HITACHI
Inspire the Next

Hitachi Advanced Data Binder
AP 開発ガイド

3000-6-502-O0

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 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 EC2，Amazon S3，Amazon Simple Storage Service 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。

Excel は，マイクロソフト 企業グループの商標です。

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。

Microsoft は，マイクロソフト 企業グループの商標です。

Oracle(R)，Java 及び MySQL 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。

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

Red Hat 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。

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

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。

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。

Visual C++は，マイクロソフト 企業グループの商標です。

Visual Studio は，マイクロソフト 企業グループの商標です。

Windows は，マイクロソフト 企業グループの商標です。
Windows Server は，マイクロソフト 企業グループの商標です。
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.

■ 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
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)

表記			製品名
		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)
	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

2023 年 7 月

変更内容

変更内容(3000-6-502-O0)　Hitachi Advanced Data Binder　05-08

追加・変更内容		変更個所
ODBC ドライバに関する変更	SQLGetData 関数で分割取得機能を使用できるようにしました。	4.3.1，4.3.2，16.9.8(6)，17.4.2，17.4.3
クラウドストレージ機能の制限事項の解除	クラウドストレージ機能の使用時に，INSERT 文，UPDATE 文，およびDELETE 文を実行できるようにしました。 それに伴い，SQL パラレル実行機能も使用できるようにしました。	5.16
ALTER TABLE 文の機能拡張	ALTER TABLE 文で実表の列を削除できるようにしました。	8.7.108(4)，15.8.2(3)，16.13
HADB クライアントのインストールに関する変更	HADB クライアントのインストール，バージョンアップ，バージョンダウン，修正版との入れ替えに関する説明を追加しました。	4.2.1(1)，4.2.2(2)，4.6.2，4.7.2，4.8
HADB サーバおよび HADB クライアントの適用 OS に Red Hat Enterprise Linux Server 9 を新規にサポートし，Red Hat Enterprise Linux Server 6 のサポートを終了しました。		4.2.2(1)，4.2.2(2)，15.1.2(2)
HADB クライアントの次の適用 OS のサポートを終了しました。 • Windows 7 • Windows 8.1 • Windows Server 2008 R2		15.1.2(1)

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 で使用する AP を開発するための基礎技術，および HADB クライアントの環境設定方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 05-08　（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 05-08　（適用 OS：Red Hat Enterprise Linux Server 9（64-bit x86_64））
- P-9W62-C311 Hitachi Advanced Data Binder Client 05-08　（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 05-08　（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 クライアントの管理者

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）
- ODBC の基本的な知識（ODBC 対応の 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）

