定がある場合

   - 逆分布関数を複数指定している場合

     （例）

```
SELECT MEDIAN(”C1”)
    FROM (SELECT ABS(100) AS ”C1”,”C2” FROM ”T1”) AS ”V1”
    HAVING MEDIAN(”C1”)>100
```

   - 列指定を含まない値式から導出された導出問合せ名の列を，ウィンドウ関数中に指定している場合

     （例）

```
SELECT ”C2”,SUM(”C2”) OVER(ORDER BY ”C1”)
     FROM (SELECT SUBSTR(’ABC’,2) AS ”C1”,”C2” FROM ”T1”) AS ”V1”
```

8. 導出問合せの最も外側の問合せ指定の選択式に，スカラ関数RANDOMまたはRANDOM_NORMALを含む値式を指定した導出問合せ名をFROM句に指定し，そのスカラ関数RANDOMまたはRANDOM_NORMALから導出された列をSQL文中に指定している場合

   （例）

```
SELECT ”C1”, ”C2” FROM (SELECT ”C1”+RANDOM() AS ”C1”, ”C2” FROM ”T2”) AS ”V1”
```

9. 導出問合せの最も外側の問合せ指定の選択式に，スカラ関数RANDOMCURSORを含む値式を指定した導出問合せ名をFROM句に指定し，そのFROM句を直接含む問合せ指定中に導出問合せのスカラ関数RANDOMCURSORを含む値式から導出された列を「選択式またはORDER BY句」以外に指定している場合

   （例）

```
SELECT ”C1”,”C2”
    FROM (SELECT ”C1”+RANDOMCURSOR(1,10,20) AS ”C1”,”C2” FROM ”T2”) AS ”V1”
      WHERE ”C1”>1000
```

10. 導出問合せの最も外側の問合せ指定の選択式に，スカラ関数RANDOMROWを含む値式を指定した導出問合せ名をFROM句に指定し，そのFROM句を直接含む問合せ指定中に次の指定がある場合

    - 導出問合せのスカラ関数RANDOMROWを含む値式から導出された列を，「選択式またはORDER BY句」以外に指定している場合

      （例）

```
SELECT "C1","C2"
    FROM (SELECT "C1"+RANDOMROW(1,10,20) AS "C1","C2" FROM "T2") AS "V1"
      WHERE "C1">1000
```

- 導出問合せのスカラ関数RANDOMROW を含む値式から導出された列を，集合関数の引数に指定している場合

  (例)

```
SELECT SUM("C1")
    FROM (SELECT "C1"+RANDOMROW(1,10,20) AS "C1","C2" FROM "T2") AS "V1"
```

- 導出問合せのスカラ関数RANDOMROW を含む値式から導出された列を，ウィンドウ関数中に指定している場合

  (例)

```
SELECT "C1",SUM("C2") OVER(ORDER BY "C1")
    FROM (SELECT "C1", "C2"+RANDOMROW(1,10,20) AS "C2" FROM "T2") AS "V1"
```

- 導出問合せのスカラ関数RANDOMROW を含む値式から導出された列を，スカラ関数RANDOMROW 中に指定している場合

  (例)

```
SELECT RANDOMROW(1,"C1","C2")
    FROM (SELECT "C1"+RANDOMROW(1,10,20) AS "C1","C2" FROM "T2") AS "V1"
```

- 表の結合（結合表を含む）を指定している場合

  (例)

```
SELECT * FROM (SELECT "C1"+RANDOMROW(1,10,20) AS "C1","C2" FROM "T2")
    AS "V1","T1"
```

11. 導出問合せの最も外側の問合せ指定の選択式に，？パラメタを含む値式を指定した導出問合せ名をFROM 句に指定し，？パラメタから導出された列を問合せ指定中に指定している場合

    (例)

```
SELECT "C1" FROM (SELECT SUBSTR("C1",?) AS "C1","C2" FROM "T1") AS "V1"
```

12. 導出問合せの最も外側の問合せ指定の選択式に，スカラ副問合せを含む値式を指定した導出問合せ名をFROM 句に指定した場合

    (例)

```
SELECT "C1" FROM (SELECT (SELECT "C1" FROM "T2") + 10 AS "C1"
                    FROM "T1") AS "V1"
```

13. 導出問合せの最も外側の問合せ指定の選択式に，逆分布関数を含む値式を指定した導出問合せ名をFROM 句に指定し，逆分布関数から導出された列を問合せ指定中で指定している場合

    (例)

```
SELECT "C1" FROM (SELECT MEDIAN("C1") AS "C1",MAX("C2")
                    FROM "T1") AS "V1"
```

14. 導出問合せの最も外側の問合せ指定の選択式に，集合関数を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合

    - 値式を含むGROUP BY 句，DISTINCT 集合関数，逆分布関数，LISTAGG 集合関数，またはARRAY_AGG 集合関数を指定している場合

      （例）

```
SELECT SUM(DISTINCT "C1")
    FROM (SELECT COUNT("C1") AS "C1" FROM "T1") AS "V1"
    GROUP BY "C1"
```

    - 表の結合（結合表を含む）を指定している場合

      （例）

```
SELECT * FROM (SELECT COUNT("C1") AS "C1" FROM "T1") AS "V1","T1"
    WHERE "V1"."C1"="T1"."C1"
```

    - 集合関数から導出された導出問合せ名の列を，外への参照を行う列として集合関数の引数に指定している場合

      （例）

```
SELECT "C1" FROM (SELECT COUNT("C1") AS "C1" FROM "T1") AS "V1"
    GROUP BY "C1"
    HAVING EXISTS(SELECT * FROM "T2" WHERE MAX("V1"."C1")="C1")
```

    - 集合関数から導出された導出問合せ名の列を，行値構成子の行値構成子要素に指定している場合

      （例）

```
SELECT "C1" FROM (SELECT COUNT("C1") AS "C1",COUNT("C2") AS "C2" FROM "T1") AS "V1"
    WHERE ("C1","C2") = (10,20)
```

15. 導出問合せの最も外側の問合せ指定の選択式にLISTAGG 集合関数またはARRAY_AGG 集合関数を含む値式を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に，LISTAGG 集合関数またはARRAY_AGG 集合関数を含む値式から導出された列を外への参照列として指定している場合

    （例）

```
SELECT "C1" FROM (SELECT LISTAGG("C1") WITHIN GROUP(ORDER BY "C2") AS "C1"
                      FROM "T1") AS "V1"
    WHERE EXISTS(SELECT * FROM "T2" WHERE "T2"."C1"="V1"."C1")
```

16. 導出問合せの最も外側の問合せ指定の選択式に，ARRAY_AGG 集合関数を含む値式を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合

    - 導出問合せのARRAY_AGG 集合関数を含む値式から導出された列を，ANY を指定した配列要素参照に指定している場合

      （例）

```
SELECT "C1" FROM (SELECT ARRAY_AGG("C1") AS "C1" FROM "T1") AS "V1"
    WHERE "C1"[ANY] = 1
```

17. 導出問合せの最も外側の問合せ指定に，ウィンドウ関数を指定した導出問合せ名を FROM 句に指定している場合

(例)

```
SELECT "C2" FROM (SELECT "C1",AVG("C1") OVER(ORDER BY "C2") AS "C2"
                    FROM "T1") AS "V1"
```

18. 導出問合せの最も外側の問合せ指定に，LIMIT 句を指定している場合

(例)

```
SELECT "C1","C2" FROM (SELECT "C1","C2" FROM "T1" LIMIT 10) AS "V1"
```

19. 導出問合せの最も外側の問合せ指定のWHERE 句に行値構成子またはANY を指定した配列要素参照を指定した導出問合せ名を，結合表でのナル値を補う側の表参照に指定している場合

(例)

```
SELECT * FROM "T1" LEFT OUTER JOIN (SELECT * FROM "T2" WHERE ("C1","C2") = (1,2)) AS "V1"
     ON "T1"."C1"="V1"."C1"
```

20. 導出問合せの最も外側の問合せ指定に複数の表を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定のWHERE 句に次の指定がある場合

- 次の指定がある述語
  - 導出問合せの最も外側の問合せ指定に指定した表の列指定を含む値式から導出された列を指定した，ANY を指定した配列要素参照
  - 導出問合せの最も外側の問合せ指定に指定した上記とは別の表の列指定を含む値式から導出された列

(例)

```
SELECT "VC1","VC2"
  FROM (SELECT "T1"."C1","T2"."C2" FROM "T1","T2") AS "V1"("VC1","VC2")
    WHERE "VC1"[ANY] = "VC2"
```

- 次の同じ識別番号を持つANY を指定した配列要素参照
  - 導出問合せの最も外側の問合せ指定に指定した表の列指定を含む値式から導出された列を指定した配列要素参照
  - 導出問合せの最も外側の問合せ指定に指定した上記とは別の表の列指定を含む値式から導出された列を指定した配列要素参照

```
SELECT "VC1","VC2"
  FROM (SELECT "T1"."C1","T2"."C2" FROM "T1","T2") AS "V1"("VC1","VC2")
    WHERE "VC1"[ANY(1)] = 1 AND "VC2"[ANY(1)] = 2
```

21. 導出問合せの最も外側の問合せ指定に，コンマ結合を指定した導出問合せ名を結合表の表参照に指定している場合

(例)

```
SELECT "VC1","VC2" FROM (SELECT "T1"."C1","T2"."C1" FROM "T1","T2","T3"
                           WHERE "T1"."C1"="T2"."C1"
                             AND "T2"."C1"="T3"."C1") AS "V1"("VC1","VC2")
                         LEFT JOIN "T3" ON "VC1" = "T3"."C1"
```

22. FULL OUTER JOIN 中の表参照に導出問合せを指定している場合

(例)

```
SELECT * FROM (SELECT "C1" FROM "T1") AS "V1"
                FULL OUTER JOIN "T2" ON "V1"."C1"="T2"."C1"
```

23. 導出問合せの最も外側の問合せ指定を演算項に持つ集合演算の結果導出される列に，データ型が 32,000 バイトを超えるVARCHAR 型のデータを含んでいる場合

(例)

```
SELECT "C1" FROM (SELECT "C1" FROM "T1"
                  UNION
                  SELECT "DEFINE_SOURCE" FROM "MASTER"."SQL_DEFINE_SOURCE"
                 ) AS "V1"
```

24. 導出問合せに指定された集合演算の演算項の問合せ指定を導出問合せと見なしたときに，上記の 1.～18.の内部導出表の展開が行われない条件を 1 つでも満たしている場合

(例)

```
SELECT "C1" FROM (SELECT DISTINCT "C1","C2" FROM "T1"
                  UNION ALL
                  SELECT "C1","C2" FROM "T2"
                 ) AS "V1"
```

集合演算の演算項の問合せ指定「SELECT DISTINCT "C1","C2" FROM "T1"」を導出問合せと見なした場合，1.の内部導出表の展開が行われない条件※を満たしているため，導出表V1 の展開は行われません。

注※

導出問合せの最も外側の問合せ指定にSELECT DISTINCT を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合

- 列指定を含む値式から導出される導出列を，選択式に単独の列指定として指定していない

25. 導出問合せの最も外側の問合せ指定を演算項に持ち，UNION ALL だけの集合演算を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合

- GROUP BY 句，HAVING 句，または集合関数を指定している場合

  (例)

```
SELECT "C1","C2"
    FROM (
          SELECT "C1","C2" FROM "T1"
          UNION ALL SELECT "C1","C2" FROM "T2"
         ) AS "V1"
    GROUP BY "C1","C2"
```

- 選択式にウィンドウ関数を指定している場合

（例）

```
SELECT ”C1”,SUM(”C1”) OVER(ORDER BY ”C2”)
    FROM (
          SELECT ”C1”,”C2” FROM ”T1”
          UNION ALL SELECT ”C1”,”C2” FROM ”T2”
          ) AS ”V1”
```

- 表の結合（結合表を含む）を指定している場合

（例）

```
SELECT * FROM (
               SELECT ”C1”, ”C2” FROM ”T1”
               UNION ALL SELECT ”C1”,”C2” FROM ”T2”
               ) AS ”V1”,”T3”
```

- 選択式に指定していない導出問合せ名の列をソートキーとして指定している場合

（例）

```
SELECT ”C1” FROM (
                  SELECT ”C1”,”C2” FROM ”T1”
                  UNION ALL SELECT ”C1”,”C2” FROM ”T2”
                  ) AS ”V1”
    ORDER BY ”C2”
```

- 選択式に指定していない値式をソートキーに指定している場合

（例）

```
SELECT ”C1” FROM (
                  SELECT ”C1”,”C2” FROM ”T1”
                  UNION ALL SELECT ”C1”,”C2” FROM ”T2”
                  ) AS ”V1”
    ORDER BY (”C1”+1)
```

26. 導出問合せの最も外側の問合せ指定を演算項に持ち，UNION ALL 以外の集合演算子を含む集合演算を指定した導出問合せ名をFROM 句に指定し，そのFROM 句を直接含む問合せ指定中に次の指定がある場合

- SELECT DISTINCT を指定している場合

（例）

```
SELECT DISTINCT ”C1” FROM (
                           SELECT ”C1” FROM ”T1”
                           EXCEPT ALL SELECT ”C1” FROM ”T2”
                           ) AS ”V1”
```

- GROUP BY 句，HAVING 句，または集合関数を指定している場合

（例）

```
SELECT ”C1”,”C2” FROM (
                       SELECT ”C1”,”C2” FROM ”T1”
                       UNION SELECT ”C1”,”C2” FROM ”T2”
                       ) AS ”V1”
    GROUP BY ”C1”,”C2”
```

- 選択式にウィンドウ関数を指定している場合

  （例）

```
SELECT ”C1”,SUM(”C1”) OVER(ORDER BY ”C2”) FROM (
                      SELECT ”C1”,”C2” FROM ”T1”
                      INTERSECT ALL SELECT ”C1”,”C2” FROM ”T2”
                      ) AS ”V1”
```

- 表の結合（結合表を含む）を指定している場合

  （例）

```
SELECT * FROM (
               SELECT ”C1”,”C2” FROM ”T1”
               EXCEPT SELECT ”C1”,”C2” FROM ”T2”
               ) AS ”V1”,”T3”
```

- 選択式にスカラ関数RANDOM，RANDOM_NORMAL，RANDOMROW，スカラ副問合せ，またはウィンドウ関数を含む値式を指定している場合

  （例）

```
SELECT ”VC1”,”VC2”,RANDOM() FROM (
                                  SELECT ”C1”,”C2” FROM ”T1”
                                  UNION SELECT ”C1”,”C2” FROM ”T2”
                                  ) AS ”V1”(”VC1”,”VC2”)
```

- 選択式に単独の列指定で指定されていない導出問合せ名の列が少なくとも 1 つ存在する場合

  （例）

```
SELECT ”VC1” FROM (
                   SELECT ”C1”,”C2” FROM ”T1”
                   INTERSECT SELECT ”C1”,”C2” FROM ”T2”
                   ) AS ”V1”(”VC1”,”VC2”)
```

  導出列VC2 が選択式に指定されていないため，導出表V1 の展開は行われません。

  （例）

```
SELECT ”VC1”*1.05,”VC2” FROM (
                              SELECT ”C1”,”C2” FROM ”T1”
                              UNION SELECT ”C1”,”C2” FROM ”T2”
                              ) AS ”V1”(”VC1”,”VC2”)
```

  導出列VC1 が，選択式に単独の列指定として指定されていないため，導出表V1 の展開は行われません。

- 選択式に指定していない導出問合せ名の列をソートキーとして指定している場合

  （例）

```
SELECT ”C1” FROM (
                  SELECT ”C1”,”C2” FROM ”T1”
                  UNION SELECT ”C1”,”C2” FROM ”T2”
                  ) AS ”V1”
    ORDER BY ”C2”
```

- 選択式に指定していない値式をソートキーに指定している場合

（例）

```
SELECT ”C1” FROM (
                  SELECT ”C1”,”C2” FROM ”T1”
                  UNION SELECT ”C1”,”C2” FROM ”T2”
                  ) AS ”V1”
    ORDER BY (”C1”+1)
```

7.32.5　導出表の展開有無

導出表の展開有無を次の表に示します。

表 7-46　導出表の展開有無（1/3）

導出問合せを操作する SQL 文の指定内容	導出問合せの指定内容								
	SEL_DIST	GRP	GRP_EXP	SEL_EXP	SEL_NCOL	SEL_RAND	SEL_RANDCRS	SEL_RANDROW	SEL_PRM
SEL_DIST	○	○	○	○	○	○※5	○	○	×
GRP	×	○※6	○※6	○※2	○※2	○※5	△	△	×
GRP_EXP	×	×	×	○	○	○※5	△	△	×
A-FUNC	×	○ ※6, ※7	○ ※6, ※7	○	○	○※5	○※8	△	×
D-FUNC	×	×	×	○	○	○※5	○※8	△	×
I-FUNC2	×	×	×	△	△	○※5	○※8	△	×
WIN_AGG	×	○	○	○	○	○※5	○※8	△	×
WIN_PAR	×	○	○	○	△	○※5	○※8	△	×
WIN_ORD	×	○	○	○	○	○※5	○※8	△	×
SEL_EXP	○	○	○	○	○	○※5	○	○※9	×
SEL_RAND	×	○	○	○	○	○※5	○	○	×
SEL_RANDROW	×	○	○	○	○	○※5	○	○※9	×
SEL_SUBQ	×	○	○	○	○	○※5	○	○	×
SEL_WINDOW	×	○	○	○	○	○※5	○	○	×
SEL_CNDRV	×	○	○	○	○	○※5	○	○	×
SEL_NCNDRV	○※11	○	○	○	○	○※5	○	○	×
JOIN	×	×	×	○	○	○※5	○	○※5	×
IN_J_TBL	×	×	×	○※4	×	○※5	○	○※5	×
J_TBL※12	×	×	×	○	○	○※5	○	○※5	×

導出問合せを操作するSQL文の指定内容	導出問合せの指定内容								
	SEL_DIST	GRP	GRP_EXP	SEL_EXP	SEL_NCOL	SEL_RAND	SEL_RANDCRS	SEL_RANDROW	SEL_PRM
FJ_TBL	×	×	×	×	×	×	×	×	×
ANY_ARRAY_ELMREF	○	○	○	○	○	○※5	○	○	×
RVC	○	×	×	○※16	○	○※5	○	○	×
S_KEY	○	○	○	○	○	○※5	○	○	×
SEXP_KEY	×	○	○	○	○	○※5	○	○※9	×
O_REF	○	○	△	○※17	○	○※5	△	△	×
O_REF_FUNC	×	×	×	△	△	○※5	△	△	×
その他	○	○	○	○	○	○※5	○※8	○ ※8, ※9	×

表 7-47　導出表の展開有無（2/3）

導出問合せを操作するSQL文の指定内容	導出問合せの指定内容								
	SUBQ	SEL_IFN	FUNC_COL	FUNC_EXP	WINDOW	JOIN	CJOIN※12	WHERE_CE	LIMIT
SEL_DIST	×	×	○	○	×	○	○	○	×
GRP	×	×	△	△	×	○	○	○	×
GRP_EXP	×	×	△	△	×	○	○	○	×
A-FUNC	×	×	○※7	○※7	×	○	○	○	×
D-FUNC	×	×	△	△	×	○	○	○	×
I-FUNC2	×	×	△	△	×	○	○	○	×
WIN_AGG	×	×	○	○	×	○	○	○	×
WIN_PAR	×	×	○	○	×	○	○	○	×
WIN_ORD	×	×	○	○	×	○	○	○	×
SEL_EXP	×	×	○	○	×	○	○	○	×
SEL_RAND	×	×	○	○	×	○	○	○	×
SEL_RANDROW	×	×	○	○	×	○	○	○	×
SEL_SUBQ	×	×	○	○	×	○	○	○	×
SEL_WINDOW	×	×	○	○	×	○	○	○	×
SEL_CNDRV	×	×	○	○	×	○	○	○	×

導出問合せを操作するSQL文の指定内容	導出問合せの指定内容								
	SUBQ	SEL_IFN	FUNC_COL	FUNC_EXP	WINDOW	JOIN	CJOIN※12	WHERE_CE	LIMIT
SEL_NCNDRV	×	×	○	○	×	○	○	○	×
JOIN	×	×	×	×	×	○	○	○	×
IN_J_TBL	×	×	×	×	×	○	×	×	×
J_TBL※12	×	×	×	×	×	○	×	○	×
FJ_TBL	×	×	×	×	×	×	×	×	×
ANY_ARRAY_ELMREF	×	×	○※13	○※13	×	○※14	○	○	×
RVC	×	×	△	△	×	○	○	○	×
S_KEY	×	×	○	○	×	○	○	○	×
SEXP_KEY	×	×	○	○	×	○	○	○	×
O_REF	×	×	○※15	△	×	○	○	○	×
O_REF_FUNC	×	×	×	×	×	○	○	○	×
その他	×	×	○	○	×	○	○	○	×

表 7-48　導出表の展開有無（3/3）

導出問合せを操作するSQL文の指定内容	導出問合せの指定内容			
	U_ALL※3, ※10	SET_OP※3, ※10	SETOP_VCH32000	その他
SEL_DIST	○	○※1	×	○
GRP	×	×	×	○
GRP_EXP	×	×	×	○
A-FUNC	×	×	×	○
D-FUNC	×	×	×	○
I-FUNC2	×	×	×	○
WIN_AGG	×	×	×	○
WIN_PAR	×	×	×	○
WIN_ORD	×	×	×	○
SEL_EXP	○	○	×	○
SEL_RAND	○	×	×	○
SEL_RANDROW	○	×	×	○

導出問合せを操作するSQL文の指定内容	導出問合せの指定内容			
	U_ALL※3，※10	SET_OP※3，※10	SETOP_VCH32000	その他
SEL_SUBQ	○	×	×	○
SEL_WINDOW	○	×	×	○
SEL_CNDRV	○	×	×	○
SEL_NCNDRV	○	×	×	○
JOIN	×	×	×	○
IN_J_TBL	×	×	×	○
J_TBL※12	×	×	×	○
FJ_TBL	×	×	×	×
ANY_ARRAY_ELMREF	○	○	×	○
RVC	○	○	×	○
S_KEY	○	○	×	○
SEXP_KEY	×	×	×	○
O_REF	○	○	×	○
O_REF_FUNC	×	×	×	○
その他	○	○	×	○

（凡例）

○：導出表の展開を行います。

△：導出表の展開を行います。ただし，導出問合せを操作するSQL文の指定内容となる個所に，導出問合せの指定内容から導出される導出列を指定した場合は，導出表の展開を行いません。

×：導出表の展開を行いません。

- A-FUNC：被集約引数に導出問合せ名の列を指定したALL集合関数を含んでいる場合
- ANY_ARRAY_ELMREF：ANYを指定した配列要素参照を指定している場合
- CJOIN：コンマ結合を含んでいる場合
- D-FUNC：被集約引数に導出問合せ名の列を指定したDISTINCT集合関数，LISTAGG集合関数，もしくはARRAY_AGG集合関数を含んでいる場合，または被集約引数に導出問合せ名の列を指定した逆分布関数を1つだけ含んでいる場合
- I-FUNC2：被集約引数に導出問合せ名の列を指定した逆分布関数を2つ以上含んでいる場合
- FJ_TBL：FULL OUTER JOINの表参照に，導出問合せ名を指定している場合
- FUNC_COL：集合関数の被集約引数に，列指定だけを指定している場合
- FUNC_EXP：集合関数の被集約引数に，列指定以外を指定している場合

- GRP：GROUP BY 句，HAVING 句，または集合関数を含んでいる場合
- GRP_EXP：GROUP BY 句に列指定以外（スカラ演算など）を含んでいる場合
- IN_J_TBL：結合表でのナル値を補う側の表参照に，該当する導出問合せ名を指定している場合
- JOIN：複数の表を含んでいる場合
- J_TBL：結合表の表参照に導出問合せ名を指定している場合
- LIMIT：LIMIT 句を指定している場合
- O_REF：外への参照列として導出問合せ名の列を指定している場合
- O_REF_FUNC：外への参照列として，導出問合せ名の列を集合関数の引数に指定している場合
- RVC：述語中に行値構成子を指定している場合
- S_KEY：ソートキーに選択式に指定した導出問合せ名の列を指定している場合
- SEL_CNDRV：列指定を含む値式から導出される導出列を，選択式に単独の列指定として指定していない場合
- SEL_IFN：選択式に逆分布関数を指定している場合
- SEL_DIST：SELECT DISTINCT の指定ありの場合
- SEL_EXP：選択式に列指定以外（スカラ演算など）を含んでいる場合（ただし，値式中に列指定を含んでいる場合）
- SEL_NCNDRV：列指定を含まない値式から導出される導出列を，選択式に単独の列指定として指定していない場合
- SEL_NCOL：選択式に列指定以外から成る値式を含んでいる場合
- SEL_PRM：選択式に？パラメタを含んでいる場合
- SEL_RAND：選択式に，スカラ関数RANDOM またはRANDOM_NORMAL を含んでいる場合
- SEL_RANDCRS：選択式に，スカラ関数RANDOMCURSOR を含んでいる場合
- SEL_RANDROW：選択式に，スカラ関数RANDOMROW を含んでいる場合
- SEL_SUBQ：選択式にスカラ副問合せを含んでいる場合
- SEL_WINDOW：選択式にウィンドウ関数を含んでいる場合
- SET_OP：U_ALL 以外のケースで集合演算を指定している場合
- SETOP_VCH32000：集合演算の結果によって導出される列に，データ型が 32,000 バイトを超える VARCHAR 型の列がある場合
- SEXP_KEY：選択式に指定した値式と異なる値式をソートキーに指定している場合

  （例）

  ```
  SELECT ”C1”+”C2”,”C2” AS ”DC1” FROM ”T1” ORDER BY ”C1”/”C2”
  ```

- SUBQ：選択式に副問合せを含んでいる場合
- U_ALL：UNION ALL だけの集合演算を指定している場合

- WHERE_CE：WHERE 句中に次のどちらかを指定している場合
  - 行値構成子
  - ANY を指定した配列要素参照
- WINDOW：ウィンドウ関数を指定している場合
- WIN_AGG：ウィンドウ関数として指定した集合関数に導出問合せ名の列を指定している場合
- WIN_PAR：ウィンドウ関数中のウィンドウ分割句に導出問合せ名の列を指定している場合
- WIN_ORD：ウィンドウ関数中のウィンドウ順序句に導出問合せ名の列を指定している場合

注※1

導出問合せに指定された集合演算で，最後に演算される集合演算子がEXCEPT ALL の場合に限り，導出表の展開は行われません。

（例）展開されない例

```
SELECT DISTINCT "C1","C2" FROM (
                  (SELECT "C1","C2" FROM "T1"
                   UNION ALL
                   SELECT "C1","C2" FROM "T2")
                   EXCEPT ALL
                   SELECT "C1","C2" FROM "T2") AS "V1"
```

上記の SQL 文の場合，最後に演算されるのがEXCEPT ALL のため，導出表の展開は行われません。

（例）展開される例

```
SELECT DISTINCT "C1","C2" FROM (
                    SELECT "C1","C2" FROM "T1"
                    UNION ALL
                   (SELECT "C1","C2" FROM "T2"
                    EXCEPT ALL
                    SELECT "C1","C2" FROM "T2")) AS "V1"
```

上記の SQL 文の場合，最後に演算されるのがUNION ALL のため，導出表の展開が行われます。

注※2

次の条件をすべて満たしている場合，導出表の展開は行われません。

1. 値式から導出された導出問合せ名の列を，グループ化列に指定している
2. 次のどちらかの個所に指定した 1.の導出問合せ名の列が，外への参照列である

- 選択式

  （例）展開されない例

```
SELECT (SELECT "C1" FROM "T2" WHERE "T2"."C1" = "V1"."DC1") "DC2"
    FROM (SELECT "C1"+1 AS "DC1" FROM "T1") AS "V1"
    GROUP BY "DC1"
```

- HAVING 句

  （例）展開されない例

```
SELECT ”DC1” FROM (SELECT ”C1”+1 AS ”DC1” FROM ”T1”) AS ”V1”
    GROUP BY ”DC1”
    HAVING EXISTS (SELECT * FROM ”T2” WHERE ”T2”.”C1” = ”V1”.”DC1”)
```

注※3

集合演算を指定した導出表を操作する問合せ指定に指定された副問合せ中にFULL OUTER JOIN が指定されている場合，集合演算導出表の展開は行われません。

（例）展開されない例

```
SELECT * FROM (SELECT ”C1”,”C2” FROM ”T1”
               UNION ALL
               SELECT ”C1”,”C2” FROM ”T2”) ”DT2”
    WHERE ”C1”=ANY(SELECT ”X”.”C1” FROM ”T3” FULL OUTER JOIN ”T4”
                                            ON ”T3”.”C2”> ”T4”.”C2”)
```

注※4

導出問合せの最も外側の問合せ指定の選択式に，次のどれかの値式が指定されている場合，導出表の展開は行われません。

- スカラ関数COALESCE
- スカラ関数ISNULL
- スカラ関数NULLIF
- スカラ関数NVL
- スカラ関数DECODE
- スカラ関数LTDECODE
- CASE 式

注※5

SQL 文中に，導出問合せの指定内容の対象となる導出列を指定した場合だけ，導出表の展開は行われません。

（例）展開されない例

```
SELECT ”DC1” FROM (SELECT RANDOM(”C1”,”C2”) AS ”DC1” FROM ”T1”) AS ”V1”
```

上記の SQL 文の場合，導出問合せに指定したスカラ関数RANDOM を含む選択式の列に対応する導出列”DC1”を，導出問合せを操作する SQL 文に指定しているため，導出表の展開は行われません。

注※6

次の条件をすべて満たしている場合に，導出表の展開が行われます。

- 導出問合せを操作する問合せ指定に指定したグループ化列数と，導出問合せに指定したグループ化列数が同じである
- 導出問合せの選択式に指定したグループ化列から導出される導出列すべてを，導出問合せを操作する問合せ指定のグループ化列に指定している

（例 1）展開される例

```
SELECT "C1","C2"
    FROM (SELECT "C1","C2" FROM "T1" GROUP BY "C1","C2") AS "V1"
    GROUP BY "C1","C2"
```

（例 2）展開される例

```
SELECT "C1","DC2"
    FROM (SELECT "C1","C2"+1 AS "DC2" FROM "T1" GROUP BY "C1","C2"+1) AS "V1"
    GROUP BY "C1","DC2"
```

注※7

次の条件をすべて満たしている場合に，導出表の展開が行われます。

- 導出問合せを操作する問合せ指定にGROUP BY 句，またはHAVING 句を指定している。
- 導出問合せを操作する問合せ指定に指定したすべての集合関数が，次の条件 1 または条件 2 のどちらかを満たす。

  条件 1

  - 導出問合せを操作する問合せ指定に指定した集合関数がCOUNT(*)※である

  注※

  引数に定数，または定数と等価な値式を指定した集合関数COUNT（ALL 指定）は，集合関数COUNT(*)へ置き換えられ，集合関数COUNT(*)と同じものとして扱われます。

  条件 2

  - 導出問合せを操作する問合せ指定に指定した集合関数が次のどれかである
    - ALL 集合関数MAX
    - ALL 集合関数MIN
    - ALL 集合関数SUM
    - ALL 集合関数AVG
  - 上記の集合関数の被集約引数として，導出問合せに指定した集合関数から成る導出列を指定している

  （例）展開される例

```
SELECT "C1", "C2",SUM("C3")
    FROM (SELECT "C1","C2",COUNT("C3") AS "C3"
              FROM "T1"
              GROUP BY "C1","C2") "V1"
    GROUP BY "C1","C2"
```

注※8

導出問合せの指定内容の対象となる導出列を含む値式を「選択式またはORDER BY 句」以外に指定している場合，導出表の展開は行われません。

注※9

導出問合せの指定内容の対象となる導出列をスカラ関数RANDOMROW 中に指定している場合，導出表の展開は行われません。

注※10

導出問合せに指定した集合演算の演算項の問合せ指定を導出問合せと見なしたときに，内部導出表が展開されない条件を 1 つでも満たす場合は，導出表の展開は行われません。

注※11

次のどれかの指定があり，かつ列指定を含まない値式から導出される導出列を，選択式に単独の列指定で指定していない場合は，導出表の展開は行われません。

- スカラ関数RANDOM
- スカラ関数RANDOM_NORMAL
- スカラ関数RANDOMROW
- 集合関数

注※12

INNER JOIN またはCROSS JOIN は，HADB サーバによってコンマ結合に変換されることがあります。そのため，導出問合せに指定したINNER JOIN またはCROSS JOIN がコンマ結合に変換される場合，導出問合せにコンマ結合が含まれると見なされます（上記の表のCJOIN に該当します）。また，導出問合せを操作する問合せ指定に指定したINNER JOIN またはCROSS JOIN がコンマ結合に変換されて，問合せ指定から結合表がなくなる場合，結合表の指定がないと見なされます（上記の表のJ_TBL に該当しなくなります）。

注※13

導出問合せのARRAY_AGG 集合関数を含む値式から導出される導出列を，導出問合せを操作する問合せ指定に次の指定をしている場合は，導出表の展開は行われません。

- ANY を指定した配列要素参照の配列値式

注※14

導出問合せを操作する問合せ指定のWHERE 句に次の指定がある場合は，導出表の展開は行われません。

- 次の指定がある述語
  - 導出問合せの最も外側の問合せ指定に指定した表の列指定を含む値式から導出された列を指定した，ANY を指定した配列要素参照
  - 導出問合せの最も外側の問合せ指定に指定した上記とは別の表の列指定を含む値式から導出された列
- 次の同じ識別番号を持つANY を指定した配列要素参照
  - 導出問合せの最も外側の問合せ指定に指定した表の列指定を含む値式から導出された列を指定した配列要素参照
  - 導出問合せの最も外側の問合せ指定に指定した上記とは別の表の列指定を含む値式から導出された列を指定した配列要素参照

注※15

導出問合せのLISTAGG 集合関数またはARRAY_AGG 集合関数を含む値式から導出される導出列を外への参照列として指定している場合は，導出表の展開は行われません。

注※16

導出問合せの配列要素参照を含む値式から導出される導出列を行値構成子の行値構成子要素中に指定している場合は，導出表の展開は行われません。

注※17

導出問合せの配列型の列指定を含む値式（ただし，単独で指定した配列型の列指定，およびスカラ関数ARRAY_MAX_CARDINALITY は除く）から導出される導出列を外への参照列として指定している場合は，導出表の展開は行われません。

7.32.6　内部導出表にスカラ関数 CONVERT を付加する条件

次の場合，内部導出表にスカラ関数CONVERT を付加します。そのため，スカラ演算の数，およびスカラ演算の入れ子の数は，その分 1 つ加算されます。

- 内部導出表に集合演算を指定した場合

  結果が集合演算の結果のデータ型となるようなスカラ関数CONVERT を，選択式に付加します。

  （例）

  - 導出表を使用した問合せ

```
SELECT ”SN”,”KI”*1.08 AS ”税込金額” FROM
(SELECT ”商品名”,”商品金額”,”原産地” FROM ”商品A”
  WHERE ”商品金額”>10000
 UNION ALL
 SELECT ”商品名”,”商品金額”*0.8,”原産地” FROM ”商品B”
  WHERE ”商品金額”>20000) AS ”X”(”SN”,”KI”,”GE”)
WHERE ”GE” IN('東京','大阪')
```

  - 導出表の展開

```
SELECT CONVERT(”商品名”,VARCHAR(100)) AS ”SN”,
       CONVERT(”商品金額”*1.08,DEC(23,2)) AS ”税込金額”
  FROM ”商品A”
 WHERE ”商品金額”>10000 AND
       ”原産地” IN('東京','大阪')
 UNION ALL
SELECT CONVERT(”商品名”,VARCHAR(100)),
       CONVERT(”商品金額”*0.8*1.08,DEC(23,2))
 FROM ”商品B”
 WHERE ”商品金額”>20000 AND
       ”原産地” IN('東京','大阪')
```

- 次のすべての条件を満たす場合
  - 「7.32.5　導出表の展開有無」の「注※7」の「条件 2」を満たす内部導出表が展開される。
  - 導出問合せを操作する問合せ指定に指定した集合関数の結果のデータ型と，導出問合せに指定した集合関数の結果のデータ型が異なる。

結果が導出問合せを操作する問合せ指定に指定した集合関数の結果のデータ型となるようなスカラ関数CONVERTを，導出問合せに指定した集合関数に付加します。

（例）

- 導出表を使用した問合せ（”温度”列のデータ型がDECIMAL(10,2)の場合）

```
SELECT ”測定地点”,”測定日”,AVG(”温度”) AS ”温度”
  FROM (SELECT ”測定地点”,”測定日”,MAX(”温度”) AS ”温度”
          FROM ”センサデータ”
            GROUP BY ”測定地点”,”測定日”) ”V1”
    WHERE ”測定日” BETWEEN DATE’2018-01-01’ AND DATE’2018-12-31’
      GROUP BY ”測定地点”,”測定日”
```

- 導出表の展開

```
SELECT ”測定地点”,”測定日”,CONVERT(MAX(”温度”),DECIMAL(38, 30)) AS ”温度”
  FROM ”センサデータ”
    WHERE ”測定日” BETWEEN DATE’2018-01-01’ AND DATE’2018-12-31’
      GROUP BY ”測定地点”,”測定日”
```

8 スカラ関数

この章では，スカラ関数の機能，指定形式，および規則について説明します。

8.1　スカラ関数の一覧

スカラ関数の一覧を次の表に示します。

表 8-1　スカラ関数の一覧

項番	分類		スカラ関数名	機能
1	数学関数	三角関数	ACOS	対象データ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。
2			ASIN	対象データ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。
3			ATAN	対象データ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。
4			ATAN2	*y*/*x* 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。
5			COS	ラジアン単位で指定された対象データの余弦（三角関数の COS）を返します。
6			COSH	対象データの双曲線余弦を返します。
7			DEGREES	指定された角度をラジアンから度数に変換して返します。
8			PI	円周率 π の値を返します。
9			RADIANS	指定された角度を度数からラジアンに変換して返します。
10			SIN	ラジアン単位で指定された対象データの正弦（三角関数の SIN）を返します。
11			SINH	対象データの双曲線正弦を返します。
12			TAN	ラジアン単位で指定された対象データの正接（三角関数の TAN）を返します。
13			TANH	対象データの双曲線正接を返します。
14		指数・対数計算	EXP	自然対数の底の値の累乗を返します。
15			LN	対象データの自然対数を返します。
16			LOG	指定値を底とする対象データ（真数）の対数を返します。
17			POWER	対象データの累乗を返します。
18		数値計算	ABS	対象データの絶対値を返します。
19			CEIL	対象データ以上の値で，最小の整数値を返します。
20			FLOOR	対象データ以下の値で，最大の整数値を返します。

項番	分類		スカラ関数名	機能
21			MOD	被除数を除数で割った余りを返します。
22			RANDOM	最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。
23			RANDOMCURSOR	最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。 1SQL 文中で，同じ識別番号を指定したRANDOMCURSORは，常に同じ値を返します。
24			RANDOMROW	最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。 1 問合せ指定中で，同じ識別番号を指定したRANDOMROWは，問合せ指定の結果の行ごとに同じ値を返します。
25			RANDOM_NORMAL	平均 μ，標準偏差 σ の正規分布に従う擬似乱数を返します。
26			ROUND	対象データを小数点以下n 桁に丸めた値を返します。
27			SIGN	対象データが正の値の場合は+1 を，負の値の場合は-1 を，0 の場合は0 を返します。
28			SQRT	対象データの平方根を返します。
29			TRUNC	指定された桁数より下の数値を切り捨てた値を返します。
30	文字列関数	文字列操作	CONCAT	2 つの文字データを連結します。
31			LEFT	文字データの先頭（左）から一部の文字列を抽出します。
32			LPAD	対象データの先頭（左側）に，指定文字数となるまで，埋め込み文字列を繰り返し埋め込みます。
33			LTRIM	対象データの文字列の先頭から順に，削除文字に指定した文字を削除します。
34			RIGHT	文字データの末尾（右）から一部の文字列を抽出します。
35			RPAD	対象データの末尾（右側）に，指定文字数となるまで，埋め込み文字列を繰り返し埋め込みます。
36			RTRIM	対象データの文字列の末尾から順に，削除文字に指定した文字を削除します。
37			SUBSTR	文字データの任意の位置から一部の文字列を抽出します。
38			TRIM	対象データの文字列から，削除文字に指定した文字を削除します。文字の削除方法を次のどれかから選択できます。

項番	分類		スカラ関数名	機能
				• 文字列の先頭から順に，削除文字に指定した文字を削除します。 • 文字列の末尾から順に，削除文字に指定した文字を削除します。 • 文字列の先頭および末尾の両方から順に，文字を削除します。
39		文字列情報の取得	CONTAINS	検索条件式を満たす文字列が対象データ中に含まれているかどうかを返します。
40			INSTR	対象データ中の任意の文字列を検索し，その文字列の開始位置を返します。
41			LENGTH	対象データの文字列の文字数を返します。
42		文字置換	REPLACE	対象データ中の任意の文字列を置換します。対象データ中に存在する置換対象文字列のすべてを置換後の文字列に置換します。
43			TRANSLATE	対象データ中の任意の文字を置換します。
44		文字変換	LOWER	文字データの英大文字（A～Z）を英小文字（a～z）に変換します。
45			UPPER	文字データの英小文字（a～z）を英大文字（A～Z）に変換します。
46	日時関数		DATEDIFF	開始日時と終了日時の差を返します。
47			DAYOFWEEK	指定した日が，週の何日目かを返します。
48			DAYOFYEAR	指定した日が，その年の第何日目かを返します。
49			EXTRACT	日時を示すデータの一部（年，月，日，時，分，または秒のどれか）を抽出します。
50			GETAGE	生年月日と基準日から満年齢を求めます。
51			LASTDAY	日時データに指定した月の最終日の日付または日時を返します。
52			ROUND	日時データを日時書式で指定した単位に丸めて返します。
53			TIMESTAMPADD	対象データに指定した日時に，日時単位に指定した単位で日時を加算します。
54			TIMESTAMPDIFF	開始日時と終了日時の差を返します。
55			TRUNC	日時データを日時書式で指定した単位で切り捨てます。
56	バイナリ列関数	バイナリデータ操作	CONCAT	2つのバイナリデータを連結します。
57			SUBSTRB	バイナリデータの任意の位置から一部のバイナリデータを抽出します。

項番	分類		スカラ関数名	機能
58		ビット演算	BITAND	2 つのバイナリデータのビットごとの論理積を返します。
59			BITLSHIFT	バイナリデータを左ビットシフトした値を返します。
60			BITNOT	バイナリデータのビットごとの論理否定を返します。
61			BITOR	2 つのバイナリデータのビットごとの論理和を返します。
62			BITRSHIFT	バイナリデータを右ビットシフトした値を返します。
63			BITXOR	2 つのバイナリデータのビットごとの排他的論理和を返します。
64	配列関数		ARRAY_MAX_CARDINALITY	対象データに指定した配列データの最大要素数を返します。
65			CARDINALITY	対象データに指定した配列データの配列要素数を返します。
66	データ変換関数		ASCII	対象データの先頭の文字の文字コードを整数値で返します。
67			BIN	バイナリデータを 2 進文字列表現（'0'，'1'で構成された文字データ）に変換します。
68			CAST	データのデータ型を変換します。
69			CHR	対象データの整数値が示す文字コードに対応する文字を返します。
70			CONVERT	データのデータ型を変換します。 また，日時書式または数値書式を指定することで，次のことができます。 日時書式を指定した場合： • 日時データを文字データに変換する際，変換後の文字データの出力形式を指定できます。 • 文字データを日時データに変換する際，変換前の文字データの入力形式を指定できます。 数値書式を指定した場合： • 数データを文字データに変換する際，変換後の文字データの出力形式を指定できます。 • 文字データを数データに変換する際，変換前の文字データの入力形式を指定できます。
71			HEX	バイナリデータを 16 進文字列表現（'0'～'9'，'A'～'F'で構成された文字データ）に変換します。
72	NULL 評価関数		COALESCE	指定した対象データを指定した順に評価し，ナル値でない最初の値を返します。
73			ISNULL	

項番	分類	スカラ関数名	機能
74		NULLIF	対象データ 1 と対象データ 2 を比較した結果，等しい場合はNULL を返し，等しくない場合は対象データ 1 を返します。
75		NVL	指定した対象データを指定した順に評価し，ナル値でない最初の値を返します。
76	情報取得関数	LENGTHB	対象データの長さをバイト数で返します。
77	比較関数	DECODE	対象データと比較データを順次比較し，一致した場合は対応する返却値を返します。対象データとすべての比較データが一致しない場合は，既定返却値を返します。
78		GREATEST	指定した対象データの値のうち，最大値を返します。
79		LEAST	指定した対象データの値のうち，最小値を返します。
80		LTDECODE	対象データと比較データを順次比較し，対象データの値が比較データの値未満となる場合は，対応する返却値を返します。対象データの値がすべての比較データの値未満とならない場合は，既定返却値を返します。

8.2　数学関数（三角関数）

ここでは，三角関数に関する数学関数の機能と指定形式について説明します。

8.2.1　ACOS

対象データ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。

(1)　指定形式

スカラ関数ACOS：：＝ACOS(*対象データ*)

対象データ：：＝*値式*

(2)　指定形式の説明

対象データ：

逆余弦を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- 対象データには，-1～1 の値を指定してください。範囲外の値は指定できません。
- 対象データには，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4)　例題

例題

表T1 のC1 列～C3 列の値の逆余弦を求めます。

```
SELECT ACOS(”C1”) ,ACOS(”C2”),ACOS(”C3”)  FROM ”T1”
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
0	-0.15	2.0000000000000001E-1

検索結果

1.5707963267948966E0	1.7213645995715827E0	1.3694384060045659E0

8.2.2 ASIN

対象データ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。

(1) 指定形式

スカラ関数ASIN：：＝ASIN(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

逆正弦を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，-1～1 の値を指定してください。範囲外の値は指定できません。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列～C3 列の値の逆正弦を求めます。

```
SELECT ASIN("C1"),ASIN("C2"),ASIN("C3") FROM "T1"
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
0	-0.15	2.0000000000000001E-1

検索結果

0.0000000000000000E0	-1.5056827277668602E-1	2.0135792079033080E-1

8.2.3 ATAN

対象データ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。

(1) 指定形式

スカラ関数ATAN：：＝ATAN(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

逆正接を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列～C3 列の値の逆正接を求めます。

```
SELECT ATAN("C1"),ATAN("C2"),ATAN("C3") FROM "T1"
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
0	-0.15	2.0000000000000001E-1

検索結果

0.0000000000000000E0	-1.4888994760949725E-1	1.9739555984988078E-1

8.2.4 ATAN2

y/x 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，*y* を*対象データ 1* とし，*x* を*対象データ 2* とし，2 つの対象データの符号によって実行結果の値の象限が決まります。

(1) 指定形式

スカラ関数ATAN2：：＝ATAN2(*対象データ1*,*対象データ2*)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2) 指定形式の説明

対象データ 1 および*対象データ 2*：

y/x の逆正接を求める数データを指定します。

指定規則を次に示します。

- *対象データ 1* および*対象データ 2* は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *対象データ 1* および*対象データ 2* には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- *対象データ 1* および*対象データ 2* には，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *対象データ 1* または*対象データ 2* のどちらかがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列とC2 列の値を*対象データ 1* と*対象データ 2* に指定して，*y/x* の逆正接を求めます。

```
SELECT ATAN2(”C1”,”C2”) FROM ”T1”
```

表T1

C1列 DOUBLE PRECISION	C2列 DOUBLE PRECISION
2.9999999999999999E-1	2.0000000000000001E-1

検索結果

9.8279372324732905E-1

8.2.5 COS

ラジアン単位で指定された対象データの余弦（三角関数の COS）を返します。

(1) 指定形式

スカラ関数COS：：＝COS(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

余弦を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列の値をラジアン単位にして，その値に対する余弦を求めます。

```
SELECT COS(”C1”*PI()/180) FROM ”T1”
```

表T1

C1列 INTEGER
0
60
90

検索結果

1.0000000000000000E0
5.0000000000000011E-1
6.1232339957367660E-17

8.2.6 COSH

対象データの双曲線余弦を返します。

(1) 指定形式

スカラ関数COSH：：＝COSH(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

双曲線余弦を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. DOUBLE PRECISION 型で実行結果を表現できない場合，オーバフローエラーになります。

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

4. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列～C3 列の値の双曲線余弦を求めます。

```
SELECT COSH("C1"),COSH("C2"),COSH("C3") FROM "T1"
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
0	-0.15	2.0000000000000001E-1

検索結果

1.0000000000000000E0	1.0112711095766704E0	1.0200667556190759E0

8.2.7 DEGREES

指定された角度をラジアンから度数に変換して返します。

(1) 指定形式

スカラ関数DEGREES：：＝DEGREES(*角度*)

角度：：＝*値式*

(2) 指定形式の説明

角度：

角度をラジアン単位で指定します。

指定規則を次に示します。

- 角度は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 角度には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 角度には，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。

2. DOUBLE PRECISION 型で実行結果を表現できない場合，オーバフローエラーになります。

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 角度がナル値の場合，実行結果はナル値になります。

(4) 例題

例題 1

円周率 π を度数に変換します。

```
SELECT DEGREES(PI()) FROM "T1"
```

検索結果

1.8000000000000000E2

例題 2

表T1 のC1 列およびC2 列の値を入力して，スカラ関数ATAN2 で逆正接の角度（ラジアン単位）を求め，それをスカラ関数DEGREES で度数に変換します。

```
SELECT DEGREES(ATAN2("C1","C2")) FROM "T1"
```

表T1

C1列 DOUBLE PRECISION	C2列 DOUBLE PRECISION
1.0000000000000000E0	1.0000000000000000E0

検索結果

4.5000000000000000E1

8.2.8 PI

円周率 π の値を返します。

(1) 指定形式

```
スカラ関数PI::=PI()
```

(2) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。ただし，常に円周率の値を返すため，ナル値を返すことはありません。

(3) 例題

例題

表T1 のC1 列の値を半径とする円の円周を求めます。

```
SELECT "C1"*2*PI() FROM "T1"
```

表T1

C1列 INTEGER
2
10

検索結果

1.2566370614359172E1
6.2831853071795862E1

8.2.9 RADIANS

指定された角度を度数からラジアンに変換して返します。

(1) 指定形式

スカラ関数RADIANS：：＝RADIANS(*角度*)

角度：：＝*値式*

(2) 指定形式の説明

角度：

角度を度数単位で指定します。

指定規則を次に示します。

- 角度は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 角度には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 角度には，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 角度がナル値の場合，実行結果はナル値になります。

(4) 例題

例題 1

表T1 のC1 列～C3 列の値（角度）を，度数からラジアンに変換します。

```
SELECT RADIANS(”C1”),RADIANS(”C2”),RADIANS(”C3”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
0	0.15	4.50000000000000000E1

検索結果

0.00000000000000000E0	2.6179938779914941E-3	7.8539816339744828E-1

例題 2

表T1 のC1 列の値（角度）の余弦を求めます。

なお，C1 列の値の単位は度数のため，スカラ関数RADIANS で角度を度数からラジアンに変換し，スカラ関数COS で余弦を求めます。

```
SELECT COS(RADIANS(”C1”)) FROM ”T1”
```

表T1

C1列 DOUBLE PRECISION
6.00000000000000000E1

検索結果

5.0000000000000011E-1

この例では，COS(60°)を求めています。

スカラ関数COS に指定する対象データは，ラジアン単位で指定する必要があります。

8.2.10 SIN

ラジアン単位で指定された対象データの正弦（三角関数の SIN）を返します。

(1) 指定形式

スカラ関数SIN：：＝SIN(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

正弦を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4)　例題

例題

表T1 のC1 列の値をラジアン単位にして，その値に対する正弦を求めます。

```
SELECT SIN(”C1”*PI()/180) FROM ”T1”
```

表T1

C1列 INTEGER
0
60
90

検索結果

0.0000000000000000E0
8.6602540378443860E-1
1.0000000000000000E0

8.2.11　SINH

対象データの双曲線正弦を返します。

(1)　指定形式

スカラ関数SINH：：＝SINH(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

双曲線正弦を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. DOUBLE PRECISION 型で実行結果を表現できない場合，オーバフローエラーになります。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列～C3 列の値の双曲線正弦を求めます。

```
SELECT SINH(”C1”),SINH(”C2”),SINH(”C3”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
0	-0.15	2.0000000000000001E-1

検索結果

0.0000000000000000E0	-1.5056313315161265E-1	2.0133600254109402E-1

8.2.12 TAN

ラジアン単位で指定された対象データの正接（三角関数の TAN）を返します。

(1) 指定形式

スカラ関数TAN：：＝TAN(*対象データ*)

対象データ：：＝*値式*

(2)　指定形式の説明

対象データ：

正接を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. DOUBLE PRECISION 型で実行結果を表現できない場合，オーバフローエラーになります。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 対象データがナル値の場合，実行結果はナル値になります。

(4)　例題

例題

表T1 のC1 列の値をラジアン単位にして，その値に対する正接を求めます。

```
SELECT TAN(”C1”*PI()/180) FROM ”T1”
```

表T1

C1列 INTEGER
0
60
90

検索結果

0.0000000000000000E0
1.7320508075688767E0
1.6331239353195370E16

8.2.13　TANH

対象データの双曲線正接を返します。

(1)　指定形式

スカラ関数TANH：：＝TANH(*対象データ*)

対象データ：：＝*値式*

(2)　指定形式の説明

対象データ：

双曲線正接を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4)　例題

例題

表T1 のC1 列～C3 列の値の双曲線正接を求めます。

```
SELECT TANH(”C1”),TANH(”C2”),TANH(”C3”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
0	-0.15	2.0000000000000001E-1

検索結果

0.0000000000000000E0	-1.4888503362331798E-1	1.9737532022490401E-1

8.3　数学関数（指数・対数）

ここでは，指数・対数に関する数学関数の機能と指定形式について説明します。

8.3.1　EXP

自然対数の底の値の累乗を返します。

(1)　指定形式

```
スカラ関数EXP：：＝EXP(指数)
  指数：：＝値式
```

(2)　指定形式の説明

指数：

指数を指定します。

指定規則を次に示します。

- 指数は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 指数には，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1) 数データ」を参照してください。
- 指数には，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. DOUBLE PRECISION 型で実行結果を表現できない場合，オーバフローエラーになります。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 指数がナル値の場合，実行結果はナル値になります。

(4)　例題

例題

自然対数の底を累乗した値を求めます。表T1 のC1 列～C3 列の値を指数に指定します。

```
SELECT EXP("C1"),EXP("C2"),EXP("C3") FROM "T1"
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
0	-0.15	2.0000000000000001E-1

検索結果

1.0000000000000000E0	8.6070797642505781E-1	1.2214027581601699E0

8.3.2 LN

対象データの自然対数を返します。

(1) 指定形式

スカラ関数LN：：＝LN(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

自然対数を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，正の値を指定してください。0 以下の値は指定できません。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列～C3 列の値の自然対数を求めます。

```
SELECT LN("C1"),LN("C2"),LN("C3") FROM "T1"
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
1	3.15	2.0000000000000001E-1

検索結果

0.0000000000000000E0	1.1474024528375417E0	-1.6094379124341003E0

8.3.3 LOG

底と真数を指定して，その対数を返します。

(1) 指定形式

スカラ関数LOG：：＝LOG(*底*,*対象データ*)

底：：＝*値式*
対象データ：：＝*値式*

(2) 指定形式の説明

底：

対数の底を指定します。

指定規則を次に示します。

- 底は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 底には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 底には，0 以下の値を指定できません。
- 底に1 を指定した場合，0 除算エラーになります。
- 底には，？パラメタを単独で指定できません。

対象データ：

対象データ（真数）を指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，0 以下の値を指定できません。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 底または対象データがナル値の場合，実行結果はナル値になります。
4. LOG(*底*,*対象データ*)の結果は，LN(*対象データ*)/LN(*底*)の結果と同じになります。

(4) 例題

例題 1

表T1 のC1 列の値の常用対数を求めます。

```
SELECT LOG(10,"C1") FROM "T1"
```

表T1

C1列 INTEGER
10
100
574266

検索結果

1.0000000000000000E0
2.0000000000000000E0
5.7591131041977697E0

例題 2

表T1 のC2 列の値の対数を求めます。C1 列の値を底とします。

```
SELECT LOG("C1","C2") FROM "T1"
```

表T1

C1列 INTEGER	C2列 INTEGER
2	8
3	3
10	1056372

検索結果

3.0000000000000000E0
1.0000000000000000E0
6.0238168813585791E0

8.3.4 POWER

対象データの累乗を返します。

(1)　指定形式

```
スカラ関数POWER：：＝POWER(対象データ,指数)

  対象データ：：＝値式
  指数：：＝値式
```

(2)　指定形式の説明

対象データ：

累乗を求める対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

指数：

指数を指定します。

指定規則を次に示します。

- 指数は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 指数には，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- 指数には，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. DOUBLE PRECISION 型で実行結果を表現できない場合，オーバフローエラーになります。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 対象データまたは指数がナル値の場合，実行結果はナル値になります。
5. 対象データに負の値を指定した場合，指数には整数を指定してください。
6. 対象データに0 を指定した場合，指数には正の値を指定してください。違反した場合，0 除算エラーになります。

(4)　例題

例題 1

表T1 のC1 列の値の 2 乗を求めます。

```
SELECT POWER(”C1”,2) FROM ”T1”
```

表T1

C1列 INTEGER
2
10
-3
5

検索結果

4.0000000000000000E0
1.0000000000000000E2
9.0000000000000000E0
2.5000000000000000E1

例題 2

表T1 のC1 列の値を対象データ，C2 列の値を指数として，対象データの累乗を求めます。

```
SELECT POWER(”C1”,”C2”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 INTEGER
2	3
10	8
-3	20
5	-2

検索結果

8.0000000000000000E0
1.0000000000000000E8
3.4867844010000000E9
4.0000000000000001E-2

8.4　数学関数（数値計算）

ここでは，数値計算に関する数学関数の機能と指定形式について説明します。

8.4.1　ABS

対象データの絶対値を返します。

(1)　指定形式

```
スカラ関数ABS::=ABS(対象データ)
  対象データ::=値式
```

(2)　指定形式の説明

対象データ：

絶対値を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3)　規則

1. 対象データのデータ型とデータ長が，実行結果のデータ型とデータ長になります。
2. 対象データのデータ型で実行結果を表現できない場合，オーバフローエラーになります。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 対象データがナル値の場合，実行結果はナル値になります。

(4)　例題

例題

表T1 のC1 列～C3 列の値の絶対値を求めます。

```
SELECT ABS("C1"),ABS("C2"),ABS("C3") FROM "T1"
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
268	7.25	1.0050000000000000E3
-475	-2.28	-3.2550000000000000E2

検索結果

268	7.25	1.0050000000000000E3
475	2.28	3.2550000000000000E2

8.4.2 CEIL

対象データ以上の値で，最小の整数値を返します。

(1) 指定形式

スカラ関数CEIL ：：＝CEIL(*対象データ*)

対象データ ：：＝*値式*

(2) 指定形式の説明

対象データ：

処理対象の数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型を次の表に示します。

表 8-2 スカラ関数 CEIL の実行結果のデータ型

対象データのデータ型	実行結果のデータ型
INTEGER	INTEGER
SMALLINT	SMALLINT
DECIMAL(*p*,*s*)	DECIMAL(*p*,0)
NUMERIC(*p*,*s*)	
DOUBLE PRECISION	DOUBLE PRECISION

対象データのデータ型	実行結果のデータ型
FLOAT	

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列の値に対して，その値以上で最小の整数値を求めます。同様にC2 列の値，C3 列の値に対しても，その値以上で最小の整数値を求めます。

```
SELECT CEIL(”C1”),CEIL(”C2”),CEIL(”C3”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
268	7.25	1.1500000000000000E-3
-475	-2.28	-3.2550000000000003E-2

検索結果

268	8.	1.0000000000000000E0
-475	-2.	0.0000000000000000E0

8.4.3 FLOOR

対象データ以下の値で，最大の整数値を返します。

(1) 指定形式

スカラ関数FLOOR：：＝FLOOR(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

処理対象の数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型を次の表に示します。

表 8-3　スカラ関数 FLOOR の実行結果のデータ型

対象データのデータ型	実行結果のデータ型
INTEGER	INTEGER
SMALLINT	SMALLINT
DECIMAL(*p*,*s*)	DECIMAL(*p*,0)
NUMERIC(*p*,*s*)	
DOUBLE PRECISION	DOUBLE PRECISION
FLOAT	

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列の値に対して，その値以下で最大の整数値を求めます。同様にC2 列の値，C3 列の値に対しても，その値以下で最大の整数値を求めます。

```
SELECT FLOOR("C1"),FLOOR("C2"),FLOOR("C3") FROM "T1"
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
268	7.25	1.1500000000000000E-3
-475	-2.28	-3.2550000000000003E-2

検索結果

268	7.	0.0000000000000000E0
-475	-3.	-1.0000000000000000E0

8.4.4 MOD

被除数を除数で割った余りを返します。

(1) 指定形式

```
スカラ関数MOD：：＝MOD(被除数,除数)
```

被除数：：＝*値式*
除数：：＝*値式*

(2) 指定形式の説明

被除数：

被除数を指定します。

指定規則を次に示します。

- 被除数は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 被除数には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 被除数には，？パラメタを単独で指定できません。

除数：

除数を指定します。

指定規則を次に示します。

- 除数は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 除数には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 除数には0 を指定できません。0 を指定した場合，0 除算エラーになります。
- 除数には，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型は，被除数および除数のデータ型によって，次の表のように決まります。

表 8-4 スカラ関数 MOD の実行結果のデータ型

被除数のデータ型	除数のデータ型	実行結果のデータ型
INTEGER	INTEGER	INTEGER
	SMALLINT	SMALLINT
	DECIMAL，NUMERIC	DECIMAL
	DOUBLE PRECISION，FLOAT	DOUBLE PRECISION
SMALLINT	INTEGER	INTEGER
	SMALLINT	SMALLINT
	DECIMAL，NUMERIC	DECIMAL
	DOUBLE PRECISION，FLOAT	DOUBLE PRECISION
DECIMAL(p,0) または	INTEGER	INTEGER

被除数のデータ型	除数のデータ型	実行結果のデータ型
NUMERIC(*p*,0)	SMALLINT	SMALLINT
	DECIMAL，NUMERIC	DECIMAL
	DOUBLE PRECISION，FLOAT	DOUBLE PRECISION
DECIMAL(*p*,*s*) または NUMERIC(*p*,*s*) *s*≧1 の場合	INTEGER	DECIMAL
	SMALLINT	
	DECIMAL，NUMERIC	
	DOUBLE PRECISION，FLOAT	DOUBLE PRECISION
DOUBLE PRECISION または FLOAT	INTEGER	DOUBLE PRECISION
	SMALLINT	
	DECIMAL，NUMERIC	
	DOUBLE PRECISION，FLOAT	

注

実行結果のデータ型がDECIMAL 型の場合，精度と位取りは次のように決まります。

精度（*p*）＝MIN（*py*−*sy*＋*s*，38）

位取り（*s*）＝MAX（*sx*，*sy*）

MOD(*x*,*y*)と指定した場合，*x* のデータ型をDECIMAL(*px*,*sx*)，*y* のデータ型をDECIMAL(*py*,*sy*)として計算してください。

また，*x* または*y* のデータ型がSMALLINT 型の場合はDECIMAL(10,0)，INTEGER 型の場合はDECIMAL(20,0)として計算してください。

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 被除数または除数がナル値の場合，実行結果はナル値になります。
4. 実行結果の値の符号は，被除数の値の符号と同じになります。
5. DOUBLE PRECISION 型またはFLOAT 型のデータは誤差を含むため，スカラ関数MOD でDOUBLE PRECISION 型またはFLOAT 型のデータを指定する場合は注意してください。例えば，次の実行結果は 0 になりません。

```
MOD(5.0E-1,1.0E-1) → 9.9999999999999978E-2
```

これは，0.1 が 2 進法では有限桁で表現できないことに起因しています（0.1 と1.0E-1 は正確には等しくありません）。剰余の結果として正確な値を必要とする場合は，DECIMAL 型またはNUMERIC 型を使用してください。

(4) 例題

例題 1

表T1 のC1 列の値を 3 で割った余りを求めます。

```
SELECT MOD(”C1”,3) FROM ”T1”
```

表T1

C1列 INTEGER
10
15
-7
-24

検索結果

1
0
-1
0

例題 2

表T1 のC1 列の値をC2 列の値で割った余りを求めます。

```
SELECT MOD(”C1”,”C2”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 DECIMAL(2,1)
10	4.0
15	5.0
-7	2.3
-24	-5.1

検索結果

2.0
0.0
-0.1
-3.6

8.4.5 RANDOM

最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。

なお，RANDOM のほかにも擬似乱数を返すスカラ関数が幾つかあります。擬似乱数を返すスカラ関数の仕様差を確認して，用途に合ったスカラ関数を使用してください。擬似乱数を返すスカラ関数の仕様差については，「(6) 擬似乱数を返すスカラ関数の一覧」を参照してください。

(1) 指定形式

```
スカラ関数RANDOM：：＝RANDOM(〔最小値,最大値〕)
```

最小値 ：：＝*値式*
最大値 ：：＝*値式*

(2) 指定形式の説明

最小値：

乱数を生成する範囲の最小値（この値を含む）を指定します。引数の指定を省略した場合，*最小値*には0 が仮定されます。

指定規則を次に示します。

- *最小値*は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *最小値*には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- *最小値*に？パラメタを単独で指定した場合，DOUBLE PRECISION 型が仮定されます。

最大値：

乱数を生成する範囲の最大値（この値を含まない）を指定します。引数の指定を省略した場合，*最大値*には1 が仮定されます。

指定規則を次に示します。

- *最大値*は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *最大値*には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- *最大値*に？パラメタを単独で指定した場合，DOUBLE PRECISION 型が仮定されます。

(3) 規則

1. 実行結果のデータ型は，DOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *最小値*または*最大値*がナル値であれば，実行結果もナル値になります。
4. *最小値*と*最大値*をそれぞれDOUBLE PRECISION 型に変換したあとで，実行結果を計算します。
5. *最小値*に指定した値A と，*最大値*に指定した値B の関係が，「$A > B$」を満たす場合，自動的に*最小値*と*最大値*を入れ替えます。そして，「B 以上A 未満」の範囲での一様分布に従う擬似乱数を返します。
6. *最小値*と*最大値*に同じ値を指定した場合，実行結果は*最小値*に指定した値になります。
7. 実行結果のデータ型で実行結果を表現できない場合，オーバフローエラーになります。
8. *最大値*に0 を指定した場合，実行結果に+0 を返すことがあります。

(4) 留意事項

このスカラ関数は，暗号用途への使用には適していません。

(5) 例題

例題

表T1 で，1 以上 10 未満の範囲での一様分布に従うDOUBLE PRECISION 型の値を求めます。

なお，SELECT 文を実行するたびに，検索結果の値が変わります。

```
SELECT RANDOM(1,10) FROM "T1"
```

表T1

C1列
1
2
3
4
5

検索結果の例

3.4152565556163950E0
5.1485283237583523E0
2.4197853386516375E0
8.0146660601528428E0
6.8577888277555266E0

(6) 擬似乱数を返すスカラ関数の一覧

擬似乱数を返すスカラ関数は，RANDOM のほかにも次の 3 つがあります。

- RANDOMCURSOR
- RANDOMROW
- RANDOM_NORMAL

各スカラ関数の仕様差を次の表に示します。用途に合ったスカラ関数を使用してください。

表 8-5 擬似乱数を返すスカラ関数の一覧

項番	項目	スカラ関数 RANDOM	スカラ関数 RANDOMCURSOR	スカラ関数 RANDOMROW	スカラ関数 RANDOM_NORMAL
1	形式	RANDOM([*最小値*，*最大値*]) *最小値*：：=*値式* *最大値*：：=*値式*	RANDOMCURSOR(*識別番号* [，*最小値*，*最大値*]) *識別番号*：：=*符号なし整数定数* *最小値*：：=*値指定* *最大値*：：=*値指定*	RANDOMROW(*識別番号* [，*最小値*，*最大値*]) *識別番号*：：=*符号なし整数定数* *最小値*：：=*値式* *最大値*：：=*値式*	RANDOM_NORMAL([*平均* μ，*標準偏差* σ]) *平均* μ：：=*値式* *標準偏差* σ：：=*値式*
2	擬似乱数の分布	一様分布	一様分布	一様分布	正規分布

項番	項目	スカラ関数 RANDOM	スカラ関数 RANDOMCURSOR	スカラ関数 RANDOMROW	スカラ関数 RANDOM_NORMAL
3	擬似乱数の範囲	最小値に指定した値以上，最大値に指定した値未満の範囲	最小値に指定した値以上，最大値に指定した値未満の範囲	最小値に指定した値以上，最大値に指定した値未満の範囲	平均 μ，標準偏差 σ に従う値
4	SQL 文中で同じ識別番号となるスカラ関数は常に同じ値を返す	－	○	×	－
5	問合せ指定中で同じ識別番号となるスカラ関数は行ごとに同じ値を返す	－	○	○	－
6	指定できる位置	*値式*	• 選択式の*値式* • ORDER BY 句※	• 選択式の*値式* • ORDER BY 句※	*値式*

（凡例）

○：該当します。

×：該当しません。

－：対象外です。識別番号は指定できません。

注※

WITHIN グループ指定，およびウィンドウ順序句中のORDER BY 句には指定できません。

8.4.6 RANDOMCURSOR

RANDOMCURSOR は，次の条件を満たす値を返します。

- 最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。
- 検索系 SQL の場合，カーソルがオープンしている間は，常に同じ値を返します。更新系 SQL の場合は，SQL 文を実行している間は，常に同じ値を返します。
- 1SQL 文中で，同じ識別番号を指定したRANDOMCURSOR は，常に同じ値を返します。

なお，RANDOMCURSOR のほかにも擬似乱数を返すスカラ関数が幾つかあります。擬似乱数を返すスカラ関数の仕様差を確認して，用途に合ったスカラ関数を使用してください。擬似乱数を返すスカラ関数の仕様差については，「8.4.5 RANDOM」の「(6) 擬似乱数を返すスカラ関数の一覧」を参照してください。

(1) 指定形式

スカラ関数RANDOMCURSOR：：＝RANDOMCURSOR(*識別番号*〔,*最小値*,*最大値*〕)

識別番号：：＝*符号なし整数定数*

最小値：：＝*値指定*
最大値：：＝*値指定*

(2) 指定形式の説明

識別番号：

1 から1000 までの整数値を指定します。1SQL 文中で，同じ識別番号を指定したRANDOMCURSOR は，常に同じ値を返します。

最小値：

乱数を生成する範囲の最小値（この値を含む）を指定します。*最小値*と*最大値*の指定を省略した場合，*最小値*には0 が仮定されます。

指定規則を次に示します。

- *最小値*は，値指定の形式で指定します。値指定については，「7.22　値指定」を参照してください。
- *最小値*には，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- *最小値*に？パラメタを指定した場合，DOUBLE PRECISION 型が仮定されます。

最大値：

乱数を生成する範囲の最大値（この値を含まない）を指定します。*最小値*と*最大値*の指定を省略した場合，*最大値*には1 が仮定されます。

指定規則を次に示します。

- *最大値*は，値指定の形式で指定します。値指定については，「7.22　値指定」を参照してください。
- *最大値*には，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- *最大値*に？パラメタを指定した場合，DOUBLE PRECISION 型が仮定されます。

(3) 規則

1. このスカラ関数は，次の個所に指定できます。
   - 問合せ指定の選択式
   - ORDER BY 句（WITHIN グループ指定，およびウィンドウ順序句中のORDER BY 句には指定できません）
2. 同じ識別番号を指定したRANDOMCURSOR を 1SQL 文中に複数指定する場合，次のどちらかの規則に従って指定してください。
   - 最小値と最大値を指定したRANDOMCURSOR を 1 つだけ指定し，それ以外のすべてのRANDOMCURSOR は最小値と最大値を省略してください。

     （例）正しい SQL 文の例

```
SELECT "C1"+RANDOMCURSOR(1,10,20),"C2"+RANDOMCURSOR(1) FROM "T1"
UNION ALL
SELECT "C3"+RANDOMCURSOR(1),"C4"+RANDOMCURSOR(1) FROM "T2"
```

（例）エラーになるSQL文の例

```
SELECT ”C1”+RANDOMCURSOR(1,10,20),”C2”+RANDOMCURSOR(1) FROM ”T1”
UNION ALL
SELECT ”C3”+RANDOMCURSOR(1,10,20),”C4”+RANDOMCURSOR(1) FROM ”T2”
```

- すべてのRANDOMCURSORの最小値と最大値を省略してください。

（例）正しいSQL文の例

```
SELECT ”C1”+RANDOMCURSOR(1),”C2”+RANDOMCURSOR(1) FROM ”T1”
UNION ALL
SELECT ”C3”+RANDOMCURSOR(1),”C4”+RANDOMCURSOR(1) FROM ”T2”
```

3. 1SQL文中で，同じ識別番号を指定したRANDOMCURSORは，常に同じ値を返します。

（例）

```
SELECT
  ”C1”+ RANDOMCURSOR(1,10,20),    ...[a]
  ”C2”+ RANDOMCURSOR(1),          ...[a]
  ”C3”+ RANDOMCURSOR(2),          ...[b]
  ”C4”+ RANDOMCURSOR(2)           ...[b]
FROM ”T1”
UNION ALL
SELECT
  ”C1”+ RANDOMCURSOR(1),          ...[a]
  ”C2”+ RANDOMCURSOR(1),          ...[a]
  ”C3”+ RANDOMCURSOR(2,20,30),    ...[b]
  ”C4”+ RANDOMCURSOR(2)           ...[b]
FROM ”T2”
```

[説明]

a. 識別番号に1を指定したRANDOMCURSORです。それぞれの関数は，常に同じ値（10以上20未満の値）を返します。

b. 識別番号に2を指定したRANDOMCURSORです。それぞれの関数は，常に同じ値（20以上30未満の値）を返します。

4. 次に示す識別番号が重複した場合，重複した識別番号をHADBが自動的に変更します。そのため，SQL文はエラーになりません。

- ビュー表の定義時（CREATE VIEW文）に指定した識別番号
- ビュー表を指定したSQL文中に指定した識別番号

（例）

ビュー表V1の定義

```
CREATE VIEW ”V1”(”VC1”,”VC2”) AS
  SELECT ”C1”+RANDOMCURSOR(1,10,20),”C2”+RANDOMCURSOR(1) FROM ”T1”
```

ビュー表V1を検索するSQL文

```
SELECT ”VC1”+RANDOMCURSOR(1,10,20),”VC2”+RANDOMCURSOR(1) FROM ”V1”
```

上記のSELECT文は，HADBが次のように等価変換します。

```
SELECT "VC1"+RANDOMCURSOR(1,10,20),"VC2"+RANDOMCURSOR(1)
  FROM (SELECT "C1"+RANDOMCURSOR(2,10,20),"C2"+RANDOMCURSOR(2)
            FROM "T1") "V1"("VC1","VC2")
```

下線部分の識別番号を，1 から2 に HADB が自動的に変更するため，このSELECT 文はエラーになりません。このように，ビュー表の定義時に指定した識別番号を HADB が自動的に変更します。

5. 1SQL 文中に指定できるRANDOMCURSOR の識別番号の種類の上限は，1,000 となります。SQL 文中にビュー表を指定している場合，CREATE VIEW 文に指定したRANDOMCURSOR の識別番号の種類の数も加算されて合計した数の上限が 1,000 となります。

6. RANDOMCURSOR とRANDOMROW に同じ識別番号を指定できます。この場合，各スカラ関数は個別に生成した擬似乱数を返します。

(例)

```
SELECT RANDOMCURSOR(1,10,20) AS "C1",
       RANDOMROW(1,10,20) AS "C2"
    FROM "T1"
```

実行結果の例

C1	C2
+1.2475764960039722E+01	+1.7828308131439851E+01
+1.2475764960039722E+01	+1.5309877916946510E+01
+1.2475764960039722E+01	+1.3148733592755859E+01

7. カーソルオープン時に擬似乱数が生成されるため，カーソルオープンのたびにRANDOMCURSOR の結果が変わります。

8. 実行結果のデータ型は，DOUBLE PRECISION 型になります。

9. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

10. *最小値*または*最大値*がナル値の場合，実行結果もナル値になります。

11. *最小値*と*最大値*をそれぞれDOUBLE PRECISION 型に変換したあとで，実行結果を計算します。

12. *最小値*に指定した値A と，*最大値*に指定した値B の関係が，「$A > B$」を満たす場合，自動的に*最小値*と*最大値*を入れ替えます。そして，「B 以上A 未満」の範囲での一様分布に従う擬似乱数を返します。

13. *最小値*と*最大値*に同じ値を指定した場合，実行結果は*最小値*に指定した値になります。

14. 実行結果のデータ型で実行結果を表現できない場合，オーバフローエラーになります。

15. *最大値*に0 を指定した場合，実行結果に+0 を返すことがあります。

(4) 留意事項

このスカラ関数は，暗号用途への使用には適していません。

(5) 例題

例題

患者の入院日と退院日を，次の条件を満たすように加工します。

- 入院期間が変わらないように，入院日と退院日を加工します。入院日と退院日に同じ日数を加算し，加工後の入院日，加工後の退院日にします。
- 加工後の入院日，加工後の退院日には，0～6 日の同じだけの日数を加算します。
- 検索結果は，加工後の入院日順に並べます。

(例)

```
SELECT ”患者ID”,”入院日”,”退院日”,
       ”入院日”+CAST(RANDOMCURSOR(1,0,7) AS INTEGER) DAY AS ”加工後の入院日”,
       ”退院日”+CAST(RANDOMCURSOR(1) AS INTEGER) DAY AS ”加工後の退院日”
    FROM ”入院履歴表”
      ORDER BY ”入院日”+CAST(RANDOMCURSOR(1) AS INTEGER) DAY
```

実行結果の例

患者ID	入院日	退院日	加工後の入院日	加工後の退院日
U0003	2018-04-01	2018-04-11	2018-04-04	2018-04-14
U0004	2018-05-01	2018-05-11	2018-05-04	2018-05-14
U0001	2018-06-01	2018-06-11	2018-06-04	2018-06-14
U0002	2018-07-01	2018-07-11	2018-07-04	2018-07-14

8.4.7 RANDOMROW

RANDOMROW は，次の条件を満たす値を返します。

- 最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。
- 1 問合せ指定中で，同じ識別番号を指定したRANDOMROW は，問合せ指定の結果の行ごとに同じ値を返します。

なお，RANDOMROW のほかにも擬似乱数を返すスカラ関数が幾つかあります。擬似乱数を返すスカラ関数の仕様差を確認して，用途に合ったスカラ関数を使用してください。擬似乱数を返すスカラ関数の仕様差については，「8.4.5 RANDOM」の「(6) 擬似乱数を返すスカラ関数の一覧」を参照してください。

(1) 指定形式

```
スカラ関数RANDOMROW : : =RANDOMROW(識別番号 〔,最小値,最大値〕 )

  識別番号 : : =符号なし整数定数
  最小値 : : =値式
  最大値 : : =値式
```

(2) 指定形式の説明

識別番号：

1 から1000 までの整数値を指定します。1 問合せ指定中で，同じ識別番号を指定したRANDOMROW は，問合せ指定の結果の行ごとに同じ値を返します。

最小値：

乱数を生成する範囲の最小値（この値を含む）を指定します。*最小値*と*最大値*の指定を省略した場合，*最小値*には0 が仮定されます。

指定規則を次に示します。

- *最小値*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *最小値*には，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- *最小値*に？パラメタを単独で指定した場合，DOUBLE PRECISION 型が仮定されます。

最大値：

乱数を生成する範囲の最大値（この値を含まない）を指定します。*最小値*と*最大値*の指定を省略した場合，*最大値*には1 が仮定されます。

指定規則を次に示します。

- *最大値*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *最大値*には，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- *最大値*に？パラメタを単独で指定した場合，DOUBLE PRECISION 型が仮定されます。

(3) 規則

1. このスカラ関数は，次の個所に指定できます。
   - 問合せ指定の選択式
   - ORDER BY 句（WITHIN グループ指定，およびウィンドウ順序句中のORDER BY 句には指定できません）
2. RANDOMROW 中の値式に，RANDOMROW を指定することはできません。

   ただし，RANDOMROW に指定した副問合せ中には，RANDOMROW を指定できます。

   （例）エラーになる SQL 文の例

   ```
   RANDOMROW(1,0,RANDOMROW(1))
   ```

   （例）正しい SQL 文の例

   ```
   RANDOMROW(1,0,(SELECT RANDOMROW(1) FROM "T1"))
   ```

3. 同じ識別番号を指定したRANDOMROW を 1 問合せ指定中に複数指定する場合，次のどちらかの規則に従って指定してください。

- 最小値と最大値を指定したRANDOMROW を 1 つだけ指定し，それ以外のすべてのRANDOMROW は最小値と最大値を省略してください。

  （例）正しい SQL 文の例

```
SELECT ”C1”+RANDOMROW(1,10,20),”C2”+RANDOMROW(1),”C3”+RANDOMROW(1) FROM ”T1”
```

  （例）エラーになる SQL 文の例

```
SELECT ”C1”+RANDOMROW(1,10,20),”C2”+RANDOMROW(1,10,20),”C3”+RANDOMROW(1) FROM ”T1”
```

- すべてのRANDOMROW の最小値と最大値を省略してください。

  （例）正しい SQL 文の例

```
SELECT ”C1”+RANDOMROW(1),”C2”+RANDOMROW(1),”C3”+RANDOMROW(1) FROM ”T1”
```

4. 問合せ指定の結果の行ごとに擬似乱数が生成されるため，問合せ指定の結果の行ごとにRANDOMROW の結果が変わります。

5. 同じ識別番号を指定したRANDOMROW は，問合せ指定の結果の行ごとに同じ値を返します。

   （例）

```
SELECT
  ”C1”+ RANDOMROW(1,10,20),    ...[a]
  ”C2”+ RANDOMROW(1),          ...[a]
  ”C3”+ RANDOMROW(2,20,30),    ...[b]
  ”C4”+ RANDOMROW(2)           ...[b]
FROM ”T1”
UNION ALL
SELECT
  ”C1”+ RANDOMROW(1,10,20),    ...[c]
  ”C2”+ RANDOMROW(1),          ...[c]
  ”C3”+ RANDOMROW(2),          ...[d]
  ”C4”+ RANDOMROW(2)           ...[d]
FROM ”T2”
```

   [説明]

   a. 識別番号に1 を指定したRANDOMROW です。問合せ指定の結果の行ごとに同じ値（10 以上 20 未満の値）を返します。

   b. 識別番号に2 を指定したRANDOMROW です。問合せ指定の結果の行ごとに同じ値（20 以上 30 未満の値）を返します。

   c. 識別番号に1 を指定したRANDOMROW です。問合せ指定の結果の行ごとに同じ値（10 以上 20 未満の値）を返します。

   d. 識別番号に2 を指定したRANDOMROW です。問合せ指定の結果の行ごとに同じ値（0 以上 1 未満の値）を返します。

6. 次の内部導出表が展開された場合，内部導出表の導出問合せ中のRANDOMROW の識別番号は，内部導出表を指定した問合せ指定中のRANDOMROW の識別番号と重複しないように HADB が自動的に変更します。

   - 導出問合せ中にRANDOMROW を指定した内部導出表

   （例）

導出表を指定したSQL文

```
SELECT "DC1"+RANDOMROW(1,10,20),"DC2"+RANDOMROW(1)
    FROM (SELECT "C1"+RANDOMROW(1,20,30),"C2"+RANDOMROW(1)
              FROM "T1") "DT"("DC1","DC2")
```

導出表を展開したSQL文

```
SELECT "C1"+RANDOMROW(2,20,30)+RANDOMROW(1,10,20),
       "C2"+RANDOMROW(2)+RANDOMROW(1)
    FROM "T1"
```

導出表"DT"中のRANDOMROWは，次のように識別番号が変更されます。

- RANDOMROW(1,20,30) → RANDOMROW(2,20,30)
- RANDOMROW(1) → RANDOMROW(2)

7. 同じ識別番号のRANDOMROWを異なる問合せ指定に指定できます。ただし，それぞれのRANDOMROWの識別番号の種類は異なるものとして扱われます。
8. SQL文中に指定できる識別番号の種類の数の上限は，1,000となります。ただし，SQL文中にビュー表を指定している場合は，そのビュー表が導出表に等価変換されたあとに，識別番号の種類の数がチェックされます。
9. RANDOMCURSORとRANDOMROWに同じ識別番号を指定できます。この場合，各スカラ関数は個別に生成した擬似乱数を返します。

   (例)

```
SELECT RANDOMCURSOR(1,10,20) AS "C1",
       RANDOMROW(1,10,20) AS "C2"
    FROM "T1"
```

実行結果の例

C1	C2
+1.2475764960039722E+01	+1.7828308131439851E+01
+1.2475764960039722E+01	+1.5309877916946510E+01
+1.2475764960039722E+01	+1.3148733592755859E+01

10. 実行結果のデータ型は，DOUBLE PRECISION型になります。
11. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
12. *最小値*または*最大値*がナル値の場合，実行結果もナル値になります。
13. *最小値*と*最大値*をそれぞれDOUBLE PRECISION型に変換したあとで，実行結果を計算します。
14. *最小値*に指定した値Aと，*最大値*に指定した値Bの関係が，「$A > B$」を満たす場合，自動的に*最小値*と*最大値*を入れ替えます。そして，「B以上A未満」の範囲での一様分布に従う擬似乱数を返します。
15. *最小値*と*最大値*に同じ値を指定した場合，実行結果は*最小値*に指定した値になります。
16. 実行結果のデータ型で実行結果を表現できない場合，オーバフローエラーになります。
17. *最大値*に0を指定した場合，実行結果に+0を返すことがあります。

(4) 留意事項

このスカラ関数は，暗号用途への使用には適していません。

(5) 例題

例題

患者の入院日と退院日を，次の条件を満たすように加工します。

- 入院期間が変わらないように，入院日と退院日を加工します。入院日と退院日に同じ日数を加算し，加工後の入院日，加工後の退院日にします。
- 加工後の入院日，加工後の退院日には，0～6 日の同じだけの日数を加算します。また，患者ごとに加算する日数を変えます。
- 検索結果は，加工後の入院日順に並べます。

(例)

```
SELECT ”患者ID”,”入院日”,”退院日”,
       ”入院日”+CAST(RANDOMROW(1,0,7) AS INTEGER) DAY AS ”加工後の入院日”,
       ”退院日”+CAST(RANDOMROW(1) AS INTEGER) DAY AS ”加工後の退院日”
    FROM ”入院履歴表”
      ORDER BY ”入院日”+CAST(RANDOMROW(1) AS INTEGER) DAY
```

患者ID	入院日	退院日	加工後の入院日	加工後の退院日
U0003	2018-04-01	2018-04-11	2018-04-04	2018-04-14
U0004	2018-05-01	2018-05-11	2018-05-06	2018-05-16
U0001	2018-06-01	2018-06-11	2018-06-07	2018-06-17
U0002	2018-07-01	2018-07-11	2018-07-03	2018-07-13

8.4.8 RANDOM_NORMAL

平均 μ，標準偏差 σ の正規分布に従う擬似乱数を返します。

なお，RANDOM_NORMAL のほかにも擬似乱数を返すスカラ関数が幾つかあります。擬似乱数を返すスカラ関数の仕様差を確認して，用途に合ったスカラ関数を使用してください。擬似乱数を返すスカラ関数の仕様差については，「8.4.5 RANDOM」の「(6) 擬似乱数を返すスカラ関数の一覧」を参照してください。

(1) 指定形式

```
スカラ関数RANDOM_NORMAL ：：＝RANDOM_NORMAL(〔平均μ,標準偏差σ〕)

  平均μ ：：＝値式
  標準偏差σ ：：＝値式
```

(2) 指定形式の説明

平均 μ：

平均 μ を指定します。引数の指定を省略した場合，*平均 μ* には0 が仮定されます。

指定規則を次に示します。

- *平均 μ* は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *平均 μ* には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- *平均 μ* に？パラメタを単独で指定した場合，DOUBLE PRECISION 型が仮定されます。

標準偏差 σ：

標準偏差 σ を指定します。引数の指定を省略した場合，*標準偏差 σ* には1（標準正規分布）が仮定されます。

指定規則を次に示します。

- *標準偏差 σ* は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *標準偏差 σ* には，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- *標準偏差 σ* には，0 以上の値を指定してください。
- *標準偏差 σ* に？パラメタを単独で指定した場合，DOUBLE PRECISION 型が仮定されます。

(3) 規則

1. 実行結果のデータ型は，DOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *平均 μ* または*標準偏差 σ* がナル値であれば，実行結果もナル値になります。
4. *平均 μ* と*標準偏差 σ* をそれぞれDOUBLE PRECISION 型に変換したあとで，実行結果を計算します。
5. 実行結果のデータ型で実行結果を表現できなくなった場合，オーバフローエラーになります。

(4) 留意事項

このスカラ関数は，暗号用途への使用には適していません。

(5) 例題

例題

表T1 で，*平均 μ* が 30 で*標準偏差 σ* が 20 の正規分布に従うDOUBLE PRECISION 型の値を求めます。

なお，SELECT 文を実行するたびに，検索結果の値が変わります。

```
SELECT RANDOM_NORMAL(30,20) FROM "T1"
```

表T1

C1列
1
2
3
4
5

検索結果の例

-1.2987575425050977E1
5.1920095220623608E1
2.3649975133459982E1
5.7409734912048918E1
3.2063684046363342E1

8.4.9 ROUND

対象データを小数点以下*n* 桁に丸めた値を返します。

日時データを丸めるスカラ関数ROUND については，「8.9.7 ROUND」を参照してください。

(1) 指定形式

```
スカラ関数ROUND：：＝ROUND(対象データ〔,桁数〕)

  対象データ：：＝値式
  桁数：：＝値式
```

(2) 指定形式の説明

対象データ：

数データ（小数点以下*n* 桁に丸める値）を指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

桁数：

桁数を指定します。

指定規則を次に示します。

- *桁数*は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。

- *桁数*には，データ型がINTEGER 型またはSMALLINT 型のデータを指定してください。
- *桁数*を省略した場合，*桁数*に 0 が仮定されます。
- *桁数*に？パラメタを指定した場合，？パラメタのデータ型にINTEGER 型が仮定されます。

(3) 規則

1. 実行結果のデータ型を次の表に示します。

表 8-6　スカラ関数 ROUND の実行結果のデータ型

対象データのデータ型		実行結果のデータ型
INTEGER		INTEGER
SMALLINT		SMALLINT
DECIMAL(p,s) または NUMERIC(p,s)	$p \leqq 37$ の場合	DECIMAL($p + 1$,s)
	$p = 38$ の場合	DECIMAL(38,s)
DOUBLE PRECISION またはFLOAT		DOUBLE PRECISION

2. 対象データのデータ型で実行結果を表現できない場合，オーバフローエラーになります。
3. 対象データのデータ型がSMALLINT 型，INTEGER 型，DECIMAL 型，またはNUMERIC 型の場合，$n + 1$ 桁の端数を四捨五入した値を返します。

   （例）ROUND(325.72,1) → 325.70

   なお，対象データが負の値の場合，次のように四捨五入されます。

   （例 1）ROUND(-2.3,0) → -2.0

   （例 2）ROUND(-2.7,0) → -3.0
4. 対象データがDOUBLE PRECISION 型またはFLOAT 型の場合，$n + 1$ 桁以下の端数に対して，最近接偶数への丸めを行った値を返します。
5. *桁数*に負の値を指定した場合は，整数部の指定桁以下を丸めます。

   （例）ROUND(325.72,-1) → 330.00
6. *桁数*を省略するか，または0 を指定した場合，小数点以下が丸められ，実行結果が整数になります。

   （例）ROUND(325.72,0) → 326.00
7. 対象データの範囲外の桁数を指定した場合，次のように処理されます。
   - *桁数*に正の値を指定した場合は，数値の丸めは実行されません。対象データの値をそのまま返します。

     （例 1）ROUND(0.12,5) → 0.12

     （例 2）ROUND(58,1) → 58
   - *桁数*に負の値を指定した場合は0 を返します。

     （例）ROUND(58,-5) → 0
8. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

9.*対象データ*または*桁数*がナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列～C3 列の値を次のように丸めます。

- C1 列：小数点第 1 位に丸めます（小数点第 2 位を四捨五入します)。
- C2 列：整数部の百の位に丸めます（十の位を四捨五入します)。
- C3 列：小数点第 1 位に最近接偶数への丸めを適用します。

この例では，HADB が動作する環境の丸めモードは，最近接偶数への丸めとします。

```
SELECT ROUND(”C1”,1),ROUND(”C2”,-2) ,ROUND(”C3”,0) FROM ”T1”
```

表T1

C1列 DECIMAL(4,3)	C2列 INTEGER	C3列 DOUBLE PRECISION
3.123	128	1.14000000000000000E1
-4.089	1051	1.15000000000000000E1
5.000	-565	1.16000000000000000E1
3.123	-1117	1.24000000000000000E1
-4.089	7	1.25000000000000000E1
5.001	-5	1.26000000000000000E1

検索結果

3.100	100	1.10000000000000000E1
-4.100	1100	1.20000000000000000E1
5.000	-600	1.20000000000000000E1
3.100	-1100	1.20000000000000000E1
-4.100	0	1.20000000000000000E1
5.000	0	1.30000000000000000E1

8.4.10 SIGN

対象データが正の値の場合は＋1 を，負の値の場合は-1 を，0 の場合は0 を返します。

(1) 指定形式

スカラ関数SIGN：：＝SIGN(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

処理対象の数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型は，対象データのデータ型によって，次の表のように決まります。

表 8-7 スカラ関数 SIGN の実行結果のデータ型

項番	対象データのデータ型	実行結果のデータ型
1	INTEGER	INTEGER
2	SMALLINT	SMALLINT
3	DECIMAL(*p*,*s*)	DECIMAL(1,0)
4	NUMERIC(*p*,*s*)	
5	DOUBLE PRECISION	DOUBLE PRECISION
6	FLOAT	

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列～C3 列の値が，正の値か，負の値か，または 0 かを求めます。

```
SELECT SIGN(”C1”),SIGN(”C2”),SIGN(”C3”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
268	7.25	1.1500000000000000E-3
-475	-2.28	-3.2550000000000003E-2
0	0.00	0.0000000000000000E0

検索結果

1	1.	1.0000000000000000E0
-1	-1.	-1.0000000000000000E0
0	0.	0.0000000000000000E0

8.4.11 SQRT

対象データの平方根を返します。

(1) 指定形式

スカラ関数SQRT：：＝SQRT(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

平方根を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1 データ型の種類」の「(1) 数データ」を参照してください。
- 対象データには，0 以上の値を指定してください。負の値は指定できません。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はDOUBLE PRECISION 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4)　例題

例題

表T1 のC1 列～C3 列の値の平方根を求めます。

```
SELECT SQRT("C1"),SQRT("C2"),SQRT("C3") FROM "T1"
```

表T1

C1列 INTEGER	C2列 DECIMAL(3,2)	C3列 DOUBLE PRECISION
9	5.15	2.1200000000000000E8

検索結果

3.0000000000000000E0	2.2693611435820435E0	1.4560219778561037E4

8.4.12　TRUNC

指定された桁数より下の数値を切り捨てた値を返します。

日時データを切り捨てるスカラ関数TRUNC については，「8.9.10　TRUNC」を参照してください。

(1)　指定形式

```
スカラ関数TRUNC：：＝TRUNC(対象データ〔,桁数〕)

  対象データ：：＝値式
  桁数：：＝値式
```

(2)　指定形式の説明

対象データ：

処理対象の数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，数データを指定してください。数データについては，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

桁数：

桁数を指定します。

指定規則を次に示します。

- *桁数*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *桁数*には，INTEGER 型またはSMALLINT 型のデータを指定してください。

*桁数*の指定値が正（*n*）の場合は，対象データの小数点第*n* 位を残して，小数点*n*+1 位以下が切り捨てられます。*桁数*の指定値が負（-*n*）の場合は，対象データの整数部*n* 桁以下が切り捨てられます。

*桁数*を省略した場合，0 が仮定されて小数点以下の値が切り捨てられます。

スカラ関数TRUNC の実行結果の例を次に示します。

（例）

数データ 123.456 の小数点第 2 位までを残し，第 3 位以下を切り捨てます。

TRUNC(123.456,2) → 123.450

(3) 規則

1. *桁数*に？パラメタを指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。
2. 実行結果のデータ型と長さは，引数に指定した*数データ*のデータ型と長さになります。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. *対象データ*または*桁数*がナル値の場合，実行結果はナル値になります。
5. DOUBLE PRECISION 型またはFLOAT 型のデータは誤差を含むため，スカラ関数TRUNC で使用する場合は注意が必要です。例えば，次のようなケースでは，計算結果が期待した結果になりません。

   TRUNC(2.172157E4,2) → 2.1721560000000001E4

   これは，2 進法では0.01 を有限桁で表現できないため，0.01 と1.0E-02 が正確には等しくないことが原因となっています。切り捨ての結果として正確な値が必要な場合は，対象データにはDECIMAL 型またはNUMERIC 型のデータを使用してください。
6. 引数に指定できるデータ型と*桁数*の有効範囲を次の表に示します。

表 8-8 引数に指定できるデータ型と桁数の有効範囲

項番	引数（数データ）に指定できるデータ型	桁数の有効範囲
1	INTEGER	-18～0
2	SMALLINT	-9～0
3	DECIMAL(*m*,*n*)	-(*m*-*n*-1)～*n*
4	NUMERIC(*m*,*n*)	
5	DOUBLE PRECISION	-308～323
6	FLOAT	

（凡例）*m*，*n*：正の整数

注

*桁数*の指定値が有効範囲外であっても，エラーとなりません。正の値で有効範囲外のときは，切り捨ては発生しません。負の値で有効範囲外のときは，結果は0 となります。

*桁数*の指定値と変換結果の例を次に示します。

例 1

C1 列のデータ型がINTEGER 型，値が123456789 の表T1 に対して，次の SQL 文を実行します。

```
SELECT TRUNC(”C1”,x) FROM ”T1”
```

このときのTRUNC(”C1”,*x*)の結果を次の表に示します。

表 8-9　x の値と SQL 文の実行結果

x の値	TRUNC(C1,x)の結果
1 以上	123456789
0	123456789
-1	123456780
-8	100000000
-9 以下	0

例 2

C2 列のデータ型がDECIMAL(5,2)，値が 123.45 の表T1 に対して，次の SQL 文を実行します。

```
SELECT TRUNC(”C2”,y) FROM ”T1”
```

このときのTRUNC(”C2”,*y*)の結果を次の表に示します。

表 8-10　y の値と SQL 文の実行結果

y の値	TRUNC(C2,y)の結果
3 以上	123.45
2	123.45
1	123.40
0	123.00
-1	120.00
-2	100.00
-3 以下	0.00

(4)　例題

例題 1

表T1 のC2 列およびC3 列のデータの小数点第 2 位までを残して，小数点第 3 位以下を切り捨てます。

```
SELECT TRUNC(”C2”,2),TRUNC(”C3”,2) FROM ”T1”
```

表T1

C2列 DECIMAL(8,4)	C3列 DOUBLE PRECISION
1502.4890	1.3547110000000000E1
217.3538	3.8875659999999999E2
738.6600	6.3456700000000001E3

検索結果

1502.4800	1.3539999999999999E1
217.3500	3.8875000000000000E2
738.6600	6.3456700000000001E3

例題 2

表T1 のC1 列～C3 列のデータの整数部 2 桁以下を切り捨てます。

```
SELECT TRUNC(”C1”,-2),TRUNC(”C2”,-2),TRUNC(”C3”,-2) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 DECIMAL(8,4)	C3列 DOUBLE PRECISION
78543	1502.4890	1.3547110000000000E1
44712	217.3538	3.8875659999999999E2
11475	738.6600	6.3456700000000001E3

検索結果

78500	1500.0000	0.0000000000000000E0
44700	200.0000	3.0000000000000000E2
11400	700.0000	6.3000000000000000E3

8.5　文字列関数（文字列操作）

ここでは，文字列操作に関する文字列関数の機能と指定形式について説明します。

8.5.1　CONCAT

2 つの文字データを連結します。

バイナリデータとバイナリデータを連結するスカラ関数については，「8.10.1　CONCAT」を参照してください。

(1)　指定形式

スカラ関数CONCAT：：＝CONCAT(*対象データ1*,*対象データ2*)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2)　指定形式の説明

対象データ 1 および*対象データ 2*：

連結する文字データを指定します。

指定規則を次に示します。

- *対象データ 1* および*対象データ 2* は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *対象データ 1* および*対象データ 2* には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *対象データ 1* および*対象データ 2* には，？パラメタを単独で指定できません。

スカラ関数CONCAT の実行結果の例を次に示します。

(例)

2 つの文字データ（ABC とXYZ）を連結します。

CONCAT('ABC','XYZ') → 'ABCXYZ'

(3)　規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-11　スカラ関数 CONCAT の実行結果のデータ型とデータ長

対象データ 1 のデータ型とデータ長	対象データ 2 のデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*m*)	CHAR(*n*)	CHAR(*m* ＋ *n*)
	VARCHAR(*n*) 実データ長：*L2*	VARCHAR(*m* ＋ *n*) 実データ長：*m* ＋ *L2*
VARCHAR(*m*) 実データ長：*L1*	CHAR(*n*)	VARCHAR(*m* ＋ *n*) 実データ長：*L1* ＋ *n*
	VARCHAR(*n*) 実データ長：*L2*	VARCHAR(*m* ＋ *n*) 実データ長：*L1* ＋ *L2*

（凡例）

m：*対象データ 1* の最大長

n：*対象データ 2* の最大長

L1：*対象データ 1* の実データ長

L2：*対象データ 2* の実データ長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

3. *対象データ 1* または*対象データ 2* のどちらかがナル値の場合，実行結果はナル値になります。

4. *対象データ 1* と*対象データ 2* を連結した結果，最大長が 32,000 バイトを超える文字データを連結することはできません。

5. CHAR 型のデータの末尾に半角空白が格納されている場合，その半角空白も連結の対象になります。

（例）

C1 列がCHAR(5)で値が'ABC△△'，C2 列がVARCHAR(10)で値が'XYZ'の場合，次のように連結されます。

CONCAT("C1","C2") → 'ABC△△XYZ'

CONCAT("C2","C1") → 'XYZABC△△'

（凡例）△：半角空白

(4)　例題

例題

表T1 のC2 列の文字データとC3 列の文字データを連結し，連結結果が'efg03v03'の行を検索します。

```
SELECT * FROM "T1"
    WHERE CONCAT("C2","C3")='efg03v03'
```

表T1

C1列 CHAR	C2列 VARCHAR	C3列 VARCHAR
A10101	abc010587	rs3354
A15014	efg03	v03
A31399	hijk99842688	wxyz22725

検索結果

A15014	efg03	v03

8.5.2 LEFT

文字データの先頭（左）から一部の文字列を抽出します。

(1) 指定形式

スカラ関数LEFT：：＝LEFT(*抽出元の文字データ*,*抽出文字数*)

抽出元の文字データ：：＝*値式*
抽出文字数：：＝*値式*

(2) 指定形式の説明

抽出元の文字データ：

抽出元の文字データを指定します。

指定規則を次に示します。

- 抽出元の文字データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 抽出元の文字データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 抽出元の文字データには，？パラメタを単独で指定できません。

抽出文字数：

抽出する文字数を指定します。抽出元の文字データの先頭から，指定した文字数分の文字列が抽出されます。

指定規則を次に示します。

- 抽出文字数は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 抽出文字数には，整数（INTEGER 型またはSMALLINT 型のデータ）を指定してください。
- 抽出文字数に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

スカラ関数LEFT の実行結果の例を次に示します。

（例）

文字列'ABCDEF'の先頭から 3 文字分のデータを抽出します。

LEFT('ABCDEF',3) → 'ABC'

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-12 スカラ関数 LEFT の実行結果のデータ型とデータ長

抽出元の文字データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

（凡例）

n：抽出元の文字データの最大長

抽出される文字数は次のようになります。

MIN（*抽出文字数*，*抽出元の文字データの文字数*）

2. 「抽出文字数＞抽出元の文字データの文字数」の場合，抽出元の文字データの文字数分のデータが返されます。
3. 次に示す場合，実行結果は実長 0 バイトのデータになります。
   - 実行結果の文字列の長さが 0 の場合
   - 抽出元の文字データが実長 0 バイトまたは実長 0 文字の場合
4. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
5. 次に示す場合，実行結果はナル値になります。
   - 抽出元の文字データまたは抽出文字数のどちらかがナル値の場合
   - 抽出文字数が負の値の場合（抽出元の文字データの指定に関係なくナル値になります）

(4) 例題

例題

表T1 のC1 列のデータで，先頭から 3 文字分のデータが'A15'の行を検索します。

```
SELECT * FROM "T1"
    WHERE LEFT("C1",3)='A15'
```

表T1

C1列 CHAR	C2列 INTEGER
A10101	300
A15014	1000
A31399	200

検索結果

A15014	1000

8.5.3 LPAD

対象データの先頭（左側）に，指定文字数となるまで，埋め込み文字列を繰り返し埋め込みます。

(1) 指定形式

スカラ関数LPAD：：＝LPAD(*対象データ*,*文字数*〔,*埋め込み文字列*〕)

対象データ：：＝*値式*
文字数：：＝*値式*
埋め込み文字列：：＝*値式*

(2) 指定形式の説明

対象データ：

文字列の埋め込みを行う対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

文字数：

文字列を埋め込んだ結果の文字列の文字数を指定します。

指定規則を次に示します。

- 文字数は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 文字数には，INTEGER 型またはSMALLINT 型のデータを指定してください。
- 文字数に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

埋め込み文字列：

対象データの先頭（左側）に埋め込む文字列を指定します。

指定規則を次に示します。

- *埋め込み文字列*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *埋め込み文字列*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *埋め込み文字列*を省略した場合，埋め込み文字列に半角空白が仮定されます。
- *埋め込み文字列*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

スカラ関数LPAD の実行結果の例を次に示します。

(例)

C1 列のデータの先頭（左側）に，全体の文字列が 10 文字になるまで，文字列'xyz'を繰り返し埋め込みます。

LPAD("C1",10,'xyz') → 'xyzxyzxABC'

C1 列はVARCHAR(20)の列で，文字列'ABC'が格納されています。

(3)　規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-13　スカラ関数 LPAD の実行結果のデータ型とデータ長

対象データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

（凡例）*n*：対象データの最大長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 次のどちらかの条件を満たす場合は，実行結果はナル値になります。
   - 対象データ，文字数，または埋め込み文字列がナル値の場合
   - 文字数に負の値を指定した場合
4. 埋め込み文字列が実長 0 バイトまたは実長 0 文字のデータの場合，文字列の埋め込みは行われません。
5. *対象データ*の文字数が，文字数の値より大きい場合，*対象データ*の文字列を先頭から指定文字数分返します。

   （例）LPAD('ABCDE',3,'xy') → 'ABC'
6. 実行結果のデータ長では，指定した文字数の文字列を表現できない場合，埋め込み文字列の埋め込みを途中で打ち切ります。そのため，実行結果の文字数が，指定した文字数と異なることがあります。指定した文字数分の文字列を取得したい場合は，スカラ関数CAST を使用して対象データのデータ長を変更してください。

   (例)

   C1 列の値とデータ型は次のとおりで，使用している文字コードは Unicode（UTF-8）とします。

・C1 列の値：Ⅰ Ⅱ

・C1 列のデータ型：VARCHAR(10)

LPAD(”C1”,5,’Ⅲ Ⅳ Ⅴ’) → ’Ⅲ Ⅰ Ⅱ’

上記の例の場合，LPAD の実行結果のデータ型はVARCHAR(10)になります。1 文字が 3 バイトのため，実行結果の文字数が，指定した文字数（5 文字）になりません。

LPAD(CAST(”C1” AS VARCHAR(15)),5,’Ⅲ Ⅳ Ⅴ’) → ’Ⅲ Ⅳ Ⅴ Ⅰ Ⅱ’

上記の例の場合，LPAD の実行結果のデータ型はVARCHAR(15)になります。1 文字が 3 バイトのため，実行結果の文字数が，指定した文字数（5 文字）になります。

(4) 例題

例題

表T1 のC1 列はVARCHAR(8)の列です。C1 列の文字データのうち，8 文字未満の文字データの先頭に’0’を埋め込み 8 文字にそろえます。

```
SELECT LPAD(”C1”,8,’0’) FROM ”T1”
```

表T1

C1列 VARCHAR(8)
10101A
1501A
9999999A

検索結果

0010101A
0001501A
9999999A

8.5.4 LTRIM

対象データの文字列の先頭から順に，削除文字に指定した文字を削除します。

文字列の先頭から順に削除文字と一致するかどうかをチェックし，削除文字と一致した場合，その文字を削除します。削除文字と異なる文字が現れた場合，そこで処理を終了します。

(1) 指定形式

スカラ関数LTRIM：：＝LTRIM(*対象データ* 〔,*削除文字*〕)

対象データ：：＝*値式*
削除文字：：＝*値式*

(2) 指定形式の説明

対象データ：

*削除文字*に指定した文字を削除するデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

削除文字：

対象データから削除する文字を指定します。

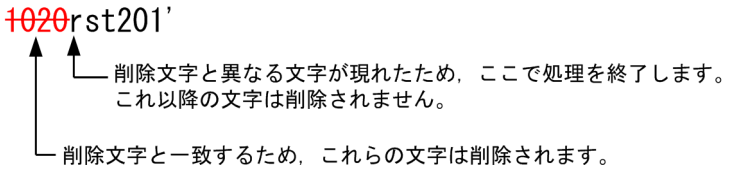
指定規則を次に示します。

- *削除文字*は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *削除文字*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *削除文字*を省略した場合，削除文字に半角空白が仮定されます。
- *削除文字*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

スカラ関数LTRIM の実行結果の例を次に示します。

(例)

LTRIM('1020rst201','012') → 'rst201'



LTRIM('aaaadatabaseaaaa','a') → 'databaseaaaa'

LTRIM('aabbccdatabase','abc') → 'database'

LTRIM('△△△database△') →'database△'

LTRIM('database','012') → 'database'

(凡例) △：半角空白

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-14 スカラ関数 LTRIM の実行結果のデータ型とデータ長

対象データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)

対象データのデータ型とデータ長	実行結果のデータ型とデータ長
VARCHAR(*n*)	

（凡例）*n*：対象データの最大長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *対象データ*または*削除文字*がナル値の場合，実行結果はナル値になります。
4. 対象データが実長 0 バイトまたは実長 0 文字のデータの場合，実行結果は実長 0 バイトのデータになります。
5. 対象データの文字列がすべて削除された場合，実行結果は実長 0 バイトのデータになります。
6. *削除文字*に実長 0 バイトまたは実長 0 文字のデータを指定した場合，実行結果の値は対象データとなります。

(4) 例題

例題

表T1 のC2 列の文字データから，先頭に付加されている数字を削除した結果を求めます。

```
SELECT "C1",LTRIM("C2",'0123456789') FROM "T1"
```

表T1

C1列 CHAR	C2列 VARCHAR
A001	205678abcdefg
A002	98742hijklmn
A003	13opqrstuvwxyz

検索結果

A001	abcdefg
A002	hijklmn
A003	opqrstuvwxyz

8.5.5 RIGHT

文字データの末尾（右）から一部の文字列を抽出します。

(1) 指定形式

```
スカラ関数RIGHT : : =RIGHT(抽出元の文字データ,抽出文字数)

  抽出元の文字データ : : =値式
  抽出文字数 : : =値式
```

(2) 指定形式の説明

抽出元の文字データ：

抽出元の文字データを指定します。

指定規則を次に示します。

- 抽出元の文字データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 抽出元の文字データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 抽出元の文字データには，？パラメタを単独で指定できません。

抽出文字数：

抽出する文字数を指定します。抽出元の文字データの末尾から，指定した文字数分の文字列が抽出されます。

指定規則を次に示します。

- 抽出文字数は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 抽出文字数には，整数（INTEGER 型またはSMALLINT 型のデータ）を指定してください。
- 抽出文字数に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

スカラ関数RIGHT の実行結果の例を次に示します。

（例）

文字列'ABCDEF'の末尾から 3 文字分のデータを抽出します。

RIGHT('ABCDEF',3) → 'DEF'

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-15 スカラ関数 RIGHT の実行結果のデータ型とデータ長

抽出元の文字データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

（凡例）

n：抽出元の文字データの最大長

抽出される文字数は次のようになります。

MIN（*抽出文字数*，*抽出元の文字データの文字数*）

2.「抽出文字数＞抽出元の文字データの文字数」の場合，抽出元の文字データの文字数分のデータが返されます。

3. 次に示す場合，実行結果は実長 0 バイトのデータになります。
   - 実行結果の文字列の長さが 0 の場合
   - 抽出元の文字データが実長 0 バイトまたは実長 0 文字の場合
4. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
5. 次に示す場合，実行結果はナル値になります。
   - 抽出元の文字データまたは抽出文字数のどちらかがナル値の場合
   - 抽出文字数が負の値の場合（抽出元の文字データの指定に関係なくナル値になります）

(4) 例題

例題

表T1 のC1 列のデータで，末尾から 3 文字分のデータが'14B'の行を検索します。

```
SELECT * FROM "T1"
    WHERE RIGHT("C1",3)='14B'
```

表T1

C1列 CHAR	C2列 INTEGER
A10101B	300
A15014B	1000
A31399B	200

検索結果

A15014B	1000

8.5.6 RPAD

対象データの末尾（右側）に，指定文字数となるまで，埋め込み文字列を繰り返し埋め込みます。

(1) 指定形式

```
スカラ関数RPAD：：＝RPAD(対象データ,文字数〔,埋め込み文字列〕)

  対象データ：：＝値式
  文字数：：＝値式
  埋め込み文字列：：＝値式
```

(2) 指定形式の説明

対象データ：

文字列の埋め込みを行う対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

文字数：

文字列を埋め込んだ結果の文字列の文字数を指定します。

指定規則を次に示します。

- 文字数は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 文字数には，INTEGER 型またはSMALLINT 型のデータを指定してください。
- 文字数に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

埋め込み文字列：

対象データの末尾（右側）に埋め込む文字列を指定します。

指定規則を次に示します。

- *埋め込み文字列*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *埋め込み文字列*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *埋め込み文字列*を省略した場合，埋め込み文字列に半角空白が仮定されます。
- *埋め込み文字列*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

スカラ関数RPAD の実行結果の例を次に示します。

（例）

C1 列のデータの末尾（右側）に，全体の文字列が 10 文字になるまで，文字列'xyz'を繰り返し埋め込みます。

RPAD("C1",10,'xyz') → 'ABCxyzxyzx'

C1 列はVARCHAR(20)の列で，文字列'ABC'が格納されています。

(3)　規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-16　スカラ関数 RPAD の実行結果のデータ型とデータ長

対象データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

（凡例）*n*：対象データの最大長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

3. 次のどちらかの条件を満たす場合は，実行結果はナル値になります。
   - 対象データ，文字数，または埋め込み文字列がナル値の場合
   - 文字数に負の値を指定した場合
4. 埋め込み文字列が実長 0 バイトまたは実長 0 文字のデータの場合，文字列の埋め込みは行われません。
5. *対象データ*の文字数が，文字数の指定値より大きい場合，*対象データ*の文字列を先頭から指定文字数分返します。

   （例）RPAD('ABCDE',3,'xy') → 'ABC'
6. 実行結果のデータ長では，指定した文字数の文字列を表現できない場合，埋め込み文字列の埋め込みを途中で打ち切ります。そのため，実行結果の文字数が，指定した文字数と異なることがあります。指定した文字数分の文字列を取得したい場合は，スカラ関数CAST を使用して対象データのデータ長を変更してください。

   （例）

   C1 列の値とデータ型は次のとおりで，使用している文字コードは Unicode（UTF-8）とします。

   ・C1 列の値：Ⅰ Ⅱ

   ・C1 列のデータ型：VARCHAR(10)

   RPAD("C1",5,'Ⅲ Ⅳ Ⅴ') → 'Ⅰ Ⅱ Ⅲ'

   上記の例の場合，RPAD の実行結果のデータ型はVARCHAR(10)になります。1 文字が 3 バイトのため，実行結果の文字数が，指定した文字数（5 文字）になりません。

   RPAD(CAST("C1" AS VARCHAR(15)),5,'Ⅲ Ⅳ Ⅴ') → 'Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ'

   上記の例の場合，RPAD の実行結果のデータ型はVARCHAR(15)になります。1 文字が 3 バイトのため，実行結果の文字数が，指定した文字数（5 文字）になります。

(4) 例題

例題

表T1 のC1 列はVARCHAR(8)の列です。C1 列の文字データのうち，8 文字未満の文字データの末尾に'0'を埋め込み 8 文字にそろえます。

```
SELECT RPAD("C1",8,'0') FROM "T1"
```

表T1

C1列 VARCHAR(8)
A10101
A1501
A9999999

検索結果

A1010100
A1501000
A9999999

8.5.7　RTRIM

対象データの文字列の末尾から順に，削除文字に指定した文字を削除します。

文字列の末尾から順に削除文字と一致するかどうかをチェックし，削除文字と一致した場合，その文字を削除します。削除文字と異なる文字が現れた場合，そこで処理を終了します。

(1)　指定形式

スカラ関数RTRIM：：＝RTRIM(*対象データ*〔,*削除文字*〕)

対象データ：：＝*値式*
削除文字：：＝*値式*

(2)　指定形式の説明

対象データ：

*削除文字*に指定した文字を削除するデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

削除文字：

対象データから削除する文字を指定します。

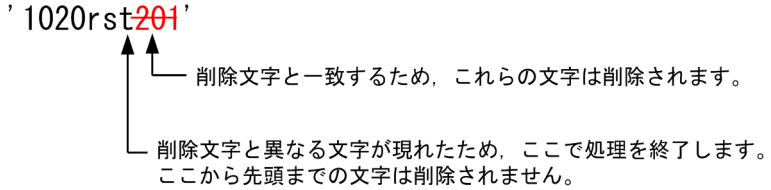
指定規則を次に示します。

- *削除文字*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *削除文字*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *削除文字*を省略した場合，削除文字に半角空白が仮定されます。
- *削除文字*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

スカラ関数RTRIM の実行結果の例を次に示します。

(例)

```
RTRIM('1020rst201','012') → '1020rst'
```



RTRIM('aaaadatabaseaaaa','a') → 'aaaadatabase'

RTRIM('aabbccdatabase','abes') → 'aabbccdat'

RTRIM('△△△database△') →'△△△database'

RTRIM('database','012') → 'database'

（凡例）△：半角空白

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-17　スカラ関数 RTRIM の実行結果のデータ型とデータ長

対象データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

（凡例）*n*：対象データの最大長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *対象データ*または*削除文字*がナル値の場合，実行結果はナル値になります。
4. 対象データが実長 0 バイトまたは実長 0 文字のデータの場合，実行結果は実長 0 バイトのデータになります。
5. 対象データの文字列がすべて削除された場合，実行結果は実長 0 バイトのデータになります。
6. *削除文字*に実長 0 バイトまたは実長 0 文字のデータを指定した場合，実行結果の値は対象データとなります。

(4) 例題

例題

表T1 のC2 列の文字データから，末尾に付加されている数字を削除した結果を求めます。

```
SELECT "C1",RTRIM("C2",'0123456789') FROM "T1"
```

表T1

C1列 CHAR	C2列 VARCHAR
A001	abcdefg205678
A002	hijklmn98742
A003	opqrstuvwxyz13

検索結果

A001	abcdefg
A002	hijklmn
A003	opqrstuvwxyz

8.5.8　SUBSTR

文字データの任意の位置から一部の文字列を抽出します。

(1)　指定形式

スカラ関数SUBSTR：：＝SUBSTR(*抽出元の文字データ*,*開始位置*〔,*抽出文字数*〕)

抽出元の文字データ：：＝*値式*
開始位置：：＝*値式*
抽出文字数：：＝*値式*

(2)　指定形式の説明

抽出元の文字データ：

抽出元の文字データを指定します。

指定規則を次に示します。

- 抽出元の文字データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 抽出元の文字データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 抽出元の文字データには，？パラメタを単独で指定できません。

開始位置：

文字データの抽出開始位置を，文字数単位で指定します。

開始位置に0 以上の値を指定した場合は，抽出元の文字データの先頭からの位置を意味します。例えば，開始位置に2 を指定した場合，先頭 2 文字目から抽出を始めるという意味になります。

開始位置に負の値を指定した場合は，抽出元の文字データの末尾からの位置を意味します。例えば，開始位置に-2 を指定した場合，末尾 2 文字目から抽出を始めるという意味になります。

指定規則を次に示します。

- 開始位置は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 開始位置には，整数（INTEGER 型またはSMALLINT 型のデータ）を指定してください。
- 開始位置に0 を指定した場合，1 が指定されたと仮定されます。
- 開始位置に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

抽出文字数：

抽出する文字数を指定します。

指定規則を次に示します。

- 抽出文字数は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 抽出文字数には，0 以上の整数（INTEGER 型またはSMALLINT 型のデータ）を指定してください。

- 抽出文字数を省略した場合，抽出元の文字データがCHAR 型のときは，開始位置から定義長の最後の文字までを抽出します。抽出元の文字データがVARCHAR 型のときは，開始位置から実データの最後の文字までを抽出します。
- 抽出文字数に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

スカラ関数SUBSTR の実行結果の例を次に示します。

(例)

- 文字列'ABCDEF'の先頭 2 文字目から 3 文字分のデータを抽出します。
  SUBSTR('ABCDEF',2,3) → 'BCD'
- 文字列'ABCDEF'の末尾 3 文字目から 2 文字分のデータを抽出します。
  SUBSTR('ABCDEF',-3,2) → 'DE'

(3) 規則

1. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
2. 次に示す場合，実行結果はナル値になります。
   - 抽出文字数が負の値の場合（抽出元の文字データ，開始位置の指定に関係なくナル値になります）
   - 抽出元の文字データ，開始位置，または抽出文字数のどれかがナル値の場合
3. 実行結果のデータ型とデータ長を次の表に示します。

表 8-18　スカラ関数 SUBSTR の実行結果のデータ型とデータ長

抽出元の文字データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

（凡例）*n*：抽出元の文字データの最大長

4. スカラ関数SUBSTR によって抽出される文字数を次の表に示します。

表 8-19　スカラ関数 SUBSTR によって抽出される文字数

スカラ関数 SUBSTR の指定		抽出される文字数
抽出文字数の指定	開始位置の指定値	
指定あり	正の値	MAX｛0，MIN（*抽出文字数*，*抽出元の文字データの文字数－開始位置*＋1)｝
	0	MIN（*抽出文字数*，*抽出元の文字データの文字数*）
	負の値	MIN（*抽出文字数*，*開始位置の絶対値*，*抽出元の文字データの文字数*）
省略	正の値	MAX（0，*抽出元の文字データの文字数－開始位置*＋1）

スカラ関数 SUBSTR の指定		抽出される文字数
抽出文字数の指定	開始位置の指定値	
	0	*抽出元の文字データの文字数*
	負の値	MIN（*開始位置の絶対値*，*抽出元の文字データの文字数*）

5. 次に示す場合，実行結果は実長 0 バイトのデータになります。
   - 実行結果の文字列の長さが 0 の場合
   - 抽出元の文字データが実長 0 バイトまたは実長 0 文字の場合
   - 開始位置に次の値を指定した場合
     開始位置＞抽出元の文字データの文字数
     開始位置＜－抽出元の文字データの文字数
6. 「抽出元の文字データの開始位置以降の文字数＜抽出文字数」の場合，抽出元の文字データの開始位置以降のすべての文字データを返します。
   （例）
   SUBSTR('ABCDEF',5,3) → 'EF'

(4) 例題

例題 1

表T1 のC1 列のデータで，先頭 2 文字目から 3 文字分のデータが'150'の行を検索します。

```
SELECT * FROM "T1"
    WHERE SUBSTR("C1",2,3)='150'
```

表T1

C1列 CHAR	C2列 INTEGER
A10101	300
A15014	1000
A31399	200

検索結果

A15014	1000

例題 2

表T1 のC1 列のデータで，末尾 2 文字目から 2 文字分のデータが'01'の行を検索します。

```
SELECT * FROM "T1"
    WHERE SUBSTR("C1",-2,2)='01'
```

表T1

C1列 CHAR	C2列 INTEGER
A10101	300
A15014	1000
A31399	200

検索結果

A10101	300

8.5.9 TRIM

対象データの文字列から，削除文字に指定した文字を削除します。文字の削除方法を次のどれかから選択できます。

- 文字列の先頭から順に，削除文字に指定した文字を削除します。
- 文字列の末尾から順に，削除文字に指定した文字を削除します。
- 文字列の先頭および末尾の両方から順に，文字を削除します。

(1) 指定形式

```
スカラ関数TRIM：：＝TRIM( 〔 {削除規則 削除文字
                            |削除規則
                            |削除文字}  FROM〕  対象データ)

  削除規則：：＝ {LEADING | TRAILING | BOTH}
  削除文字：：＝値式
  対象データ：：＝値式
```

(2) 指定形式の説明

削除規則：

文字を削除する際の規則を指定します。指定を省略した場合，BOTH が仮定されます。

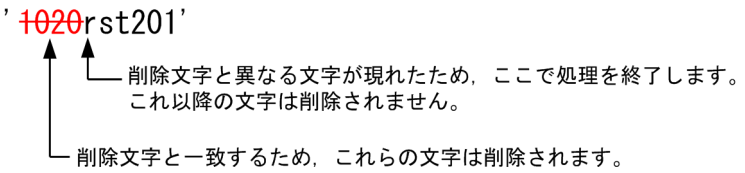
LEADING：

LEADING を指定した場合，文字列の先頭から順に削除文字と一致するかどうかをチェックし，削除文字と一致した場合はその文字を削除します。削除文字と異なる文字が現れた場合，そこで処理を終了します。

LEADING を指定したときの実行結果の例を次に示します。

(例)

TRIM(LEADING ’012’ FROM ’1020rst201’) → ’rst201’



TRIM(LEADING 'a' FROM 'aaaadatabaseaaaa') → 'databaseaaaa'

TRIM(LEADING 'abc' FROM 'aabbccdatabase') → 'database'

TRIM(LEADING FROM '△△△database△') →'database△'

TRIM(LEADING '012' FROM 'database') → 'database'

（凡例）△：半角空白

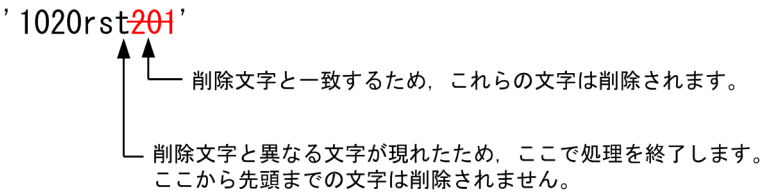
TRAILING：

TRAILING を指定した場合，文字列の末尾から順に削除文字と一致するかどうかをチェックし，削除文字と一致した場合はその文字を削除します。削除文字と異なる文字が現れた場合，そこで処理を終了します。

TRAILING を指定したときの実行結果の例を次に示します。

（例）

TRIM(TRAILING '012' FROM '1020rst201') → '1020rst'



TRIM(TRAILING 'a' FROM 'aaaadatabaseaaaa') → 'aaaadatabase'

TRIM(TRAILING 'abes' FROM 'aabbccdatabase') → 'aabbccdat'

TRIM(TRAILING FROM '△△△database△') →'△△△database'

TRIM(TRAILING '012' FROM 'database') → 'database'

（凡例）△：半角空白

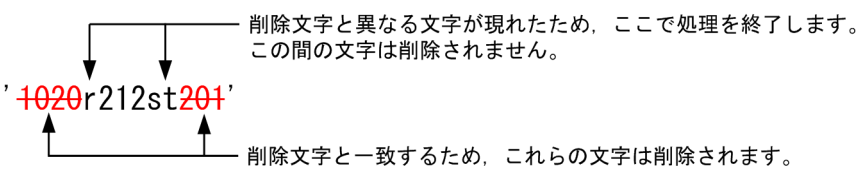
BOTH：

BOTH を指定した場合，文字列の先頭および末尾の両方から順に削除文字と一致するかどうかをチェックし，削除文字と一致した場合はその文字を削除します。削除文字と異なる文字が現れた場合，そこで処理を終了します。

BOTH を指定したときの実行結果の例を次に示します。

（例）

TRIM(BOTH '012' FROM '1020r212st201') → 'r212st'



TRIM(BOTH 'a' FROM 'aaaadatabaseaaaa') → 'database'

TRIM(BOTH 'abces' FROM 'aabbccdatabase') → 'dat'

TRIM(BOTH FROM '△△△database△') →'database'

TRIM(BOTH '012' FROM 'database') → 'database'

（凡例）△：半角空白

削除文字：

対象データから削除する文字を指定します。

指定規則を次に示します。

- *削除文字*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *削除文字*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *削除文字*を省略した場合，削除文字に半角空白が仮定されます。
- *削除文字*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

対象データ：

*削除文字*に指定した文字を削除するデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-20　スカラ関数 TRIM の実行結果のデータ型とデータ長

対象データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

（凡例）*n*：対象データの最大長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *対象データ*または*削除文字*がナル値の場合，実行結果はナル値になります。
4. 対象データが実長 0 バイトまたは実長 0 文字のデータの場合，実行結果は実長 0 バイトのデータになります。
5. 対象データの文字列がすべて削除された場合，実行結果は実長 0 バイトのデータになります。
6. *削除文字*に実長 0 バイトまたは実長 0 文字のデータを指定した場合，実行結果の値は対象データとなります。

(4) 例題

例題

表T1 のC2 列の文字データから，先頭および末尾に付加されている数字を削除した結果を求めます。

```
SELECT "C1",TRIM(BOTH '0123456789' FROM "C2") FROM "T1"
```

表T1

C1列 CHAR	C2列 VARCHAR
A001	581abcdefg205678
A002	611hijklmn98742
A003	32opqrstuvwxyz13

検索結果

A001	abcdefg
A002	hijklmn
A003	opqrstuvwxyz

8.6　文字列関数（文字列情報の取得）

ここでは，文字列情報の取得に関する文字列関数の機能と指定形式について説明します。

8.6.1　CONTAINS

検索条件式指定で指定した条件を満たす文字列が，対象データ中に含まれているかどうかを返します。検索条件式指定で指定した条件を満たす文字列が，対象データ中に含まれている場合は1 を返します。含まれていない場合は0 を返します。

スカラ関数CONTAINS は探索条件に指定できます。ただし，CASE 式の探索条件には指定できません。

(1)　指定形式

```
スカラ関数CONTAINS：：＝CONTAINS(対象データ,検索条件式指定)

  対象データ：：＝値式
  検索条件式指定：：＝文字列定数
```

(2)　指定形式の説明

対象データ：

検索対象のデータを指定します。

指定規則を次に示します。

- *対象データ*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *対象データ*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *対象データ*には，？パラメタを単独で指定できません。

検索条件式指定：

検索条件を指定します。

指定規則を次に示します。

- *検索条件式指定*は，文字列定数の形式で指定します。文字列定数については，「6.3　定数」を参照してください。
- *検索条件式指定*に指定した検索文字列および同義語辞書名以外の文字列は，半角英大文字として扱われます。
- 分離符号は指定できません。
- *検索条件式指定*には，単純文字列指定，表記ゆれ補正検索指定，同義語検索指定，およびワード検索指定の４つの指定方法があります。

```
検索条件式指定：：＝｛単純文字列指定｜表記ゆれ補正検索指定｜同義語検索指定
                      ｜ワード検索指定｝
```

単純文字列指定：：＝”*検索文字列*”

表記ゆれ補正検索指定：：＝ {IGNORECASE(*単純文字列指定*)｜SORTCODE(*単純文字列指定*)}

同義語検索指定：：＝SYNONYM(”*同義語辞書名*”,｛*単純文字列指定*｜*表記ゆれ補正検索指定*｝)

ワード検索指定：：＝ {WORDCONTEXT({*単純文字列指定*｜*表記ゆれ補正検索指定*
｜*同義語検索指定*｝)
｜WORDCONTEXT_PREFIX({*単純文字列指定*｜*表記ゆれ補正検索指定*｝)}

単純文字列指定：

検索する文字列を次の形式で指定します。

単純文字列指定：：＝”*検索文字列*”

単純文字列指定の指定例を次に示します。

(例) ”COMPUTER”または”computer”

検索する文字列（COMPUTER またはcomputer）を二重引用符（”）で囲みます。

留意事項を次に示します。

- *検索文字列*は，半角英大文字と半角英小文字が区別されます。
- *検索文字列*中に二重引用符（”）がある場合は，二重引用符（”）を 2 個続けて指定してください。
- *検索文字列*が 0 バイトの文字データの場合，実行結果には1 が返されます。*対象データ*に関係なく，*対象データ*中に*検索文字列*が含まれていると判定されます。

表記ゆれ補正検索指定：

表記ゆれ補正検索をする場合に指定します。表記ゆれ補正検索については，マニュアル『HADB システム構築・運用ガイド』の『表記ゆれ補正検索』を参照してください。

検索する文字列を次のどちらかの形式で指定します。

表記ゆれ補正検索指定：：＝ {IGNORECASE(*単純文字列指定*)｜SORTCODE(*単純文字列指定*)}

- IGNORECASE(*単純文字列指定*)：

  IGNORECASE を指定した場合，半角英大文字と半角英小文字の表記ゆれだけが，表記ゆれ補正検索の対象になります。

  なお，次の形式で指定することもできます。

  I(*単純文字列指定*)

- SORTCODE(*単純文字列指定*)：

  表記ゆれ補正検索をする場合に指定します。

  なお，次の形式で指定することもできます。

  S(*単純文字列指定*)

同義語検索指定：

同義語辞書中の同義語グループに指定された同義語を一括して検索する場合に指定します。指定形式を次に示します。

同義語検索指定：：＝SYNONYM(”*同義語辞書名*”, {*単純文字列指定*｜*表記ゆれ補正検索指定*})

- *同義語辞書名*：

  同義語辞書名を指定します。

同義語検索指定の指定例を次に示します。

同義語辞書（Dictionary1）に登録されている文字列

```
コンピュータ,コンピューター,計算機,計算器,電子計算機,COMPUTER,PC
```

（例 1）単純文字列指定の場合

```
SYNONYM(”Dictionary1”,”COMPUTER”)
```

この場合，同義語辞書に登録されている「コンピュータ，コンピューター，計算機，計算器，電子計算機，COMPUTER，PC」が検索対象の文字列になります。

（例 2）表記ゆれ補正検索指定の場合

```
SYNONYM(”Dictionary1”,IGNORECASE(”COMPUTER”))
```

この場合，同義語辞書に登録されている「コンピュータ，コンピューター，計算機，計算器，電子計算機，COMPUTER，PC」のほかに，「Computer，computer，pc」などが検索対象の文字列になります。

重要

同義語辞書を登録，更新する際の表記ゆれ補正オプションにCASESENSITIVE（表記ゆれ補正検索に対応した同義語辞書を作成しない）を指定した場合，同義語検索指定で表記ゆれ補正検索を指定することはできません。

ワード検索指定：

ワード検索をする場合に指定します。ワード検索については，マニュアル『HADB　システム構築・運用ガイド』の『ワード検索』を参照してください。ワード検索指定の指定形式を次に示します。

```
ワード検索指定：：＝ {WORDCONTEXT( {単純文字列指定｜表記ゆれ補正検索指定
                                  |同義語検索指定} )
                    |WORDCONTEXT_PREFIX( {単純文字列指定｜表記ゆれ補正検索指定} )}
```

単語単位で完全一致検索を行う場合は，WORDCONTEXT を指定します。単語単位で前方一致検索を行う場合は，WORDCONTEXT_PREFIX を指定します。

(3) 規則

1. スカラ関数CONTAINS は，比較述語の左側の比較演算項として指定できます。比較演算子と右側の比較演算項には，「>0」を指定してください。
2. HADB サーバで使用する文字コードが Shift-JIS の場合，表記ゆれ補正検索指定は指定できません。
3. 実行結果のデータ型はINTEGER 型になります。
4. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

5. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

表T1 のC2 列（VARCHAR 型）には文書情報が格納されています。この文書情報中に含まれている文字列を，スカラ関数CONTAINS を使用して検索します。

例題 1（単純文字列指定の検索を行う場合）

文書情報中に「COMPUTER」が含まれている行を検索します。

```
SELECT ”C1” FROM ”T1”
    WHERE CONTAINS(”C2”,’”COMPUTER”’) > 0
```

この場合，半角英大文字と半角英小文字が区別されます。そのため，computer などの文字列は検索対象になりません。

例題 2（表記ゆれ補正検索を行う場合）

文書情報中に「COMPUTER」などが含まれている行を検索します。

```
SELECT ”C1” FROM ”T1”
    WHERE CONTAINS(”C2”,’IGNORECASE(”COMPUTER”)’) > 0
```

この場合，表記ゆれ補正検索によって，「COMPUTER」，「computer」，または「Computer」などの文字列が含まれている行が検索対象になります。

例題 3（表記ゆれ補正検索を行う場合）

文書情報中に「コンピュータ」などが含まれている行を検索します。

```
SELECT ”C1” FROM ”T1”
    WHERE CONTAINS(”C2”,’SORTCODE(”コンピュータ”)’) > 0
```

この場合，表記ゆれ補正検索によって，「コンピュータ」，「ｺﾝﾋﾟｭｰﾀ」，または「こんぴゅーた」などの文字列が含まれている行が検索対象になります。

例題 4（同義語検索を行う場合）

同義語辞書Dictionary1 に登録している文字列（コンピュータ，コンピューター，計算機，計算器，電子計算機，COMPUTER，PC）を一括して検索します。

```
SELECT ”C1” FROM ”T1”
    WHERE CONTAINS(”C2”,’SYNONYM(”Dictionary1”,”COMPUTER”)’) > 0
```

例題 5（同義語検索と表記ゆれ補正検索を同時に行う場合）

同義語辞書Dictionary1 に登録している文字列（コンピュータ，コンピューター，計算機，計算器，電子計算機，COMPUTER，PC）を一括して検索します。さらに，同義語辞書に登録されている文字列の表記ゆれ補正検索も行います。

同義語辞書Dictionary1 は，表記ゆれ補正検索に対応している辞書とします。

```
SELECT ”C1” FROM ”T1”
    WHERE CONTAINS(”C2”,’SYNONYM(”Dictionary1”,SORTCODE(”COMPUTER”))’) > 0
```

上記のSELECT 文を実行した場合，同義語辞書に登録している文字列（コンピュータ，コンピューター，計算機，計算器，電子計算機，COMPUTER，PC）のほかに，「Computer」，「computer」，「pc」，「ｺﾝﾋﾟｭｰﾀ」，「こんぴゅーた」などの文字列が含まれている行が検索対象になります。

例題６（ワード検索の単語単位の完全一致検索を行う場合）

C2 列に格納されている英語文書中に，「COMPUTER」などの英単語がある行を検索します。

```
SELECT ”C1” FROM ”T1”
    WHERE CONTAINS(”C2”,’WORDCONTEXT(IGNORECASE(”COMPUTER”))’) > 0
```

この場合，表記ゆれ補正検索の指定（IGNORECASE）によって，「COMPUTER」，「computer」，または「Computer」などの英単語がある行が検索対象になります。

例題７（ワード検索の単語単位の前方一致検索を行う場合）

C2 列に格納されている英語文書中に，「COMP」で始まる英単語がある行を検索します。

```
SELECT ”C1” FROM ”T1”
    WHERE CONTAINS(”C2”,’WORDCONTEXT_PREFIX(”COMP”)’) > 0
```

この場合，「COMPUTER」，「COMPUTERS」，または「COMPANY」などの英単語がある行が検索対象になります。

8.6.2 INSTR

対象データ中の任意の文字列を検索し，その文字列の開始位置を返します。

対象データを検索する際，検索開始位置を指定できます。

また，例えば，対象データ中の’ABC’という文字列の開始位置を求める際，３番目に出現する’ABC’の開始位置を求めるなどの指定もできます。

(1) 指定形式

```
スカラ関数INSTR ： ：＝INSTR(対象データ,検索文字列 〔,検索開始位置 〔,出現回数〕 〕 )

  対象データ ： ：＝値式
  検索文字列 ： ：＝値式
  検索開始位置 ： ：＝値式
  出現回数 ： ：＝値式
```

(2) 指定形式の説明

対象データ：

文字列を検索する対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。

- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

検索文字列：

検索する文字列を指定します。

指定規則を次に示します。

- *検索文字列*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *検索文字列*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *検索文字列*には，？パラメタを単独で指定できません。

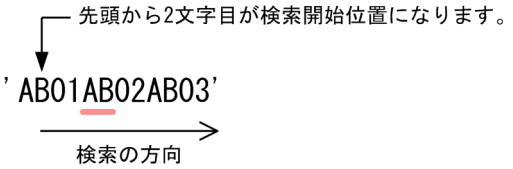
検索開始位置：

対象データの検索開始位置を文字数単位で指定します。

- ***検索開始位置*に正の整数を指定した場合**

  対象データの先頭からの位置を意味し，その位置から順方向（右方向）に検索が行われます。例えば，*検索開始位置*に2 を指定した場合，対象データの先頭 2 文字目から順方向（右方向）に検索が行われます。

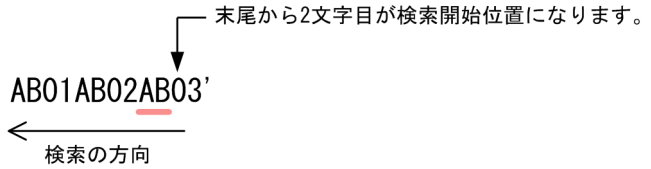
  （例）INSTR('AB01AB02AB03','AB',2) → 5



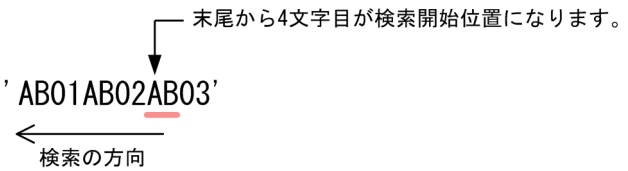
- ***検索開始位置*に負の整数を指定した場合**

  対象データの末尾からの位置を意味し，その位置から逆方向（左方向）に検索が行われます。例えば，*検索開始位置*に-2 を指定した場合，対象データの末尾 2 文字目から逆方向（左方向）に検索が行われます。

  （例 1）INSTR('AB01AB02AB03','AB',-2) → 9



  （例 2）INSTR('AB01AB02AB03','AB',-4) → 9



  上記の例の場合，検索開始位置にA があり，末尾に向けてB が並んでいるため，実行結果に9 が返ります。

指定規則を次に示します。

- *検索開始位置*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *検索開始位置*には，INTEGER 型またはSMALLINT 型のデータを指定してください。
- *検索開始位置*を省略した場合，1 が仮定されます。
- *検索開始位置*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

出現回数：

検索対象の文字列の出現回数を指定します。例えば，*出現回数*に3 を指定した場合，対象データ中で 3 番目に出現する文字列の開始位置が返ります。

指定規則を次に示します。

- *出現回数*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *出現回数*には正の整数を指定してください。
- *出現回数*には，INTEGER 型またはSMALLINT 型のデータを指定してください。
- *出現回数*を省略した場合，1 が仮定されます。
- *出現回数*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

スカラ関数INSTR の実行結果の例を次に示します。

（例）

- INSTR('AB01AB02AB03','AB') → 1

  上記の例の場合，検索文字列'AB'が，対象データの文字列の 1 文字目にあるため，1 が返ります。

- INSTR('AB01AB02AB03','AB',3) → 5

  上記の例の場合，対象データの先頭 3 文字目から検索が始まります。検索文字列'AB'が，対象データの文字列の 5 文字目にあるため，5 が返ります。

- INSTR('AB01AB02AB03','AB',3,2) → 9

  上記の例の場合，対象データの先頭 3 文字目から検索が始まります。また，*出現回数*に2 を指定しているため，2 つ目の'AB'の開始位置が返ります。2 つ目の'AB'が，対象データの文字列の 9 文字目にあるため，9 が返ります。

- INSTR('AB01AB02AB03','AB',-2,3) → 1

  上記の例の場合，対象データの末尾 2 文字目から先頭に向かって検索が始まります。また，*出現回数*に3 を指定しているため，3 つ目の'AB'の開始位置が返ります。3 つ目の'AB'が，対象データの文字列の 1 文字目にあるため，1 が返ります。

(3)　規則

1. 実行結果の値の単位は文字数です。

2. *検索開始位置*の値に関係なく，対象データの先頭から（左から）の出現位置を実行結果の値として返します。
3. 検索文字列に指定した文字列が見つからない場合，実行結果の値に0 が返ります。
4. 実行結果のデータ型はINTEGER 型になります。
5. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
6. 次のどれかの場合，実行結果はナル値になります。
   - *対象データ*，*検索文字列*，*検索開始位置*，または*出現回数*のどれかがナル値の場合
   - *検索開始位置*に0 を指定した場合
   - *出現回数*に0 または負の値を指定した場合
7. *対象データ*または*検索文字列*のどちらかが実長0 バイトまたは実長0 文字のデータの場合，実行結果の値は0 になります。ただし，次の場合を除きます。
   - *対象データ*または*検索文字列*のどちらかがナル値の場合
   - *検索開始位置*に0 を指定した場合
   - *出現回数*に0 または負の値を指定した場合
8. *検索文字列*に指定した文字列と*対象データ*中の文字列の比較は文字単位に行われます。この比較処理は，検索対象の文字列が*出現回数*に指定した回数見つかるか，または検索する文字列がなくなるまで実行されます。

(4) 例題

例題

表T1 のC1 列には，メールアドレスの情報が格納されています。メールアドレスの@より前にある文字列を抽出します。

```
SELECT LEFT(”C1”,INSTR(”C1”,’@’)-1) FROM ”T1”
```

表T1

C1列 VARCHAR
abcdefg@hhhh.co.jp
abcdefghijk@iiii.co.jp
lmn@bbbb.co.jp

検索結果

abcdefg
abcdefghijk
lmn

8.6.3 LENGTH

対象データの文字列の文字数を返します。

(1) 指定形式

スカラ関数LENGTH：：＝LENGTH(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

文字数をカウントする対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はINTEGER 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。
4. 対象データが実長 0 バイトまたは実長 0 文字のデータの場合，実行結果の値は0 になります。

(4) 例題

例題

表T1 のC1 列およびC2 列のデータの文字数を求めます。

なお，使用している文字コードは，Unicode（UTF-8）とします。

```
SELECT LENGTH(”C1”),LENGTH(”C2”) FROM ”T1”
```

表T1

C1列 VARCHAR(10)	C2列 CHAR(10)
abc	abc△△△△△△△
Ⅰ	Ⅰ△△△△△△△
ⅠⅡ	ⅠⅡ△△△△

検索結果

3	10
1	8
2	6

（凡例）△：半角空白

上記の例のように，1 つの半角空白を 1 文字とカウントします。

8.7　文字列関数（文字置換）

ここでは，文字置換に関する文字列関数の機能と指定形式について説明します。

8.7.1　REPLACE

対象データ中の任意の文字列を置換します。対象データ中に存在する置換対象文字列のすべてを置換後の文字列に置換します。

(1)　指定形式

```
スカラ関数REPLACE：：＝REPLACE(対象データ,置換対象文字列〔,置換後の文字列〕)

  対象データ：：＝値式
  置換対象文字列：：＝値式
  置換後の文字列：：＝値式
```

(2)　指定形式の説明

対象データ：

対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

置換対象文字列：

置換対象の文字列を指定します。

指定規則を次に示します。

- *置換対象文字列*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *置換対象文字列*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *置換対象文字列*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

置換後の文字列：

置換後の文字列を指定します。

指定規則を次に示します。

- *置換後の文字列*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *置換後の文字列*には，CHAR 型またはVARCHAR 型のデータを指定してください。

- *置換後の文字列*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。
- *置換後の文字列*を省略した場合，実長０バイトのデータが仮定されます。

スカラ関数REPLACE の実行結果の例を次に示します。

（例）

対象データ中のすべての文字列BCD をYZ に置換します。

REPLACE('ABCDEBCD','BCD','YZ') → 'AYZEYZ'

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-21　スカラ関数 REPLACE の実行結果のデータ型とデータ長

対象データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

（凡例）*n*：対象データの最大長

2. 置換した結果，実行結果のデータ長を超えた場合，エラーになります。そのため，実行結果のデータ長を大きくしたい場合は，スカラ関数CAST を使用して対象データのデータ長を変更してください。

（例）

C1 列はVARCHAR(5)の列で，文字列'ABCD'が格納されているとします。

REPLACE("C1",'AB','WXYZ') → エラー

上記の例の場合，REPLACE の実行結果のデータ長はVARCHAR(5)になるため，データ長不足によってエラーになります。

REPLACE(CAST("C1" AS VARCHAR(10)),'AB','WXYZ') → 'WXYZCD'

上記の例の場合，REPLACE の実行結果のデータ長はVARCHAR(10)になるため，エラーになりません。

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 対象データ，置換対象文字列，または置換後の文字列のどれかがナル値の場合，実行結果はナル値になります。
5. 置換した結果，対象データの文字列がすべて取り除かれた場合，実行結果は実長０バイトのデータになります。
6. 置換対象文字列に実長０バイトまたは実長０文字のデータを指定した場合，対象データ中の文字は置換されません。
7. 置換後の文字列に実長０バイトまたは実長０文字のデータを指定した場合，対象データ中の置換対象文字列がすべて削除されます。

(4) 例題

例題

表T1 のC1 列（CHAR 型）には，日付に関する情報が *YYYY*.*MM*.*DD*（*YYYY* が年，*MM* が月，*DD* が日）の形式で格納されています。

YYYY が2013 年の場合，2014 年に一括置換します。

```
SELECT REPLACE(”C1”,’2013’,’2014’) FROM ”T1”
```

表T1

C1列 CHAR(10)
2012.10.18
2013.01.25
2013.02.05

検索結果

2012.10.18
2014.01.25
2014.02.05

8.7.2 TRANSLATE

対象データ中の任意の文字を置換します。

(1) 指定形式

```
スカラ関数TRANSLATE：：＝TRANSLATE(対象データ,置換対象文字,置換後の文字)

  対象データ：：＝値式
  置換対象文字：：＝値式
  置換後の文字：：＝値式
```

(2) 指定形式の説明

対象データ：

対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

置換対象文字：

置換対象の文字を指定します。

指定規則を次に示します。

- *置換対象文字*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *置換対象文字*に同じ文字を 2 つ以上指定した場合，最初に指定した置換対象文字を有効とします。
- *置換対象文字*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

置換後の文字：

置換後の文字を指定します。

指定規則を次に示します。

- *置換後の文字*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *置換後の文字*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はVARCHAR(32000)になります。

ヒント

複数の文字を置換する場合，*置換対象文字*と*置換後の文字*の先頭からの文字位置が同じになるように指定します。例えば，A をa，B をb，C をc に置換する場合，*置換対象文字*には'ABC'，*置換後の文字*には'abc'と指定します。

スカラ関数TRANSLATE の実行結果の例を次に示します。

(例)

文字列中のA をa に，B をb に，C をc に置換します。

TRANSLATE('AXBYCZ','ABC','abc') → 'aXbYcZ'

(3)　規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-22　スカラ関数 TRANSLATE の実行結果のデータ型とデータ長

対象データのデータ型とデータ長	実行結果のデータ型とデータ長
CHAR(*n*)	VARCHAR(*n*)
VARCHAR(*n*)	

(凡例) *n*：対象データの最大長

2. 置換した結果，実行結果のデータ長を超えた場合，エラーになります。そのため，実行結果のデータ長を大きくしたい場合は，スカラ関数CAST を使用して対象データのデータ長を変更してください。

(例)

使用している文字コードは Unicode（UTF-8）とします。

C1 列はVARCHAR(5)の列で，文字列'ABC'が格納されているとします。

TRANSLATE(”C1”,’ABC’,’Ⅰ Ⅱ Ⅲ’) → エラー

上記の例の場合，TRANSLATE の実行結果のデータ長はVARCHAR(5)になるため，データ長不足によってエラーになります。

TRANSLATE(CAST(”C1” AS VARCHAR(9)),’ABC’,’Ⅰ Ⅱ Ⅲ’) → ’Ⅰ Ⅱ Ⅲ’

上記の例の場合，TRANSLATE の実行結果のデータ長はVARCHAR(9)になるため，エラーになりません。

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. *対象データ*，*置換対象文字*，または*置換後の文字*がナル値の場合，実行結果はナル値になります。
5. *対象データ*中の*置換対象文字*を，*置換後の文字*に置換します。置換対象文字の指定にはない対象データ中の文字は置換されません。
6. *置換後の文字数*が，*置換対象文字数*より少ない場合，余分に指定されている*置換対象文字*が対象データ中に存在するときは，対象データ中からその*置換対象文字*を取り除きます。

   （例）TRANSLATE(’ABCD’,’ABC’,’ab’) → ’abD’
7. *置換後の文字数*が，*置換対象文字数*より多い場合，余分に指定されている*置換後の文字*は無視されます。

   （例）TRANSLATE(’ABCD’,’AB’,’abc’) → ’abCD’
8. 置換した結果，対象データの文字列がすべて取り除かれた場合，実行結果は実長 0 バイトのデータになります。

(4) 例題

例題

表T1 のC1 列（CHAR 型）には，日付に関する情報が *YYYY*.*MM*.*DD*（*YYYY* が年，*MM* が月，*DD* が日）の形式で格納されています。これを *YYYY*/*MM*/*DD* の形式に変更します。

```
SELECT TRANSLATE(”C1”,’.’,’/’) FROM ”T1”
```

表T1

C1列 CHAR(10)
2014.01.25
2014.02.05

検索結果

2014/01/25
2014/02/05

8.8　文字列関数（文字変換）

ここでは，文字変換に関する文字列関数の機能と指定形式について説明します。

8.8.1　LOWER

文字データの英大文字（A～Z）を英小文字（a～z）に変換します。半角英文字および全角英文字に対応しています。

(1)　指定形式

スカラ関数LOWER：：＝LOWER(*変換対象の文字データ*)

変換対象の文字データ：：＝*値式*

(2)　指定形式の説明

変換対象の文字データ：

変換対象の文字データを指定します。

指定規則を次に示します。

- *変換対象の文字データ*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *変換対象の文字データ*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *変換対象の文字データ*には，？パラメタを単独で指定できません。

変換対象となる文字コードおよび文字範囲を次の表に示します。

表 8-23　スカラ関数 LOWER の対象となる文字コードおよび文字範囲

文字コード	変換対象の文字範囲	変換後の文字範囲	半角/全角
Unicode(UTF-8)	A(0x41)～Z(0x5a)	a(0x61)～z(0x7a)	半角文字
	Ａ(0xefbca1)～Ｚ(0xefbcba)	ａ(0xefbd81)～ｚ(0xefbd9a)	全角文字
Shift-JIS	A(0x41)～Z(0x5a)	a(0x61)～z(0x7a)	半角文字
	Ａ(0x8260)～Ｚ(0x8279)	ａ(0x8281)～ｚ(0x829a)	全角文字

スカラ関数LOWER の実行結果の例を次に示します。

(例)

文字列aBc123XyZ の英大文字を英小文字に変換します。

LOWER('aBc123XyZ') → 'abc123xyz'

(3) 規則

1. 実行結果のデータ型と長さは，*変換対象の文字データ*のデータ型と長さになります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *変換対象の文字データ*がナル値の場合，実行結果はナル値になります。

(4) 例題

例題

社員表（EMPLIST）のNAME 列のデータをすべて小文字に変換します。

```
SELECT "USERID",LOWER("NAME")
   FROM "EMPLIST"
```

EMPLIST

USERID	NAME	SEX	SCODE
U00555	Taro Tanaka	M	S001
U00358	Nancy White	F	S003
U00212	Maria Gomez	F	S001
U00687	Taro Tanaka	M	S002
U00869	NULL	M	S003

検索結果

U00555	taro tanaka
U00358	nancy white
U00212	maria gomez
U00687	taro tanaka
U00869	NULL

8.8.2 UPPER

文字データの英小文字（a～z）を英大文字（A～Z）に変換します。半角英文字および全角英文字に対応しています。

(1) 指定形式

スカラ関数UPPER：：＝UPPER(*変換対象の文字データ*)

変換対象の文字データ：：＝*値式*

(2) 指定形式の説明

変換対象の文字データ：

変換対象の文字データを指定します。

指定規則を次に示します。

- *変換対象の文字データ*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *変換対象の文字データ*には，CHAR 型またはVARCHAR 型のデータを指定してください。
- *変換対象の文字データ*には，？パラメタを単独で指定できません。

変換対象となる文字コードおよび文字範囲を次の表に示します。

表 8-24　スカラ関数 UPPER の変換対象となる文字コードおよび文字範囲

文字コード	変換対象の文字範囲	変換後の文字範囲	半角/全角
Unicode(UTF-8)	a(0x61)～z(0x7a)	A(0x41)～Z(0x5a)	半角文字
	ａ(0xefbd81)～ｚ(0xefbd9a)	Ａ(0xefbca1)～Ｚ(0xefbcba)	全角文字
Shift-JIS	a(0x61)～z(0x7a)	A(0x41)～Z(0x5a)	半角文字
	ａ(0x8281)～ｚ(0x829a)	Ａ(0x8260)～Ｚ(0x8279)	全角文字

スカラ関数UPPER の実行結果の例を次に示します。

(例)

文字列aBc123XyZ の英小文字を，英大文字に変換します。

UPPER('aBc123XyZ') → 'ABC123XYZ'

(3)　規則

1. 実行結果のデータ型と長さは，*変換対象の文字データ*のデータ型と長さになります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *変換対象の文字データ*がナル値の場合，実行結果はナル値になります。

(4)　例題

例題

社員表（EMPLIST）のNAME 列のデータをすべて大文字に変換します。

```
SELECT "USERID",UPPER("NAME")
   FROM "EMPLIST"
```

EMPLIST

USERID	NAME	SEX	SCODE
U00555	Taro Tanaka	M	S001
U00358	Nancy White	F	S003
U00212	Maria Gomez	F	S001
U00687	Taro Tanaka	M	S002
U00869	NULL	M	S003

検索結果

U00555	TARO TANAKA
U00358	NANCY WHITE
U00212	MARIA GOMEZ
U00687	TARO TANAKA
U00869	NULL

8.9 日時関数

ここでは，日時関数の機能と指定形式について説明します。

8.9.1 DATEDIFF

開始日時と終了日時の差を返します。

> **メモ**
>
> スカラ関数DATEDIFF とスカラ関数TIMESTAMPDIFF に機能差はありません。

(1) 指定形式

```
スカラ関数DATEDIFF：：＝DATEDIFF(日時単位,開始日時,終了日時)

  日時単位：：＝ {YEAR | QUARTER | MONTH | WEEK | DAY
                | DAYOFYEAR | HOUR | MINUTE | SECOND
                | MILLISECOND | MICROSECOND | NANOSECOND | PICOSECOND}
  開始日時：：＝値式
  終了日時：：＝値式
```

(2) 指定形式の説明

日時単位：

*開始日時*と*終了日時*の差を求めるときの単位を指定します。次に示すどれかの値を指定してください。

- YEAR

  *開始日時*と*終了日時*の差を年単位で求める場合に指定します。

  (例)

  DATEDIFF(YEAR,'2011-05-05','2013-07-10') → 2

  DATEDIFF(YEAR,'2013-05-05','2013-07-10') → 0

  DATEDIFF(YEAR,'2012-12-31 23:59:59','2013-01-01 00:00:00') → 1

- QUARTER

  *開始日時*と*終了日時*の差を四半期単位で求める場合に指定します。1 月 1 日を基準に，3 か月単位に分けた範囲で計算します。

  ・第 1 四半期：01 月 01 日～03 月 31 日

  ・第 2 四半期：04 月 01 日～06 月 30 日

  ・第 3 四半期：07 月 01 日～09 月 30 日

  ・第 4 四半期：10 月 01 日～12 月 31 日

  (例)

DATEDIFF(QUARTER,'2013-01-05','2013-07-10') → 2

DATEDIFF(QUARTER,'2013-01-05','2013-03-10') → 0

DATEDIFF(QUARTER,'2012-12-31 23:59:59','2013-01-01 00:00:00') → 1

- MONTH

  *開始日時*と*終了日時*の差を月単位で求める場合に指定します。

  (例)

  DATEDIFF(MONTH,'2013-01-05','2013-07-10') → 6

  DATEDIFF(MONTH,'2013-01-05','2013-01-10') → 0

  DATEDIFF(MONTH,'2012-12-31 23:59:59','2013-01-01 00:00:00') → 1

- WEEK

  *開始日時*と*終了日時*の差を週単位で求める場合に指定します。週の最初を日曜日として計算します。

  (例)

  DATEDIFF(WEEK,'2013-07-05','2013-07-10') → 1

  2013 年 7 月 7 日が日曜日で週が変わっているため，1 が返ります。

  DATEDIFF(WEEK,'2012-12-30','2013-01-01') → 0

  2012 年 12 月 30 日が日曜日で週が変わっていないため，0 が返ります。

- DAY

  *開始日時*と*終了日時*の差を日単位で求める場合に指定します。

  (例)

  DATEDIFF(DAY,'2013-07-05','2013-07-10') → 5

  DATEDIFF(DAY,'2013-07-05 08:02:25','2013-07-05 17:55:18') → 0

  DATEDIFF(DAY,'2012-12-31 23:59:59','2013-01-01 00:00:00') → 1

- DAYOFYEAR

  *開始日時*と*終了日時*の差を通算日単位で求める場合に指定します。DAY を指定した場合と同じ結果が返ります。

  (例)

  DATEDIFF(DAYOFYEAR,'2013-07-05','2013-07-10') → 5

  DATEDIFF(DAYOFYEAR,'2013-07-05 08:02:25','2013-07-05 17:55:18') → 0

  DATEDIFF(DAYOFYEAR,'2012-12-31 23:59:59','2013-01-01 00:00:00') → 1

- HOUR

  *開始日時*と*終了日時*の差を時単位で求める場合に指定します。

  (例)

  DATEDIFF(HOUR,'2013-07-10 08:02:25','2013-07-10 11:37:55') → 3

  DATEDIFF(HOUR,'2013-07-10 08:02:25','2013-07-10 08:45:15') → 0

  DATEDIFF(HOUR,'2012-12-31 23:59:59','2013-01-01 00:00:00') → 1

- MINUTE

*開始日時*と*終了日時*の差を分単位で求める場合に指定します。

（例）

```
DATEDIFF(MINUTE,'2013-07-10 08:02:25','2013-07-10 08:07:25') → 5
DATEDIFF(MINUTE,'2013-07-10 08:02:25','2013-07-10 08:02:32') → 0
DATEDIFF(MINUTE,'2012-12-31 23:59:59','2013-01-01 00:00:00') → 1
```

- SECOND

  *開始日時*と*終了日時*の差を秒単位で求める場合に指定します。

  （例）

```
DATEDIFF(SECOND,'2013-07-10 08:02:25','2013-07-10 08:02:33') → 8
DATEDIFF(SECOND,'2012-12-31 23:59:59','2013-01-01 00:00:00') → 1
```

- MILLISECOND

  *開始日時*と*終了日時*の差をミリ秒（1/1,000 秒）単位で求める場合に指定します。

  （例）

```
DATEDIFF(MILLISECOND,'08:02:25.000','08:02:25.003') → 3
DATEDIFF(MILLISECOND,'08:02:24.000','08:02:25.001') → 1001
DATEDIFF(MILLISECOND,'08:02:25.000000','08:02:25.003111') → 3
```

- MICROSECOND

  *開始日時*と*終了日時*の差をマイクロ秒（1/1,000,000 秒）単位で求める場合に指定します。

  （例）

```
DATEDIFF(MICROSECOND,'08:02:25.000000','08:02:25.000012') → 12
```

- NANOSECOND

  *開始日時*と*終了日時*の差をナノ秒（1/1,000,000,000 秒）単位で求める場合に指定します。

  （例）

```
DATEDIFF(NANOSECOND,'08:02:25.000000000','08:02:25.000000123') → 123
```

- PICOSECOND

  *開始日時*と*終了日時*の差をピコ秒（1/1,000,000,000,000 秒）単位で求める場合に指定します。

  （例）

```
DATEDIFF(PICOSECOND,'08:02:25.000000000000','08:02:25.000000000003') → 3
```

開始日時：

開始日時を指定します。

指定規則を次に示します。

- *開始日時*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *開始日時*のデータ型は，DATE 型，TIME 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。

- *開始日時*には，？パラメタを単独で指定できません。

終了日時：

終了日時を指定します。

指定規則を次に示します。

- *終了日時*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *終了日時*のデータ型は，DATE 型，TIME 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- *終了日時*には，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果には，*終了日時*から*開始日時*を減算した値を返します。*終了日時*が*開始日時*よりも前の日時の場合は，負の値を返します。
2. *開始日時*のデータ型がDATE 型で，*終了日時*のデータ型がTIMESTAMP 型のように，片方の日時に時，分，秒の要素がない場合，時，分，秒には00:00:00 が仮定されます。小数秒の桁数が不足している場合は，不足しているすべての桁に 0 が仮定されます。

   （例）DATEDIFF(SECOND,'2013-07-10','2013-07-10 00:00:07') → 7

   上記の例の場合，*開始日時*には'2013-07-10 00:00:00'が仮定されます。
3. *開始日時*にDATE 型，TIMESTAMP 型，日付を表す既定の文字列表現，または時刻印を表す既定の文字列表現を指定した場合，*終了日時*にもDATE 型，TIMESTAMP 型，日付を表す既定の文字列表現，または時刻印を表す既定の文字列表現を指定してください。
4. *開始日時*にTIME 型，または時刻を表す既定の文字列表現を指定した場合，*終了日時*にもTIME 型，または時刻を表す既定の文字列表現を指定してください。
5. *開始日時*および*終了日時*に，TIME 型または時刻を表す既定の文字列表現を指定し，かつ*日時単位*にYEAR，QUARTER，MONTH，DAYOFYEAR，DAY，またはWEEK を指定した場合，実行結果の値は 0 になります。
6. 実行結果のデータ型はINTEGER 型になります。実行結果の値がINTEGER 型で表せる範囲を超えた場合，エラーになります。INTEGER 型で表せる範囲については，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
7. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
8. *開始日時*または*終了日時*がナル値の場合，実行結果はナル値になります。

(4)　例題

例題

表T1 のC1 列とC2 列の日時データの差を，日単位で求めます。

```
SELECT DATEDIFF(DAY,"C1","C2") FROM "T1"
```

表T1

C1列（DATE型）	C2列（DATE型）
2013-06-12	2013-06-20
2013-05-31	2013-06-21
2013-06-20	2013-06-18

検索結果

8
21
-2

8.9.2 DAYOFWEEK

指定した日が，週の何日目かを返します。なお，週の 1 日目は日曜日とします。

(1) 指定形式

スカラ関数DAYOFWEEK：：＝ {DAYOFWEEK | DOW}（*対象データ*）

対象データ：：=*値式*

注 DAYOFWEEK はDOW と省略して指定できます。

(2) 指定形式の説明

対象データ：

日を表すデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データのデータ型は，DATE 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3 既定の文字列表現」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

スカラ関数DAYOFWEEK の実行結果の例を次に示します。

(例)

2012 年 9 月 12 日が，週の何日目かを示す整数値を返します。

DAYOFWEEK(DATE'2012-09-12') → 4

2012 年 9 月 12 日は水曜日のため，4 が返却されます。

(3) 規則

1. 実行結果のデータ型はINTEGER 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。
4. 実行結果の値と曜日の関係を次の表に示します。

表 8-25　実行結果の値と曜日の関係

実行結果の値	曜日
1	日曜
2	月曜
3	火曜
4	水曜
5	木曜
6	金曜
7	土曜

(4) 例題

例題

表T1 のC2 列のデータが，週の何日目かを示す整数値を返します。

```
SELECT ”C1”,DAYOFWEEK(”C2”) FROM ”T1”
```

表T1

C1列 CHAR	C2列 DATE
A001	2011-12-28
A002	2012-01-21
A003	2012-02-23

検索結果

A001	4
A002	7
A003	5

8.9.3 DAYOFYEAR

指定した日が，その年の第何日目かを返します。

(1) 指定形式

スカラ関数DAYOFYEAR：：＝ {DAYOFYEAR | DOY}（*対象データ*）

対象データ：：=*値式*

注　DAYOFYEAR はDOY と省略して指定できます。

(2) 指定形式の説明

対象データ：

日を表すデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データのデータ型は，DATE 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

スカラ関数DAYOFYEAR の実行結果の例を次に示します。

(例)

2013 年 1 月 15 日が，その年の第何日目かを返します。

DAYOFYEAR(DATE'2013-01-15') → 15

(3) 規則

1. 実行結果の値は，その年の第何日目かを示す 1～366 の整数値になります。
2. 実行結果のデータ型はINTEGER 型になります。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC2 列の対象データが，その年の第何日目かを求めます。

```
SELECT "C1",DAYOFYEAR("C2") FROM "T1"
```

表T1

C1列 CHAR	C2列 DATE
A001	2013-01-03
A002	2013-02-01
A003	2013-12-31
A004	2012-12-31

検索結果

A001	3
A002	32
A003	365
A004	366

8.9.4 EXTRACT

日時を示すデータの一部（年，月，日，時，分，または秒のどれか）を抽出します。

(1) 指定形式

スカラ関数EXTRACT：：＝EXTRACT(*抽出部分* FROM *抽出元データ*)

抽出部分：：＝ {YEAR | MONTH | DAY | HOUR | MINUTE | SECOND}
抽出元データ：：＝*値式*

(2) 指定形式の説明

抽出部分：

抽出元データのデータ抽出部分を指定します。次に示すどれかの値を指定してください。ただし，抽出元データに時刻を示すデータがない場合は，HOUR，MINUTE，およびSECOND は指定できません。

- YEAR

  抽出元データの年の部分を抽出する場合に指定します。実行結果の値の範囲は，1～9,999 になります。

- MONTH

  抽出元データの月の部分を抽出する場合に指定します。実行結果の値の範囲は，1～12 になります。

- DAY

  抽出元データの日の部分を抽出する場合に指定します。実行結果の値の範囲は，1～31 になります。

- HOUR

  抽出元データの時の部分を抽出する場合に指定します。実行結果の値の範囲は，0～23 になります。

- MINUTE

  抽出元データの分の部分を抽出する場合に指定します。実行結果の値の範囲は，0～59 になります。

- SECOND

  抽出元データの秒の部分を抽出する場合に指定します。実行結果の値の範囲は，抽出元データの小数秒精度によって次の表に示すとおりに変わります。

  表 8-26　スカラ関数 EXTRACT の実行結果の値の範囲（抽出部分に SECOND を指定した場合）

抽出元データの小数秒精度	実行結果の値の範囲
0	0～59
3	0.000～59.999
6	0.000000～59.999999
9	0.000000000～59.999999999
12	0.000000000000～59.999999999999

抽出元データ：

抽出元のデータを指定します。

指定規則を次に示します。

- 抽出元データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *抽出部分*にYEAR，MONTH，またはDAY を指定した場合，*抽出元データ*のデータ型は，DATE 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，日付を表す既定の入力表現，または時刻印を表す既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- *抽出部分*にHOUR，MINUTE，またはSECOND を指定した場合，*抽出元データ*のデータ型は，TIME 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，時刻を表す既定の入力表現，または時刻印を表す既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- 抽出元データには，？パラメタを単独で指定できません。

スカラ関数EXTRACT の実行結果の例を次に示します。

（例）

DATE 型のデータ（DATE'2012-03-15'）の，年の部分のデータを抽出します。

EXTRACT(YEAR FROM DATE'2012-03-15') → 2012

(3)　規則

1. *抽出部分*にSECOND 以外を指定した場合，実行結果のデータ型はINTEGER 型になります。
2. *抽出部分*にSECOND を指定した場合，抽出元データの小数秒精度によって実行結果のデータ型が次の表に示すとおりに変わります。

表 8-27　スカラ関数 EXTRACT の実行結果のデータ型（抽出部分に SECOND を指定した場合）

抽出元データの小数秒精度	実行結果のデータ型
0	INTEGER
3	DECIMAL(5,3)
6	DECIMAL(8,6)
9	DECIMAL(11,9)
12	DECIMAL(14,12)

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 抽出元データがナル値の場合，実行結果はナル値になります。

(4)　例題

例題 1

表T1 のC2 列が 2012 年のデータを検索します。

```
SELECT ”C1”,”C2” FROM ”T1”
    WHERE EXTRACT(YEAR FROM ”C2”)=2012
```

表T1

C1列 CHAR	C2列 DATE
A001	2011-12-28
A002	2012-01-21
A003	2012-02-23

検索結果

A002	2012-01-21
A003	2012-02-23

例題 2

表T1 の行のうち，C2 列が 3 月以外の行をすべて削除します。

```
DELETE FROM ”T1”
    WHERE EXTRACT(MONTH FROM ”C2”)<>3
```

表T1

C1列 CHAR	C2列 DATE
A001	2012-01-15
A002	2012-02-21
A003	2012-03-09

実行結果

A003	2012-03-09

8.9.5　GETAGE

生年月日と基準日から満年齢を求めます。

(1)　指定形式

```
スカラ関数GETAGE：：＝GETAGE(生年月日,基準日)

  生年月日：：＝値式
  基準日：：＝値式
```

(2)　指定形式の説明

生年月日：

生年月日を指定します。

指定規則を次に示します。

- *生年月日*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *生年月日*のデータ型は，DATE 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，日付を表す既定の入力表現，または時刻印を表す既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- *生年月日*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はDATE 型になります。

基準日：

満年齢を求める基準日を指定します。

指定規則を次に示します。

- *基準日*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *基準日*のデータ型は，DATE 型，TIMESTAMP 型，CHAR 型，または VARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，日付を表す既定の入力表現，または時刻印を表す既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- *基準日*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はDATE 型になります。

スカラ関数GETAGE の実行結果の例を次に示します。

(例)

生年月日が 1986 年 1 月 15 日の人の，2014 年 9 月 30 日時点の満年齢を求めます。

```
GETAGE(DATE'1986-01-15',DATE'2014-09-30') → 28
```

(3) 規則

1. 実行結果のデータ型はINTEGER 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *生年月日*または*基準日*がナル値の場合，実行結果はナル値になります。
4. *基準日*が*生年月日*よりも前の日時の場合，実行結果は0 になります。
5. スカラ関数GETAGE は，基準日時点での満年齢を返します。生年月日の 1 年後の同じ日に満 1 歳と数えます。なお，生年月日が 2 月 29 日の場合，うるう年以外の年は 3 月 1 日を生年月日と仮定します。

(4) 例題

例題

社員表（EMPLIST）から，2015 年 1 月 1 日時点の満年齢が 30 歳以上の社員数を求めます。BIRTH 列には，社員の生年月日が格納されています。

```
SELECT COUNT(*) FROM "EMPLIST"
    WHERE GETAGE("BIRTH",DATE'2015-01-01')>=30
```

検索結果

247

8.9.6 LASTDAY

日時データに指定した月の最終日の日付または日時を返します。

(1) 指定形式

スカラ関数LASTDAY：：＝ {LASTDAY | LAST_DAY}（*日時データ*）

日時データ：：=*値式*

(2) 指定形式の説明

日時データ：

処理対象の日時データを指定します。

指定規則を次に示します。

- *日時データ*は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *日時データ*のデータ型は，DATE 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，日付を表す既定の入力表現または時刻印を表す既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3 既定の文字列表現」を参照してください。

- *日時データ*には，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型は次のようになります。
   - *日時データ*がDATE 型または日付を表す既定の入力表現（CHAR 型，VARCHAR 型）の場合，実行結果のデータ型はDATE 型になります。
   - *日時データ*がTIMESTAMP 型または時刻印を表す既定の入力表現（CHAR 型，VARCHAR 型）の場合，実行結果のデータ型はTIMESTAMP 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *日時データ*がナル値の場合，実行結果はナル値になります。
4. *日時データ*がTIMESTAMP 型または時刻印を表す既定の入力表現（CHAR 型，VARCHAR 型）の場合，時，分，秒，および小数秒の部分は入力値をそのまま返します。

   （例）

   LASTDAY(TIMESTAMP'2014-07-03 15:30:45.123') → '2014-07-31 15:30:45.123'

(4) 例題

例題

表T1 のC2 列の日時データに対して，その月の最終日の日付を求めます。

```
SELECT "C1",LASTDAY("C2") FROM "T1"
```

表T1

C1列 CHAR	C2列 DATE
A001	2013-01-03
A002	2013-02-01
A003	2012-02-01

検索結果

A001	2013-01-31
A002	2013-02-28
A003	2012-02-29

8.9.7 ROUND

日時データを日時書式で指定した単位に丸めて返します。

数データを丸めるスカラ関数ROUND については，「8.4.9 ROUND」を参照してください。

(1)　指定形式

スカラ関数ROUND：：＝ROUND(*日時データ*,*日時書式*)

日時データ：：＝*値式*
日時書式：：＝*定数*

(2)　指定形式の説明

日時データ：

処理対象の日時データを指定します。

指定規則を次に示します。

- 日時データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 日時データのデータ型は，DATE 型，TIME 型，またはTIMESTAMP 型のどれかにしてください。
- 日時データには，？パラメタを単独で指定できません。

日時書式：

日時データを丸める単位を指定します。

指定規則を次に示します。

- *日時書式*には，文字列定数を指定します。文字列定数については，「6.3　定数」を参照してください。
- 日時書式に指定できる要素を次の表に示します。

表 8-28　日時書式に指定できる要素

項番	日時書式に指定できる要素	単位	説明
1	CC	世紀	日時データが各世紀の 51 年目以降の場合は，翌世紀の最初の年の 1 月 1 日 0 時 0 分 0 秒に切り上げます。50 年目以前の場合は，同世紀の最初の年の 1 月 1 日 0 時 0 分 0 秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'1951-10-04 15:25:38','CC') →TIMESTAMP'2001-01-01 00:00:00' • 切り捨ての例 ROUND(TIMESTAMP'1950-10-04 15:25:38','CC') →TIMESTAMP'1901-01-01 00:00:00'
2	YYYY YYYYN YY YYN	年	日時データが 7 月 1 日以降の場合は，翌年の 1 月 1 日 0 時 0 分 0 秒に切り上げます。6 月 30 日以前の場合は，同年の 1 月 1 日 0 時 0 分 0 秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2013-07-01 15:25:38','YYYY') →TIMESTAMP'2014-01-01 00:00:00' • 切り捨ての例 ROUND(TIMESTAMP'2013-06-30 15:25:38','YYYY')

項番	日時書式に指定できる要素	単位	説明
			→TIMESTAMP'2013-01-01 00:00:00'
3	Q	四半期	日時データが各四半期の2番目の月（2月，5月，8月，または11月）の16日以降の場合は，翌四半期の最初の月の1日0時0分0秒に切り上げます。15日以前の場合は，同四半期の最初の月の1日0時0分0秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2013-11-16 15:25:38','Q') →TIMESTAMP'2014-01-01 00:00:00' • 切り捨ての例 ROUND(TIMESTAMP'2013-11-15 15:25:38','Q') →TIMESTAMP'2013-10-01 00:00:00' 1月1日を基準にして，3か月単位を四半期とします。 • 第1四半期：1月1日～3月31日 • 第2四半期：4月1日～6月30日 • 第3四半期：7月1日～9月30日 • 第4四半期：10月1日～12月31日
4	MONTH MON MM	月	日時データが16日以降の場合は，翌月の1日0時0分0秒に切り上げます。15日以前の場合は，同月の1日0時0分0秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2014-01-16 15:25:38','MONTH') →TIMESTAMP'2014-02-01 00:00:00' • 切り捨ての例 ROUND(TIMESTAMP'2014-01-15 15:25:38','MONTH') →TIMESTAMP'2014-01-01 00:00:00'
5	WW	週	同年の最初の曜日を週の開始日とします。 日時データが，週の開始日から4日目の正午12時以降の場合は，翌週の開始日の0時0分0秒に切り上げます。4日目の正午12時より前の場合は，同週の開始日の0時0分0秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2014-01-04 15:25:38','WW') →TIMESTAMP'2014-01-08 00:00:00' 2014年1月1日は水曜日のため，週の開始日は水曜日になります。そのため，その週の4日目（1月4日の土曜日）の正午12時以降の場合は，翌週の開始日（1月8日）の0時0分0秒に切り上げます。 • 切り捨ての例 ROUND(TIMESTAMP'2014-01-04 10:25:38','WW') →TIMESTAMP'2014-01-01 00:00:00' 2014年1月1日は水曜日のため，週の開始日は水曜日になります。そのため，その週の4日目（1月4日の土曜日）の正午12

項番	日時書式に指定できる要素	単位	説明
			時より前の場合は，同週の開始日（1 月 1 日）の 0 時 0 分 0 秒に切り捨てます。
6	W	週	同月の最初の曜日を週の開始日とします。 日時データが，週の開始日から 4 日目の正午 12 時以降の場合は，翌週の開始日の 0 時 0 分 0 秒に切り上げます。4 日目の正午 12 時より前の場合は，同週の開始日の 0 時 0 分 0 秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2014-02-04 12:25:38','W') →TIMESTAMP'2014-02-08 00:00:00' 2014 年 2 月 1 日は土曜日のため，週の開始日は土曜日になります。そのため，その週の 4 日目（2 月 4 日の火曜日）の正午 12 時以降の場合は，翌週の開始日（2 月 8 日）の 0 時 0 分 0 秒に切り上げます。 • 切り捨ての例 ROUND(TIMESTAMP'2014-02-04 11:55:38','W') →TIMESTAMP'2014-02-01 00:00:00' 2014 年 2 月 1 日は土曜日のため，週の開始日は土曜日になります。そのため，その週の 4 日目（2 月 4 日の火曜日）の正午 12 時より前の場合は，同週の開始日（2 月 1 日）の 0 時 0 分 0 秒に切り捨てます。
7	DAY DAYN DY DYN D	週	日曜日を週の開始日とします。 日時データが，週の開始日から 4 日目（水曜日）の正午 12 時以降の場合は，翌週の開始日の 0 時 0 分 0 秒に切り上げます。4 日目の正午 12 時より前の場合は，同週の開始日の 0 時 0 分 0 秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2014-02-05 12:25:38','DAY') →TIMESTAMP'2014-02-09 00:00:00' 2014 年 2 月 5 日は水曜日です。そのため，2 月 9 日（日曜日）の 0 時 0 分 0 秒に切り上げます。 • 切り捨ての例 ROUND(TIMESTAMP'2014-02-05 11:55:38','DAY') →TIMESTAMP'2014-02-02 00:00:00' 2014 年 2 月 5 日は水曜日です。そのため，2 月 2 日（日曜日）の 0 時 0 分 0 秒に切り捨てます。
8	DD DDD	日	日時データが正午 12 時以降の場合は，翌日の 0 時 0 分 0 秒に切り上げます。正午 12 時より前の場合は，同日の 0 時 0 分 0 秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2014-01-16 15:25:38','DD') →TIMESTAMP'2014-01-17 00:00:00' • 切り捨ての例 ROUND(TIMESTAMP'2014-01-16 10:25:38','DD')

項番	日時書式に指定できる要素	単位	説明
			→TIMESTAMP'2014-01-16 00:00:00'
9	HH HH12 HH24	時	日時データが 30 分以降の場合は，翌時の 0 分 0 秒に切り上げます。29 分以前の場合は，同時の 0 分 0 秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2014-01-16 15:35:38','HH') →TIMESTAMP'2014-01-16 16:00:00' • 切り捨ての例 ROUND(TIMESTAMP'2014-01-16 15:25:38','HH') →TIMESTAMP'2014-01-16 15:00:00'
10	MI	分	日時データが 30 秒以降の場合は，翌分の 0 秒に切り上げます。29 秒以前の場合は，同分の 0 秒に切り捨てます。 • 切り上げの例 ROUND(TIMESTAMP'2014-01-16 15:35:33','MI') →TIMESTAMP'2014-01-16 15:36:00' • 切り捨ての例 ROUND(TIMESTAMP'2014-01-16 15:35:28','MI') →TIMESTAMP'2014-01-16 15:35:00'
11	SSSSS SS	秒	日時データが 500 ミリ秒以降の場合は，翌秒に切り上げます。500 ミリ秒未満の場合は，小数秒部分を切り捨てます。 • 切り上げの例 ROUND(TIME'11:59:30.596123','SS') →TIME'11:59:31.000000' • 切り捨ての例 ROUND(TIME'11:59:30.488123','SS') →TIME'11:59:30.000000'

- 日時書式に指定する文字データは，半角文字で指定してください。また，大文字，小文字は区別されません。
- 日時書式に指定できる要素が複数ある場合，どれか 1 つを指定してください。どの要素を指定しても同じ実行結果になります。例えば，日時書式の要素としてYYYY とYYYYN のどちらを指定しても実行結果は同じになります。
- 日時書式の要素の前後の空白は無視されます。
- 日時書式の長さは，64 バイト以内にしてください。

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-29　スカラ関数 ROUND の実行結果のデータ型とデータ長

日時データのデータ型とデータ長	実行結果のデータ型とデータ長
DATE	DATE
TIME(*p*)	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)

（凡例）*p*：小数秒精度

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 日時データがナル値の場合，実行結果はナル値になります。
4. *日時データ*がDATE 型の場合に，日時書式に時刻を丸める指定（DDD，DD，HH，HH12，HH24，MI，SSSSS，またはSS）をしたときは，入力された日時データをそのまま返します。
5. 日時データにTIME 型のデータを指定した場合，日時書式に世紀，年，四半期，月，週，日を丸める指定（CC，YYYY，YYYYN，YY，YYN，Q，MONTH，MON，MM，WW，W，DAY，DAYN，DY，DYN，D，DDD，またはDD）はできません。
6. *日時データ*がDATE 型の場合，時刻要素には 00:00:00 が仮定されます。そのため，次のような指定をしたときは，週単位の切り捨てが行われます。

   （例）

   ROUND(DATE'2013-10-04','W') → DATE'2013-10-01'
7. 実行結果のデータ型がDATE 型の場合，実行結果が 1 年 1 月 1 日～9999 年 12 月 31 日の範囲に収まらないとエラーになります。
8. 実行結果のデータ型がTIME 型の場合，実行結果が 0 時 0 分 0.000000000000 秒～23 時 59 分 59.999999999999 秒の範囲に収まらないとエラーになります。
9. 実行結果のデータ型がTIMESTAMP 型の場合，実行結果が 1 年 1 月 1 日の 0 時 0 分 0.000000000000 秒～9999 年 12 月 31 日の 23 時 59 分 59.999999999999 秒の範囲に収まらないとエラーになります。

(4)　例題

例題

販売履歴表（SALESLIST）から，商品コード（PUR-CODE）P001 の商品の販売個数を半期ごと（2013 年 1/1～6/30 までと，2013 年 7/1～12/31 まで）に集計します。

```
SELECT SUM("PUR-NUM") FROM "SALESLIST"
    WHERE "PUR-DATE" BETWEEN DATE'2013-01-01' AND DATE'2013-12-31'
    AND "PUR-CODE"='P001'
    GROUP BY ROUND("PUR-DATE",'YYYY')
```

販売履歴表(SALESLIST)

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P001	5	2013-01-23
U00358	P002	3	2013-02-04
U00555	P001	4	2013-03-07
U00212	P003	2	2013-04-12
U00687	P001	2	2013-06-13
U00555	P001	4	2013-06-30
U00687	P001	3	2013-07-01
U00555	P003	4	2013-07-10
U00212	P001	3	2013-09-24
U00555	P002	4	2013-10-26
U00358	P001	8	2013-11-30
U00555	P003	2	2013-12-31

検索結果

15	← 上半期(1/1～6/30)の販売個数
14	← 下半期(7/1～12/31)の販売個数

日時書式の要素にYYYY を指定した場合，7 月 1 日以降のデータは切り上げ対象となります。また，6/30 以前のデータは切り捨ての対象となります。そのため，1/1～6/30 を上半期，7/1～12/31 を下半期とした半期ごとの集計ができます。

8.9.8 TIMESTAMPADD

対象データに指定した日時に，日時単位に指定した単位で日時を加算します。

(1) 指定形式

```
スカラ関数TIMESTAMPADD：：＝TIMESTAMPADD(日時単位,加算値,対象データ)

  日時単位：：＝ {YEAR | QUARTER | MONTH | WEEK | DAY | DAYOFYEAR | HOUR | MINUTE
                 | SECOND | MILLISECOND | MICROSECOND | NANOSECOND | PICOSECOND}
  加算値：：＝値式
  対象データ：：＝値式
```

(2) 指定形式の説明

日時単位：

対象データに加算する日時の単位を指定します。

日時単位に指定できる単位と，加算値に指定できる数データの値の範囲を次の表に示します。

表 8-30 日時単位に指定できる単位と，加算値に指定できる数データの値の範囲

日時単位の指定	説明	加算値に指定できる数データの値の範囲
YEAR	年	-9,998～9,998

日時単位の指定	説明	加算値に指定できる数データの値の範囲
QUARTER	四半期 1QUARTER は，3MONTH（3 か月）として計算されます。	-39,995～39,995
MONTH	月	-119,987～119,987
WEEK	週 1WEEK は，7DAY（7 日）として計算されます。	-521,722～521,722
DAY	日	-3,652,058～3,652,058
DAYOFYEAR	通算日 DAY を指定したときと同じ結果が返ります。	-3,652,058～3,652,058
HOUR	時	-87,649,415～87,649,415
MINUTE	分	-5,258,964,959～5,258,964,959
SECOND	秒	-315,537,897,599～315,537,897,599
MILLISECOND	ミリ秒（1/1,000 秒） 仮定される小数秒精度は 3 です。	-315,537,897,599,999～315,537,897,599,999
MICROSECOND	マイクロ秒（1/1,000,000 秒） 仮定される小数秒精度は 6 です。	-315,537,897,599,999,999～315,537,897,599,999,999
NANOSECOND	ナノ秒（1/1,000,000,000 秒） 仮定される小数秒精度は 9 です。	-9,223,372,036,854,775,807～9,223,372,036,854,775,807
PICOSECOND	ピコ秒（1/1,000,000,000,000 秒） 仮定される小数秒精度は 12 です。	

加算値：

対象データの日時に加算する値を指定します。

指定規則を次に示します。

- 加算値は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 加算値のデータ型は，INTEGER 型またはSMALLINT 型のどちらかにしてください。
- 対象データの日時を減算したい場合は，加算値に負の値を指定します。

  （例）

  ```
  TIMESTAMPADD(DAY,-1,DATE'2020-06-20') → DATE'2020-06-19'
  ```

- 加算値に指定できる数データの値の範囲については，「表 8-30　日時単位に指定できる単位と，加算値に指定できる数データの値の範囲」を参照してください。
- 加算値に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

対象データ：

日時を加算する対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，？パラメタを単独で指定できません。
- 対象データのデータ型は，DATE 型，TIME 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- 日時単位と対象データの指定可能な組み合わせを次の表に示します。

表 8-31　日時単位と対象データの指定可能な組み合わせ

日時単位の指定	指定可能な対象データのデータ型		
	DATE 型または日付を表す既定の入力表現	TIME 型または時刻を表す既定の入力表現	TIMESTAMP 型または時刻印を表す既定の入力表現
YEAR	○	×	○
QUARTER	○	×	○
MONTH	○	×	○
WEEK	○	×	○
DAY	○	×	○
DAYOFYEAR	○	×	○
HOUR	×	○	○
MINUTE	×	○	○
SECOND	×	○	○
MILLISECOND	×	○	○
MICROSECOND	×	○	○
NANOSECOND	×	○	○
PICOSECOND	×	○	○

（凡例）

○：指定できます。

×：指定できません。

スカラ関数TIMESTAMPADD の実行結果の例を次に示します。

（例 1）

```
TIMESTAMPADD(DAY,2,DATE'2020-03-01') → DATE'2020-03-03'
```

（例 2）

```
TIMESTAMPADD(MILLISECOND,1,TIMESTAMP'2019-12-31 23:59:59.999')
→ TIMESTAMP'2020-01-01 00:00:00.000'
```

(3) 規則

1. 実行結果には，指定した日時単位で加算値を，対象データに加算した値が返されます。
2. 実行結果のデータ型は次のようになります。
   - 対象データがDATE 型または日付を表す既定の入力表現（CHAR 型，VARCHAR 型）の場合，実行結果のデータ型はDATE 型になります。
   - 対象データがTIME 型または時刻を表す既定の入力表現（CHAR 型，VARCHAR 型）の場合，実行結果のデータ型はTIME 型になります。
   - 対象データがTIMESTAMP 型または時刻印を表す既定の入力表現（CHAR 型，VARCHAR 型）の場合，実行結果のデータ型はTIMESTAMP 型になります。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 加算値または対象データがナル値の場合，実行結果はナル値になります。
5. 実行結果の値が，DATE 型，TIME 型，またはTIMESTAMP 型の値の範囲に収まらない場合，エラーになります。DATE 型，TIME 型，およびTIMESTAMP 型の値の範囲については，「6.2.1　データ型の種類」の「(3)　日時データ」を参照してください。
6. 年，月を加減算した結果が，存在しない日付（小の月の 31 日，またはうるう年以外の 2 月 29 日）になった場合，その月の最終日に補正されます。

   （例）

   ```
   TIMESTAMPADD(YEAR,1,DATE'2020-02-29') → DATE'2021-02-28'
   ```

   2021 年はうるう年ではないため，実行結果が上記のように補正されます。
7. 日を加減算した結果が，その月の最終日を超えたり，または初日（1 日）より前になったりした場合，年，月の桁上げまたは桁下げが行われます。

   （例）

   ```
   TIMESTAMPADD(DAY,1,DATE'2020-12-31') → DATE'2021-01-01'
   TIMESTAMPADD(DAY,-1,DATE'2020-07-01') → DATE'2020-06-30'
   ```
8. 対象データの小数秒精度と，加算する時刻または時刻印の小数秒精度が異なる場合，精度が高い方の小数秒精度に合わせます（精度が低い方に 0 を補います）。例えば，対象データの小数秒精度が 0 で，日時単位がMILLISECOND の場合，対象データの小数秒精度を 3 に拡張したあとに加算値を加算します。
9. 時を加減算した結果が，23 時 59 分 59.999999999999 秒を超えたり，00 時 00 分 00.000000000000 秒より前になったりした場合，日の桁上げまたは桁下げが行われます。

   （例）

```
TIMESTAMPADD(SECOND,1,TIMESTAMP'2020-02-01 23:59:59') → TIMESTAMP'2020-02-02 00:00:00'
TIMESTAMPADD(SECOND,-1,TIMESTAMP'2020-02-02 00:00:00') → TIMESTAMP'2020-02-01 23:59:59'
```

10. スカラ関数TIMESTAMPADD は，次の表に示す日時演算に変換されたあとに実行されます。

表 8-32　スカラ関数 TIMESTAMPADD から日時演算への変換規則

TIMESTAMPADD の指定形式	変換後の日時演算
TIMESTAMPADD(YEAR,*加算値*,*対象データ*)	*対象データ* + (*加算値*)YEAR
TIMESTAMPADD(QUARTER,*加算値*,*対象データ*)	*対象データ* + ((*加算値*)*3)MONTH
TIMESTAMPADD(MONTH,*加算値*,*対象データ*)	*対象データ* + (*加算値*)MONTH
TIMESTAMPADD(WEEK,*加算値*,*対象データ*)	*対象データ* + ((*加算値*)*7)DAY
TIMESTAMPADD(DAY,*加算値*,*対象データ*)	*対象データ* + (*加算値*)DAY
TIMESTAMPADD(DAYOFYEAR,*加算値*,*対象データ*)	*対象データ* + (*加算値*)DAY
TIMESTAMPADD(HOUR,*加算値*,*対象データ*)	*対象データ* + (*加算値*)HOUR
TIMESTAMPADD(MINUTE,*加算値*,*対象データ*)	*対象データ* + (*加算値*)MINUTE
TIMESTAMPADD(SECOND,*加算値*,*対象データ*)	*対象データ* + (*加算値*)SECOND
TIMESTAMPADD(MILLISECOND,*加算値*,*対象データ*)	*対象データ* + (*加算値*)MILLISECOND
TIMESTAMPADD(MICROSECOND,*加算値*,*対象データ*)	*対象データ* + (*加算値*)MICROSECOND
TIMESTAMPADD(NANOSECOND,*加算値*,*対象データ*)	*対象データ* + (*加算値*)NANOSECOND
TIMESTAMPADD(PICOSECOND,*加算値*,*対象データ*)	*対象データ* + (*加算値*)PICOSECOND

(4)　例題

例題

表T1 のC2 列の対象データ（日付）に，5 日を加算した日付を求めます。

```
SELECT "C1",TIMESTAMPADD(DAY,5,"C2") FROM "T1"
```

表T1

C1列 CHAR	C2列 DATE
A001	2020-07-01
A002	2020-07-10
A003	2020-07-31
A004	2020-12-31

検索結果

A001	2020-07-06
A002	2020-07-15
A003	2020-08-05
A004	2021-01-05

8.9.9 TIMESTAMPDIFF

開始日時と終了日時の差を返します。

> **メモ**
>
> スカラ関数TIMESTAMPDIFF とスカラ関数DATEDIFF に機能差はありません。

(1) 指定形式

```
スカラ関数TIMESTAMPDIFF：：＝TIMESTAMPDIFF(日時単位,開始日時,終了日時)

  日時単位：：＝ {YEAR | QUARTER | MONTH | WEEK | DAY | DAYOFYEAR | HOUR | MINUTE
                 | SECOND | MILLISECOND | MICROSECOND | NANOSECOND | PICOSECOND}
  開始日時：：＝値式
  終了日時：：＝値式
```

(2) 指定形式の説明

日時単位：

開始日時と終了日時の差を求めるときの単位を指定します。次に示すどれかの値を指定してください。

- YEAR

  開始日時と終了日時の差を年単位で求める場合に指定します。

  (例)

  TIMESTAMPDIFF(YEAR,'2018-05-05','2020-07-10') → 2

  TIMESTAMPDIFF(YEAR,'2020-05-05','2020-07-10') → 0

  TIMESTAMPDIFF(YEAR,'2019-12-31 23:59:59','2020-01-01 00:00:00') → 1

- QUARTER

  開始日時と終了日時の差を四半期単位で求める場合に指定します。1 月 1 日を基準に，3 か月単位に分けた範囲で計算します。

  ・第 1 四半期：01 月 01 日～03 月 31 日

  ・第 2 四半期：04 月 01 日～06 月 30 日

  ・第 3 四半期：07 月 01 日～09 月 30 日

  ・第 4 四半期：10 月 01 日～12 月 31 日

  (例)

  TIMESTAMPDIFF(QUARTER,'2020-01-05','2020-07-10') → 2

  TIMESTAMPDIFF(QUARTER,'2020-01-05','2020-03-10') → 0

  TIMESTAMPDIFF(QUARTER,'2019-12-31 23:59:59','2020-01-01 00:00:00') → 1

- MONTH

  開始日時と終了日時の差を月単位で求める場合に指定します。

（例）

TIMESTAMPDIFF(MONTH,'2020-01-05','2020-07-10') → 6

TIMESTAMPDIFF(MONTH,'2020-06-05','2020-06-10') → 0

TIMESTAMPDIFF(MONTH,'2019-12-31 23:59:59','2020-01-01 00:00:00') → 1

- WEEK

  開始日時と終了日時の差を週単位で求める場合に指定します。週の最初を日曜日として計算します。

  （例）

  TIMESTAMPDIFF(WEEK,'2020-07-03','2020-07-06') → 1

  2020 年 7 月 5 日が日曜日で週が変わっているため，1 が返ります。

  TIMESTAMPDIFF(WEEK,'2019-12-30','2020-01-01') → 0

  2019 年 12 月 30 日が月曜日で週が変わっていないため，0 が返ります。

- DAY

  開始日時と終了日時の差を日単位で求める場合に指定します。

  （例）

  TIMESTAMPDIFF(DAY,'2020-07-05','2020-07-10') → 5

  TIMESTAMPDIFF(DAY,'2020-07-05 08:02:25','2020-07-05 17:55:18') → 0

  TIMESTAMPDIFF(DAY,'2019-12-31 23:59:59','2020-01-01 00:00:00') → 1

- DAYOFYEAR

  開始日時と終了日時の差を通算日単位で求める場合に指定します。DAY を指定した場合と同じ結果が返ります。

  （例）

  TIMESTAMPDIFF(DAYOFYEAR,'2020-07-05','2020-07-10') → 5

  TIMESTAMPDIFF(DAYOFYEAR,'2020-07-05 08:02:25','2020-07-05 17:55:18') → 0

  TIMESTAMPDIFF(DAYOFYEAR,'2019-12-31 23:59:59','2020-01-01 00:00:00') → 1

- HOUR

  開始日時と終了日時の差を時単位で求める場合に指定します。

  （例）

  TIMESTAMPDIFF(HOUR,'2020-07-10 08:02:25','2020-07-10 11:37:55') → 3

  TIMESTAMPDIFF(HOUR,'2020-07-10 08:02:25','2020-07-10 08:45:15') → 0

  TIMESTAMPDIFF(HOUR,'2019-12-31 23:59:59','2020-01-01 00:00:00') → 1

- MINUTE

  開始日時と終了日時の差を分単位で求める場合に指定します。

  （例）

  TIMESTAMPDIFF(MINUTE,'2020-07-10 08:02:25','2020-07-10 08:07:25') → 5

  TIMESTAMPDIFF(MINUTE,'2020-07-10 08:02:25','2020-07-10 08:02:32') → 0

  TIMESTAMPDIFF(MINUTE,'2019-12-31 23:59:59','2020-01-01 00:00:00') → 1

- SECOND

  開始日時と終了日時の差を秒単位で求める場合に指定します。

  （例）

  TIMESTAMPDIFF(SECOND,'2020-07-10 08:02:25','2020-07-10 08:02:33') → 8

  TIMESTAMPDIFF(SECOND,'2019-12-31 23:59:59','2020-01-01 00:00:00') → 1

- MILLISECOND

  開始日時と終了日時の差をミリ秒（1/1,000 秒）単位で求める場合に指定します。

  （例）

  TIMESTAMPDIFF(MILLISECOND,'08:02:25.000','08:02:25.003') → 3

  TIMESTAMPDIFF(MILLISECOND,'08:02:24.000','08:02:25.001') → 1001

  TIMESTAMPDIFF(MILLISECOND,'08:02:25.000000','08:02:25.003111') → 3

- MICROSECOND

  開始日時と終了日時の差をマイクロ秒（1/1,000,000 秒）単位で求める場合に指定します。

  （例）

  TIMESTAMPDIFF(MICROSECOND,'08:02:25.000000','08:02:25.000012') → 12

- NANOSECOND

  開始日時と終了日時の差をナノ秒（1/1,000,000,000 秒）単位で求める場合に指定します。

  （例）

  TIMESTAMPDIFF(NANOSECOND,'08:02:25.000000000','08:02:25.000000123') → 123

- PICOSECOND

  開始日時と終了日時の差をピコ秒（1/1,000,000,000,000 秒）単位で求める場合に指定します。

  （例）

  TIMESTAMPDIFF(PICOSECOND,'08:02:25.000000000000','08:02:25.000000000003') → 3

開始日時：

開始日時を指定します。

指定規則を次に示します。

- 開始日時は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 開始日時のデータ型は，DATE 型，TIME 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- 開始日時には，？パラメタを単独で指定できません。

終了日時：

終了日時を指定します。

指定規則を次に示します。

- 終了日時は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 終了日時のデータ型は，DATE 型，TIME 型，TIMESTAMP 型，CHAR 型，またはVARCHAR 型のどれかにしてください。ただし，CHAR 型またはVARCHAR 型の場合は，既定の入力表現の形式に従っている文字列定数だけを指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- 終了日時には，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果には，終了日時から開始日時を日時単位で減算した値を返します。終了日時が開始日時よりも前の日時の場合は，負の値を返します。
2. 開始日時または終了日時に指定した日時データの欠落している要素の値には，既定値を設定して差が求められます。時刻要素が欠落している日付データの時刻要素には，00:00:00 が仮定されます。小数秒部分の桁数が不足している場合は，不足しているすべての桁に0 が仮定されます。

   例えば，開始日時のデータ型がDATE 型で，終了日時のデータ型がTIMESTAMP 型のように，片方の日時データに時，分，秒の要素がない場合，時，分，秒には00:00:00 が仮定されます。小数秒部分の桁数が不足している場合は，不足しているすべての桁に0 が仮定されます。

   （例）TIMESTAMPDIFF(SECOND,'2020-07-10','2020-07-10 00:00:07') → 7

   上記の例の場合，開始日時には'2020-07-10 00:00:00'が仮定されます。
3. 開始日時にDATE 型，TIMESTAMP 型，日付を表す既定の入力表現，または時刻印を表す既定の入力表現を指定した場合，終了日時にもDATE 型，TIMESTAMP 型，日付を表す既定の入力表現，または時刻印を表す既定の入力表現を指定してください。
4. 開始日時にTIME 型，または時刻を表す既定の入力表現を指定した場合，終了日時にもTIME 型，または時刻を表す既定の入力表現を指定してください。
5. 開始日時および終了日時に，TIME 型または時刻を表す既定の入力表現を指定し，かつ日時単位にYEAR，QUARTER，MONTH，WEEK，DAY，またはDAYOFYEAR を指定した場合，実行結果の値は0 になります。
6. 実行結果のデータ型はINTEGER 型になります。実行結果の値がINTEGER 型で表せる範囲を超えた場合，エラーになります。INTEGER 型で表せる範囲については，「6.2.1　データ型の種類」の「(1)　数データ」を参照してください。
7. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
8. 開始日時または終了日時がナル値の場合，実行結果はナル値になります。
9. スカラ関数TIMESTAMPDIFF の実行結果の値が1 になる例を次に示します。この例では，開始日時と終了日時の差は 1 秒となっています。

日時単位の指定	開始日時の指定	終了日時の指定	実行結果
YEAR	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1
QUARTER	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1
MONTH	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1

日時単位の指定	開始日時の指定	終了日時の指定	実行結果
WEEK	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1※
DAY	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1
DAYOFYEAR	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1
HOUR	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1
MINUTE	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1
SECOND	'2011-12-31 23:59:59'	'2012-01-01 00:00:00'	1

注※

2011 年 12 月 31 日は土曜日で，2012 年 1 月 1 日は日曜日のため，実行結果の値は1 になります。開始日時と終了日時に指定した年が異なる場合であっても，開始日時と終了日時の両方が同じ週の場合，実行結果の値は0 になります。

(4) 例題

例題

表T1 のC1 列とC2 列の日時データの差を，日単位で求めます。

```
SELECT TIMESTAMPDIFF(DAY,"C1","C2") FROM "T1"
```

表T1

C1列（DATE型）	C2列（DATE型）
2020-06-12	2020-06-20
2020-05-31	2020-06-21
2020-06-20	2020-06-18

検索結果

8
21
-2

8.9.10 TRUNC

日時データを日時書式で指定した単位で切り捨てます。

数データを切り捨てるスカラ関数TRUNC については，「8.4.12 TRUNC」を参照してください。

(1) 指定形式

スカラ関数TRUNC ：：＝TRUNC(*日時データ*,*日時書式*)

日時データ ：：＝*値式*
日時書式 ：：＝*定数*

(2) 指定形式の説明

日時データ：

処理対象の日時データを指定します。

指定規則を次に示します。

- 日時データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 日時データのデータ型は，DATE 型，TIME 型，またはTIMESTAMP 型のどれかにしてください。
- 日時データには，？パラメタを単独で指定できません。

日時書式：

日時データを切り捨てる単位を指定します。

指定規則を次に示します。

- 日時書式には，文字列定数を指定します。文字列定数については，「6.3 定数」を参照してください。
- 日時書式に指定できる要素を次の表に示します。

表 8-33 日時書式に指定できる要素

項番	日時書式に指定できる要素	単位	説明
1	CC	世紀	日時データを同世紀の最初の年の 1 月 1 日 0 時 0 分 0 秒に切り捨てます。 (例) TRUNC(TIMESTAMP'2014-03-14 15:25:38','CC') →TIMESTAMP'2001-01-01 00:00:00'
2	YYYY YYYYN YY YYN	年	日時データを同年の 1 月 1 日 0 時 0 分 0 秒に切り捨てます。 (例) TRUNC(TIMESTAMP'2014-03-14 15:25:38','YYYY') →TIMESTAMP'2014-01-01 00:00:00'
3	Q	四半期	日時データを同四半期の最初の月の 1 日 0 時 0 分 0 秒に切り捨てます。 (例) TRUNC(TIMESTAMP'2014-03-14 15:25:38','Q') →TIMESTAMP'2014-01-01 00:00:00' 1 月 1 日を基準にして，3 か月単位を四半期とします。 • 第 1 四半期：1 月 1 日～3 月 31 日 • 第 2 四半期：4 月 1 日～6 月 30 日 • 第 3 四半期：7 月 1 日～9 月 30 日 • 第 4 四半期：10 月 1 日～12 月 31 日
4	MONTH MON MM	月	日時データを同月の 1 日 0 時 0 分 0 秒に切り捨てます。 (例) TRUNC(TIMESTAMP'2014-03-14 15:25:38','MONTH') →TIMESTAMP'2014-03-01 00:00:00'

項番	日時書式に指定できる要素	単位	説明
5	WW	週	日時データを同週の開始日の０時０分０秒に切り捨てます。同年の最初の曜日を週の開始日とし，その週の開始日の０時０分０秒に切り捨てます。 （例） TRUNC(TIMESTAMP'2014-03-14 15:25:38','WW') →TIMESTAMP'2014-03-12 00:00:00' 2014年1月1日は水曜日のため，週の開始日は水曜日になります。そのため，その週の開始日（水曜日）である3月12日の０時０分０秒に切り捨てます。
6	W	週	日時データを同週の開始日の０時０分０秒に切り捨てます。同月の最初の曜日を週の開始日とし，その週の開始日の０時０分０秒に切り捨てます。 （例） TRUNC(TIMESTAMP'2014-03-14 15:25:38','W') →TIMESTAMP'2014-03-08 00:00:00' 2014年3月1日は土曜日のため，週の開始日は土曜日になります。そのため，その週の開始日（土曜日）である3月8日の０時０分０秒に切り捨てます。
7	DAY DAYN DY DYN D	週	日時データを同週の開始日の０時０分０秒に切り捨てます。日曜日を週の開始日とし，その週の開始日の０時０分０秒に切り捨てます。 （例） TRUNC(TIMESTAMP'2014-03-14 15:25:38','DAY') →TIMESTAMP'2014-03-09 00:00:00' 2014年3月14日は金曜日です。そのため，その週の開始日（日曜日）である3月9日の０時０分０秒に切り捨てます。
8	DD DDD	日	日時データを同日の０時０分０秒に切り捨てます。 （例） TRUNC(TIMESTAMP'2014-03-14 15:25:38','DD') →TIMESTAMP'2014-03-14 00:00:00'
9	HH HH12 HH24	時	日時データを同時の０分０秒に切り捨てます。 （例） TRUNC(TIMESTAMP'2014-03-14 15:25:38','HH') →TIMESTAMP'2014-03-14 15:00:00'
10	MI	分	日時データを同分の０秒に切り捨てます。 （例） TRUNC(TIMESTAMP'2014-03-14 15:25:38','MI') →TIMESTAMP'2014-03-14 15:25:00'
11	SSSSS SS	秒	日時データの小数秒を切り捨てます。 （例） TRUNC(TIME'11:58:31.784','SS') →TIME'11:58:31.000'

- 日時書式に指定する文字データは，半角文字で指定してください。また，大文字，小文字は区別されません。
- 日時書式に指定できる要素が複数ある場合，どれか 1 つを指定してください。どの要素を指定しても同じ実行結果になります。例えば，日時書式の要素としてYYYY とYYYYN のどちらを指定しても実行結果は同じになります。
- 日時書式の要素の前後の空白は無視されます。
- 日時書式の長さは，64 バイト以内にしてください。

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-34 スカラ関数 TRUNC の実行結果のデータ型とデータ長

日時データのデータ型とデータ長	実行結果のデータ型とデータ長
DATE	DATE
TIME(*p*)	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)

（凡例）*p*：小数秒精度

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 日時データがナル値の場合，実行結果はナル値になります。
4. 日時データがDATE 型の場合に，日時書式に時刻を切り捨てる指定（DDD，DD，HH，HH12，HH24，MI，SSSSS，またはSS）をしたときは，入力された日時データをそのまま返します。
5. 日時データにTIME 型のデータを指定した場合，日時書式に世紀，年，四半期，月，週，日を丸める指定（CC，YYYY，YYYYN，YY，YYN，Q，MONTH，MON，MM，WW，W，DAY，DAYN，DY，DYN，D，DDD，またはDD）はできません。
6. 日時書式にDAY，DAYN，DY，DYN，またはD を指定した場合，実行結果が 0001 年 1 月 1 日よりも前になるとエラーになります。

(4) 例題

例題

販売履歴表（SALESLIST）から，商品コード（PUR-CODE）P001 の商品の販売個数を集計します。2013 年 11 月の販売個数を週ごとに集計します。

```
SELECT SUM("PUR-NUM") FROM "SALESLIST"
    WHERE "PUR-DATE" BETWEEN DATE'2013-11-01' AND DATE'2013-11-30'
    AND "PUR-CODE"='P001'
    GROUP BY TRUNC("PUR-DATE",'DAY')
```

販売履歴表(SALESLIST)

USERID	PUR-CODE	PUR-NUM	PUR-DATE
U00212	P001	5	2013-11-01
U00358	P002	3	2013-11-04
U00555	P001	4	2013-11-07
U00212	P003	2	2013-11-12
U00687	P001	2	2013-11-13
U00555	P001	4	2013-11-13
U00687	P001	1	2013-11-15
U00555	P001	4	2013-11-21
U00212	P001	3	2013-11-24
U00555	P002	4	2013-11-26
U00358	P001	1	2013-11-26
U00555	P003	2	2013-11-29

検索結果

5	← 11月第1週(11/1～11/2)の販売個数
4	← 11月第2週(11/3～11/9)の販売個数
7	← 11月第3週(11/10～11/16)の販売個数
4	← 11月第4週(11/17～11/23)の販売個数
4	← 11月第5週(11/24～11/30)の販売個数

日時書式の要素にDAY を指定した場合，日曜日が週の開始日になり，週ごとの販売個数を集計できます。

8.10　バイナリ列関数（バイナリデータ操作）

ここでは，バイナリデータを操作するバイナリ列関数の機能と指定形式について説明します。

8.10.1　CONCAT

2 つのバイナリデータを連結します。

文字データと文字データを連結するスカラ関数については，「8.5.1　CONCAT」を参照してください。

(1)　指定形式

スカラ関数CONCAT：：＝CONCAT(*対象データ1*,*対象データ2*)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2)　指定形式の説明

対象データ 1 および*対象データ 2*：

連結するバイナリデータを指定します。

指定規則を次に示します。

- *対象データ 1* および*対象データ 2* は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *対象データ 1* および*対象データ 2* には，BINARY 型またはVARBINARY 型のデータを指定してください。
- *対象データ 1* および*対象データ 2* には，？パラメタを単独で指定できません。

スカラ関数CONCAT の実行結果の例を次に示します。

(例)

2 つのバイナリデータ（X'ABC1230000'とX'DEF456'）を連結します。

CONCAT(X'ABC1230000',X'DEF456'）→ X'ABC1230000DEF456'

(3)　規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-35　スカラ関数 CONCAT の実行結果のデータ型とデータ長

対象データ 1 のデータ型とデータ長	対象データ 2 のデータ型とデータ長	実行結果のデータ型とデータ長
BINARY(m)	BINARY(n)	BINARY($m + n$)

対象データ 1 のデータ型とデータ長	対象データ 2 のデータ型とデータ長	実行結果のデータ型とデータ長
	VARBINARY(*n*) 実データ長：*L2*	VARBINARY(*m* ＋*n*) 実データ長：*m* ＋*L2*
VARBINARY(*m*) 実データ長：*L1*	BINARY(*n*)	VARBINARY(*m* ＋*n*) 実データ長：*L1* ＋*n*
	VARBINARY(*n*) 実データ長：*L2*	VARBINARY(*m* ＋*n*) 実データ長：*L1* ＋*L2*

（凡例）

m：*対象データ 1* の最大長

n：*対象データ 2* の最大長

L1：*対象データ 1* の実データ長

L2：*対象データ 2* の実データ長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *対象データ 1* または*対象データ 2* のどちらかがナル値の場合，実行結果はナル値になります。
4. *対象データ 1* と*対象データ 2* を連結した結果，最大長が 32,000 バイトを超えるバイナリデータを連結することはできません。

8.10.2 SUBSTRB

バイナリデータの任意の位置から一部のバイナリデータを抽出します。

(1) 指定形式

```
スカラ関数SUBSTRB：：＝SUBSTRB(抽出元のバイナリデータ,開始位置 〔,抽出バイト数〕)

  抽出元のバイナリデータ：：＝値式
  開始位置：：＝値式
  抽出バイト数：：＝値式
```

(2) 指定形式の説明

抽出元のバイナリデータ：

抽出元のバイナリデータを指定します。

指定規則を次に示します。

- 抽出元のバイナリデータは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 抽出元のバイナリデータには，BINARY 型またはVARBINARY 型のデータを指定してください。

- 抽出元のバイナリデータには，？パラメタを単独で指定できません。

開始位置：

バイナリデータの抽出開始位置を，バイト数単位で指定します。

開始位置に0 以上の値を指定した場合は，抽出元のバイナリデータの先頭からの位置を意味します。例えば，開始位置に2 を指定した場合，先頭 2 バイト目から抽出を始めるという意味になります。

開始位置に負の値を指定した場合は，抽出元のバイナリデータの末尾からの位置を意味します。例えば，開始位置に-2 を指定した場合，末尾 2 バイト目から抽出を始めるという意味になります。

指定規則を次に示します。

- 開始位置は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 開始位置には，整数（INTEGER 型またはSMALLINT 型のデータ）を指定してください。
- 開始位置に0 を指定した場合，1 が指定されたと仮定されます。
- 開始位置に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

抽出バイト数：

抽出するバイナリデータの長さを指定します。

指定規則を次に示します。

- *抽出バイト数*は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *抽出バイト数*には，0 以上の整数（INTEGER 型またはSMALLINT 型のデータ）を指定してください。
- *抽出バイト数*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

スカラ関数SUBSTRB の実行結果の例を次に示します。

（例）

- バイナリデータX'ABCDEF1234567890'の先頭 2 バイト目から 3 バイト分のデータを抽出します。
  SUBSTRB(X'ABCDEF1234567890',2,3) → X'CDEF12'
- バイナリデータX'ABCDEF1234567890'の末尾 3 バイト目から 2 バイト分のデータを抽出します。
  SUBSTRB(X'ABCDEF1234567890',-3,2) → X'5678'

(3)　規則

1. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
2. 次に示す場合，実行結果はナル値になります。
   - *抽出バイト数*が負の値の場合（抽出元のバイナリデータ，開始位置の指定に関係なくナル値になります）
   - 抽出元のバイナリデータ，開始位置，または抽出バイト数のどれかがナル値の場合
3. 実行結果のデータ型とデータ長を次の表に示します。

表 8-36　スカラ関数 SUBSTRB の実行結果のデータ型とデータ長

抽出元のバイナリデータのデータ型とデータ長	実行結果のデータ型とデータ長
BINARY(*n*)	VARBINARY(*n*)
VARBINARY(*n*)	

（凡例）*n*：抽出元のバイナリデータの最大長

4. スカラ関数SUBSTRB によって抽出されるバイナリデータのバイト数を次の表に示します。

表 8-37　スカラ関数 SUBSTRB によって抽出されるバイナリデータのバイト数

スカラ関数 SUBSTRB の指定		抽出されるバイナリデータのバイト数
抽出バイト数の指定	開始位置の指定値	
指定あり	正の値	MAX｛0，MIN（抽出バイト数，*抽出元のバイナリデータのバイト数－開始位置*＋1)｝
	0	MIN（*抽出バイト数*，*抽出元のバイナリデータのバイト数*）
	負の値	MIN（*抽出バイト数*，*開始位置の絶対値*，*抽出元のバイナリデータのバイト数*）
省略	正の値	MAX（0，*抽出元のバイナリデータのバイト数－開始位置*＋1）
	0	*抽出元のバイナリデータのバイト数*
	負の値	MIN（*開始位置の絶対値*，*抽出元のバイナリデータのバイト数*）

5. 次に示す場合，実行結果は実長 0 バイトのデータになります。
   - 実行結果のバイナリデータの長さが 0 の場合
   - 抽出元のバイナリデータが実長 0 バイトの場合
   - 開始位置に次の値を指定した場合
     開始位置＞抽出元のバイナリデータのバイト数
     開始位置＜－抽出元のバイナリデータのバイト数
6. 「抽出元のバイナリデータの開始位置以降のバイナリデータのバイト数＜抽出バイト数」の場合，抽出元のバイナリデータの開始位置以降のすべてのバイナリデータを返します。
   （例）
   SUBSTRB(X'ABCDEF',2,5) → X'CDEF'

8.11　バイナリ列関数（ビット演算）

ここでは，ビット演算を行うバイナリ列関数の機能と指定形式について説明します。

8.11.1　BITAND

2 つのバイナリデータのビットごとの論理積を返します。

(1)　指定形式

スカラ関数BITAND：：＝BITAND(*対象データ1*,*対象データ2*)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2)　指定形式の説明

対象データ 1 および*対象データ 2*：

バイナリデータを指定します。

指定規則を次に示します。

- *対象データ 1* および*対象データ 2* は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *対象データ 1* および*対象データ 2* には，BINARY 型またはVARBINARY 型のデータを指定してください。
- *対象データ 1* と*対象データ 2* のデータ長（対象データがBINARY 型の場合），または実長（対象データがVARBINARY 型の場合）が同じになるようにしてください。
- *対象データ 1* には，？パラメタを単独で指定できません。
- *対象データ 2* に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型とデータ長は，*対象データ 1* のデータ型とデータ長になります。

スカラ関数BITAND の実行結果の例を次に示します。

（例）

2 つのバイナリデータのビットごとの論理積を返します。

```
BITAND(B'01011011',B'01001110') → B'01001010'
BITAND(B'01011011',X'FF') → B'01011011'
BITAND(X'0F',X'FF') → X'0F'
```

(3)　規則

1. スカラ関数BITAND の実行結果（第*n* ビット目の値）は，次の表に示すようになります。

表 8-38　スカラ関数 BITAND の実行結果（第 n ビット目の値）

対象データ 1 の第 n ビット目の値	対象データ 2 の第 n ビット目の値	スカラ関数 BITAND の実行結果（第 n ビット目の値）
0	0	0
	1	0
1	0	0
	1	1

2. 実行結果のデータ型とデータ長は，*対象データ 1* および*対象データ 2* のデータ型とデータ長によって決まります。スカラ関数BITAND の実行結果のデータ型とデータ長を次の表に示します。

表 8-39　スカラ関数 BITAND の実行結果のデータ型とデータ長

対象データ 1 のデータ型およびデータ長	対象データ 2 のデータ型およびデータ長	実行結果のデータ型とデータ長
BINARY(*m*)	BINARY(*m*)	BINARY(*m*)
	VARBINARY(*Y*) 対象データの実長：*m*	VARBINARY(*Y*) 対象データの実長：*m*
VARBINARY(*X*) 対象データの実長：*m*	BINARY(*m*)	VARBINARY(*X*) 対象データの実長：*m*
	VARBINARY(*Y*) 対象データの実長：*m*	VARBINARY(MAX(*X*,*Y*)) 対象データの実長：*m*

（凡例）

m：データ長または実長

X：データ長（*X*≧*m* の場合）

Y：データ長（*Y*≧*m* の場合）

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. *対象データ 1* または*対象データ 2* のどちらかがナル値の場合，実行結果はナル値になります。
5. *対象データ 1* および*対象データ 2* の実長が 0 バイトの場合，実行結果は実長 0 バイトのバイナリデータになります。

8.11.2　BITLSHIFT

バイナリデータを左ビットシフトした値を返します。

(1)　指定形式

スカラ関数BITLSHIFT：：＝BITLSHIFT(*対象データ*,*シフトビット数*)

対象データ：：＝*値式*
シフトビット数：：＝*値式*

(2) 指定形式の説明

対象データ：

バイナリデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，BINARY 型またはVARBINARY 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

シフトビット数：

シフトビット数を指定します。

指定規則を次に示します。

- *シフトビット数*は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *シフトビット数*には，整数（INTEGER 型またはSMALLINT 型のデータ）を指定してください。
- *シフトビット数*に正の値を指定した場合，対象データを左ビットシフトした値を返します。
- *シフトビット数*に負の値を指定した場合，対象データを右ビットシフトした値を返します。
- *シフトビット数*に 0 を指定した場合，対象データと同じバイナリデータを返します。
- *シフトビット数*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER 型になります。

スカラ関数BITLSHIFT の実行結果の例を次に示します。

(例)

指定したバイナリデータを左ビットシフトした値を返します。

```
BITLSHIFT(B'01011011',1) → B'10110110'
BITLSHIFT(B'01011011',8) → B'00000000'
BITLSHIFT(B'01011011',0) → B'01011011'
BITLSHIFT(X'0F0F',8) → X'0F00'
```

*シフトビット数*に負の値を指定した場合，対象データを右ビットシフトした値を返します。

```
BITLSHIFT(B'01011011',-3) → B'00001011'
BITLSHIFT(X'0F0F',-16) → X'0000'
```

(3) 規則

1. ビットがシフトしたことによって空いたビットにはB'0' が補われます。

2. 対象データのデータ長（対象データがVARBINARY 型の場合は実長）がm バイトであり，シフトビット数がn の場合，BITLSHIFT の実行結果は次の表に示すようになります。

表 8-40　BITLSHIFT の実行結果

条件		BITLSHIFT の実行結果
n ＞ 0 の場合	8×m ≦n の場合	対象データのデータ長（対象データがVARBINARY 型の場合は実長）のバイト数分，X'00' となるバイナリデータを返します。
	0 ＜n ＜ 8×m の場合	n ビット分，対象データを左ビットシフトしたバイナリデータを返します。
n=0 の場合		対象データと同じバイナリデータを返します。
n ＜ 0 の場合	0 ＜｜n ｜＜ 8×m	｜n ｜ビット分，対象データを右ビットシフトしたバイナリデータを返します。
	上記以外	対象データのデータ長（対象データがVARBINARY 型の場合は実長）のバイト数分，X'00' となるバイナリデータを返します。

3. 対象データのデータ型とデータ長が，実行結果のデータ型とデータ長になります。
4. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
5. 対象データまたは*シフトビット数*のどちらかがナル値の場合，実行結果はナル値になります。
6. 対象データが実長 0 バイトのバイナリデータの場合，実行結果は実長 0 バイトのバイナリデータになります。

8.11.3　BITNOT

バイナリデータのビットごとの論理否定を返します。

(1)　指定形式

スカラ関数BITNOT：：＝BITNOT(*対象データ*)

対象データ：：＝*値式*

(2)　指定形式の説明

対象データ：

バイナリデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，BINARY 型またはVARBINARY 型のデータを指定してください。

- 対象データには，？パラメタを単独で指定できません。

スカラ関数BITNOT の実行結果の例を次に示します。

（例）

バイナリデータのビットごとの論理否定を返します。

BITNOT（B'01011011'）→ B'10100100'

BITNOT（B'11010001'）→ B'00101110'

BITNOT（X'0F'）→ X'F0'

(3) 規則

1. 対象データのデータ型とデータ長が，実行結果のデータ型とデータ長になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。
4. 対象データが実長 0 バイトのバイナリデータの場合，実行結果は実長 0 バイトのバイナリデータになります。
5. スカラ関数BITNOT の実行結果（第*n* ビット目の値）は，次の表に示すようになります。

表 8-41 スカラ関数 BITNOT の実行結果（第 n ビット目の値）

対象データの第 n ビット目の値	スカラ関数 BITNOT の実行結果（第 n ビット目の値）
0	1
1	0

8.11.4 BITOR

2 つのバイナリデータのビットごとの論理和を返します。

(1) 指定形式

スカラ関数BITOR：：＝BITOR(*対象データ1*,*対象データ2*)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2) 指定形式の説明

対象データ 1 および*対象データ 2*：

バイナリデータを指定します。

指定規則を次に示します。

- *対象データ 1* および*対象データ 2* は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *対象データ 1* および*対象データ 2* には，BINARY 型またはVARBINARY 型のデータを指定してください。
- *対象データ 1* と*対象データ 2* のデータ長（対象データがBINARY 型の場合），または実長（対象データがVARBINARY 型の場合）が同じになるようにしてください。
- *対象データ 1* には，？パラメタを単独で指定できません。
- *対象データ 2* に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型とデータ長は，*対象データ 1* のデータ型とデータ長になります。

スカラ関数BITOR の実行結果の例を次に示します。

（例）

2 つのバイナリデータのビットごとの論理和を返します。

```
BITOR(B'01011011',B'01001110') → B'01011111'
BITOR(B'01011011',X'FF') → B'11111111'
BITOR(X'0F',X'FF') → X'FF'
```

(3)　規則

1. スカラ関数 BITOR の実行結果（第 n ビット目の値）は，次の表に示すようになります。

表 8-42　スカラ関数 BITOR の実行結果（第 n ビット目の値）

対象データ 1 の第 n ビット目の値	対象データ 2 の第 n ビット目の値	スカラ関数 BITOR の実行結果（第 n ビット目の値）
0	0	0
	1	1
1	0	1
	1	1

2. 実行結果のデータ型とデータ長は，*対象データ 1* および*対象データ 2* のデータ型とデータ長によって決まります。スカラ関数BITOR の実行結果のデータ型とデータ長を次の表に示します。

表 8-43　スカラ関数 BITOR の実行結果のデータ型とデータ長

対象データ 1 のデータ型およびデータ長	対象データ 2 のデータ型およびデータ長	実行結果のデータ型とデータ長
BINARY(*m*)	BINARY(*m*)	BINARY(*m*)
	VARBINARY(*Y*) 対象データの実長：*m*	VARBINARY(*Y*) 対象データの実長：*m*
VARBINARY(*X*) 対象データの実長：m	BINARY(*m*)	VARBINARY(*X*) 対象データの実長：*m*

対象データ 1 のデータ型およびデータ長	対象データ 2 のデータ型およびデータ長	実行結果のデータ型とデータ長
	VARBINARY(*Y*) 対象データの実長：*m*	VARBINARY(MAX(*X*,*Y*)) 対象データの実長：*m*

（凡例）

m：データ長または実長

X：データ長（*X*≧*m* の場合）

Y：データ長（*Y*≧*m* の場合）

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. *対象データ 1* または*対象データ 2* のどちらかがナル値の場合，実行結果はナル値になります。
5. *対象データ 1* および*対象データ 2* の実長が 0 バイトの場合，実行結果は実長 0 バイトのバイナリデータになります。

8.11.5 BITRSHIFT

バイナリデータを右ビットシフトした値を返します。

(1) 指定形式

```
スカラ関数BITRSHIFT：：＝BITRSHIFT(対象データ,シフトビット数)

  対象データ：：＝値式
  シフトビット数：：＝値式
```

(2) 指定形式の説明

対象データ：

バイナリデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，BINARY 型またはVARBINARY 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

シフトビット数：

シフトビット数を指定します。

指定規則を次に示します。

- *シフトビット数*は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *シフトビット数*には，整数（INTEGER 型またはSMALLINT 型のデータ）を指定してください。

- *シフトビット数*に正の値を指定した場合，対象データを右ビットシフトした値を返します。
- *シフトビット数*に負の値を指定した場合，対象データを左ビットシフトした値を返します。
- *シフトビット数*に 0 を指定した場合，対象データと同じバイナリデータを返します。
- *シフトビット数*に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型はINTEGER型になります。

スカラ関数BITRSHIFT の実行結果の例を次に示します。

(例)

指定したバイナリデータを右ビットシフトした値を返します。

```
BITRSHIFT (B'01011011',1) → B'00101101'
BITRSHIFT (B'01011011',8) → B'00000000'
BITRSHIFT (B'01011011',0) → B'01011011'
BITRSHIFT (X'0F0F',8) → X'000F'
```

*シフトビット数*に負の値を指定した場合，対象データを左ビットシフトした値を返します。

```
BITRSHIFT (B'01011011',-3) → B'11011000'
BITRSHIFT (X'0F0F',-16) → X'0000'
```

(3) 規則

1. ビットがシフトしたことによって空いたビットにはB'0'が補われます。
2. 対象データのデータ長（対象データがVARBINARY 型の場合は実長）がm バイトであり，シフトビット数がn の場合，BITRSHIFT の実行結果は次の表に示すようになります。

表 8-44　BITRSHIFT の実行結果

条件		BITRSHIFT の実行結果
$n > 0$ の場合	$8 \times m \leqq n$ の場合	対象データのデータ長（対象データがVARBINARY 型の場合は実長）のバイト数分，X'00'となるバイナリデータを返します。
	$0 < n < 8 \times m$ の場合	n ビット分，対象データを右ビットシフトしたバイナリデータを返します。
n=0 の場合		対象データと同じバイナリデータを返します。
$n < 0$ の場合	$0 < \lvert n \rvert < 8 \times m$	$\lvert n \rvert$ ビット分，対象データを左ビットシフトしたバイナリデータを返します。
	上記以外	対象データのデータ長（対象データがVARBINARY 型の場合は実長）のバイト数分，X'00'となるバイナリデータを返します。

3. 対象データのデータ型とデータ長が，実行結果のデータ型とデータ長になります。
4. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

5. 対象データまたは*シフトビット数*のどちらかがナル値の場合，実行結果はナル値になります。
6. 対象データが実長０バイトのバイナリデータの場合，実行結果は実長０バイトのバイナリデータになります。

8.11.6 BITXOR

２つのバイナリデータのビットごとの排他的論理和を返します。

(1) 指定形式

スカラ関数BITXOR：：＝BITXOR(*対象データ1*,*対象データ2*)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2) 指定形式の説明

対象データ 1 および*対象データ 2*：

バイナリデータを指定します。

指定規則を次に示します。

- *対象データ 1* および*対象データ 2* は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *対象データ 1* および*対象データ 2* には，BINARY 型またはVARBINARY 型のデータを指定してください。
- *対象データ 1* と*対象データ 2* のデータ長（対象データがBINARY 型の場合），または実長（対象データがVARBINARY 型の場合）が同じになるようにしてください。
- *対象データ 1* には，？パラメタを単独で指定できません。
- *対象データ 2* に？パラメタを単独で指定した場合，？パラメタに仮定されるデータ型とデータ長は，*対象データ 1* のデータ型とデータ長になります。

スカラ関数BITXOR の実行結果の例を次に示します。

（例）

２つのバイナリデータのビットごとの排他的論理和を返します。

```
BITXOR(B'01011011',B'01001110') → B'00010101'
BITXOR(B'01011011',X'FF') → B'10100100'
BITXOR(X'0F',X'FF') → X'F0'
```

(3) 規則

1. スカラ関数BITXOR の実行結果（第*n* ビット目の値）は，次の表に示すようになります。

表 8-45　スカラ関数 BITXOR の実行結果（第 n ビット目の値）

対象データ 1 の第 n ビット目の値	対象データ 2 の第 n ビット目の値	スカラ関数 BITXOR の実行結果（第 n ビット目の値）
0	0	0
	1	1
1	0	1
	1	0

2. 実行結果のデータ型とデータ長は，*対象データ 1* および*対象データ 2* のデータ型とデータ長によって決まります。スカラ関数BITXOR の実行結果のデータ型とデータ長を次の表に示します。

表 8-46　スカラ関数 BITXOR の実行結果のデータ型とデータ長

対象データ 1 のデータ型およびデータ長	対象データ 2 のデータ型およびデータ長	実行結果のデータ型とデータ長
BINARY(*m*)	BINARY(*m*)	BINARY(*m*)
	VARBINARY(*Y*) 対象データの実長：*m*	VARBINARY(*Y*) 対象データの実長：*m*
VARBINARY(*X*) 対象データの実長：*m*	BINARY(*m*)	VARBINARY(*X*) 対象データの実長：*m*
	VARBINARY(*Y*) 対象データの実長：*m*	VARBINARY(MAX(*X*,*Y*)) 対象データの実長：*m*

（凡例）

m：データ長または実長

X：データ長（*X*≧*m* の場合）

Y：データ長（*Y*≧*m* の場合）

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

4. *対象データ 1* または*対象データ 2* のどちらかがナル値の場合，実行結果はナル値になります。

5. *対象データ 1* および*対象データ 2* の実長が 0 バイトの場合，実行結果は実長 0 バイトのバイナリデータになります。

8.12　配列関数

ここでは，配列関数の機能と指定形式について説明します。

8.12.1　ARRAY_MAX_CARDINALITY

対象データに指定した配列データの最大要素数を返します。

(1)　指定形式

```
スカラ関数ARRAY_MAX_CARDINALITY：：＝ARRAY_MAX_CARDINALITY(対象データ)
  対象データ：：＝値式
```

(2)　指定形式の説明

対象データ：

最大要素数を求める配列データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，配列データを指定してください。配列データについては，「6.2.1　データ型の種類」の「(5)　配列データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型はINTEGER 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。ただし，常に最大要素数の値が返却されるため，実行結果の値がナル値になることはありません。

(4)　例題

例題

表T1 の配列型の各列（C1 列，C2 列，C3 列）の最大要素数を求めます。

```
SELECT ARRAY_MAX_CARDINALITY("C1"),
       ARRAY_MAX_CARDINALITY("C2"),
       ARRAY_MAX_CARDINALITY("C3")
    FROM "T1"
    LIMIT 1
```

表T1

C1列 ARRAY[10]	C2列 ARRAY[20]	C3列 ARRAY[30]
{1, 2, 3}	{2, 3, 4}	{3, 4, 5}
{1, 2}	{2, 3}	{3, 4}
{1, 2, NULL}	{2, 3, NULL}	{3, 4, NULL}
{}	{}	{}
NULL	NULL	NULL

検索結果

10	20	30

8.12.2 CARDINALITY

対象データに指定した配列データの配列要素数を返します。

(1) 指定形式

スカラ関数CARDINALITY：：＝CARDINALITY(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

配列要素数を求める配列データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，配列データを指定してください。配列データについては，「6.2.1 データ型の種類」の「(5) 配列データ」を参照してください。
- 対象データには，？パラメタを単独で指定できません。

(3) 規則

1. 実行結果のデータ型はINTEGER 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 の配列型の列（C2 列）の各行の配列要素数を求めます。

```
SELECT ”C1”,CARDINALITY(”C2”) FROM ”T1”
```

表T1

C1列 CHAR	C2列 ARRAY[10]
A10101	{1,2,3}
A10201	{1,2}
A10202	{1,2,NULL}
A10301	{}
A10401	NULL

検索結果

A10101	3
A10201	2
A10202	3
A10301	0
A10401	NULL

8.13　データ変換関数

ここでは，データ変換関数の機能と指定形式について説明します。

8.13.1　ASCII

対象データの先頭の文字の文字コードを整数値で返します。

(1)　指定形式

スカラ関数ASCII：：＝ASCII(*対象データ*)

対象データ：：＝*値式*

(2)　指定形式の説明

対象データ：

対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，CHAR 型またはVARCHAR 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

スカラ関数ASCII の実行結果の例を次に示します。使用している文字コードは Unicode（UTF-8）とします。

(例)

```
ASCII('A') → 65
ASCII('ABCD') → 65
ASCII('Ⅱ') → 14845345
```

(3)　規則

1. 実行結果のデータ型はINTEGER 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 次のどちらかの場合，実行結果はナル値になります。
   - 対象データがナル値の場合
   - 対象データが実長 0 バイトまたは実長 0 文字の場合

(4) 例題

例題

表T1 のC1 列の文字データのうち，先頭の文字が ASCII コードの範囲にある文字データを検索します。

```
SELECT ”C1” FROM ”T1” WHERE ASCII(”C1”)<128
```

表T1

C1列 VARCHAR
A10101
Ⅱ3345
A3139922
Ⅲ84775

検索結果

A10101
A3139922

8.13.2 BIN

バイナリデータを 2 進文字列表現（'0'，'1'で構成された文字データ）に変換します。

(1) 指定形式

スカラ関数BIN：：＝BIN(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

バイナリデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，BINARY 型またはVARBINARY 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。
- 定義長が 4,001 バイト以上のバイナリデータは，対象データに指定できません。

スカラ関数BIN の実行結果の例を次に示します。

（例）

```
BIN(B’10100100’) → ’10100100’
```

BIN(X'A4') → '10100100'

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-47 スカラ関数 BIN の実行結果のデータ型とデータ長

対象データのデータ型とデータ長			実行結果のデータ型とデータ長		
データ型	定義長	実長	データ型	定義長	実長
BINARY(*n*)	1≦*n*≦4,000	該当しません。	VARCHAR	*n*×8	*n*×8
VARBINARY(*n*)	1≦*n*≦4,000	*r*			*r*×8

（凡例）

n：対象データの定義長

r：対象データの実長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。
4. 対象データの実長が 0 バイトの場合，実行結果は実長 0 バイトのデータになります。

8.13.3 CAST

データのデータ型を変換します。

(1) 指定形式

```
スカラ関数CAST：：＝CAST(変換対象データ AS 変換後のデータ型)

  変換対象データ：：＝ {値式 | NULL}
  変換後のデータ型：：＝データ型
```

(2) 指定形式の説明

変換対象データ：

データ型を変換するデータを指定します。

変換対象データは，値式の形式で指定するか，またはNULL を指定します。値式については，「7.21 値式」を参照してください。

なお，変換対象データには，配列データを指定できません。

変換後のデータ型：

変換後のデータ型を指定します。指定例を次に示します。

- INTEGER
  INTEGER 型のデータに変換されます。
- DECIMAL(5,2)
  精度が 5，位取りが 2 のDECIMAL 型のデータに変換されます。
- CHAR(8)
  データ長 8 バイトのCHAR 型のデータに変換されます。

各データ型の指定形式については，「6.2.1　データ型の種類」を参照してください。
なお，変換後のデータ型には，次のデータ型を指定できません。

- データ長が 32,000 バイトを超えるVARCHAR 型
- 配列型

スカラ関数CAST の実行結果の例を次に示します。

(例)

DECIMAL 型のデータ（-12.37）を，INTEGER 型に変換します。

CAST(-12.37 AS INTEGER) → -12

(3)　規則

(a)　共通の規則

1. 実行結果のデータ型は，*変換後のデータ型*に指定したデータ型になります。ただし，*変換後のデータ型*にNUMERIC 型が指定された場合，実行結果のデータ型はDECIMAL 型になります。また，*変換後のデータ型*にFLOAT 型が指定された場合，実行結果のデータ型はDOUBLE PRECISION 型になります。
2. *変換対象データ*に？パラメタを単独で指定した場合，*変換後のデータ型*が？パラメタのデータ型として仮定されます。ただし，*変換後のデータ型*にNUMERIC 型が指定された場合は，？パラメタのデータ型にはDECIMAL 型が仮定されます。また，*変換後のデータ型*にFLOAT 型が指定された場合は，？パラメタのデータ型にはDOUBLE PRECISION 型が仮定されます。
3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 変換対象データがナル値の場合，または*変換対象データ*にNULL を指定した場合，実行結果はナル値になります。
5. 変換対象データが，実長 0 バイトまたは実長 0 文字の文字データの場合，次のように変換されます。
   - CHAR 型に変換する場合：半角空白に変換されます。CHAR(3)の場合，'△△△'に変換されます。△は半角空白を意味しています。
   - VARCHAR 型に変換する場合：実長 0 バイトまたは実長 0 文字のVARCHAR 型のデータに変換されます。
   - BINARY 型に変換する場合：X'00'に変換されます。BINARY(3)の場合，X'000000'に変換されます。
   - VARBINARY 型に変換する場合：実長 0 バイトのVARBINARY 型のデータに変換されます。
   - 上記以外のデータ型の場合：ナル値に変換されます。

6. データ型の変換可否を次の表に示します。

表 8-48　データ型の変換可否

変換対象データのデータ型	変換後のデータ型					
	INTEGER, SMALLINT	DECIMAL, NUMERIC, DOUBLE PRECISION, FLOAT	CHAR, VARCHAR	DATE, TIMESTAMP	TIME	BINARY, VARBINARY
INTEGER, SMALLINT	○	○	○	○	×	×
DECIMAL, NUMERIC, DOUBLE PRECISION, FLOAT	○	○	○	×	×	×
CHAR, VARCHAR	○	○	○	○	○	○
DATE, TIMESTAMP	○	×	○	○	×	×
TIME	×	×	○	×	○	×
BINARY, VARBINARY	×	×	○	×	×	○

（凡例）

○：変換できます。

×：変換できません。

(b)　数データに変換する場合の規則

■数データを数データに変換する場合

数データを数データに変換する場合，「6.2.2　変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」の「数データの格納代入」で説明している規則が適用されます。

■文字データを数データに変換する場合

- 変換対象の文字データ（文字データの前後の半角空白を取り除いた結果）が，数定数の記述形式の規則を満たしている必要があります。数定数の記述形式の規則については，「6.3.2　定数の記述形式」を参照してください。

  ＜変換できる文字データの例＞

  ’219’，’+56’，’-3547’，’-11.35’，’887△△’，’△95△’

  ＜変換できない文字データの例＞

'a89', '77g9', '33△49'

（凡例）△：半角空白

- 文字データが半角空白だけで構成されている場合，ナル値を返します。
- 数定数の文字列表現を数値に変換したあとに，変換後のデータ型に変換します。その際，「6.2.2 変換，代入，比較できるデータ型」の「(2) 格納代入できるデータ型」の「数データの格納代入」で説明している規則が適用されます。

  （例）

  CAST('11.35' AS INTEGER) → 11

  いったん文字列'11.35'がDECIMAL 型の数値 11.35 に変換され，そのあとにINTEGER 型の数値に変換されます。その際，数データの格納代入の規則が適用され，この例の場合，小数点以下が切り捨てられます。

■日時データを数データに変換する場合

西暦 1 年 1 月 1 日からの通算日に変換されます。西暦 1 年 1 月 1 日の場合，通算日は 1 になります。西暦 1 年 1 月 2 日の場合，通算日は 2 になります。

（例）

CAST(DATE'0001-01-03' AS INTEGER) → 3

CAST(TIMESTAMP'0001-01-05 11:03:58' AS INTEGER) → 5

(c) 文字データに変換する場合の規則

文字データへの変換規則（データ長に関する規則）を次の表に示します。

表 8-49 文字データへの変換規則（データ長に関する規則）

変換時の条件	文字データへの変換規則	
	変換対象データのデータ型が文字データまたはバイナリデータの場合	変換対象データのデータ型が左記以外の場合
A ＜ *B* の場合	変換後のデータ型がCHAR 型の場合は，データを左詰めにして，余りに半角空白が格納されます。	
A ＝ *B* の場合	変換されます。	
A ＞ *B* の場合	データを左詰めにして，余った部分を切り捨てます。※1	変換できません。エラーになります。※2

（凡例）

A：変換対象データを文字データに変換した長さ

B：変換後のデータ型のデータ長

注※1

マルチバイト文字の途中で切り捨てが発生した場合，マルチバイト文字の一部分が実行結果の値として返されます。

注※2

変換対象データのデータ型がDOUBLE PRECISION 型またはFLOAT 型の場合は，*変換後のデータ型*に指定したデータ長に収まるように仮数の小数点以下を切り捨てるため（最近接偶数への丸めを行うため），エラーにはなりません。ただし，仮数の小数点以下すべてを切り捨てても，*変換後のデータ型*に指定したデータ長を超える場合は，エラーになります。

■INTEGER 型，SMALLINT 型，DECIMAL 型，またはNUMERIC 型の数データを文字データに変換する場合

- 数データを数定数の形式に変換した結果を文字データとして出力します。その際，数定数で表現可能な形式のうち，最も短い形式で結果が出力されます。

  ただし，DECIMAL 型またはNUMERIC 型のデータの場合は，次のように変換されます。

  ・小数点以下の桁数は，数データのデータ型の位取りと同じになり，末尾からの 0 は削除されません。

  ・「数データのデータ型の精度＞位取り」の場合，整数部の桁数は 0 にはなりません。

  ・小数点は必ず付加されます。

  （例）+0025.100 → '25.100'

  上記のように，＋の符号は削除されます。また，整数部分の先頭からの 0 は削除されます。

- 変換対象データが0 未満の場合，先頭に負符号（-）が付加されます。

■DOUBLE PRECISION 型またはFLOAT 型の数データを文字データに変換する場合

- 数データを浮動小数点数定数の形式に変換した結果を文字データとして出力します。その際，浮動小数点数定数で表現可能な形式のうち，最も短い形式で結果が出力されます。

  （例）

  +1.0000000000000000E+010 → '1E10'

  +3.2000000000000000E+001 → '3.2E1'

  +0.1000000000000000E+001 → '1E0'

  +0.0000000000000000E+000 → '0E0'

  上記のように，仮数の＋の符号は削除され，小数点以下の末尾からの0 も削除されます。指数の＋の符号は削除され，指数の先頭からの0 も削除されます。

- 変換対象データが0 未満の場合は，先頭に負符号（-）が付加されます。
- 0 未満の指数には，先頭に負符号（-）が付加されます。

■日時データを文字データに変換する場合

- 日時データを文字データに変換する場合，既定の出力表現の形式に変換されます。DATE 型のデータを文字データに変換する場合は，日付を表す既定の出力表現の形式に変換されます。TIME 型のデータを文字データに変換する場合は，時刻を表す既定の出力表現の形式に変換されます。TIMESTAMP 型のデータを文字データに変換する場合は，時刻印を表す既定の出力表現の形式に変換されます。既定の出力表現については，「6.3.3　既定の文字列表現」を参照してください。

  （例）

  CAST(DATE'2013-06-30' AS CHAR(10)) → '2013-06-30'

  CAST(DATE'0001-01-01' AS CHAR(10)) → '0001-01-01'

  CAST(TIME'05:33:48.123' AS CHAR(12)) → '05:33:48.123'

CAST(TIMESTAMP'2013-06-30 11:03:58' AS CHAR(19)) → '2013-06-30 11:03:58'

- 日時データをCHAR(*n*)またはVARCHAR(*n*)に変換する場合，次の条件を満たす必要があります。

変換対象データのデータ型		変換後のデータ長の条件
DATE		*n*≧10
TIME(*p*)	*p* ＝0 の場合	*n*≧8
	p ＞0 の場合	*n*≧9 ＋*p*
TIMESTAMP(*p*)	*p* ＝0 の場合	*n*≧19
	p ＞0 の場合	*n*≧20 ＋*p*

n が上記の長さより短い場合は変換できません。

- DATE 型のデータをCHAR 型に変換する場合，変換後のデータのデータ長が 11 バイト以上のときは，データを左詰めにして，余りに半角空白が格納されます。

  （例）

  CAST(DATE'2013-06-30' AS CHAR(15)) → '2013-06-30△△△△△'

  （凡例）△：半角空白

- 小数秒精度が*p* のTIME 型のデータをCHAR 型に変換する場合，変換後のデータのデータ長が 10 ＋*p* バイト以上（*p*=0 のときは 9 バイト以上）のときは，データを左詰めにして，余りに半角空白が格納されます。

  （例）

  CAST(TIME'11:03:58.123' AS CHAR(13)) → '11:03:58.123△'

  （凡例）△：半角空白

- 小数秒精度が*p* のTIMESTAMP 型のデータをCHAR 型に変換する場合，変換後のデータのデータ長が 21 ＋*p* バイト以上（*p*=0 のときは 20 バイト以上）のときは，データを左詰めにして，余りに半角空白が格納されます。

  （例）

  CAST(TIMESTAMP'2013-06-30 11:03:58' AS CHAR(20)) → '2013-06-30 11:03:58△'

  （凡例）△：半角空白

■バイナリデータを文字データに変換する場合

- データ型が変換されるだけで，データの内容（文字コード自体）は変換されません。

  （例）

  CAST(X'61626364' AS CHAR(4)) → 'abcd'

- 「*変換対象データ*のデータ長＞*変換後のデータ型*のデータ長」の場合，末尾が切り捨てられます。

  （例）

  CAST(X'6162636<u>4</u>' AS CHAR(3)) → 'abc'

  下線部分が切り捨てられます。

- 「*変換対象データのデータ長*＜*変換後のデータ型*のデータ長」の場合，末尾に半角空白が格納されます。

  （例）

  CAST(X'61626364' AS CHAR(5)) → 'abcd△'

  （凡例）△：半角空白

(d)　日時データに変換する場合の規則

■INTEGER 型またはSMALLINT 型の数データを日時データに変換する場合

- 西暦 1 年 1 月 1 日を起点として変換されます。
- TIMESTAMP 型の時刻部分は'00:00:00'に変換され，小数秒部分には 0 が補われます。例を次に示します。

  （例）

  CAST(2 AS DATE) → DATE'0001-01-02'

  CAST(2 AS TIMESTAMP(3)) → TIMESTAMP'0001-01-02 00:00:00.000'
- INTEGER 型またはSMALLINT 型のデータが 1～3,652,059 の場合に変換できます。範囲外の場合はエラーになります。

■文字データを日時データに変換する場合

- 変換対象の文字データ（文字データの前後の半角空白を取り除いた結果）が，日付を表す既定の入力表現の形式に従っている場合に限り，文字データをDATE 型のデータに変換できます。日付を表す既定の入力表現については，「6.3.3　既定の文字列表現」の「(1)　日付を表す既定の文字列表現」の「(a)　既定の入力表現」を参照してください。

  （例）

  CAST('2014-07-22△△' AS DATE) → DATE'2014-07-22'

  ＜変換できる文字データの例＞

  '2014-06-30', '0001-01-02', '△△2014-07-30', '△2014/07/30△△'

  ＜変換できない文字データの例＞

  '2013△06△30', '2013.06.30'

  （凡例）△：半角空白
- 変換対象の文字データ（文字データの前後の半角空白を取り除いた結果）が，時刻を表す既定の入力表現の形式に従っている場合に限り，文字データをTIME 型のデータに変換できます。時刻を表す既定の入力表現については，「6.3.3　既定の文字列表現」の「(2)　時刻を表す既定の文字列表現」の「(a)　既定の入力表現」を参照してください。

  （例）

  CAST('△19:46:23.123456' AS TIME(6)) → TIME'19:46:23.123456'

  ＜変換できる文字データの例＞

  '18:05:22', '10:21:44.123', '△△10:21:44.123456△'

  ＜変換できない文字データの例＞

'18△05△22', '10:21:44△123456'

(凡例) △：半角空白

- 変換対象の文字データ（文字データの前後の半角空白を取り除いた結果）が，時刻印を表す既定の入力表現の形式に従っている場合に限り，文字データをTIMESTAMP 型のデータに変換できます。時刻印を表す既定の入力表現については，「6.3.3　既定の文字列表現」の「(3)　時刻印を表す既定の文字列表現」の「(a)　既定の入力表現」を参照してください。

  (例)

  CAST('2014/08/02 11:03:58.123456△' AS TIMESTAMP(6)) → TIMESTAMP'2014-08-02 11:03:58.123456'

  ＜変換できる文字データの例＞

  '2014-06-30 11:03:58', '2014/07/30 11:03:58.123', '△2014/07/30 11:03:58.123456789△△'

  ＜変換できない文字データの例＞

  '2014-06-30 11-03-58', '2014/07/30 11:03:58:123456'

  (凡例) △：半角空白

- 「変換対象の文字データの小数秒の桁数＞*変換後のデータ型*の小数秒の桁数」の場合，*変換後のデータ型*の小数秒の桁数を超えた部分の小数秒は切り捨てられます。

  (例)

  CAST('19:46:23.123456' AS TIME(3)) → TIME'19:46:23.123'

- 「変換対象の文字データの小数秒の桁数＜*変換後のデータ型*の小数秒の桁数」の場合，足りない小数秒部分に 0 が補われます。

  (例)

  CAST('2014-08-02 11:03:58.123' AS TIMESTAMP(9)) → TIMESTAMP'2014-08-02 11:03:58.123000000'

- 文字データが半角空白だけで構成されている場合，ナル値を返します。

■日時データを日時データに変換する場合

日時データを日時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 8-50　日時データを日時データに変換する場合の変換規則

変換対象データのデータ型	変換後のデータ型の指定	変換規則
DATE	DATE	変換されません。
	TIMESTAMP(*p2*)	• 時刻部分は'00:00:00'に変換されます。 • 小数秒部分には 0 が補われます。
TIME(*p1*)	TIME(*p2*)	• *p1* ＝*P2* の場合 変換されません。 • *p1* ＞*p2* の場合 *p2* を超えた部分の小数秒は切り捨てられます。 • *p1* ＜*p2* の場合 足りない小数秒部分には 0 が補われます。

変換対象データのデータ型	変換後のデータ型の指定	変換規則
TIMESTAMP(*p1*)	DATE	日付部分だけが変換されます。
	TIMESTAMP(*p2*)	• *p1* ＝*P2* の場合 変換されません。 • *p1* ＞*p2* の場合 *p2* を超えた部分の小数秒は切り捨てられます。 • *p1* ＜*p2* の場合 足りない小数秒部分には 0 が補われます。

（凡例）

p1，*p2*：小数秒精度

(e) バイナリデータに変換する場合の規則

■文字データをバイナリデータに変換する場合

- データ型が変換されるだけで，データの内容（文字コード自体）は変換されません。

  （例）

  CAST('abcd' AS BINARY(4)) →X'61626364'

- 「*変換対象データのデータ長*＞*変換後のデータ型のデータ長*」の場合，末尾が切り捨てられます。

  （例）

  CAST('abcd' AS BINARY(3)) → X'616263'

  下線部分が切り捨てられます。

  マルチバイト文字の途中で切り捨てが発生した場合，マルチバイト文字の一部分が実行結果の値として返されます。

- 「*変換対象データのデータ長*＜*変換後のデータ型のデータ長*」の場合，末尾にX'00'が格納されます。

  （例）

  CAST('abcd' AS BINARY(5)) → X'6162636400'

■バイナリデータをバイナリデータに変換する場合

- 「*変換対象データのデータ長*＞*変換後のデータ型のデータ長*」の場合，末尾が切り捨てられます。

  （例）

  CAST(X'6162636400' AS BINARY(3)) → X'616263'

  下線部分が切り捨てられます。

  マルチバイト文字の途中で切り捨てが発生した場合，マルチバイト文字の一部分が実行結果の値として返されます。

- 「*変換対象データのデータ長*＜*変換後のデータ型のデータ長*」の場合，末尾にX'00'が格納されます。

  （例）

  CAST(X'61626364' AS BINARY(5)) → X'6162636400'

(4) 例題

例題

表T1 のC2 列のデータをTIMESTAMP 型からDATE 型に変換し，C2 列が 2013 年 7 月 21 日の行を検索します。

```
SELECT * FROM "T1"
    WHERE CAST("C2" AS DATE)=DATE'2013-07-21'
```

表T1

C1列 CHAR	C2列 TIMESTAMP
A10101	2013-06-30 14:55:03
A15014	2013-07-21 16:05:17
A31399	2013-07-21 03:24:33

検索結果

A15014	2013-07-21 16:05:17
A31399	2013-07-21 03:24:33

8.13.4 CHR

対象データの整数値が示す文字コードに対応する文字を返します。

(1) 指定形式

スカラ関数CHR：：＝CHR(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

対象データを指定します。

*対象データ*に指定する値は，1 つの文字に対応する文字コードを 0 以上の整数で指定します。例えば，16 進数で0xE38182 というマルチバイト文字の場合，0xE38182 に等しい 14909826 を指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，INTEGER 型またはSMALLINT 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。

スカラ関数CHR の実行結果の例を次に示します。使用している文字コードは Unicode（UTF-8）とします。

(例)

CHR(65) → 'A'

CHR(97) → 'a'

CHR(14845345) → 'Ⅱ'

(3) 規則

1. 実行結果のデータ型はVARCHAR(8)になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値，または負の値の場合，実行結果はナル値になります。
4. 対象データの値が 255 より大きい場合は，マルチバイト文字として扱います。例えば，14909826 は，16 進数で表すと0xE38182 のため，先頭 1 バイトが0xE3，2 バイト目が0x81，3 バイト目が0x82 で構成されるマルチバイト文字として扱います。

(4) 例題

例題

表T1 のC1 列の末尾に，NL（改行）を含む文字データを検索します。

```
SELECT "C1" FROM "T1" WHERE SUBSTR("C1",-1)=CHR(10)
```

表T1

C1列 VARCHAR
A10101
A99455↲
A3139922

検索結果

A99455↲

（凡例）↲：NL（改行）

8.13.5 CONVERT

データのデータ型を変換します。

また，日時書式または数値書式を指定することで，次のことができます。

- 日時書式を指定した場合
  - 日時データを文字データに変換する際，変換後の文字データの出力形式を指定できます。
  - 文字データを日時データに変換する際，変換前の文字データの入力形式を指定できます。
- 数値書式を指定した場合
  - 数データを文字データに変換する際，変換後の文字データの出力形式を指定できます。
  - 文字データを数データに変換する際，変換前の文字データの入力形式を指定できます。

スカラ関数CONVERT の実行結果の例を次に示します。

（例 1）

- DECIMAL 型のデータ（-12.37）を，INTEGER 型に変換します。

  CONVERT(-12.37,INTEGER) → -12

（例 2）日時書式を指定した例

- TIMESTAMP 型のデータ（TIMESTAMP'2013-07-30 11:03:58'）を，CHAR(10)のデータに変換します。

  CONVERT(TIMESTAMP'2013-07-30 11:03:58',CHAR(10),'YYYY/MM/DD'） → '2013/07/30'

- 日時を意味しているCHAR 型のデータ（'07/15/2013 12:34:56'）を，TIMESTAMP 型に変換します。

  CONVERT('07/15/2013 12:34:56',TIMESTAMP,'MM/DD/YYYY HH:MI:SS'） → TIMESTAMP'2013-07-15 12:34:56'

（例 3）数値書式を指定した例

- INTEGER 型のデータをCHAR(7)のデータに変換します。変換の際，先頭に通貨記号$を付けます。また，3 桁区切りのコンマも付けます。

  CONVERT(1000,CHAR(7),'$9,999'） → '△$1,000'

  CONVERT(-1000,CHAR(7),'$9,999'） → '-$1,000'

  △は半角空白を意味しています。

- 通貨記号$および 3 桁区切りのコンマが付いているCHAR 型のデータをINTEGER 型に変換します。

  CONVERT('$1,000,000',INTEGER,'$9,999,999'） → 1000000

  CONVERT('-$1,000',INTEGER,'$9,999,999'） → -1000

(1) 指定形式

スカラ関数CONVERT：：＝CONVERT(*変換対象データ*,*変換後のデータ型*〔,*書式指定*〕）

変換対象データ：：＝｛*値式*｜NULL｝
変換後のデータ型：：＝*データ型*
書式指定：：＝｛*日時書式*｜*数値書式*｝
日時書式：：＝*定数*
数値書式：：＝*定数*

(2) 指定形式の説明

変換対象データ：

データ型を変換するデータを指定します。

変換対象データは，値式の形式で指定するか，またはNULL を指定します。値式については，「7.21 値式」を参照してください。

なお，変換対象データには，配列データを指定できません。

変換後のデータ型：

変換後のデータ型を指定します。指定例を次に示します。

- INTEGER

  INTEGER 型のデータに変換されます。

- DECIMAL(5,2)

  精度が 5，位取りが 2 のDECIMAL 型のデータに変換されます。

- CHAR(8)

  データ長 8 バイトのCHAR 型のデータに変換されます。

- TIMESTAMP(3)

  小数秒精度が 3 のTIMESTAMP 型のデータに変換されます。

各データ型の指定形式については，「6.2.1　データ型の種類」を参照してください。

なお，変換後のデータ型には，次のデータ型を指定できません。

- データ長が 32,000 バイトを超えるVARCHAR 型
- 配列型

書式指定：

日時書式または数値書式を指定します。

日時書式：

次のどちらかの場合に日時書式を指定します。

- 日時データを文字データに変換する際，変換後の文字データの出力形式を指定します。
- 文字データを日時データに変換する際，変換前の文字データの入力形式を指定します。

日時書式には，文字列定数を指定します。文字列定数については，「6.3　定数」を参照してください。

日時書式の指定例を次に示します。

(例)

```
'YYYY-MM-DD HH:MI:SS'
'YYYY/MM/DD HH MI SS FF3'
'YYYY.MM.DD-HH:MI:SS.FF6'
'YYYY:MM'
'MM/DD-HH'
```

上記の指定例のYYYY，MM，およびDD などを**日時書式の要素**といいます。日時書式に指定できる要素については，「(3)　日時書式に指定できる要素および規則」を参照してください。

日時書式を指定したときのスカラ関数CONVERT の実行結果の例を次に示します。

- 日時データを文字データに変換する場合の例

CONVERT の指定例	実行結果
CONVERT(DATE'2013-01-01',VARCHAR(20),'YYYY/MM/DD')	'2013/01/01'
CONVERT(DATE'2013-01-01',VARCHAR(20),'CC"世紀"')	'21世紀'
CONVERT(DATE'2013-01-01',VARCHAR(20),'EYYN/Q"Q"')	'H25/1Q'

CONVERT の指定例	実行結果
CONVERT(DATE'2013-01-01',VARCHAR(20),'YY-WW')	'13-01'
CONVERT(DATE'2013-01-01',VARCHAR(20),'DD-Mon-YY')	'01-Jan-13'
CONVERT(DATE'2013-01-01',VARCHAR(20),'YYYY/MM/DD"("DY")"')	'2013/01/01(TUE)'
CONVERT(TIME'09:15:20.12',VARCHAR(20),'FMHH:MI:SS.FF6')	'9:15:20.120000'

- 文字データを日時データに変換する場合の例

CONVERT の指定例	実行結果
CONVERT('01/02/2012 12:34:56',TIMESTAMP,'mm/dd/yyyy hh:mi:ss')	TIMESTAMP'2012-01-02 12:34:56'
CONVERT('平成25年1月1日午前10時23分5秒',TIMESTAMP, 'fmeeyyn"年"mm"月"dd"日"pmnhh12"時"mi"分"ss"秒"')	TIMESTAMP'2013-01-01 10:23:05'
CONVERT('1 2 3 45',TIME(6),'FMHH MI SS FF2')	TIME'01:02:03.450000'

数値書式：

次のどちらかの場合に数値書式を指定します。

- 数データを文字データに変換する際，変換後の文字データの出力形式を指定します。
- 文字データを数データに変換する際，変換前の文字データの入力形式を指定します。

数値書式には，文字列定数を指定します。文字列定数については，「6.3　定数」を参照してください。

数値書式の指定例を次に示します。

(例)

'$9,999,999'

'00,000.00'

上記の指定例の$，0，9，3 桁区切りのコンマ（,），ピリオド（.）などを数値書式の**要素**といいます。数値書式に指定できる要素については，「(4)　数値書式に指定できる要素および規則」を参照してください。

数値書式を指定したときのスカラ関数CONVERT の実行結果の例を次に示します。

- 数データを文字データに変換する場合の例

CONVERT の指定例	実行結果
CONVERT(1234567,CHAR(10),'9,999,999')	'△1,234,567'
CONVERT(1234,CHAR(10),'0,000,000')	'△0,001,234'
CONVERT(-1000,CHAR(7),'$9,999')	'-$1,000'
CONVERT(1000,VARCHAR(12),'LJ9,999"dollars"')	'1,000dollars'

△は半角空白を意味しています。

- 文字データを数データに変換する場合の例

CONVERT の指定例	実行結果
CONVERT('1,234,567',INTEGER,'9,999,999')	1234567
CONVERT('12',INTEGER,'9,999,999')	12
CONVERT('$1,000,000',INTEGER,'$9,999,999')	1000000
CONVERT('1,000dollars',INTEGER,'9,999"dollars"')	1000
CONVERT('+1.23E+10*浮動小数点文字列*',DOUBLE PRECISION,'9.99EEEE"*浮動小数点文字列*"')	1.23000000000000000E10

(3) 日時書式に指定できる要素および規則

(a) 日時書式の要素

日時書式に指定できる要素を次の表に示します。

表 8-51 日時書式に指定できる要素

項番	日時書式の持つ意味	日時書式に指定できる要素	説明	
1	世紀	CC	世紀を意味しています。値の範囲は00～99 です。 なお，00 は，100 世紀（西暦 9901～9999 年）を意味しています。	
2	西暦年	YYYY	4 桁の西暦年を意味しています。0001～9999 を使用できます。 なお，文字データを日時データに変換する際にYYYY を指定するときは，時代名を指定することはできません。	
3		YY	2 桁の西暦年（西暦年の下 2 桁）を意味しています。値の範囲は00～99 です。	
4	時代名	E	日本の時代名の省略形を意味しています。 • 'M'：明治を意味しています。 • 'T'：大正を意味しています。 • 'S'：昭和を意味しています。 • 'H'：平成を意味しています。 • 'R'：令和を意味しています。	文字データを日時データに変換する際に時代名を指定するときは，和暦年も一緒に指定する必要があります。
5		EE	日本の時代名を意味しています。 • '明治' • '大正' • '昭和' • '平成' • '令和'	
6	和暦年	YYYYN	4 桁の和暦年を意味しています。各時代の対応範囲を次に示します。 • 明治（0006～0045）	文字データを日時データに変換する際に和暦年を指定するとき

項番	日時書式の持つ意味	日時書式に指定できる要素	説明	
			• 大正（0001～0015） • 昭和（0001～0064） • 平成（0001～8011） • 令和（0001～7981）	は，時代名も一緒に指定する必要があります。
7		YYN	2桁の和暦年を意味しています。00～99を使用できます。 なお，日時データを文字データに変換する場合は，3桁以上の年数については，下2桁だけを変換対象にします。	
8	四半期	Q	四半期を意味しています。値の範囲は1～4です。 • 1：第1四半期（1月1日～3月31日）を意味しています。 • 2：第2四半期（4月1日～6月30日）を意味しています。 • 3：第3四半期（7月1日～9月30日）を意味しています。 • 4：第4四半期（10月1日～12月31日）を意味しています。	
9	月	MM	月を意味しています。01～12を使用できます。	
10		MON	月の英語名の省略形を意味しています。 • 'JAN'：1月 • 'FEB'：2月 • 'MAR'：3月 • 'APR'：4月 • 'MAY'：5月 • 'JUN'：6月 • 'JUL'：7月 • 'AUG'：8月 • 'SEP'：9月 • 'OCT'：10月 • 'NOV'：11月 • 'DEC'：12月	
11		MONTH	月の英語名を意味しています。 • 'JANUARY△△' • 'FEBRUARY△' • 'MARCH△△△△' • 'APRIL△△△△' • 'MAY△△△△△△' • 'JUNE△△△△△' • 'JULY△△△△△' • 'AUGUST△△△' • 'SEPTEMBER' • 'OCTOBER△△' • 'NOVEMBER△'	

項番	日時書式の持つ意味	日時書式に指定できる要素	説明
			• 'DECEMBER△' △は，半角空白を意味しています。
12	週	W	月内の週を意味しています。値の範囲は1～5 です。 • 1：各月の 1～7 日を意味しています。 • 2：各月の 8～14 日を意味しています。 • 3：各月の 15～21 日を意味しています。 • 4：各月の 22～28 日を意味しています。 • 5：各月の 29 日以降を意味しています。 なお，2 月（うるう年を除く）は1～4 となります。
13		WW	年内の週を意味しています。値の範囲は01～53 です。 例えば，01 は 1 月 1 日～1 月 7 日を，02 は 1 月 8 日～1 月 14 日を意味しています。
14	日	DD	月初からの通算日を意味しています。01～該当する月の最終日を使用できます。
15		DDD	年初からの通算日を意味しています。001～365（うるう年の場合は001～366）を使用できます。 例えば，001 は 1 月 1 日を，002 は 1 月 2 日を意味しています。032 は 2 月 1 日を意味しています。
16	曜日	D	数字で表した曜日を意味しています。1～7 を使用できます。 • 1：日曜日 • 2：月曜日 • 3：火曜日 • 4：水曜日 • 5：木曜日 • 6：金曜日 • 7：土曜日
17		DAY	英語で表した曜日を意味しています。 • 'SUNDAY△△△' • 'MONDAY△△△' • 'TUESDAY△△' • 'WEDNESDAY' • 'THURSDAY△' • 'FRIDAY△△△' • 'SATURDAY△' △は，半角空白を意味しています。
18		DY	英語で表した曜日の省略形を意味しています。 • 'SUN'：日曜日 • 'MON'：月曜日 • 'TUE'：火曜日

項番	日時書式の持つ意味	日時書式に指定できる要素	説明	
			• 'WED'：水曜日 • 'THU'：木曜日 • 'FRI'：金曜日 • 'SAT'：土曜日	
19		DAYN	日本語で表した曜日を意味しています。'日曜日'，'月曜日'，'火曜日'，'水曜日'，'木曜日'，'金曜日'，'土曜日'を使用できます。	
20		DYN	日本語で表した曜日の省略形を意味しています。'日'，'月'，'火'，'水'，'木'，'金'，'土'を使用できます。	
21	時	HH	時を意味しています。00～23 を使用できます。 なお，文字データを日時データに変換する際にHH またはHH24 を指定するときは，午前/午後を指定することはできません。	
22		HH24		
23		HH12	時を意味しています。01～12 を使用できます。 なお，文字データを日時データに変換する際にHH12 を指定するときは，午前/午後も一緒に指定する必要があります。	
24	午前/午後	AM	英語の午前または午後を意味しています。※1	文字データを日時データに変換する際に午前/午後を指定するときは，HH12 も一緒に指定する必要があります。
25		A.M.		
26		PM		
27		P.M.		
28		AMN	日本語の午前または午後を意味しています。※2	
29		PMN		
30	分	MI	分を意味しています。00～59 を使用できます。	
31	秒	SS	秒を意味しています。00～59 を使用できます。	
32		SSSSS	秒を意味しています。00000～86399 を使用できます。 深夜 0 時 0 分 0 秒からの経過秒数を意味しています。例えば，午前 1 時 0 分 0 秒は03600 になります。	
33	小数秒	FF	小数秒を意味しています。 文字データを日時データに変換する場合は，*変換後のデータ型*の小数秒の桁数になります。 日時データを文字データに変換する場合は，*変換対象データ*の小数秒の桁数になります。 • 小数秒の桁数が 0 の場合，指定を無視します。 • 小数秒の桁数が 3 の場合，FF3 と同じ意味です。 • 小数秒の桁数が 6 の場合，FF6 と同じ意味です。 • 小数秒の桁数が 9 の場合，FF9 と同じ意味です。 • 小数秒の桁数が 12 の場合，FF12 と同じ意味です。	
34		FF1	小数秒 1 桁（0～9）を意味しています。	

項番	日時書式の持つ意味	日時書式に指定できる要素	説明
35		FF2	小数秒 2 桁（00～99）を意味しています。
36		FF3	小数秒 3 桁（000～999）を意味しています。
37		FF4	小数秒 4 桁（0000～9999）を意味しています。
38		FF5	小数秒 5 桁（00000～99999）を意味しています。
39		FF6	小数秒 6 桁（000000～999999）を意味しています。
40		FF7	小数秒 7 桁（0000000～9999999）を意味しています。
41		FF8	小数秒 8 桁（00000000～99999999）を意味しています。
42		FF9	小数秒 9 桁（000000000～999999999）を意味しています。
43		FF10	小数秒 10 桁（0000000000～9999999999）を意味しています。
44		FF11	小数秒 11 桁（00000000000～99999999999）を意味しています。
45		FF12	小数秒 12 桁（000000000000～999999999999）を意味しています。
46	区切り文字	ハイフン（-）	各要素を区切る際の文字です。 （例） 'YYYY-MM-DD HH:MI:SS'
47		スラッシュ（/）	
48		コンマ（,）	
49		ピリオド（.）	
50		コロン（:）	
51		セミコロン（;）	
52		空白（ ）	
53	そのほか	"*文字列*"	任意の文字列を二重引用符（"）で囲んで指定できます。 （例） CONVERT(DATE'2013-01-01',VARCHAR(20),'CC"世紀"') →'21世紀' 下線部分が該当個所です。 なお，二重引用符で囲んだ文字列中に二重引用符を指定する場合は，連続する 2 つの二重引用符（""）を指定します。 （例）文字列としてAB"CD を指定する場合 "AB""CD"
54		FM	MONTH およびDAY に対応する文字列の後ろの半角空白を削除するかどうか，YYYY などの数字のゼロサプレスをするかどうかを制御します。指定方法については，「(c)　日時書式の要素 FM の指定方法」を参照してください。

注※1

- 文字データを日時データに変換する場合，AM，A.M.，PM，またはP.M.のどれを指定しても変換結果はすべて同じになります。対象データ中でAM，A.M.，PM，またはP.M.のどれが使用されていても，日時

書式の要素にAM，A.M.，PM，またはP.M.のどれかを指定すれば，変換結果はすべて同じになります。また，大文字，小文字も区別されません。

- 日時データを文字データに変換する場合，AM またはPM のどちらを指定しても変換結果は同じになります。また，A.M.またはP.M.のどちらを指定しても変換結果は同じになります。A.M.とAM，PM とP.M.は，変換後の文字列中にピリオドが付くかどうかだけが異なります。

注※2

AMN またはPMN のどちらを指定しても変換結果は同じになります。対応する文字列は，午前の場合は'午前'，午後の場合は'午後'になります。

メモ

日時データを文字データに変換する場合は，日時書式のAM，PM などの指定に関係なく，変換対象の日時データの値によって変換されます。

また，文字データを日時データに変換する場合は，日時書式のAM，PM などの指定に関係なく，変換対象の文字データ中にあるAM，PM などの指定に従い変換されます。

(b) 日時書式に関する規則

- 日時書式の長さは，64 バイト以内にしてください。
- 日時書式に指定する文字は，二重引用符（”）で囲まれた文字列以外は，半角文字で指定してください。
- 日時書式に指定する文字の大文字，小文字は区別されません。ただし，次の文字列については，大文字，小文字が区別されます。
  - AM，A.M.，PM，およびP.M.の 1 文字目
  - MON，MONTH，DAY，およびDY の 1～2 文字目
  - 二重引用符（”）で囲まれた文字列中の文字
- 文字データを日時データに変換する場合，次の日時書式の要素は指定できません。
  - CC（世紀）
  - Q（四半期）
  - WW（年内の週）
  - W（月内の週）
  - YY（2 桁で表した西暦年）
- TIME 型のデータを文字データに変換する場合，次の日時書式の要素は指定できません。
  - CC（世紀）
  - YYYY，YY（西暦年）
  - E，EE（時代名）
  - YYYYN，YYN（和暦年）

- Q（四半期）
- MM，MON，MONTH（月）
- W，WW（週）
- DD，DDD（日）
- D，DAY，DAYN，DY，DYN（曜日）

- 文字データを日時データに変換する場合，同じ意味を持つ日時書式の要素を 2 つ以上指定できません。例えば，次のような指定はできません。

（例 1）'YYYY-MM-DD-YYYY'

YYYY を 2 つ指定できません。

（例 2）'YYYY-MM-DD-EYYN'

YYYY とYYN は，同じ意味を持つ日時書式の要素のため，指定できません。

同じ意味を持つ日時書式の要素を次の表に示します。

表 8-52　同じ意味を持つ日時書式の要素

項番	日時書式の持つ意味	同じ意味を持つ日時書式の要素
1	年	YYYY
2		YYYYN
3		YYN
4	時代名	E
5		EE
6	月	MM
7		MON
8		MONTH
9		DDD
10	日	DD
11		DDD
12	時	HH
13		HH24
14		HH12
15		SSSSS
16	午前/午後	AM
17		A.M.
18		PM

項番	日時書式の持つ意味	同じ意味を持つ日時書式の要素
19		P.M.
20		AMN
21		PMN
22	分	MI
23		SSSSS
24	秒	SS
25		SSSSS
26	小数秒	FF
27		FF1
28		FF2
29		FF3
30		FF4
31		FF5
32		FF6
33		FF7
34		FF8
35		FF9
36		FF10
37		FF11
38		FF12

- 文字データを日時データに変換する際に曜日（D，DAY，DY，DAYN，またはDYN）を指定した場合，曜日と日付の指定に矛盾があってもエラーにはなりません。
- 日時データを文字データに変換する際に，日時書式にAM，A.M.，PM，またはP.M.を指定した場合，1 文字目を大文字で指定したときにはすべて大文字で，1 文字目を小文字で指定したときにはすべて小文字で出力されます。
- 日時書式の要素に時代名を使用する場合，対応する西暦年の範囲は，西暦 1873 年 1 月 1 日（明治 6 年 1 月 1 日）～西暦 9999 年 12 月 31 日（令和 7981 年 12 月 31 日）となります。対応する和暦年の範囲は次のとおりです。
  - 明治：06 年 01 月 01 日～45 年 07 月 29 日
  - 大正：01 年 07 月 30 日～15 年 12 月 24 日
  - 昭和：01 年 12 月 25 日～64 年 01 月 07 日
  - 平成：01 年 01 月 08 日～31 年 04 月 30 日

- 令和：01 年 05 月 01 日～7981 年 12 月 31 日

ただし，時代名に平成を指定して文字データを日時データに変換する場合は，平成 8011 年 12 月 31 日まで指定できます。

(例)

CONVERT('05/01/0031/平成',DATE,'MM/DD/YYYYN/EE') → 2019-05-01

CONVERT('12/31/8011/H',DATE,'MM/DD/YYYYN/E') → 9999-12-31

- HH24 で表す 0 時は，午前/午後とHH12 を使用して表すと午前 12 時になります。また，HH24 で表す 12 時を，午前/午後とHH12 を使用して表すと午後 12 時になります。
- 日時書式に指定する文字列は，先頭（左）から順に要素を切り出します。切り出す際，長い要素名として切り出すことも，短い要素名として切り出すこともできる場合は，長い要素名として切り出します。例えば，'DDD'を指定した場合，1 つ目の要素をD やDD ではなく，DDD として切り出します。
- 文字データを日時データに変換する際に二重引用符で囲まれた文字列を指定する場合，二重引用符で囲まれた文字列中の英字と，変換対象データ中の英字の大文字，小文字を一致させてください。また，日時データを文字データに変換する場合，二重引用符で囲まれた文字列中の大文字と小文字は区別されて出力されます。
- 文字データを日時データに変換する際にE を指定する場合，変換対象データ中の大文字，小文字はどちらであっても同じように変換されます。日時データを文字データに変換する場合は，大文字で出力されます。

(c) 日時書式の要素 FM の指定方法

■日時データを文字データに変換する場合

FM の指定がない場合，日時書式の要素にMONTH またはDAY が指定されているときは，変換後の文字列は 9 文字固定になります。9 文字に足りない分は，半角空白を埋め込みます。

また，年，月，日などの数字のゼロサプレスはしません。

(例) FM の指定がない場合

```
CONVERT(DATE'2014-01-05',CHAR(17),'YYYY-MONTH-DD')
→'2014-JANUARY△△-05'
```

JANUARY の後ろの半角空白を削除しません。2 つの半角空白を埋め込んで 9 文字にします。また，日を表す05 のゼロサプレスをしません。

(例) FM の指定がある場合

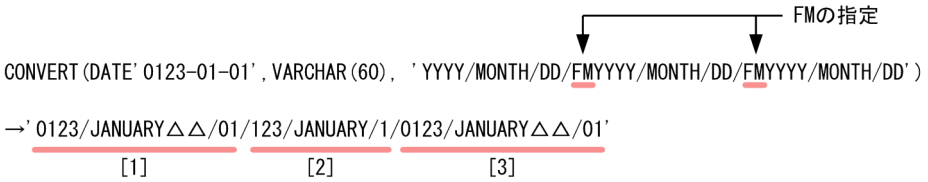
```
CONVERT(DATE'2014-01-05',CHAR(14),'FMYYYY-MONTH-DD')
→'2014-JANUARY-5'
```

JANUARY の後ろの半角空白を削除します。また，日を表す05 のゼロサプレスをします。

このように，変換後の文字データの半角空白の削除，および数字のゼロサプレスをしたい場合は，日時書式の要素にFM を指定します。

なお，FM を途中で指定することによって，指定したところから処理を切り替えることができます。

(例)



（凡例）△：半角空白

[説明]

1. MONTH に対応する文字列（この例ではJANUARY△△）の半角空白を削除しません。また，YYYY（この例では0123），DD（この例では01）に対応する数字のゼロサプレスを行いません。
2. MONTH に対応する文字列の後ろの半角空白を削除して，JANUARY と変換します。また，YYYY，DD に対応する数字のゼロサプレスを行い，YYYY を123 と，DD を1 と変換します。
3. MONTH に対応する文字列（この例ではJANUARY△△）の半角空白を削除しません。また，YYYY（この例では0123），DD（この例では01）に対応する数字のゼロサプレスを行いません。

■文字データを日時データに変換する場合

- 日時書式の要素にMONTH またはDAY を指定する場合，変換対象の文字データ中に半角空白（1 月の場合はJANUARY△△）が必要になります。変換対象の文字データ中に半角空白がない場合は（1 月の場合はJANUARY），日時書式の要素にFM を指定すると，半角空白がなくても変換できます。

  ＜エラーになる例＞

  CONVERT('2014-JANUARY-05',DATE,'YYYY-MONTH-DD') → エラー

  JANUARY の後ろに半角空白が 2 つないため，エラーになります。

  ＜エラーにならない例＞

  CONVERT('2014-JANUARY-05',DATE,'FMYYYY-MONTH-DD') → DATE'2014-01-05'

  FM を指定しているため，JANUARY の後ろの半角空白 2 つは必要ありません。

  なお，FM を指定した場合に，JANUARY△△のように半角空白があるときは，エラーになります。

  > **重要**
  >
  > FM を指定した場合，次のようなケースでは，意図したとおりの変換結果が返らないことがあります。例えば，文字列'2014111'をDATE 型（2014 年 1 月 11 日）に変換する場合，次のように指定すると，意図したとおりの変換結果が返りません。
  >
  > CONVERT('2014111',DATE,'FMYYYYMMDD') → DATE'2014-11-01'
  >
  > 上記の例の場合，2014 年 11 月 1 日に変換されます。

- 日時書式に数字で指定する要素（「表 8-53　数字で指定する日時書式の要素の前ゼロを含めた最大文字数」に示す要素）を指定する場合，変換対象の文字データ中の数字が「表 8-53　数字で指定する日時書式の要素の前ゼロを含めた最大文字数」に示す最大文字数になっている必要があります。例えば，要素MM を指定した場合，文字データ中の 1 月～9 月を表す数字は，01～09 となっている必要があります（前ゼロが必要になります）。変換対象の文字データ中に前ゼロがない場合は，日時書

式の要素にFM を指定すると，前ゼロがなくても変換できます（前ゼロがあってもなくてもどちらでもエラーにはなりません）。

＜エラーになる例＞

CONVERT('2014:1:5',DATE,'YYYY:MM:DD') → エラー

＜エラーにならない例＞

CONVERT('2014:1:5',DATE,'FMYYYY:MM:DD') → DATE'2014-01-05'

CONVERT('2014:01:05',DATE,'FMYYYY:MM:DD') → DATE'2014-01-05'

- 数字で指定する要素（「表 8-53　数字で指定する日時書式の要素の前ゼロを含めた最大文字数」に示す要素）の値が0 であっても，変換対象データの各要素に 1 文字以上の指定が必要です。例えば，日時書式に'FMHH:MI:SS' を指定して，0 時 0 分 0 秒のデータを変換する場合，変換対象データが'0:0:0' のときは変換できますが，変換対象データが'0:0:' のときはエラーになります。

  ただし，FM の適用対象外であるFF およびFF1～FF12 の場合は，0 の指定を省略できます。例えば，日時書式に'FMHH:MI:SS.FF3' を指定して，0 時 0 分 0.000 秒のデータを変換する場合，変換対象データが'0:0:0.' であっても変換できます。

- FM を途中で指定することによって，指定したところから処理を切り替えることができます。

- FM が指定されている場合，HADB はMM，DD などの要素に対応する数字の範囲を，数字以外の文字が出現するか，または指定した日時書式の最大文字数に達するかのどちらかで認識しています。数字で指定する日時書式の要素の前ゼロを含めた最大文字数を次の表に示します。

表 8-53　数字で指定する日時書式の要素の前ゼロを含めた最大文字数

項番	数字で指定する日時書式の要素	前ゼロを含めた最大文字数
1	YYYY	4
2	YYYYN	4
3	YYN	2
4	MM	2
5	DD	2
6	DDD	3
7	D	1
8	HH	2
9	HH24	2
10	HH12	2
11	MI	2
12	SS	2
13	SSSSS	5
14	FF	対象外

項番	数字で指定する日時書式の要素	前ゼロを含めた最大文字数
15	FF1	
16	FF2	
17	FF3	
18	FF4	
19	FF5	
20	FF6	
21	FF7	
22	FF8	
23	FF9	
24	FF10	
25	FF11	
26	FF12	

(4) 数値書式に指定できる要素および規則

(a) 数値書式の要素の指定形式

数値書式の要素の指定形式を次に示します。省略した要素は詰めて記述してください。記述順序を間違えたり，指定できない要素を指定したりした場合はエラーになります。

```
数値書式 ： ： ＝ {固定小数点表記 | 浮動小数点表記 | 最短表記 | 16進数表記}


  固定小数点表記 ： ： ＝
        〔”文字列”〕〔〔修飾要素〕〔符号要素〕〔B〕〔通貨要素〕
        〔数要素〔〔{区切り文字要素 | 数要素}〕…数要素〕〕〔.〕
        〔数要素〕…〔符号要素〕〔”文字列”〕〕

  浮動小数点表記 ： ： ＝
        〔”文字列”〕〔修飾要素〕〔数要素〕…〔.〕〔数要素〕…
        浮動小数点要素〔”文字列”〕

  最短表記 ： ： ＝〔”文字列”〕{TM | TM9 | TME}〔”文字列”〕

  16進数表記 ： ： ＝
        〔”文字列”〕〔修飾要素〕〔0〕…16進数要素〔16進数要素〕…〔”文字列”〕

    修飾要素 ： ： ＝ {LJ | LS}
    符号要素 ： ： ＝ {MI | S | PR}
    通貨要素 ： ： ＝ {$ | ¥}
    数要素 ： ： ＝ {0 | 9}
    区切り文字要素 ： ： ＝ {, | △}
```

浮動小数点要素：：＝ {EEEE | eeee}
16進数要素：：＝ {X | x}

注

- 区切り文字要素の△は半角空白を意味しています。
- ”*文字列*”は二重引用符で囲んだ任意の文字列を意味しています。

(b) 数値書式の要素

数値書式に指定できる要素を次の表に示します。

なお，表中の△は半角空白を意味しています。

表 8-54 数値書式に指定できる要素

項番	要素の種類	数値書式に指定できる要素	説明
1	区切り文字要素	,（コンマ）	**■数データを文字データに変換する場合** • 変換後の文字データに，数要素を区切るためのコンマを挿入する場合に指定します。コンマを指定した位置にコンマが挿入されます。 （例） CONVERT(1234567,CHAR(10),'9,999,999') →'△1,234,567' • 「数データの整数部の桁数＜数値書式に指定した整数部の桁数」の場合，桁数が多い部分にコンマは挿入されません。 （例） CONVERT(1234,CHAR(10),'9,999,999') →'△△△△△1,234' **■文字データを数データに変換する場合** • 変換対象の文字データ中にコンマがある場合に指定します。コンマを指定した位置からコンマを削除して，数データに変換します。 （例） CONVERT('1,234,567',INTEGER,'9,999,999') →1234567 • 変換対象の文字データ中に，数値書式で指定した位置にコンマがない場合はエラーになります。 （例） CONVERT('1234',INTEGER,'9,999') →エラー • 「文字データの整数部の桁数＜数値書式に指定した整数部の桁数」の場合，桁数が多い部分に指定されたコンマは無視されます。 （例） CONVERT('1,234',INTEGER,'9,999,999') →1234

項番	要素の種類	数値書式に指定できる要素	説明
			指定規則 小数点を意味するピリオドの右側にコンマは指定できません。 （例）エラーになる数値書式の指定例 '999,999.9,99' ',999,999,999'
2		半角空白	■数データを文字データに変換する場合 • 変換後の文字データに，数要素を区切るための半角空白を挿入する場合に指定します。半角空白を指定した位置に半角空白が挿入されます。 （例） CONVERT(1234567,CHAR(10),'9 999 999') →'△1△234△567' ■文字データを数データに変換する場合 • 変換対象の文字データ中に半角空白がある場合に指定します。半角空白を指定した位置から半角空白を削除して，数データに変換します。 （例） CONVERT('1 234 567',INTEGER,'9 999 999') →1234567 • 変換対象の文字データ中に，数値書式で指定した位置に半角空白がない場合はエラーになります。 （例） CONVERT('1234',INTEGER,'9 999') →エラー •「文字データの整数部の桁数＜数値書式に指定した整数部の桁数」の場合，桁数が多い部分に指定された半角空白は無視されます。 （例） CONVERT('1 234',INTEGER,'9 999 999') →1234 指定規則 小数点を意味するピリオドの右側に半角空白は指定できません。 （例）エラーになる数値書式の指定例 '9.99 9'
3	小数点文字	.（ピリオド）	小数点の位置をピリオドで指定します。数値書式に指定したピリオドの前の数要素が整数部を表し，ピリオドの後ろの数要素が小数点以下を表します。 ■数データを文字データに変換する場合 • 数データを数値書式に指定した整数部と小数点以下に変換します。 （例） CONVERT(1234.56,CHAR(9),'9,999.99') →'△1,234.56' •「数データの小数点以下の桁数＞数値書式に指定した小数点以下の桁数」の場合，数値を丸めて変換します。数値の丸め方は，スカラ関数

項番	要素の種類	数値書式に指定できる要素	説明
			ROUND の丸め方と同じになります。スカラ関数ROUND については，「8.4.9　ROUND」を参照してください。 （例） CONVERT(1.56,CHAR(4),'9.9') →'△1.6' ■文字データを数データに変換する場合 • 文字データのピリオドの左側を整数，ピリオドの右側を小数点以下として，数データに変換します。 （例） CONVERT('1,234.56',DECIMAL(6,2),'9,999.99') →1234.56 指定規則 ピリオドは，数要素の直前，直後，または数要素の間に 1 個だけ指定できます。 （例）エラーになる数値書式の指定例 '.$999'
4	通貨要素	$	■数データを文字データに変換する場合 • 変換後の文字データに，ドル記号（$）を付加する場合に指定します。 （例） CONVERT(1000,CHAR(7),'$9,999') →'△$1,000' CONVERT(-1000,CHAR(7),'$9,999') →'-$1,000' ■文字データを数データに変換する場合 • 変換対象の文字データ中にドル記号（$）がある場合に指定します。ドル記号を削除して，数データに変換します。 （例） CONVERT('$1,000',INTEGER,'$9,999') →1000 • 数字の前にドル記号がない場合はエラーになります。 （例） CONVERT('1,000',INTEGER,'$9,999') →エラー • ドル記号と最上位の数字の間に半角空白がないようにしてください。 （例） CONVERT('$ 1,000',INTEGER,'$9,999') →エラー
5		¥	■数データを文字データに変換する場合 • 変換後の文字データに，円記号（¥）を付加する場合に指定します。 （例） CONVERT(1000,CHAR(7),'¥9,999')

項番	要素の種類	数値書式に指定できる要素	説明
			→'△¥1,000' CONVERT(-1000,CHAR(7),'¥9,999') →'-¥1,000' **■文字データを数データに変換する場合** • 変換対象の文字データ中に円記号（¥）がある場合に指定します。円記号を削除して，数データに変換します。 （例） CONVERT('¥1,000',INTEGER,'¥9,999') →1000 • 数字の前に円記号がない場合はエラーになります。 （例） CONVERT('1,000',INTEGER,'¥9,999') →エラー • 円記号と最上位の数字の間に半角空白がないようにしてください。 （例） CONVERT('¥ 1,000',INTEGER,'¥9,999') →エラー
6	数要素	0	数値の 1 つの桁を意味し，変換対象データの数値に対応する桁を変換します。ここでは，固定小数点表記で指定する数要素0 について説明しています。浮動小数点表記，および 16 進数表記で指定する数要素0 については，浮動小数点要素および 16 進数要素で説明します。 **■数データを文字データに変換する場合** • 数要素0 および9 の個数の合計は，変換後の数字の最大桁数を示しています。 （例） CONVERT(1234.5,CHAR(10),'00,000.00') →'△01,234.50' • 「数データの小数点以下の桁数＞数値書式に指定した小数点以下の桁数」の場合，数値を丸めて変換します。数値の丸め方は，スカラ関数ROUND の丸め方と同じになります。スカラ関数ROUND については，「8.4.9　ROUND」を参照してください。 （例） CONVERT(2.34567,CHAR(5),'0.00') →'△2.35' • 数値書式に指定した数要素，区切り文字，ピリオドの指定に従って変換された文字データの先頭に符号を付加します。0 または正の値の場合は半角空白を付加します。負の値の場合はマイナス記号（-）を付加します。ただし，符号要素を指定した場合は，符号要素の指定に従って符号を示す文字列を付加します。 （例） CONVERT(-1234.5,CHAR(8),'0,000.0') →'-1,234.5' CONVERT(0,CHAR(8),'0,000.0')

項番	要素の種類	数値書式に指定できる要素	説明
			→'△0,000.0' • 「数データの整数部または小数点以下の桁数＜数値書式に指定した整数部または小数点以下の桁数」の場合，対応しない桁は文字列の 0 に変換されます。 (例) CONVERT(1.1,CHAR(10),'0,000.000') →'△0,001.100' • 「数データの整数部の桁数＞数値書式に指定した整数部の桁数」の場合，文字列#に変換されます。 (例) CONVERT(1234,CHAR(3),'00') →'###' ■文字データを数データに変換する場合 • 数要素0 または9 のどちらを指定しても同じように変換されます。文字データの整数部の有効桁数を超えた桁に半角空白がある場合でも，数要素0 を指定して変換できます。また，文字データの整数部の有効桁数を超えた桁に文字列0 がある場合でも，数要素9 を指定して変換できます。 (例) CONVERT('0,123',INTEGER,'9,999') →123 • 文字データと数値書式の桁の対応付けは，小数点を基点として，整数桁は左側に向かって 1 の位，10 の位，…の順に，小数点以下の桁は右側に向かって小数点第 1 位，小数点第 2 位，…の順に行います。また，文字データの桁数と数値書式の桁数が合っていなくても，「文字データの整数部の桁数＜数値書式に指定した整数部の桁数」，「文字データの小数点以下の桁数＜数値書式に指定した小数点以下の桁数」の場合は変換できます。 (例) CONVERT('1,234.56',DECIMAL(8,3),'00,000.000') →1234.560 次の場合はエラーになります。 • 「文字データの整数部の桁数＞数値書式に指定した整数部の桁数」の場合 (例) CONVERT('1234',INTEGER,'00') →エラー • 「文字データの小数点以下の桁数＞数値書式に指定した小数点以下の桁数」の場合 (例) CONVERT('1.234',DECIMAL(2,1),'0.0') →エラー • 文字データ中の符号と最上位の数字の間に半角空白がある場合 (例)

項番	要素の種類	数値書式に指定できる要素	説明
			CONVERT('- 1,234',INTEGER,'0,000') →エラー 指定規則 数要素0 および9 は，合計で最大 38 個指定できます。
7		9	数値の 1 つの桁を意味し，変換対象データの数値に対応する桁を変換します。ここでは，固定小数点表記で指定する数要素9 について説明しています。浮動小数点表記，および 16 進数表記で指定する数要素9 については，浮動小数点要素および 16 進数要素で説明します。 **■数データを文字データに変換する場合** • 要素 0 および 9 の個数の合計は，変換後の数字の最大桁数を示しています。 (例) CONVERT(1234.5,CHAR(10),'99,999.99') →'△△1,234.50' • 「数データの小数点以下の桁数＞数値書式に指定した小数点以下の桁数」の場合，数値を丸めて変換します。数値の丸め方は，スカラ関数ROUND の丸め方と同じになります。スカラ関数ROUND については，「8.4.9　ROUND」を参照してください。 (例) CONVERT(2.34567,CHAR(5),'9.99') →'△2.35' • 数値書式に指定した数要素，区切り文字，ピリオドの指定に従って変換された文字データの先頭に符号を付加します。0 または正の値の場合は半角空白を付加します。負の値の場合はマイナス記号（-）を付加します。ただし，符号要素を指定した場合は，符号要素の指定に従って符号を示す文字列を付加します。 (例) CONVERT(1234.5,CHAR(8),'9,999.9') →'△1,234.5' CONVERT(-1234.5,CHAR(8),'9,999.9') →'-1,234.5' • 「数データの整数部の桁数＜数値書式に指定した整数部の桁数」の場合，対応しない桁は半角空白に変換されます。また，「数データの小数点以下の桁数＜数値書式に指定した小数点以下の桁数」の場合，対応しない桁は文字列の 0 に変換されます。 (例) CONVERT(1.1,CHAR(10),'9,999.999') →'△△△△△1.100' • 小数点以下に数要素の指定がない場合，数データを数値書式の桁数で丸めた結果が0 のときは，文字列の0 に変換されます。 (例) CONVERT(0.1,CHAR(2),'9') →'△0'

項番	要素の種類	数値書式に指定できる要素	説明
			小数点以下に数要素の指定がある場合，変換後の文字データの整数部は，0 ではなく半角空白に変換されます。 (例) CONVERT(0.1,CHAR(5),'9.99') →'△△.10' CONVERT(0,CHAR(5),'9.99') →'△△.00' また，負の値の場合は，次のように変換されます。 (例) CONVERT(-0.1,CHAR(5),'9.99') →'△-.10' • 数要素 9 の前に数要素 0 を指定した場合，指定した数要素 0 以降の数要素 9 は，数要素 0 として扱われます。 (例) CONVERT(1,CHAR(5),'0999') →'△0001' •「数データの整数部の桁数＞数値書式に指定した整数部の桁数」の場合，文字列#に変換されます。 (例) CONVERT(1234,CHAR(3),'99') →'###' ■文字データを数データに変換する場合 • 数要素0 または9 のどちらを指定しても同じように変換されます。文字データの整数部の有効桁数を超えた桁に半角空白がある場合でも，数要素0 を指定して変換できます。また，文字データの整数部の有効桁数を超えた桁に文字列0 がある場合でも，数要素9 を指定して変換できます。 (例) CONVERT('0,123',INTEGER,'9,999') →123 • 文字データと数値書式の桁の対応付けは，小数点を基点として，整数桁は左側に向かって 1 の位，10 の位，…の順に，小数点以下の桁は右側に向かって小数点第 1 位，小数点第 2 位，…の順に行います。また，文字データの桁数と数値書式の桁数が合っていなくても，「文字データの整数部の桁数＜数値書式に指定した整数部の桁数」，「文字データの小数点以下の桁数＜数値書式に指定した小数点以下の桁数」の場合は変換できます。 (例) CONVERT('1,234.56',DECIMAL(8,3),'99,999.999') →1234.560 次の場合はエラーになります。 •「文字データの整数部の桁数＞数値書式に指定した整数部の桁数」の場合 (例)

項番	要素の種類	数値書式に指定できる要素	説明
			CONVERT('1234',INTEGER,'99') →エラー • 「文字データの小数点以下の桁数＞数値書式に指定した小数点以下の桁数」の場合 （例） CONVERT('1.234',DECIMAL(4,3),'9.9') →エラー • 文字データ中の符号と最上位の数字の間に半角空白がある場合 （例） CONVERT('- 1,234',INTEGER,'9,999') →エラー 指定規則 数要素0 および9 は，合計で最大 38 個指定できます。
8	浮動小数点要素	EEEE eeee	浮動小数点数定数を示すときに指定します。数要素0 または9 のどちらを指定しても，同じように変換されます。数値書式の指定例を次に示します。 （例） • '9.999EEEE' • '9.999eeee' • '9.EEEE' • '9EEEE' • '.9EEEE' ※ • '99.9EEEE' ※ 注※　文字データを数データに変換する場合に限り指定できます。 ■数データを文字データに変換する場合 • 数値書式の指定に従って，浮動小数点数定数の形式に変換されます。 （例） CONVERT(12.3,CHAR(9),'9.99EEEE') →'△1.23E+01' CONVERT(0.01,CHAR(9),'9.99EEEE') →'△1.00E-02' 変換後の文字データの指数が 0 または正の値の場合は，指数にプラス符号（+）が付加されます。 変換後の文字データの指数は，2～3 桁になります。値が0 の場合，指数は00 になります。 • 「数データの小数点以下の桁数＞数値書式に指定した小数点以下の桁数」の場合，数値を丸めて変換します。数値の丸め方は，スカラ関数ROUND の丸め方と同じになります。スカラ関数ROUND については，「8.4.9　ROUND」を参照してください。 （例） CONVERT(34.56,CHAR(9),'9.99EEEE') →'△3.46E+01'

項番	要素の種類	数値書式に指定できる要素	説明
			• 数値書式に指定した数要素，ピリオド，要素EEEE の指定に従って変換された文字データの先頭に符号を付加します。0 または正の値の場合は半角空白を付加します。負の値の場合はマイナス記号（-）を付加します。 （例） CONVERT(0,CHAR(9),'9.99EEEE') →'△0.00E+00' CONVERT(-1,CHAR(9),'9.99EEEE') →'-1.00E+00' • 要素を小文字eeee で指定した場合，浮動小数点数定数の表記のE は，小文字のe に変換されます。 （例） CONVERT(1,CHAR(9),'9.99eeee') →'△1.00e+00' • 整数部に数要素を 1 個指定してください。0 個または 2 個以上指定した場合はエラーになります。 （例） CONVERT(1,CHAR(9),'99.9EEEE') →エラー ■文字データを数データに変換する場合 • 浮動小数点数定数の表記で表現されている文字データを浮動小数点数データに変換します。 （例） CONVERT('1.23E+10*浮動小数点文字列*',DOUBLE PRECISION,'9.99EEEE"*浮動小数点文字列*"') →1.23000000000000000E10 CONVERT('-1.23E+10*浮動小数点文字列*',DOUBLE PRECISION,'9.99EEEE"*浮動小数点文字列*"') →-1.23000000000000000E10 • 要素EEEE と変換対象データ中のE の文字は，大文字，小文字の区別をしません。 （例） CONVERT('1.23e+10',DOUBLE PRECISION,'9.99EEEE') →1.23000000000000000E10 次の場合はエラーになります。 •「文字データの整数部の桁数＞数値書式に指定した整数部の桁数」の場合 （例） CONVERT('12.3E+1',DOUBLE PRECISION,'9.9EEEE') →エラー •「文字データの小数点以下の桁数＞数値書式に指定した小数点以下の桁数」の場合 （例） CONVERT('1.234E+1',DOUBLE PRECISION,'9.9EEEE')

項番	要素の種類	数値書式に指定できる要素	説明
			→エラー • 文字データ中の仮数の符号と最上位の数字の間に半角空白がある場合 (例) CONVERT('- 1.23E+1',DOUBLE PRECISION,'9.99EEEE') →エラー 指定規則 • 整数部に指定した数要素と，小数点以下に指定した数要素の合計が，最大 17 個指定できます。 • EEEE またはeeee と指定してください。大文字と小文字を混在して指定することはできません。
9	符号要素	MI	■数データを文字データに変換する場合 • 数値書式に指定した数要素，区切り文字，ピリオドの指定に従って変換された文字データの後ろに，負の値の場合はマイナス符号（-）を付加し，0 または正の値の場合は半角空白を付加します。 (例) CONVERT(-123,CHAR(4),'999MI') →'123-' CONVERT(123,CHAR(4),'999MI') →'123△' ■文字データを数データに変換する場合 • 要素MI の指定位置を符号と見なし，数データに変換します。マイナス符号（-）の場合は負の値に変換し，プラス符号（+）または半角空白の場合は，0 以上の値に変換します。 (例) CONVERT('123-',INTEGER,'999MI') →-123 また，変換対象データから"*文字列*"で指定した部分を除いたデータの終端が数字またはピリオドの場合は，0 または正の値に変換します。 (例) CONVERT('123$',INTEGER,'999MI"$"') →123
10		S（先頭にS を指定した場合）	■数データを文字データに変換する場合 • 数値書式に指定した数要素，区切り文字，ピリオドの指定に従って変換された文字データの前に符号を付加します。負の値の場合はマイナス符号（-）を，0 または正の値の場合はプラス符号（+）を付加します。 (例) CONVERT(123,CHAR(4),'S999') →'＋123' • 数値書式に通貨要素を指定した場合は，通貨記号の前に符号を付加します。 (例) CONVERT(123,CHAR(5),'S$999')

項番	要素の種類	数値書式に指定できる要素	説明
			→'＋$123' ■文字データを数データに変換する場合 • 文字データの数字の前にある符号に従って数データに変換します。符号がない場合はエラーになります。 (例) CONVERT('+$123',INTEGER,'S$999') →123 CONVERT('123',INTEGER,'S999') →エラー • 文字データ中の符号と最上位の数字の間に半角空白があるとエラーになります。 (例) CONVERT('+ 123',INTEGER,'S999') →エラー • 数値書式に通貨要素を指定した場合，文字データ中の通貨記号の前に符号がないとエラーになります。 (例) CONVERT('+$123',INTEGER,'S$999') →123 CONVERT('$123',INTEGER,'S$999') →エラー
11		S（末尾にS を指定した場合）	■数データを文字データに変換する場合 • 数値書式に指定した数要素，区切り文字，ピリオドの指定に従って変換された文字データの末尾に符号を付加します。負の値の場合はマイナス符号（-）を，0 または正の値の場合はプラス符号（+）を付加します。 (例) CONVERT(123,CHAR(4),'999S') →'123＋' ■文字データを数データに変換する場合 • 数値書式に指定した要素S の位置の符号に従って，数データに変換します。符号がない場合は，エラーになります。 (例) CONVERT('123+',INTEGER,'999S') →123 CONVERT('123',INTEGER,'999S') →エラー
12		PR	■数データを文字データに変換する場合 • 数データが負の値の場合，数値書式に従って変換された文字データを<>の中に設定します。 (例) CONVERT(-123,CHAR(5),'999PR')

項番	要素の種類	数値書式に指定できる要素	説明
			→'<123>' • 数データが 0 または正の値の場合は，< >の替わりに半角空白が挿入されます。 (例) CONVERT(123,CHAR(5),'999PR') →'△123△' • 数値書式に通貨要素または"*文字列*"を指定した場合，数データが負の値のときは< >の中に通貨要素または"文字列"に対応する文字が設定されます。 (例) CONVERT(-123,CHAR(6),'$999PR') →'<$123>' CONVERT(-123,CHAR(12),'999PR"dollars"') →'<123dollars>' **■文字データを数データに変換する場合** • 文字データ中の数字が< >で囲まれている場合は，< >を削除して負の値に変換します。< >で囲まれていない場合は，0 または正の値に変換します。 (例) CONVERT('<123>',INTEGER,'999PR') →-123 CONVERT('123',INTEGER,'999PR') →123
13	文字列	"*文字列*"（二重引用符で囲まれた文字列）	数値書式の先頭または末尾に，二重引用符（"）で囲んだ文字列を指定できます。全角文字も指定できます。 **■数データを文字データに変換する場合** • 二重引用符で囲んだ文字列を，変換後の文字データの先頭または末尾に挿入します。 (例) CONVERT(123,CHAR(11),'999"dollars"') →'△123dollars' • 二重引用符で囲まれた文字列中の英字は，大文字，小文字を区別します。 **■文字データを数データに変換する場合** • 文字データから二重引用符で囲まれた文字列を削除して，数データに変換します。 (例) CONVERT('123dollars',INTEGER,'999"dollars"') →123 • 二重引用符で囲まれた文字列が文字データ中にない場合はエラーになります。 (例) CONVERT('123',INTEGER,'999"dollars"')

項番	要素の種類	数値書式に指定できる要素	説明
			→エラー ただし，数値書式の先頭に指定した二重引用符で囲まれた文字列の先頭が半角空白（1 個または連続している半角空白），または末尾に指定した二重引用符で囲まれた文字列の末尾が半角空白（1 個または連続している半角空白）の場合は，半角空白が文字データになくてもエラーになりません。 （例） CONVERT('dollars123',INTEGER,'" dollars"999') →123 • 二重引用符で囲まれた文字列中の英字と，変換対象データ中の英字の大文字，小文字を一致させてください。 指定規則 • 数値書式の先頭と末尾の両方に二重引用符で囲んだ文字列を指定することができます。 • ほかの要素の間に二重引用符で囲んだ文字列を指定することはできません。 • 二重引用符で囲んだ文字列中に二重引用符を指定する場合は，二重引用符を 2 つ連続して指定してください。
14	16 進数要素	X x	■数データを文字データに変換する場合 • 数データを指定された桁数の 16 進数字に変換します。 （例） CONVERT(10,CHAR(5),'XXXX') →'△△△△A' CONVERT(10,CHAR(5),'0XXX') →'△000A' • 変換後の文字データの 16 進数字の前に，半角空白が 1 個挿入されます。 （例） CONVERT(10,CHAR(2),'X') →'△A' • 要素0，X，およびx は，16 進数字に変換した 1 文字に対応します。 • 変換できる整数値の最大は，DECIMAL 型（固定小数点）で表せる正の値の最大値になります。 • 文字データに変換した 16 進数字が，指定した桁数（要素X およびx の合計数）に満たない場合，変換した文字データを右詰に格納し，不足分に対して半角空白を挿入します。 （例） CONVERT(10,CHAR(5),'XXXX') →'△△△△A' 半角空白ではなく文字列の0 を挿入したい場合は，先頭に要素0 を指定してください。要素 0 は複数個連続して指定できますが，要素X またはx の前に指定してください。 （例） CONVERT(10,CHAR(5),'0XXX')

項番	要素の種類	数値書式に指定できる要素	説明
			→'△000A' • 指定された数値が整数でない場合は，整数に丸められます。丸め方はスカラ関数ROUND と同じです。 (例) CONVERT(10.5,CHAR(6),'0XXXX') →'△0000B' • 数データが負の値の場合，文字列#に変換されます。 (例) CONVERT(-20,CHAR(6),'0XXXX') →'######' • 最初に指定した要素X が大文字の場合は，変換後の 16 進数字は大文字（A～F）になります。小文字（x）の場合は，変換後の 16 進数字は小文字（a～f）になります。 (例) CONVERT(10,CHAR(5),'xXXX') →'△△△△a' •「変換対象データを 16 進数字に変換した文字数＞数値書式に指定した桁数（要素0，X，x の合計数)」の場合，#に変換します。 (例) CONVERT(1024,CHAR(5),'XX') →'###△△' ■文字データを数データに変換する場合 • 文字データの 16 進数字（0～9，A～F，a～f）を数データに変換します。 (例) CONVERT('AB',INTEGER,'XXXX') →171 • 要素0，X，およびx は，16 進数字を 16 進数に変換する際の要素として同等に扱われます。ただし，要素0 を指定する場合は，要素X またはx の前に指定してください。 • 要素X またはx のどちらを指定しても変換結果は同じになります。また，文字データ中の 16 進数字に大文字と小文字が混在していても変換できます。 (例) CONVERT('Ab',INTEGER,'xXXx') →171 • 文字データの 16 進数字の前に半角空白がある場合でも，要素 0 を指定できます。 (例) CONVERT('*16 進文字列 A*',INTEGER,'"*16 進文字列*"0XXX') →10 • 数値書式に要素0 を指定していなくても，有効桁数より上位の桁に0 があっても変換できます。ただし，変換できるのは「上位の桁の0 を含め

項番	要素の種類	数値書式に指定できる要素	説明
			た 16 進数字の桁数≦数値書式に指定した桁数（要素X およびx の合計数)」のときだけです。 (例) CONVERT('00A',INTEGER,'XXXXX') →10 •「文字データの桁数<数値書式に指定した桁数（要素0，X，x の合計数)」の場合でも変換できます。 (例) CONVERT('A',INTEGER,'XXX') →10 • 文字データの 16 進数字は，0 または正の整数値として扱われます。 次の場合はエラーになります。 • 文字データ中に 16 進数字（0～9，A～F，a～f）以外の文字がある場合 •「文字データの桁数>数値書式に指定した桁数（要素0，X，x の合計数)」の場合 (例) CONVERT('0001',INTEGER,'XX') →エラー 指定規則 要素0，X，およびx の合計数が，最大 32 個指定できます。
15	修飾要素	LS	■数データを文字データに変換する場合 • 変換後の文字データの先頭から連続している半角空白を削除します。削除した半角空白は，文字列の後方に挿入します。 (例) CONVERT(1,CHAR(4),'LS000') →'001△' CONVERT(1,CHAR(4),'LS999') →'1△△△' • 二重引用符で囲まれた文字列中の半角空白は編集対象になりません。 ■文字データを数データに変換する場合 • 要素LS を指定しても無視されます。後続の数値書式に従って数データに変換されます。 指定規則 LS は数値書式に 1 回だけ指定できます。
16		LJ	■数データを文字データに変換する場合 • 変換後の文字データの前後に半角空白がある場合，半角空白を削除します。 (例) CONVERT(123,VARCHAR(3),'LJ999') →'123'

項番	要素の種類	数値書式に指定できる要素	説明
			• 二重引用符（”）で囲まれた文字列中の半角空白は編集対象になりません。 ■文字データを数データに変換する場合 • 要素LJ を指定しても無視されます。後続の数値書式に従って数データに変換されます。 指定規則 LJ は数値書式に 1 回だけ指定できます。
17	そのほか	B	■数データを文字データに変換する場合 • 変換対象データの値（数値書式の桁数で丸めた結果）が 0 の場合，半角空白を設定します。符号要素および通貨要素も半角空白を設定します。 （例） CONVERT(0,CHAR(4),'B999') →'△△△△' CONVERT(0,VARCHAR(4),'LJB999') →'' ■文字データを数データに変換する場合 • 要素B を指定しても無視されます。後続の数値書式に従って数データに変換されます。
18		TM TM9 TME	■数データを文字データに変換する場合 • 数データを指定された要素（TM, TM9, またはTME）に従って変換します。 TM またはTM9：整数定数または 10 進数定数の表記に従って変換します。 TME：浮動小数点数定数の表記に従って変換します。 （例） CONVERT(1.28E2,CHAR(3),'TM') →'128' CONVERT(128,CHAR(6),'TME') →'1.28E2' 数データを数定数の形式に変換した結果を文字データとして出力します。その際，数定数で表現可能な形式のうち，最も短い形式に変換されます。 • 数データが，整数定数または 10 進数定数の表記で表現できない値の場合は，TM またはTM9 を指定しても浮動小数点数定数の表記に変換されます。 ■文字データを数データに変換する場合 • TM, TM9, またはTME のどれを指定しても変換結果は同じになります。 • 文字データは，次に示す定数の表記法に従っている必要があります。 ・整数定数 ・10 進数定数 ・浮動小数点数定数 また，文字データ中に分離符号が指定できます。ただし，分離符号は変換対象にはなりません。 （例）

項番	要素の種類	数値書式に指定できる要素	説明
			CONVERT('結果 /* 注釈 */ 12345',INTEGER,'"結果"TM') →12345

(c) 数値書式に関する規則

- 数値書式の長さは，64 バイト以内にしてください。
- 数値書式に指定する文字は，二重引用符（"）で囲まれた文字列以外は，半角文字で指定してください。
- EEEE，X，または二重引用符で囲まれた文字列中の文字以外の文字は，大文字，小文字のどちらで指定しても，同じ数値書式の要素として扱われます。
- 文字データを数データに変換する場合，数値書式中に数要素を指定してください。ただし，最短表記の要素TM，TM9，TME，または 16 進数要素を指定した場合は，数要素の指定は必要ありません。
- 小数点文字（.）の直前に数要素を指定しない場合は，小数点文字（.）の直後に数要素を指定してください。
- 通貨要素，小数点文字（.），またはB を数値書式に指定した場合，数要素を指定してください。
- 数値書式に修飾要素を指定した場合，数要素または 16 進数要素を指定してください。
- 符号要素S は，数要素の前方または後方のどちらにも指定できます。
- 符号要素MI またはPR は数要素の後方にだけ指定できます。
- 符号要素を指定する場合は，S，MI，またはPR のどれか 1 つを指定してください。
- 数値書式に指定する文字列には，次に示す要素に限り 2 つ以上指定できます。そのほかの要素については，2 つ以上指定できません。
  - 区切り文字要素のコンマ
  - 区切り文字要素の半角空白
  - 数要素（0 および9）
  - "*文字列*"（二重引用符で囲まれた文字列）
  - 16 進数要素（X およびx）

(5) 規則

(a) 共通の規則

1. 実行結果のデータ型は，*変換後のデータ型*に指定したデータ型になります。ただし，*変換後のデータ型*にNUMERIC 型が指定された場合，実行結果のデータ型はDECIMAL 型になります。また，*変換後のデータ型*にFLOAT 型が指定された場合，実行結果のデータ型はDOUBLE PRECISION 型になります。
2. *変換対象データ*に？パラメタを単独で指定した場合，*変換後のデータ型*が？パラメタのデータ型として仮定されます。ただし，*変換後のデータ型*にNUMERIC 型が指定された場合は，？パラメタのデータ型に

はDECIMAL 型が仮定されます。また，*変換後のデータ型*にFLOAT 型が指定された場合は，？パラメタのデータ型にはDOUBLE PRECISION 型が仮定されます。

3. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
4. 変換対象データがナル値の場合，または変換対象データにNULL を指定した場合，実行結果はナル値になります。
5. 変換対象データが，実長 0 バイトまたは実長 0 文字の文字データの場合，次のように変換されます。
   - CHAR 型に変換する場合：半角空白に変換されます。CHAR(3)の場合，'△△△'に変換されます。△は半角空白を意味しています。
   - VARCHAR 型に変換する場合：実長 0 バイトまたは実長 0 文字のVARCHAR 型のデータに変換されます。
   - BINARY 型に変換する場合：X'00'に変換されます。BINARY(3)の場合，X'000000'に変換されます。
   - VARBINARY 型に変換する場合：実長 0 バイトのVARBINARY 型のデータに変換されます。
   - 上記以外のデータ型の場合：ナル値に変換されます。
6. データ型の変換可否（書式指定なしの場合）を次の表に示します。

表 8-55　データ型の変換可否（書式指定なしの場合）

変換対象データのデータ型	変換後のデータ型					
	INTEGER, SMALLINT	DECIMAL, NUMERIC, DOUBLE PRECISION, FLOAT	CHAR, VARCHAR	DATE, TIMESTAMP	TIME	BINARY, VARBINARY
INTEGER, SMALLINT	○	○	○	○	×	×
DECIMAL, NUMERIC, DOUBLE PRECISION, FLOAT	○	○	○	×	×	×
CHAR, VARCHAR	○	○	○	○	○	○
DATE, TIMESTAMP	○	×	○	○	×	×
TIME	×	×	○	×	○	×
BINARY, VARBINARY	×	×	○	×	×	○

（凡例）

○：変換できます。

×：変換できません。

7. データ型の変換可否（書式指定ありの場合）を次の表に示します。

表 8-56　データ型の変換可否（書式指定ありの場合）

変換対象データのデータ型	変換後のデータ型					
	INTEGER, SMALLINT	DECIMAL, NUMERIC, DOUBLE PRECISION, FLOAT	CHAR, VARCHAR	DATE, TIMESTAMP	TIME	BINARY, VARBINARY
INTEGER, SMALLINT	×	×	○※1	×	×	×
DECIMAL, NUMERIC, DOUBLE PRECISION, FLOAT	×	×	○※1	×	×	×
CHAR, VARCHAR	○※1	○※1	×	○※2	○※2	×
DATE, TIMESTAMP	×	×	○※2	×	×	×
TIME	×	×	○※2	×	×	×
BINARY, VARBINARY	×	×	×	×	×	×

（凡例）

○：変換できます。

×：変換できません。

注※1

数値書式を指定した場合に変換できます。

注※2

日時書式を指定した場合に変換できます。

8. 書式指定を指定した場合，書式指定の指定に従って変換したあとに，格納代入の規則に従って変換後のデータ型に変換します。

書式指定については，日時書式の場合は「(3)　日時書式に指定できる要素および規則」を，数値書式の場合は「(4)　数値書式に指定できる要素および規則」を参照してください。

格納代入の規則については，「6.2.2　変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」を参照してください。

(b)　数データに変換する場合の規則

■数データを数データに変換する場合

数データを数データに変換する場合，「6.2.2　変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」の「数データの格納代入」で説明している規則が適用されます。

■文字データを数データに変換する場合（数値書式の指定なしの場合）

- 変換対象の文字データ（文字データの前後の半角空白を取り除いた結果）が，数定数の記述形式の規則を満たしている必要があります。数定数の記述形式の規則については，「6.3.2　定数の記述形式」を参照してください。

  ＜変換できる文字データの例＞

  ’219’，’+56’，’-3547’，’-11.35’，’887△△’，’△95△’

  ＜変換できない文字データの例＞

  ’a89’，’77g9’，’33△49’

  （凡例）△：半角空白

- 文字データが半角空白だけで構成されている場合，ナル値を返します。
- 数定数の文字列表現を数値に変換したあとに，変換後のデータ型に変換します。その際，「6.2.2 変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」の「数データの格納代入」で説明している規則が適用されます。

  （例）

  CONVERT(’11.35’,INTEGER) → 11

  いったん文字列’11.35’がDECIMAL 型の数値 11.35 に変換され，そのあとにINTEGER 型の数値に変換されます。その際，数データの格納代入の規則が適用され，この例の場合，小数点以下が切り捨てられます。

■文字データを数データに変換する場合（数値書式の指定ありの場合）

- 変換対象の文字データの形式と，数値書式の指定を合わせてください。なお，変換対象の文字データの前後に半角空白があっても変換できます。また，数値書式の前後に半角空白があっても変換できます。

  （例）

  CONVERT(’△1,234△’,INTEGER,’9,999’) → 1234

  CONVERT(’△△1,234’,INTEGER,’△9,999△’) → 1234

  （凡例）△：半角空白

- 文字データが半角空白だけで構成されている場合，ナル値を返します。
- 数値書式に二重引用符で囲まれた文字列がある場合，二重引用符で囲まれた文字列，および二重引用符で囲まれた文字列の前後にある半角空白を除いた文字データを数値書式に従って数データに変換します。

- 数値書式に従って数値に変換したあとに，変換後のデータ型に変換します。その際，「6.2.2　変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」の「数データの格納代入」で説明している規則が適用されます。

  （例）

  CONVERT('1,000.22',INTEGER,'9,999.99') → 1000

  いったん文字列'1,000.22'がDECIMAL 型の数値1000.22 に変換され，そのあとにINTEGER 型の数値に変換されます。その際，数データの格納代入の規則が適用され，この例の場合，小数点以下が切り捨てられます。

■日時データを数データに変換する場合

西暦 1 年 1 月 1 日からの通算日に変換されます。西暦 1 年 1 月 1 日の場合，通算日は 1 になります。西暦 1 年 1 月 2 日の場合，通算日は 2 になります。

（例）

CONVERT(DATE'0001-01-03',INTEGER) → 3

CONVERT(TIMESTAMP'0001-01-05 11:03:58',INTEGER) → 5

(c)　文字データに変換する場合の規則

文字データへの変換規則（データ長に関する規則）を次の表に示します。

表 8-57　文字データへの変換規則（データ長に関する規則）

変換時の条件	文字データへの変換規則	
	変換対象データのデータ型が文字データまたはバイナリデータの場合	変換対象データのデータ型が左記以外の場合
A ＜ *B* の場合	変換後のデータ型がCHAR 型の場合は，データを左詰めにして，余りに半角空白が格納されます。	
A ＝ *B* の場合	変換されます。	
A ＞ *B* の場合	データを左詰めにして，余った部分を切り捨てます。※1	変換できません。エラーになります。※2

（凡例）

A：変換対象データを文字データに変換した長さ

B：変換後のデータ型のデータ長

注※1

マルチバイト文字の途中で切り捨てが発生した場合，マルチバイト文字の一部分が実行結果の値として返されます。

注※2

変換対象データのデータ型がDOUBLE PRECISION 型またはFLOAT 型で，かつ数値書式を指定していない場合は，*変換後のデータ型*に指定したデータ長に収まるように仮数の小数点以下を切り捨てるため（最近接偶数への丸めを行うため），エラーにはなりません。ただし，仮数の小数点以下すべてを切り捨てても，*変換後のデータ型*に指定したデータ長を超える場合は，エラーになります。

■INTEGER 型，SMALLINT 型，DECIMAL 型，またはNUMERIC 型の数データを文字データに変換する場合（数値書式の指定なしの場合）

- 数データを数定数の形式に変換した結果を文字データとして出力します。その際，数定数で表現可能な形式のうち，最も短い形式で結果が出力されます。

  ただし，DECIMAL 型またはNUMERIC 型のデータの場合は，次のように変換されます。

  ・小数点以下の桁数は，数データのデータ型の位取りと同じになり，末尾からの０は削除されません。

  ・「数データのデータ型の精度＞位取り」の場合，整数部の桁数は０にはなりません。

  ・小数点は必ず付加されます。

  （例）+0025.100 → ’25.100’

  上記のように，＋の符号は削除されます。また，整数部分の先頭からの０は削除されます。

- 変換対象データが0 未満の場合，先頭に負符号（-）が付加されます。

■DOUBLE PRECISION 型またはFLOAT 型の数データを文字データに変換する場合（数値書式の指定なしの場合）

- 数データを浮動小数点数定数の形式に変換した結果を文字データとして出力します。その際，浮動小数点数定数で表現可能な形式のうち，最も短い形式で結果が出力されます。

  （例）

  +1.0000000000000000E+010 → ’1E10’

  +3.2000000000000000E+001 → ’3.2E1’

  +0.1000000000000000E+001 → ’1E0’

  +0.0000000000000000E+000 → ’0E0’

  上記のように，仮数の＋の符号は削除され，小数点以下の末尾からの０も削除されます。指数の＋の符号は削除され，指数の先頭からの０も削除されます。

- 変換対象データが0 未満の場合は，先頭に負符号（-）が付加されます。
- 0 未満の指数には，先頭に負符号（-）が付加されます。

■数データを文字データに変換する場合（数値書式の指定ありの場合）

- 数データを数定数の形式に変換し，指定された数値書式に従って文字データに変換します。

  CONVERT(1000,VARCHAR(6),’LJ$9,999’) → ’$1,000’

- 数データが数値書式に従って変換できない場合は，#を埋め込んだ文字列を返します。数値書式に従って，区切り文字，通貨要素，小数点文字，符号，数要素，二重引用符で囲まれた文字列を#に置き換えます。文字列に全角文字を指定した場合は，文字サイズ（バイト）分#に置き換えます。

  （例）

  CONVERT(1000,CHAR(3),’99’) → ’###’

■日時データを文字データに変換する場合（日時書式の指定なしの場合）

- 日時データを文字データに変換する場合，既定の出力表現の形式に変換されます。DATE 型のデータを文字データに変換する場合は，日付を表す既定の出力表現の形式に変換されます。TIME 型のデータを文字データに変換する場合は，時刻を表す既定の出力表現の形式に変換されます。TIMESTAMP

型のデータを文字データに変換する場合は，時刻印を表す既定の出力表現の形式に変換されます。既定の出力表現については，「6.3.3　既定の文字列表現」を参照してください。

（例）

CONVERT(DATE'2013-06-30',CHAR(10)) → '2013-06-30'

CONVERT(DATE'0001-01-01',CHAR(10)) → '0001-01-01'

CONVERT(TIME'05:33:48.123',CHAR(12)) → '05:33:48.123'

CONVERT(TIMESTAMP'2013-06-30 11:03:58',CHAR(19)) → '2013-06-30 11:03:58'

- 日時データをCHAR(*n*)またはVARCHAR(*n*)に変換する場合，次の条件を満たす必要があります。

変換対象データのデータ型		変換後のデータ長の条件
DATE		$n \geqq 10$
TIME(p)	$p = 0$ の場合	$n \geqq 8$
	$p > 0$ の場合	$n \geqq 9 + p$
TIMESTAMP(p)	$p = 0$ の場合	$n \geqq 19$
	$p > 0$ の場合	$n \geqq 20 + p$

n が上記の長さより短い場合は変換できません。

- DATE 型のデータをCHAR 型に変換する場合，変換後のデータのデータ長が 11 バイト以上のときは，データを左詰めにして，余りに半角空白が格納されます。

（例）

CONVERT(DATE'2013-06-30',CHAR(15)) → '2013-06-30△△△△△'

（凡例）△：半角空白

- 小数秒精度がp のTIME 型のデータをCHAR 型に変換する場合，変換後のデータのデータ長が $10 + p$ バイト以上（$p = 0$ のときは 9 バイト以上）のときは，データを左詰めにして，余りに半角空白が格納されます。

（例）

CONVERT(TIME'11:03:58.123',CHAR(13)) → '11:03:58.123△'

（凡例）△：半角空白

- 小数秒精度がp のTIMESTAMP 型のデータをCHAR 型に変換する場合，変換後のデータのデータ長が 21 $+ p$ バイト以上（$p = 0$ のときは 20 バイト以上）のときは，データを左詰めにして，余りに半角空白が格納されます。

（例）

CONVERT(TIMESTAMP'2013-06-30 11:03:58',CHAR(20)) → '2013-06-30 11:03:58△'

（凡例）△：半角空白

■日時データを文字データに変換する場合（日時書式の指定ありの場合）

- 指定した日時書式に従って，日時データを文字データに変換します。

- 変換対象の日時データにない日時書式の要素を指定した場合，その要素の部分については文字列が補われます。

  （例）

  CONVERT(DATE'2013-07-30',CHAR(16),'YYYY/MM/DD HH:MI') → '2013/07/30 00:00'

  上記の例の場合，変換対象の日時データがDATE 型のため，時刻の要素がありませんが，日時書式の要素に時刻（HH およびMI）を指定しているため，その部分については'00'が補われます。

  補われる文字列を次の表に示します。

表 8-58　日時書式の要素に補われる文字列

項番	日時書式の要素		補われる文字列
1	時	HH	'00'
2		HH24	
3		HH12	'12'
4	午前/午後	AM	'AM'
5		A.M.	'A.M.'
6		PM	'AM'
7		P.M.	'A.M.'
8		AMN	'午前'
9		PMN	
10	分	MI	'00'
11	秒	SS	'00'
12		SSSSS	'00000'
13	小数秒	FF1	対象データの小数秒精度より右側の桁に 0 が補われます。
14		FF2	
15		FF3	
16		FF4	
17		FF5	
18		FF6	
19		FF7	
20		FF8	
21		FF9	
22		FF10	
23		FF11	

項番	日時書式の要素		補われる文字列
24		FF12	

- 日時データをCHAR 型に変換する際，*変換後のデータ型*に指定したデータ長が変換後のデータ長より長い場合は，データを左詰めにして，余りに半角空白が格納されます。

  （例）

  CONVERT(DATE'2013-07-30',CHAR(12),'YYYY/MM/DD') → '2013/07/30△△'

  （凡例）△：半角空白

- 日時データを文字データに変換する際，日時書式の要素にMON，MONTH，DAY，またはDY を指定した場合，1 文字目と 2 文字目に指定した文字が大文字であるか小文字であるかによって，変換後の文字列が次のように変わります。

  ・1 文字目が小文字の場合，変換後の文字列は，すべて小文字になります。

  ・1 文字目が大文字で 2 文字目が小文字の場合，変換後の文字列は，1 文字目が大文字，2 文字目以降は小文字になります。

  ・1 文字目と 2 文字目が大文字の場合，変換後の文字列は，すべて大文字になります。

  例を次に示します。

日時書式の要素の指定	変換後の文字列
mon	'jan'
Mon	'Jan'
MON またはMOn	'JAN'

  上記は，1 月の場合を例にしています。

- 日時書式にFF1～FF11 を指定した場合，「変換対象データの小数秒の桁数＞日時書式の要素の小数秒の桁数」のときは，日時書式の要素の小数秒の桁数を超えた部分は切り捨てられます。

  （例）

  CONVERT(TIME'15:16:17.123456',CHAR(9),'HHMISS.FF2') → '151617.12'

■バイナリデータを文字データに変換する場合

- データ型が変換されるだけで，データの内容（文字コード自体）は変換されません。

  （例）

  CONVERT(X'61626364',CHAR(4)) → 'abcd'

- 「*変換対象データ*のデータ長＞*変換後のデータ型*のデータ長」の場合，末尾が切り捨てられます。

  （例）

  CONVERT(X'6162636<u>4</u>',CHAR(3)) → 'abc'

  下線部分が切り捨てられます。

- 「*変換対象データ*のデータ長＜*変換後のデータ型*のデータ長」の場合，末尾に半角空白が格納されます。

（例）

CONVERT(X'61626364',CHAR(5)) → 'abcd△'

（凡例）△：半角空白

(d)　日時データに変換する場合の規則

■INTEGER 型またはSMALLINT 型の数データを日時データに変換する場合

- 西暦 1 年 1 月 1 日を起点として，DATE 型またはTIMESTAMP 型のデータに変換されます。

  （例）

  CONVERT(2,DATE) → DATE'0001-01-02'

- TIMESTAMP 型の時刻部分は'00:00:00'に変換され，小数秒部分には 0 が補われます。

  （例）

  CONVERT(2,TIMESTAMP(3)) → TIMESTAMP'0001-01-02 00:00:00.000'

- INTEGER 型またはSMALLINT 型のデータが1～3,652,059 の場合に変換できます。範囲外の場合はエラーになります。

■文字データを日時データに変換する場合（日時書式の指定なしの場合）

- 変換対象の文字データ（文字データの前後の半角空白を取り除いた結果）が，日付を表す既定の入力表現の形式に従っている場合に限り，文字データをDATE 型のデータに変換できます。日付を表す既定の入力表現については，「6.3.3　既定の文字列表現」の「(1)　日付を表す既定の文字列表現」の「(a)　既定の入力表現」を参照してください。

  （例）

  CONVERT('2014-07-22△△',DATE) → DATE'2014-07-22'

  ＜変換できる文字データの例＞

  '2014-06-30', '0001-01-02', '△△2014-07-30', '△2014/07/30△△'

  ＜変換できない文字データの例＞

  '2013△06△30', '2013.06.30'

  （凡例）△：半角空白

- 変換対象の文字データ（文字データの前後の半角空白を取り除いた結果）が，時刻を表す既定の入力表現の形式に従っている場合に限り，文字データをTIME 型のデータに変換できます。時刻を表す既定の入力表現については，「6.3.3　既定の文字列表現」の「(2)　時刻を表す既定の文字列表現」の「(a)　既定の入力表現」を参照してください。

  （例）

  CONVERT('△19:46:23.123456',TIME(6)) → TIME'19:46:23.123456'

  ＜変換できる文字データの例＞

  '18:05:22', '10:21:44.123', '△△10:21:44.123456△'

  ＜変換できない文字データの例＞

  '18△05△22', '10:21:44△123456'

（凡例）△：半角空白

- 変換対象の文字データ（文字データの前後の半角空白を取り除いた結果）が，時刻印を表す既定の入力表現の形式に従っている場合に限り，文字データをTIMESTAMP 型のデータに変換できます。時刻印を表す既定の入力表現については，「6.3.3　既定の文字列表現」の「(3)　時刻印を表す既定の文字列表現」の「(a)　既定の入力表現」を参照してください。

  （例）

  CONVERT('2014/08/02 11:03:58.123456△',TIMESTAMP(6)) → TIMESTAMP'2014-08-02 11:03:58.123456'

  ＜変換できる文字データの例＞

  '2014-06-30 11:03:58', '2014/07/30 11:03:58.123', '△2014/07/30 11:03:58.123456789△△'

  ＜変換できない文字データの例＞

  '2014-06-30 11-03-58', '2014/07/30 11:03:58:123456'

  （凡例）△：半角空白

- 「変換対象の文字データの小数秒の桁数＞*変換後のデータ型*の小数秒の桁数」の場合，*変換後のデータ型*の小数秒の桁数を超えた部分の小数秒は切り捨てられます。

  （例）

  CONVERT('19:46:23.123456',TIME(3)) → TIME'19:46:23.123'

- 「変換対象の文字データの小数秒の桁数＜*変換後のデータ型*の小数秒の桁数」の場合，足りない小数秒部分に 0 が補われます。

  （例）

  CONVERT('2014-08-02 11:03:58.123',TIMESTAMP(9)) → TIMESTAMP'2014-08-02 11:03:58.123000000'

- 文字データが半角空白だけで構成されている場合，ナル値を返します。

■文字データを日時データに変換する場合（日時書式の指定ありの場合）

- 文字データをDATE 型に変換する場合は，日時書式に年，月，日の要素を指定してください。それ以外の要素（例えば，時）を指定しても結果には反映されません。日時書式の要素については，「表 8-52　同じ意味を持つ日時書式の要素」を参照してください。
- 文字データをTIME 型に変換する場合は，日時書式に時，分，秒の要素を指定してください。それ以外の要素（例えば，日）を指定しても結果には反映されません。日時書式の要素については，「表 8-52　同じ意味を持つ日時書式の要素」を参照してください。
- 文字データをTIMESTAMP 型に変換する場合は，日時書式に年，月，日，時，分，秒の要素を指定してください。日時書式の要素については，「表 8-52　同じ意味を持つ日時書式の要素」を参照してください。
- 変換対象の文字データの先頭と末尾の連続する半角空白を取り除いた結果を，日時書式に従って日時データに変換します。また，日時書式中の文字データの先頭と末尾の連続する半角空白に該当する部分は無視されます。そのため，次のケースはエラーになりません。

  （例）

CONVERT('△19△46△23△△△',TIME(12),'"△△△"FM△HH△MI△SS△FF△')

→ TIME'19:46:23.000000000000'

（凡例）△：半角空白

メモ

上記の例の場合，半角空白の扱いは次のようになります。

1. 変換対象の文字データの先頭と末尾の連続する半角空白は無視されます。

   '△19△46△23△△△' → '19△46△23'

2. 日時書式の先頭と末尾の連続する半角空白は無視されます。

   '"△△△"FM△HH△MI△SS△FF△' → 'FM△HH△MI△SS△FF'

   "△△△"部分は，文字データの先頭の連続する半角空白に該当するため，無視されます。最後の半角空白は，文字データの末尾の連続する半角空白に該当するため，無視されます。

3. FM に対応する文字はないため，FM の後の半角空白は文字データの先頭の連続する半角空白に該当し，無視されます。

   'FM△HH△MI△SS△FF' → 'FMHH△MI△SS△FF'

4. 変換対象の文字データに小数秒がないため，FF の前の半角空白は文字データの末尾の連続する半角空白に該当し，無視されます。

   'FMHH△MI△SS△FF' → 'FMHH△MI△SSFF'

   小数秒が指定されている場合は，FF の前の半角空白は文字データの末尾の連続する半角空白に該当しません。

- 文字データが半角空白だけで構成されている場合，ナル値を返します。
- 日時書式に小数秒の要素を指定しないで，小数秒精度が3 以上のTIME 型またはTIMESTAMP 型のデータに変換する場合，変換後の小数秒の値は 0 になります。

  （例）

  CONVERT('151617',TIME(3),'HHMISS') → TIME'15:16:17.000'
- 日時書式の要素を大文字または小文字のどちらで指定しても，同じように変換されます。また，変換対象データが大文字または小文字のどちらでも，同じように変換されます。ただし，二重引用符（"）で囲んだ文字列については，大文字，小文字が区別されます。
- 日時書式にFF，またはFF1～FF12 のどれかを指定した場合，日時書式に対応する文字列中の数字を切り出して変換します。このとき，数字以外が出現するか，または日時書式の各要素に対応する長さに達するまで数字を切り出して変換します。文字列中の数字が，日時書式の各要素に対応する長さよりも短い場合，足りない部分は0 に変換されます。

  （例）

  CONVERT('151617.12',TIME(3),'HHMISS.FF3') → TIME'15:16:17.120'

- 日時書式にFF，またはFF1～FF12のどれかを指定し，かつ「文字データの小数秒の桁数＜日時データの小数秒精度」の場合は，小数秒の桁数が足りない部分は0に変換されます。

  （例）

  CONVERT('151617.123',TIME(6),'HHMISS.FF3') → TIME'15:16:17.123000'

- 日時書式にFF，またはFF1～FF12のどれかを指定し，かつ「文字データの小数秒の桁数＞日時データの小数秒精度」の場合は，日時データの小数秒精度を超える部分は変換されません。

  （例）

  CONVERT('151617.123456',TIME(3),'HHMISS.FF6') → TIME'15:16:17.123'

■日時データを日時データに変換する場合

日時データを日時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 8-59　日時データを日時データに変換する場合の変換規則

変換対象データのデータ型	変換後のデータ型の指定	変換規則
DATE	DATE	変換されません。
	TIMESTAMP(*p2*)	• 時刻部分は'00:00:00'に変換されます。 • 小数秒部分には0が補われます。
TIME(*p1*)	TIME(*p2*)	• *p1* ＝*P2* の場合 変換されません。 • *p1* ＞*p2* の場合 *p2* を超えた部分の小数秒は切り捨てられます。 • *p1* ＜*p2* の場合 足りない小数秒部分には0が補われます。
TIMESTAMP(*p1*)	DATE	日付部分だけが変換されます。
	TIMESTAMP(*p2*)	• *p1* ＝*P2* の場合 変換されません。 • *p1* ＞*p2* の場合 *p2* を超えた部分の小数秒は切り捨てられます。 • *p1* ＜*p2* の場合 足りない小数秒部分には0が補われます。

（凡例）

p1，*p2*：小数秒精度

(e)　バイナリデータに変換する場合の規則

■文字データをバイナリデータに変換する場合

- データ型が変換されるだけで，データの内容（文字コード自体）は変換されません。

  （例）

CONVERT('abcd',BINARY(4)) → X'61626364'

- 「*変換対象データのデータ長*>*変換後のデータ型のデータ長*」の場合，末尾が切り捨てられます。

  （例）

  CONVERT('abcd',BINARY(3)) → X'616263'

  下線部分が切り捨てられます。

  マルチバイト文字の途中で切り捨てが発生した場合，マルチバイト文字の一部分が実行結果の値として返されます。

- 「*変換対象データのデータ長*<*変換後のデータ型のデータ長*」の場合，末尾にX'00'が格納されます。

  （例）

  CONVERT('abcd',BINARY(5)) → X'6162636400'

■バイナリデータをバイナリデータに変換する場合

- 「*変換対象データのデータ長*>*変換後のデータ型のデータ長*」の場合，末尾が切り捨てられます。

  （例）

  CONVERT(X'6162636400',BINARY(3)) → X'616263'

  下線部分が切り捨てられます。

  マルチバイト文字の途中で切り捨てが発生した場合，マルチバイト文字の一部分が実行結果の値として返されます。

- 「*変換対象データのデータ長*<*変換後のデータ型のデータ長*」の場合，末尾にX'00'が格納されます。

  （例）

  CONVERT(X'61626364',BINARY(5)) → X'6162636400'

(6) 例題

例題 1

表T1 のC2 列のデータをCHAR 型からDATE 型に変換し，C2 列が 2013 年 7 月 20 日の行を検索します。C2 列には，日時を示すCHAR 型のデータが「月/日/年」の形式で格納されています。

```
SELECT * FROM "T1"
    WHERE CONVERT("C2",DATE,'MM/DD/YYYY')=DATE'2013-07-20'
```

表T1

C1列 CHAR	C2列 CHAR
A10101	06/14/2013
A15014	07/20/2013
A31399	07/11/2013

検索結果

A15014	07/20/2013

例題 2

表T1 のC1 列のデータがA10101 の行を検索し，該当行のC2 列のデータをINTEGER 型からCHAR 型に変換します。変換の際，先頭に通貨記号$を付けて，3 桁区切りのコンマも付けます。

```
SELECT ”C1”,CONVERT(”C2”,CHAR(13),’$999,999,999’) FROM ”T1”
    WHERE ”C1”=’A10101’
```

表T1

C1列 CHAR	C2列 INTEGER
A10101	123000000
A15014	555550000
A31399	277965400

検索結果

A10101	△$123,000,000

注　△は半角空白を意味しています。

例題 3

表T1 のC2 列には，通貨記号$と 3 桁区切りのコンマが付いた価格を表すCHAR 型のデータが格納されています。C2 列のデータをCHAR 型からINTEGER 型に変換し，3 割引した価格が$1,000 以上の行を検索します。

```
SELECT * FROM ”T1”
    WHERE CONVERT(”C2”,INTEGER,’$9,999’)*0.7>=1000
```

表T1

C1列 CHAR	C2列 CHAR
A10101	$1,000
A15014	$2,000
A31399	$3,000

検索結果

A15014	$2,000
A31399	$3,000

8.13.6　HEX

バイナリデータを 16 進文字列表現（'0'～'9', 'A'～'F’で構成された文字データ）に変換します。

(1)　指定形式

スカラ関数HEX：：＝HEX(*対象データ*)

対象データ：：＝*値式*

(2) 指定形式の説明

対象データ：

バイナリデータを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データには，BINARY 型またはVARBINARY 型のデータを指定してください。
- 対象データには，？パラメタを単独で指定できません。
- 定義長が 16,001 バイト以上のバイナリデータは，対象データに指定できません。

スカラ関数HEX の実行結果の例を次に示します。

（例）

HEX(B'10100100') → 'A4'

HEX(X'1234') → '1234'

(3) 規則

1. 実行結果のデータ型とデータ長を次の表に示します。

表 8-60 スカラ関数 HEX の実行結果のデータ型とデータ長

対象データのデータ型とデータ長			実行結果のデータ型とデータ長		
データ型	定義長	実長	データ型	定義長	実長
BINARY(*n*)	1≦*n*≦16,000	該当しません。	VARCHAR	*n*×2	*n*×2
VARBINARY(*n*)	1≦*n*≦16,000	*r*			*r*×2

（凡例）

n：対象データの定義長

r：対象データの実長

2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。
4. 対象データの実長が 0 バイトの場合，実行結果は実長 0 バイトのデータになります。

8.14　NULL 評価関数

ここでは，NULL 評価関数の機能と指定形式について説明します。

8.14.1　COALESCE

指定した対象データを*対象データ 1*，*対象データ 2*，…の順に評価し，ナル値でない最初の値を返します。

(1)　指定形式

```
スカラ関数COALESCE：：＝COALESCE(対象データ1〔,対象データ2〕…)

  対象データ1：：＝値式
  対象データ2：：＝値式
```

(2)　指定形式の説明

対象データ 1，*対象データ 2*…：

対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，配列データを指定できません。
- 対象データには，比較できるデータ型のデータを指定してください。比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。ただし，次のデータは比較できません。
  - DATE 型のデータと，日付を表す既定の入力表現の文字データ
  - TIME 型のデータと，時刻を表す既定の入力表現の文字データ
  - TIMESTAMP 型のデータと，時刻印を表す既定の入力表現の文字データ
- *対象データ 1* には，？パラメタを単独で指定できません。
- *対象データ 2* 以降に？パラメタを指定した場合，*対象データ* 1 のデータ型が？パラメタのデータ型として仮定されます。
- 対象データは 255 個まで指定できます。

(3)　規則

1. 実行結果のデータ型とデータ長は，「7.21.2　値式の結果のデータ型」で説明している規則に従って決まります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

3. すべての対象データがナル値の場合，実行結果はナル値になります。

4. COALESCE(*対象データ 1*,*対象データ 2*)は，次のCASE 式と同じになります。

```
CASE
   WHEN 対象データ1 IS NOT NULL THEN 対象データ1
   ELSE 対象データ2
END
```

5. COALESCE(*対象データ 1*,*対象データ 2*,…,*対象データ n*)は，次の CASE 式と同じになります（n は 3 以上)。

```
CASE
   WHEN 対象データ1 IS NOT NULL THEN 対象データ1
   ELSE COALESCE(対象データ2,…,対象データn)
END
```

(4) 例題

例題

表T1 のC1 列～C3 列の値に対して，スカラ関数COALESCE を実行します。

```
SELECT COALESCE(”C1”,”C2”,”C3”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 INTEGER	C3列 INTEGER
10	20	30
10	NULL	30
NULL	20	30
NULL	NULL	30
NULL	NULL	NULL

検索結果

10
10
20
30
NULL

8.14.2 ISNULL

指定した対象データを*対象データ 1*，*対象データ 2* の順に評価し，ナル値でない最初の値を返します。

メモ

スカラ関数ISNULL とスカラ関数NVL に機能差はありません。

(1)　指定形式

スカラ関数ISNULL：：＝ISNULL(*対象データ1*,*対象データ2*)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2)　指定形式の説明

対象データ 1，*対象データ 2*：

対象データを指定します。

指定規則を次に示します。

- *対象データ 1* および*対象データ 2* は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *対象データ 1* および*対象データ 2* には，配列データを指定できません。
- *対象データ 1* と*対象データ 2* には，比較できるデータ型のデータを指定してください。比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。
- *対象データ 1* のデータ型がDATE 型，TIME 型，またはTIMESTAMP 型の場合，既定の入力表現の形式に従っている文字列定数を*対象データ 2* に指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- *対象データ 1* には，？パラメタを単独で指定できません。
- *対象データ 2* に？パラメタを指定した場合，*対象データ 1* のデータ型が？パラメタのデータ型として仮定されます。
- *対象データ 2* には，*対象データ 1* のデータ型に格納代入できる値を指定する必要があります。格納代入については，「6.2.2　変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」を参照してください。

(3)　規則

1. 実行結果のデータ型とデータ長は，*対象データ 1* のデータ型とデータ長になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *対象データ 1* および*対象データ 2* がナル値の場合，実行結果はナル値になります。
4. *対象データ 1* がナル値の場合，*対象データ 2* の値を*対象データ 1* のデータ型とデータ長に変換します。

(4)　例題

例題

表T1 のC1 列，C2 列の値に対して，スカラ関数ISNULL を実行します。

```
SELECT ISNULL("C1","C2") FROM "T1"
```

表T1

C1列 DECIMAL(3,1)	C2列 INTEGER
10.5	20
10.5	NULL
NULL	20
NULL	NULL

検索結果

10.5
10.5
20.0
NULL

8.14.3 NULLIF

対象データ1 と*対象データ2* を比較した結果，等しい場合はナル値を返し，等しくない場合は*対象データ1* を返します。

(1) 指定形式

```
スカラ関数NULLIF：：＝NULLIF(対象データ1,対象データ2)

  対象データ1：：＝値式
  対象データ2：：＝値式
```

(2) 指定形式の説明

対象データ1，*対象データ2*：

比較する対象データを指定します。

指定規則を次に示します。

- *対象データ1* および*対象データ2* は，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- *対象データ1* および*対象データ2* には，配列データを指定できません。
- *対象データ1* と*対象データ2* には，比較できるデータ型のデータを指定してください。比較できるデータ型については，「6.2.2 変換，代入，比較できるデータ型」の「(1) 比較できるデータ型」を参照してください。ただし，次のデータは比較できません。
  - ・DATE 型のデータと，日付を表す既定の入力表現の文字データ
  - ・TIME 型のデータと，時刻を表す既定の入力表現の文字データ
  - ・TIMESTAMP 型のデータと，時刻印を表す既定の入力表現の文字データ

- *対象データ 1* と*対象データ 2* の両方に，？パラメタを単独で指定できません。
- *対象データ 1* または*対象データ 2* のどちらかに？パラメタを指定した場合，？パラメタを指定していない対象データのデータ型が，？パラメタのデータ型として仮定されます。

(3) 規則

1. 実行結果のデータ型とデータ長は，「7.21.2 値式の結果のデータ型」で説明している規則に従って決まります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *対象データ 1* がナル値の場合，実行結果はナル値になります。
4. NULLIF(*対象データ 1*,*対象データ 2*)は，次のCASE 式と同じになります。

```
CASE
   WHEN 対象データ1 = 対象データ2 THEN NULL
   ELSE 対象データ1
END
```

(4) 例題

例題

表T1 のC1 列とC2 列の値を比較します。

```
SELECT NULLIF(”C1”,”C2”) FROM ”T1”
```

表T1

C1列 INTEGER	C2列 INTEGER
10	10
10	20

検索結果

NULL
10

8.14.4 NVL

指定した対象データを*対象データ 1*，*対象データ 2* の順に評価し，ナル値でない最初の値を返します。

メモ

スカラ関数NVL とスカラ関数ISNULL に機能差はありません。

(1)　指定形式

スカラ関数NVL ：：＝NVL(*対象データ1*,*対象データ2*)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2)　指定形式の説明

対象データ 1，*対象データ 2*：

対象データを指定します。

指定規則を次に示します。

- *対象データ 1* および*対象データ 2* は，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- *対象データ 1* および*対象データ 2* には，配列データを指定できません。
- *対象データ 1* と*対象データ 2* には，比較できるデータ型のデータを指定してください。比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。
- *対象データ 1* のデータ型がDATE 型，TIME 型，またはTIMESTAMP 型の場合，既定の入力表現の形式に従っている文字列定数を*対象データ 2* に指定できます。既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- *対象データ 1* には，？パラメタを単独で指定できません。
- *対象データ 2* に？パラメタを指定した場合，*対象データ 1* のデータ型が？パラメタのデータ型として仮定されます。
- *対象データ 2* には，*対象データ 1* のデータ型に格納代入できる値を指定する必要があります。格納代入については，「6.2.2　変換，代入，比較できるデータ型」の「(2)　格納代入できるデータ型」を参照してください。

(3)　規則

1. 実行結果のデータ型とデータ長は，*対象データ 1* のデータ型とデータ長になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. *対象データ 1* および*対象データ 2* がナル値の場合，実行結果はナル値になります。
4. *対象データ 1* がナル値の場合，*対象データ 2* の値を*対象データ 1* のデータ型とデータ長に変換します。

(4)　例題

例題

表T1 のC1 列，C2 列の値に対して，スカラ関数NVL を実行します。

```
SELECT NVL("C1","C2") FROM "T1"
```

表T1

C1列 DECIMAL(3,1)	C2列 INTEGER
10.5	20
10.5	NULL
NULL	20
NULL	NULL

検索結果

10.5
10.5
20.0
NULL

8.15　情報取得関数

ここでは，情報取得関数の機能と指定形式について説明します。

8.15.1　LENGTHB

対象データの長さをバイト数で返します。

(1)　指定形式

スカラ関数LENGTHB：：＝LENGTHB(*対象データ*)

対象データ：：＝*値式*

(2)　指定形式の説明

対象データ：

長さを求める対象データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。
- 対象データには，配列データを指定できません。
- 対象データには，？パラメタを単独で指定できません。

(3)　規則

1. 実行結果のデータ型はINTEGER 型になります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 対象データがナル値の場合，実行結果はナル値になります。
4. 対象データのデータ型ごとの実行結果の値を次の表に示します。

表 8-61　対象データのデータ型ごとの実行結果の値

項番	対象データのデータ型		実行結果の値（バイト）
1	INTEGER		8
2	SMALLINT		4
3	DECIMAL または NUMERIC	精度が 1～4 桁の場合	2
4		精度が 5～8 桁の場合	4
5		精度が 9～16 桁の場合	8

項番	対象データのデータ型		実行結果の値（バイト）
6		精度が 17～38 桁の場合	16
7	DOUBLE PRECISION またはFLOAT		8
8	CHAR(*n*)		*n*
9	VARCHAR		実長
10	DATE		4
11	TIME(*p*)		3 ＋ ↑*p*÷2↑
12	TIMESTAMP(*p*)		7 ＋ ↑*p*÷2↑
13	BINARY(*n*)		*n*
14	VARBINARY		実長

(4) 例題

例題 1（対象データが文字データの場合）

表T1 のC1 列（VARCHAR 型）のデータの実長を求めます。

なお，使用している文字コードは，Unicode（UTF-8）とします。

```
SELECT LENGTHB(”C1”) FROM ”T1”
```

表T1

C1列 VARCHAR
ABC
ⅠⅡⅢ
ABCⅠⅡⅢ

検索結果

3
9
12

例題 2（対象データがバイナリデータの場合）

表T1 のC1 列（VARBINARY(5)）とC2 列（BINARY(5)）の，各行のデータの実長を求めます。

```
SELECT LENGTHB(”C1”), LENGTHB(”C2”) FROM ”T1”
```

表T1

C1列 VARBINARY(5)	C2列 BINARY(5)
X'ABCD'	X'ABCD000000'
X'ABCDEF'	X'ABCDEF0000'
X'ABCDEF10'	X'ABCDEF1000'

検索結果

2	5
3	5
4	5

8.16　比較関数

ここでは，比較関数の機能と指定形式について説明します。

8.16.1　DECODE

対象データと比較データを順次比較し，一致した場合は対応する返却値を返します。対象データとすべての比較データが一致しない場合は，既定返却値を返します。

複数の比較データを指定した場合は，最初に一致した比較データに対応する返却値を返します。

(1)　指定形式

```
スカラ関数DECODE：：＝DECODE(対象データ,比較データ,返却値
                          〔,比較データ,返却値〕…
                          〔,既定返却値〕)

  対象データ：：＝ {値式 | NULL}
  比較データ：：＝ {値式 | NULL}
  返却値：：＝ {値式 | NULL}
  既定返却値：：＝ {値式 | NULL}
```

(2)　指定形式の説明

対象データ：

対象データを指定します。対象データは，値式の形式で指定するか，またはNULL を指定します。値式については，「7.21　値式」を参照してください。

比較データ：

比較データを指定します。

指定規則を次に示します。

- 比較データは，値式の形式で指定するか，またはNULL を指定します。値式については，「7.21　値式」を参照してください。
- 最初に指定する*比較データ*には，NULL を指定できません。
- 最初に指定する*比較データ*には，？パラメタを単独で指定できません。
- 2 つ目以降の*比較データ*に？パラメタを単独で指定した場合，？パラメタのデータ型には，最初の*比較データ*のデータ型が仮定されます。

返却値：

対象データが比較データと一致したときの返却値を指定します。

指定規則を次に示します。

- 返却値は，値式の形式で指定するか，またはNULL を指定します。値式については，「7.21　値式」を参照してください。
- 最初に指定する返却値には，NULL を指定できません。
- 最初に指定する返却値には，？パラメタを単独で指定できません。
- 2 つ目以降の返却値に？パラメタを単独で指定した場合，？パラメタのデータ型には，最初の返却値のデータ型が仮定されます。

既定返却値：

対象データとすべての比較データが一致しないときの返却値（既定返却値）を指定します。*既定返却値*の指定を省略した場合は，既定返却値としてNULL が指定されたと仮定されます。

指定規則を次に示します。

- 既定返却値は，値式の形式で指定するか，またはNULL を指定します。値式については，「7.21　値式」を参照してください。
- *既定返却値*に？パラメタを単独で指定した場合，？パラメタのデータ型には，最初の返却値のデータ型が仮定されます。

(3)　規則

1. *対象データ*，*比較データ*，*返却値*，および*既定返却値*には，数データ，文字データまたは日時データを指定してください。
2. *対象データ*，*比較データ*，*返却値*，または*既定返却値*に指定するNULL は，ナル値を意味しています。
3. *対象データ*および*比較データ*には，比較できるデータ型を指定してください（NULL の指定を除く）。比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

   ただし，*対象データ*および*比較データ*が，文字データまたは日時データである場合は，次の表を基にデータ型の組み合わせを指定してください。

表 8-62　スカラ関数 DECODE での，対象データおよび比較データに指定できるデータ型の組み合わせ

対象データ		比較データ						
		文字データ				日時データ		
		日付データの既定の入力表現である文字列定数	時刻データの既定の入力表現である文字列定数	時刻印データの既定の入力表現である文字列定数	そのほかのデータ	日付データ	時刻データ	時刻印データ
文字データ	日付データの既定の入力表現である文字列定数	○	○	○	○	×	×	×

対象データ		比較データ						
		文字データ				日時データ		
		日付データの既定の入力表現である文字列定数	時刻データの既定の入力表現である文字列定数	時刻印データの既定の入力表現である文字列定数	そのほかのデータ	日付データ	時刻データ	時刻印データ
	時刻データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	時刻印データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	そのほかのデータ	○	○	○	○	×	×	×
日時データ	日付データ	○	×	○	×	○	×	○
	時刻データ	×	○	×	×	×	○	×
	時刻印データ	○	×	○	×	○	×	○

（凡例）

○：指定できます。

×：指定できません。

4. *返却値*および*既定返却値*には，比較できるデータ型を指定してください（NULL の指定を除く）。比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

   ただし，*返却値*および*既定返却値*が，文字データまたは日時データである場合は，次の表を基にデータ型の組み合わせを指定してください。

表 8-63　スカラ関数 DECODE での，返却値および既定返却値に指定できるデータ型の組み合わせ

返却値		既定返却値，または返却値※						
		文字データ				日時データ		
		日付データの既定の入力表現である文字列定数	時刻データの既定の入力表現である文字列定数	時刻印データの既定の入力表現である文字列定数	そのほかのデータ	日付データ	時刻データ	時刻印データ
文字データ	日付データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	時刻データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	時刻印データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	そのほかのデータ	○	○	○	○	×	×	×
日時データ	日付データ	×	×	×	×	○	×	○
	時刻データ	×	×	×	×	×	○	×
	時刻印データ	×	×	×	×	○	×	○

（凡例）

○：指定できます。

×：指定できません。

注※

*返却値*を複数指定していて，すべての*返却値*が文字データまたは日時データである場合，それぞれの*返却値*に指定できるデータ型の組み合わせは，「表 8-63　スカラ関数 DECODE での，返却値および既定返却値に指定できるデータ型の組み合わせ」のとおりになります。

5. *比較データ*および*返却値*の組は，127 個まで指定できます。
6. 実行結果のデータ型とデータ長は，*返却値*および*既定返却値*の結果のデータ型によって，「7.21.2　値式の結果のデータ型」で説明している規則に従って決まります。

なお，*返却値*および*既定返却値*のNULL の指定は，実行結果のデータ型とデータ長の決定に関係しません。

7. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。

8. 対象データがナル値の場合，比較データにNULL を指定したときに，対応する返却値が返されます。

(4) 例題

例題 1

表T1 のC2 列に格納されている国名の略号を次のように変換します。

- JPN → Japan
- IND → India
- ナル値 → NODATA
- その他 → Other

```
SELECT "C1",DECODE("C2",'JPN','Japan','IND','India',NULL,'NODATA','Other')
    FROM "T1"
```

表T1

C1列	C2列
N001	JPN
N003	JPN
N010	IND
N050	CHN
N085	IND
N100	NULL

検索結果

N001	Japan
N003	Japan
N010	India
N050	Other
N085	India
N100	NODATA

例題 2

社員表（EMPLIST）を検索して次に示す値を求めます。

- 部署コード（SCODE）ごとに，所属する男性と女性の人数を求める

```
SELECT "SCODE",SUM(DECODE("SEX",'M',1,0)) AS "Men",
               SUM(DECODE("SEX",'F',1,0)) AS "Women"
    FROM "EMPLIST"
    GROUP BY "SCODE"
```

検索結果

SCODE	Men	Women
S001	12	5
S002	21	18
S003	19	33

8.16.2 GREATEST

指定した対象データの値のうち，最大値を返します。

数データ同士の比較だけではなく，文字データ同士の比較，日時データ同士の比較などもできます。

(1) 指定形式

スカラ関数GREATEST：：＝GREATEST(*対象データ1*〔,*対象データ2*〕…)

対象データ1：：＝*値式*
対象データ2：：＝*値式*

(2) 指定形式の説明

対象データ 1，*対象データ 2*，…：

最大値を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データは 255 個まで指定できます。
- 対象データに指定できるデータ型は，数データ，文字データ，または日時データです。
- *対象データ 1*，*対象データ 2*，…には，比較できるデータ型を指定してください。比較できるデータ型については，「6.2.2 変換，代入，比較できるデータ型」の「(1) 比較できるデータ型」を参照してください。ただし，次のデータは比較できません。

  ・DATE 型のデータと，日付を表す既定の入力表現の文字データ

  ・TIME 型のデータと，時刻を表す既定の入力表現の文字データ

  ・TIMESTAMP 型のデータと，時刻印を表す既定の入力表現の文字データ

  既定の入力表現については，「6.3.3 既定の文字列表現」を参照してください。
- *対象データ 1* には，？パラメタを単独で指定できません。
- *対象データ 2* 以降に？パラメタを単独で指定した場合，？パラメタのデータ型には，*対象データ 1* のデータ型が仮定されます。

(3) 規則

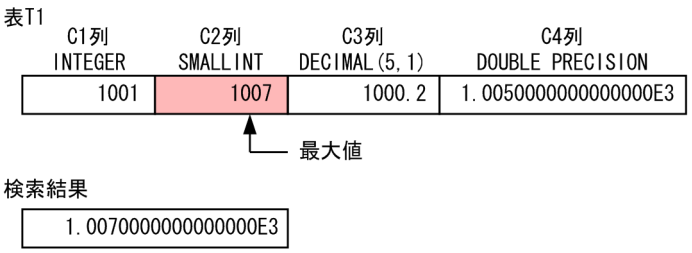
1. 実行結果のデータ型とデータ長は，「7.21.2 値式の結果のデータ型」で説明している規則に従って決まります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 指定した対象データのどれかがナル値の場合，実行結果はナル値になります。

(4) 例題

例題

表T1 のC1 列～C4 列の値を比べて，その中の最大値を求めます。

```
SELECT GREATEST(”C1”,”C2”,”C3”,”C4”) FROM ”T1”
```



8.16.3 LEAST

指定した対象データの値のうち，最小値を返します。

数データ同士の比較だけではなく，文字データ同士の比較，日時データ同士の比較などもできます。

(1) 指定形式

```
スカラ関数LEAST：：＝LEAST(対象データ1〔,対象データ2〕…)

  対象データ1：：＝値式
  対象データ2：：＝値式
```

(2) 指定形式の説明

対象データ 1，*対象データ 2*，…：

最小値を求める数データを指定します。

指定規則を次に示します。

- 対象データは，値式の形式で指定します。値式については，「7.21 値式」を参照してください。
- 対象データは 255 個まで指定できます。

- 対象データに指定できるデータ型は，数データ，文字データ，または日時データです。
- *対象データ 1*，*対象データ 2*，…には，比較できるデータ型を指定してください。比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。ただし，次のデータは比較できません。

  ・DATE 型のデータと，日付を表す既定の入力表現の文字データ

  ・TIME 型のデータと，時刻を表す既定の入力表現の文字データ

  ・TIMESTAMP 型のデータと，時刻印を表す既定の入力表現の文字データ

  既定の入力表現については，「6.3.3　既定の文字列表現」を参照してください。
- *対象データ 1* には，？パラメタを単独で指定できません。
- *対象データ 2* 以降に？パラメタを単独で指定した場合，？パラメタのデータ型には，*対象データ 1* のデータ型が仮定されます。

(3)　規則

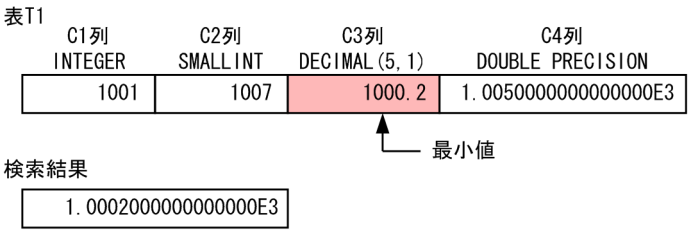
1. 実行結果のデータ型とデータ長は，「7.21.2　値式の結果のデータ型」で説明している規則に従って決まります。
2. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
3. 指定した対象データのどれかがナル値の場合，実行結果はナル値になります。

(4)　例題

例題

表T1 のC1 列～C4 列の値を比べて，その中の最小値を求めます。

```
SELECT LEAST(”C1”,”C2”,”C3”,”C4”) FROM ”T1”
```



8.16.4　LTDECODE

対象データと比較データを順次比較し，対象データの値が比較データの値未満となる場合は，対応する返却値を返します。対象データの値がすべての比較データの値未満とならない場合は，既定返却値を返します。

複数の比較データを指定した場合は，最初に対象データの値が比較データの値未満となる比較データに対応する返却値を返します。

(1) 指定形式

```
スカラ関数LTDECODE : : =LTDECODE(対象データ,比較データ,返却値
                              〔,比較データ,返却値〕…
                              〔,既定返却値〕)

  対象データ : : =値式
  比較データ : : =値式
  返却値 : : = {値式 | NULL}
  既定返却値 : : = {値式 | NULL}
```

(2) 指定形式の説明

対象データ：

対象データを指定します。対象データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。

比較データ：

比較データを指定します。比較データは，値式の形式で指定します。値式については，「7.21　値式」を参照してください。

返却値：

対象データの値が比較データの値未満となるときの返却値を指定します。返却値は，値式の形式で指定するか，またはNULL を指定します。値式については，「7.21　値式」を参照してください。

既定返却値：

対象データの値がすべての比較データの値以上となるときの返却値（既定返却値）を指定します。既定返却値は，値式の形式で指定するか，またはNULL を指定します。値式については，「7.21　値式」を参照してください。

なお，*既定返却値*の指定を省略した場合は，既定返却値としてNULL が指定されたと仮定されます。

(3) 規則

1. *対象データ*，*比較データ*，*返却値*，および*既定返却値*には，数データ，文字データ，または日時データを指定してください。
2. *返却値*，または*既定返却値*に指定するNULL は，ナル値を意味しています。
3. *対象データ*および*比較データ*には，比較できるデータ型を指定してください。比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

   ただし，*対象データ*および*比較データ*が，文字データまたは日時データである場合は，次の表を基にデータ型の組み合わせを指定してください。

表 8-64　スカラ関数 LTDECODE での，対象データおよび比較データに指定できるデータ型の組み合わせ

対象データ		比較データ						
		文字データ				日時データ		
		日付データの既定の入力表現である文字列定数	時刻データの既定の入力表現である文字列定数	時刻印データの既定の入力表現である文字列定数	そのほかのデータ	日付データ	時刻データ	時刻印データ
文字データ	日付データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	時刻データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	時刻印データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	そのほかのデータ	○	○	○	○	×	×	×
日時データ	日付データ	○	×	○	×	○	×	○
	時刻データ	×	○	×	×	×	○	×
	時刻印データ	○	×	○	×	○	×	○

（凡例）

○：指定できます。

×：指定できません。

4.*返却値*および*既定返却値*には，比較できるデータ型を指定してください（NULL の指定を除く）。比較できるデータ型については，「6.2.2　変換，代入，比較できるデータ型」の「(1)　比較できるデータ型」を参照してください。

ただし，*返却値*および*既定返却値*が，文字データまたは日時データである場合は，次の表を基にデータ型の組み合わせを指定してください。

表 8-65　スカラ関数 LTDECODE での，返却値および既定返却値に指定できるデータ型の組み合わせ

返却値		既定返却値，または返却値※						
		文字データ				日時データ		
		日付データの既定の入力表現である文字列定数	時刻データの既定の入力表現である文字列定数	時刻印データの既定の入力表現である文字列定数	そのほかのデータ	日付データ	時刻データ	時刻印データ
文字データ	日付データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	時刻データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	時刻印データの既定の入力表現である文字列定数	○	○	○	○	×	×	×
	そのほかのデータ	○	○	○	○	×	×	×
日時データ	日付データ	×	×	×	×	○	×	○
	時刻データ	×	×	×	×	×	○	×
	時刻印データ	×	×	×	×	○	×	○

（凡例）

○：指定できます。

×：指定できません。

注※

*返却値*を複数指定していて，すべての*返却値*が文字データまたは日時データである場合，それぞれの*返却値*に指定できるデータ型の組み合わせは，「表 8-65　スカラ関数 LTDECODE での，返却値および既定返却値に指定できるデータ型の組み合わせ」のとおりになります。

5. *対象データ*，*比較データ*，*返却値*，および*既定返却値*は，最大 256 個まで指定できます。
6. 実行結果のデータ型とデータ長は，*返却値*および*既定返却値*の結果のデータ型によって，「7.21.2　値式の結果のデータ型」で説明している規則に従って決まります。

なお，*返却値*および*既定返却値*の次に示す指定は，実行結果のデータ型とデータ長の決定に関係しません。

- 単独で指定した？パラメタ
- NULL

7. 実行結果の値は，非ナル値制約なし（ナル値を許す）となります。
8. *比較データ*と*返却値*で 1 組となるように指定してください。*比較データ*に対応する*返却値*は，その*比較データ*の次に指定した値です。
9. *対象データ*または*比較データ*に，単独で指定した？パラメタ以外の値式を，少なくとも 1 つ指定してください。
10. *返却値*または*既定返却値*に，下記以外の値式を，少なくとも 1 つ指定してください。
    - 単独で指定した？パラメタ
    - NULL
11. *対象データ*に？パラメタを単独で指定した場合，*対象データ*の？パラメタのデータ型には，最初の*比較データ*のデータ型が仮定されます。ただし，最初の*比較データ*に？パラメタを単独で指定した場合は，2 つ目以降に指定した？パラメタの単独指定ではない*比較データ*のデータ型が仮定されます。
12. *比較データ*に？パラメタを単独で指定した場合，？パラメタのデータ型には，*対象データ*のデータ型が仮定されます。ただし，*対象データ*に？パラメタを単独で指定している場合は，最初の*比較データ*のデータ型が仮定されます。また，最初の*比較データ*に？パラメタを単独で指定している場合は，2 つ目以降に指定した？パラメタの単独指定ではない*比較データ*のデータ型が仮定されます。

    仮定されるデータ型について，次の指定例を基に説明します。

    - **指定例 1**

      ```
      LTDECODE(?, 10, 'A', 20, 'C')
      ```

      最初の*比較データ*（10）がINTEGER 型のため，*対象データ*の？パラメタのデータ型に，INTEGER 型が仮定されます。

    - **指定例 2**

      ```
      LTDECODE(CURRENT_DATE, ?, 'A', DATE'2017/01/01', 'C')
      ```

      対象データ（CURRENT_DATE）がDATE 型のため，*比較データ*の？パラメタのデータ型に，DATE 型が仮定されます。

    - **指定例 3**

      ```
      LTDECODE(?, ?, 'A', 'B', 'C')
      ```

      最初の*比較データ*に？パラメタを単独で指定していて，2 つ目に指定した*比較データ*（'B'）のデータ型がCHAR(1)となっています。そのため，*対象データ*の？パラメタのデータ型，および*比較データ*の？パラメタのデータ型に，それぞれCHAR(1)が仮定されます。

13. *返却値*または*既定返却値*に？パラメタを単独で指定した場合，？パラメタのデータ型には，このスカラ関数の実行結果のデータ型が仮定されます。

仮定されるデータ型について，次に示す指定例を基に説明します。

- 指定例 1

```
LTDECODE(”C1”, 10, -1, 20, 0, 30, ?)
```

LTDECODE の実行結果のデータ型がINTEGER 型になるため，*返却値*の？パラメタのデータ型にINTEGER 型が仮定されます。

- 指定例 2

```
LTDECODE(”C1”, DATE’2017/01/01’, ’A’, DATE’2017/02/01’,
         ?, DATE’2017/02/01’, ’BB’)
```

LTDECODE の実行結果のデータ型がVARCHAR(2)になるため，*返却値*の？パラメタのデータ型にVARCHAR(2)が仮定されます。

- 指定例 3

```
LTDECODE(”C1”, 10, ’A’, 20, ?, ?)
```

LTDECODE の実行結果のデータ型がVARCHAR(1)になるため，*返却値*および*既定返却値*の？パラメタのデータ型に，それぞれVARCHAR(1)が仮定されます。

14. 対象データの値が比較データの値未満となる（次に示す比較述語が真となる），比較データに対応する返却値を返します。

```
対象データ ＜ 比較データ
```

15. 対象データの値が比較データの値未満となる比較データが複数ある場合，最初の比較データに対応する返却値を返します。

(4) 例題

例題 1

表T1 のC1 列に格納されている値を次のように変換します。そして，変換後の値をC2 列に格納します。

- 0 未満の値 → ナル値
- 1 以上の値 → 2

```
SELECT ”C1”, LTDECODE(”C1”, 0, NULL, 1, ”C1”, 2) ”C2”
    FROM ”T1”
```

表T1

C1列
-1.6
-0.5
0.2
0.5
0.6
1.0
1.2

検索結果

C1列	C2列
-1.6	NULL
-0.5	NULL
0.2	0.2
0.5	0.5
0.6	0.6
1.0	2.0
1.2	2.0

例題 2

社員表（EMPLIST）の身長（HEIGHT）の値を次のように変換します。そして，変換後の値をHEIGHT2 列に格納します。

- 150 未満の値 → 150
- 190 以上の値 → 190

```
SELECT "USERID", LTDECODE("HEIGHT", 150, 150, 190, "HEIGHT", 190) "HEIGHT2"
    FROM "EMPLIST"
```

社員表（EMPLIST）

USERID	HEIGHT
U00001	148
U00002	158
U00003	166
U00004	171
U00005	178
U00006	182
U00007	195

検索結果

USERID	HEIGHT2
U00001	150
U00002	158
U00003	166
U00004	171
U00005	178
U00006	182
U00007	190

例題３

社員表（EMPLIST）を検索して，次に示す値を求めます。

- 社員の年齢（AGE）を基に，世代別の社員数を求める

```
SELECT "GEN", COUNT("GEN") "GEN-NUM"
   FROM "EMPLIST"
     GROUP BY LTDECODE("AGE", 20, '20歳未満'
                              , 30, '20代'
                              , 40, '30代', '40歳以上')
   "GEN"
```

社員表（EMPLIST）

USERID	AGE
U00001	18
U00002	22
U00003	26
U00004	30
U00005	35
U00006	38
U00007	45

検索結果

GEN	GEN-NUM
20歳未満	1
20代	2
30代	3
40歳以上	1

付録

付録 A　SQL 逆引きリファレンス

関連する SQL の構文を目的別に次の表に示します。

表 A-1　目的別の関連する SQL の構文

項番	分類	目的	関連する SQL の構文
1	データの検索	範囲を指定してデータを検索する	BETWEEN 述語
2		指定した複数の値のどれかに一致するデータを検索する	IN 述語
3		特定の文字列が含まれているデータを検索する	LIKE 述語
4		正規表現を使用してデータを検索する	LIKE_REGEX 述語
5		ナル値のデータを検索する	NULL 述語
6		検索結果の重複を排除する	SELECT DISTINCT
7		検索結果を昇順または降順に並べる	ORDER BY 句
8		検索結果の最大行数を指定する	LIMIT 句
9		同じ導出表をSELECT 文中に複数回指定する	WITH 句
10		検索結果の列名を変更する	AS 句
11		分岐条件を指定して検索する	CASE 式
12		複数の表を結合して検索する	結合表
13		副問合せをする	副問合せ
14			EXISTS 述語
15			IN 述語
16			比較述語
17			限定述語
18		複数の表からの問合せ結果の和集合を求める	UNION ALL UNION DISTINCT
19	データの削除	実表内のすべての行を削除する	TRUNCATE TABLE 文
20		実表のチャンク内の全行を削除する	PURGE CHUNK 文
21	データの集計	合計値を求める	一般集合関数SUM
22		最大値を求める	一般集合関数MAX
23		最小値を求める	一般集合関数MIN
24		平均値を求める	一般集合関数AVG
25		行数（件数）を求める	一般集合関数COUNT
26			集合関数COUNT(*)

項番	分類	目的	関連する SQL の構文
27		母集団標準偏差を求める	一般集合関数STDDEV_POP
28		標本標準偏差を求める	一般集合関数STDDEV_SAMP
29		母集団分散を求める	一般集合関数VAR_POP
30		標本分散を求める	一般集合関数VAR_SAMP
31		順序付けされた一連の値の中央値を求める	逆分布関数MEDIAN
32		順序付けされた一連の値のパーセンタイル（百分位数）を求める	逆分布関数PERCENTILE_CONT
33			逆分布関数PERCENTILE_DISC
34		順序付けされた一連の値を連結し，値と値の間に区切り文字列を挿入した文字列を求める	LISTAGG 集合関数
35		値式によって集計された値を，先頭から順に配列要素とする配列データを作成する	ARRAY_AGG 集合関数
36		データの集計時に，集計範囲を設定する	ウィンドウ関数
37		グループごとにデータを集計する	GROUP BY 句
38			HAVING 句
39	文字	検索条件式を満たす文字列が対象データ中に含まれているかどうかを求める	スカラ関数CONTAINS
40		2 つの文字データを連結する	スカラ関数CONCAT
41			連結演算
42		文字データから，特定の文字を削除する	スカラ関数TRIM
43			スカラ関数LTRIM
44			スカラ関数RTRIM
45		文字データの一部分を抽出する	スカラ関数SUBSTR
46			スカラ関数LEFT
47			スカラ関数RIGHT
48		文字データの先頭，または末尾に任意の文字列を埋め込む	スカラ関数LPAD
49			スカラ関数RPAD
50		対象データ中の任意の文字列を置換する	スカラ関数REPLACE
51		文字データ中の任意の文字を置換する	スカラ関数TRANSLATE
52		文字データの文字数を求める	スカラ関数LENGTH
53		対象データ中の任意の文字列を検索し，その文字列の開始位置を求める	スカラ関数INSTR
54		英大文字を英小文字に変換する	スカラ関数LOWER

項番	分類	目的	関連するSQLの構文
55		英小文字を英大文字に変換する	スカラ関数UPPER
56	バイナリデータ	2つのバイナリデータを連結する	スカラ関数CONCAT
57		バイナリデータの一部分を抽出する	スカラ関数SUBSTRB
58		バイナリデータを左ビットシフトした値を求める	スカラ関数BITLSHIFT
59		バイナリデータを右ビットシフトした値を求める	スカラ関数BITRSHIFT
60		2つのバイナリデータのビットごとの論理積を求める	スカラ関数BITAND
61		2つのバイナリデータのビットごとの論理和を求める	スカラ関数BITOR
62		バイナリデータのビットごとの論理否定を求める	スカラ関数BITNOT
63		2つのバイナリデータのビットごとの排他的論理和を求める	スカラ関数BITXOR
64		バイナリデータを2進文字列表現（'0'，'1'で構成された文字データ）に変換する	スカラ関数BIN
65		バイナリデータを16進文字列表現（'0'～'9'，'A'～'F'で構成された文字データ）に変換する	スカラ関数HEX
66	数値計算	余りを求める	スカラ関数MOD
67		絶対値を求める	スカラ関数ABS
68		平方根を求める	スカラ関数SQRT
69		データの符号を求める（データが正の値か，負の値か，0かを調べる）	スカラ関数SIGN
70		最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を求める※	スカラ関数RANDOM
71			スカラ関数RANDOMCURSOR
72			スカラ関数RANDOMROW
73		平均 μ，標準偏差 σ の正規分布に従う擬似乱数を求める	スカラ関数RANDOM_NORMAL
74	数値丸め	数値を丸める	スカラ関数ROUND
75		数値を切り捨てる	スカラ関数TRUNC
76		指定した数値以下で，最大の整数値を求める	スカラ関数FLOOR
77		指定した数値以上で，最小の整数値を求める	スカラ関数CEIL
78	指数・対数計算	累乗を求める	スカラ関数POWER
79		底と真数を指定して，その対数を求める	スカラ関数LOG
80		自然対数を求める	スカラ関数LN
81		自然対数の底の値の累乗を求める	スカラ関数EXP
82	三角関数	正弦（三角関数のSIN）を求める	スカラ関数SIN

項番	分類	目的	関連する SQL の構文
83		余弦（三角関数の COS）を求める	スカラ関数COS
84		正接（三角関数の TAN）を求める	スカラ関数TAN
85		逆正弦（逆三角関数）を求める	スカラ関数ASIN
86		逆余弦（逆三角関数）を求める	スカラ関数ACOS
87		逆正接（逆三角関数）を求める	スカラ関数ATAN
88			スカラ関数ATAN2
89		双曲線正弦を求める	スカラ関数SINH
90		双曲線余弦を求める	スカラ関数COSH
91		双曲線正接を求める	スカラ関数TANH
92		角度をラジアンから度数に変換する	スカラ関数DEGREES
93		角度を度数からラジアンに変換する	スカラ関数RADIANS
94		円周率を求める	スカラ関数PI
95	日付・時刻	日付または時刻の一部分を抽出する（例えば，月だけを抽出する）	スカラ関数EXTRACT
96		指定した日が，その年の第何日目かを求める	スカラ関数DAYOFYEAR
97		指定した日が，その週の何日目かを求める	スカラ関数DAYOFWEEK
98		指定した月の最終日の日付を求める	スカラ関数LASTDAY
99		開始日時と終了日時の差を求める	スカラ関数DATEDIFF
100			スカラ関数TIMESTAMPDIFF
101		日時データに日時を加算する	スカラ関数TIMESTAMPADD
102		生年月日と基準日から満年齢を求める	スカラ関数GETAGE
103		日時を年，月，日，時，分などの単位で丸める	スカラ関数ROUND
104		日時を年，月，日，時，分などの単位で切り捨てる	スカラ関数TRUNC
105		現在の日付を求める	日時情報取得関数CURRENT_DATE
106		現在の時刻を求める	日時情報取得関数CURRENT_TIME
107		現在の日付と時刻を求める	日時情報取得関数 CURRENT_TIMESTAMP
108		日時データの演算をする	日時演算
109			ラベル付き間隔
110	ナル値	指定したデータのうち，ナル値でない最初の値を求める	スカラ関数COALESCE
111			スカラ関数ISNULL

項番	分類	目的	関連する SQL の構文
112			スカラ関数NVL
113	データの比較	2 つのデータが等しいかどうかを調べる	スカラ関数NULLIF
114		対象データと比較データを順次比較し，一致した場合は対応する返却値を返す	スカラ関数DECODE
115		対象データと比較データを順次比較し，対象データの値が比較データの値未満となる場合は，対応する返却値を返す	スカラ関数LTDECODE
116		最大値を求める	スカラ関数GREATEST
117		最小値を求める	スカラ関数LEAST
118	データ型	データ型を変換する	スカラ関数CAST
119			スカラ関数CONVERT
120	データの情報取得	対象データのバイト数を求める	スカラ関数LENGTHB
121		文字データの先頭の文字の文字コードを求める	スカラ関数ASCII
122		対象データの数値に対応する文字コードの文字を求める	スカラ関数CHR
123	配列データの情報取得	配列データの最大要素数を求める	スカラ関数ARRAY_MAX_CARDINALITY
124		配列データの配列要素数を求める	スカラ関数CARDINALITY
125	ユーザ情報	実行中の HADB ユーザの認可識別子を求める	ユーザ情報取得関数CURRENT_USER

注※

スカラ関数RANDOM，RANDOMCURSOR，およびRANDOMROW には仕様差があります。仕様差については，「8.4.5　RANDOM」の「(6)　擬似乱数を返すスカラ関数の一覧」を参照してください。

付録 B　関数一覧

関数の一覧を次の表に示します。

表 B-1　関数の一覧

項番	関数			機能
1	集合関数	MAX		最大値を求めます。
2		MIN		最小値を求めます。
3		SUM		合計値を求めます。
4		AVG		平均値を求めます。
5		COUNT		行数（件数）を求めます。
6		COUNT(*)		行数（件数）を求めます。
7		STDDEV_POP		母集団標準偏差を求めます。
8		STDDEV_SAMP		標本標準偏差を求めます。
9		VAR_POP		母集団分散を求めます。
10		VAR_SAMP		標本分散を求めます。
11		MEDIAN		順序付けされた一連の値の中央値を求めます。
12		PERCENTILE_CONT		順序付けされた一連の値のパーセンタイル（百分位数）を求めます。
13		PERCENTILE_DISC		
14		LISTAGG		順序付けされた一連の値を連結し，値と値の間に区切り文字列を挿入した文字列を求めます。
15		ARRAY_AGG		値式によって集計された値を，先頭から順に配列要素とする配列データを作成します。
16	数学関数	三角関数	SIN	ラジアン単位で指定された対象データの正弦（三角関数の SIN）を返します。
17			COS	ラジアン単位で指定した対象データの余弦（三角関数の COS）を返します。
18			TAN	ラジアン単位で指定された対象データの正接（三角関数の TAN）を返します。
19			ASIN	対象データの逆正弦である角度を，$-\pi/2 \sim \pi/2$ の範囲（ラジアン単位）で返します。
20			ACOS	対象データの逆余弦である角度を，$0 \sim \pi$ の範囲（ラジアン単位）で返します。
21			ATAN	対象データの逆正接である角度を，$-\pi/2 \sim \pi/2$ の範囲（ラジアン単位）で返します。

項番	関数			機能
22			ATAN2	y/x の逆正接である角度を，$-\pi$ ～ π の範囲（ラジアン単位）で返します。
23			SINH	対象データの双曲線正弦を返します。
24			COSH	対象データの双曲線余弦を返します。
25			TANH	対象データの双曲線正接を返します。
26			DEGREES	指定された角度をラジアンから度数に変換して返します。
27			RADIANS	指定された角度を度数からラジアンに変換して返します。
28			PI	円周率 π の値を返します。
29		指数・対数計算	POWER	対象データの累乗を返します。
30			LOG	指定値を底とする対象データ（真数）の対数を返します。
31			LN	対象データの自然対数を返します。
32			EXP	自然対数の底の値の累乗を返します。
33		数値計算	MOD	被除数を除数で割った余りを返します。
34			ABS	対象データの絶対値を返します。
35			SQRT	対象データの平方根を返します。
36			SIGN	対象データが正の値の場合は+1 を，負の値の場合は-1 を，0 の場合は0 を返します。
37			RANDOM	最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。
38			RANDOMCURSOR	最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。 1SQL 文中で，同じ識別番号を指定した RANDOMCURSOR は，常に同じ値を返します。
39			RANDOMROW	最小値に指定した値以上，かつ最大値に指定した値未満の範囲での一様分布に従う擬似乱数を返します。 1 問合せ指定中で，同じ識別番号を指定した RANDOMROW は，問合せ指定の結果の行ごとに同じ値を返します。
40			RANDOM_NORMAL	平均 μ，標準偏差 σ の正規分布に従う擬似乱数を返します。
41		数値丸め	ROUND	対象データを小数点以下n 桁に丸めた値を返します。
42			TRUNC	指定された桁数より下の数値を切り捨てた値を返します。

項番	関数			機能
43			FLOOR	対象データ以下の値で，最大の整数値を返します。
44			CEIL	対象データ以上の値で，最小の整数値を返します。
45	文字列関数	文字列の検索	CONTAINS	検索条件式を満たす文字列が対象データ中に含まれているかどうかを返します。
46		文字データの連結	CONCAT	2 つの文字データを連結します。
47		文字データの一部抽出	SUBSTR	文字データの任意の位置から一部の文字列を抽出します。
48			LEFT	文字データの先頭（左）から一部の文字列を抽出します。
49			RIGHT	文字データの末尾（右）から一部の文字列を抽出します。
50		文字データからの文字の削除	TRIM	対象データの文字列から，削除文字に指定した文字を削除します。文字の削除方法を次のどれかから選択できます。 • 文字列の先頭から順に，削除文字に指定した文字を削除します。 • 文字列の末尾から順に，削除文字に指定した文字を削除します。 • 文字列の先頭および末尾の両方から順に，文字を削除します。
51			LTRIM	対象データの文字列の先頭から順に，削除文字に指定した文字を削除します。
52			RTRIM	対象データの文字列の末尾から順に，削除文字に指定した文字を削除します。
53		文字列の埋め込み	LPAD	対象データの先頭（左側）に，指定文字数となるまで，埋め込み文字列を繰り返し埋め込みます。
54			RPAD	対象データの末尾（右側）に，指定文字数となるまで，埋め込み文字列を繰り返し埋め込みます。
55		文字データ中の文字列の置換	REPLACE	対象データ中の任意の文字列を置換します。対象データ中に存在する置換対象文字列のすべてを置換後の文字列に置換します。
56		文字データ中の文字の置換	TRANSLATE	対象データ中の任意の文字を置換します。
57		文字データの文字数	LENGTH	対象データの文字列の文字数を返します。
58		文字データ中の文字列の開始位置	INSTR	対象データ中の任意の文字列を検索し，その文字列の開始位置を返します。
59		英大文字から英小文字への変換，またはその逆	LOWER	文字データの英大文字（A～Z）を英小文字（a～z）に変換します。

項番	関数			機能
60			UPPER	文字データの英小文字（a～z）を英大文字（A～Z）に変換します。
61	日時関数	DATEDIFF		開始日時と終了日時の差を返します。
62		DAYOFWEEK		指定した日が，週の何日目かを返します。
63		DAYOFYEAR		指定した日が，その年の第何日目かを返します。
64		EXTRACT		日時を示すデータの一部（年，月，日，時，分，または秒のどれか）を抽出します。
65		GETAGE		生年月日と基準日から満年齢を求めます。
66		LASTDAY		日時データに指定した月の最終日の日付または日時を返します。
67		ROUND		日時データを日時書式で指定した単位に丸めて返します。
68		TIMESTAMPADD		対象データに指定した日時に，日時単位に指定した単位で日時を加算します。
69		TIMESTAMPDIFF		開始日時と終了日時の差を返します。
70		TRUNC		日時データを日時書式で指定した単位で切り捨てます。
71	バイナリ列関数	バイナリデータの連結	CONCAT	2 つのバイナリデータを連結します。
72		バイナリデータの一部抽出	SUBSTRB	バイナリデータの任意の位置から一部のバイナリデータを抽出します。
73		バイナリデータのビット演算	BITAND	2 つのバイナリデータのビットごとの論理積を返します。
74			BITOR	2 つのバイナリデータのビットごとの論理和を返します。
75			BITNOT	バイナリデータのビットごとの論理否定を返します。
76			BITXOR	2 つのバイナリデータのビットごとの排他的論理和を返します。
77			BITLSHIFT	バイナリデータを左ビットシフトした値を返します。
78			BITRSHIFT	バイナリデータを右ビットシフトした値を返します。
79	配列関数	ARRAY_MAX_CARDINALITY		対象データに指定した配列データの最大要素数を返します。
80		CARDINALITY		対象データに指定した配列データの配列要素数を返します。
81	データ変換関数	CAST		データのデータ型を変換します。
82		CONVERT		データのデータ型を変換します。

項番	関数		機能
			また，日時書式または数値書式を指定することで，次のことができます。 日時書式を指定した場合： • 日時データを文字データに変換する際，変換後の文字データの出力形式を指定できます。 • 文字データを日時データに変換する際，変換前の文字データの入力形式を指定できます。 数値書式を指定した場合： • 数データを文字データに変換する際，変換後の文字データの出力形式を指定できます。 • 文字データを数データに変換する際，変換前の文字データの入力形式を指定できます。
83		ASCII	対象データの先頭の文字の文字コードを整数値で返します。
84		CHR	対象データの整数値が示す文字コードに対応する文字を返します。
85		BIN	バイナリデータを 2 進文字列表現（'0'，'1'で構成された文字データ）に変換します。
86		HEX	バイナリデータを 16 進文字列表現（'0'～'9'，'A'～'F'で構成された文字データ）に変換します。
87	NULL 評価関数	COALESCE	指定した対象データを指定した順に評価し，ナル値でない最初の値を返します。
88		ISNULL	
89		NULLIF	対象データ 1 と対象データ 2 を比較した結果，等しい場合はNULL を返し，等しくない場合は対象データ 1 を返します。
90		NVL	指定した対象データを指定した順に評価し，ナル値でない最初の値を返します。
91	情報取得関数	LENGTHB	対象データの長さをバイト数で返します。
92	比較関数	DECODE	対象データと比較データを順次比較し，一致した場合は対応する返却値を返します。対象データとすべての比較データが一致しない場合は，既定返却値を返します。
93		LTDECODE	対象データと比較データを順次比較し，対象データの値が比較データの値未満となる場合は，対応する返却値を返します。対象データの値がすべての比較データの値未満とならない場合は，既定返却値を返します。
94		GREATEST	指定した対象データの値のうち，最大値を返します。
95		LEAST	指定した対象データの値のうち，最小値を返します。

項番	関数		機能
96	日時情報取得関数	CURRENT_DATE	現在の日付を返します。
97		CURRENT_TIME	現在の時刻を返します。
98		CURRENT_TIMESTAMP	現在の時刻印（日付と時刻）を返します。
99	ユーザ情報取得関数	CURRENT_USER	実行中の HADB ユーザの認可識別子を返します。

索引

記号

数字

A

B

C

D

E

F

G

H

T

U

V

W

あ

い

う

え

お

か

き

く

け

こ

さ

し

に

は

ひ

ふ

へ

ほ

ま

み

も

ゆ

よ

ら

り

れ

ろ

わ

株式会社 日立製作所
〒 100-8280　東京都千代田区丸の内一丁目 6 番 6 号