- Linux または Windows のシステム管理の基本的な知識

■ マニュアルの構成

このマニュアルは，次に示す編，章と付録から構成されています。

第 1 編　環境設定編【共通】

第 1 章　AP 開発の概要

AP 開発の流れ，AP を開発する前に知っておく必要がある前提条件，および AP の実行形態について説明しています。

第 2 章　クライアント定義の設計

クライアント定義のオペランドの指定形式，内容，および文法規則について説明しています。

第 3 章　JDBC ドライバの環境設定

JDBC ドライバのインストールや環境変数の設定など，JDBC ドライバの環境設定方法について説明しています。

第 4 章　HADB クライアントの環境設定（ODBC ドライバおよび CLI 関数を使用する場合）

HADB クライアントのインストールや環境変数の設定など，HADB クライアントの環境設定方法について説明しています。

第 2 編　AP 作成編【共通】

第 5 章　AP の性能向上に関する設計

AP の性能向上に関する設計について説明しています。

第 6 章　AP のチューニング

アクセスパスの見方について説明しています。

第 3 編　AP 作成編【JDBC】

第 7 章　AP の作成

JDBC ドライバを使用した AP の作成方法について説明しています。

第 8 章　JDBC 1.2 API

JDBC 1.2 API の各インタフェースとメソッドについて説明しています。

第 9 章　JDBC 2.1 コア API

JDBC 2.1 コア API で追加された機能の HADB でのサポート範囲について説明しています。

第 10 章　JDBC 2.0 Optional Package

JDBC 2.0 Optional Package の各インタフェースとメソッドについて説明しています。

第 11 章　JDBC 3.0 API

JDBC 3.0 API の各インタフェースとメソッドについて説明しています。

第 12 章　JDBC 4.0 API

JDBC 4.0 API の各インタフェースとメソッドについて説明しています。

第 13 章　JDBC 4.1 API

JDBC 4.1 API で追加された機能に対する HADB でのサポート範囲について説明しています。

第 14 章　JDBC 4.2 API

JDBC 4.2 API で追加された機能に対する HADB でのサポート範囲について説明しています。

第 4 編　AP 作成編【ODBC】

第 15 章　AP の作成

HADB ODBC ドライバの環境設定，および ODBC 対応の AP 作成時の留意事項について説明しています。

第 16 章　ODBC 関数

HADB が提供している ODBC 関数の機能と文法について説明しています。

第 17 章　トラブルシュート

ODBC インタフェース使用時のトラブルシュートについて説明しています。

第 5 編　AP 作成編【CLI 関数】

第 18 章　AP の作成

C 言語および C++言語で AP を設計，作成するときに考慮する必要がある基本事項について説明しています。

第 19 章　CLI 関数

HADB が提供している CLI 関数の機能と文法について説明しています。

付録 A　サンプル AP

サンプル AP の概要，およびサンプル AP を実行するための準備と手順について説明しています。

付録 B　HADB クライアントのディレクトリの構成

HADB クライアントのクライアントディレクトリ（インストール時），およびクライアントディレクトリ（運用時）の構成について説明しています。

付録 C　HADB クライアントのメモリ所要量の見積もり

HADB クライアントが使用するメモリ所要量の見積もり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 コマンドリファレンス』（3000-6-503）

- 『Hitachi Advanced Data Binder SQL リファレンス』（3000-6-504）
- 『Hitachi Advanced Data Binder メッセージ』（3000-6-505）
- 『日立コード変換ユーザーズガイド』（3000-7-415）
- 『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
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

英略語	英字での表記
BOM	Byte Order Mark
CLI	Call Level Interface
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

英略語	英字での表記
LDAP	Lightweight Directory Access Protocol
LOB	Large Object
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

■ 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。

構文要素記号	意味
〈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。
〈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。

■ メソッドの略記について

- 先頭に「get」が付くメソッドをまとめて表す場合，get*XXX* メソッドと表記しています。
- 先頭に「set」が付くメソッドをまとめて表す場合，set*XXX* メソッドと表記しています。
- 先頭に「execute」が付くメソッドをまとめて表す場合，execute*XXX* メソッドと表記しています。

■ 関数の略記について

- 先頭に「SQL」が付く関数をまとめて表す場合，SQL*xxx* 関数と表記しています。
- 先頭に「SQL」が付き，最後に「W」が付く関数をまとめて表す場合，SQL*xxx*W 関数と表記しています。

■ 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。

目次

第 2 編　AP 作成編【共通】

第 3 編　AP 作成編【JDBC】

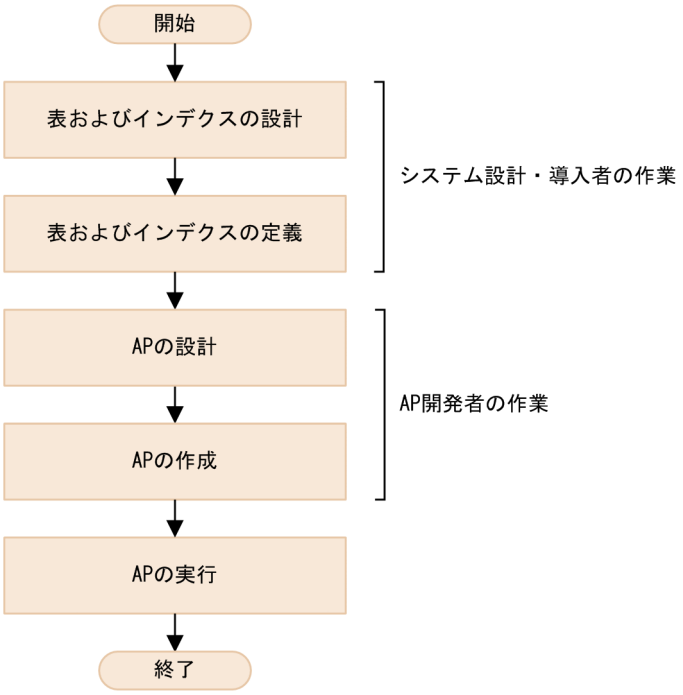
1 AP 開発の概要

この章では，AP 開発の流れ，AP を開発する前に知っておく必要がある前提条件，および AP の実行形態について説明します。

1.1　AP 開発の流れと前提条件

AP 開発の流れを次の図に示します。

図 1-1　AP 開発の流れ



AP を開発する前に知っておく必要がある前提条件について説明します。

1.1.1　AP の記述言語

AP の記述言語は次に示すどれかになります。

- Java 言語
- C 言語
- C++言語

Java 言語の場合，JDBC の API を使用してデータベースにアクセスできます。

C または C++言語の場合，CLI 関数（C または C++言語対応の HADB が提供している API），または ODBC 関数を使用してデータベースにアクセスできます。

また，ODBC 対応の AP からも，ODBC 関数を使用してデータベースにアクセスできます。

1.1.2　文字コード

HADB サーバおよび HADB クライアントでは，次に示す文字コードが使用できます。

- UTF-8
- Shift-JIS

1.1.3　AP の開発環境

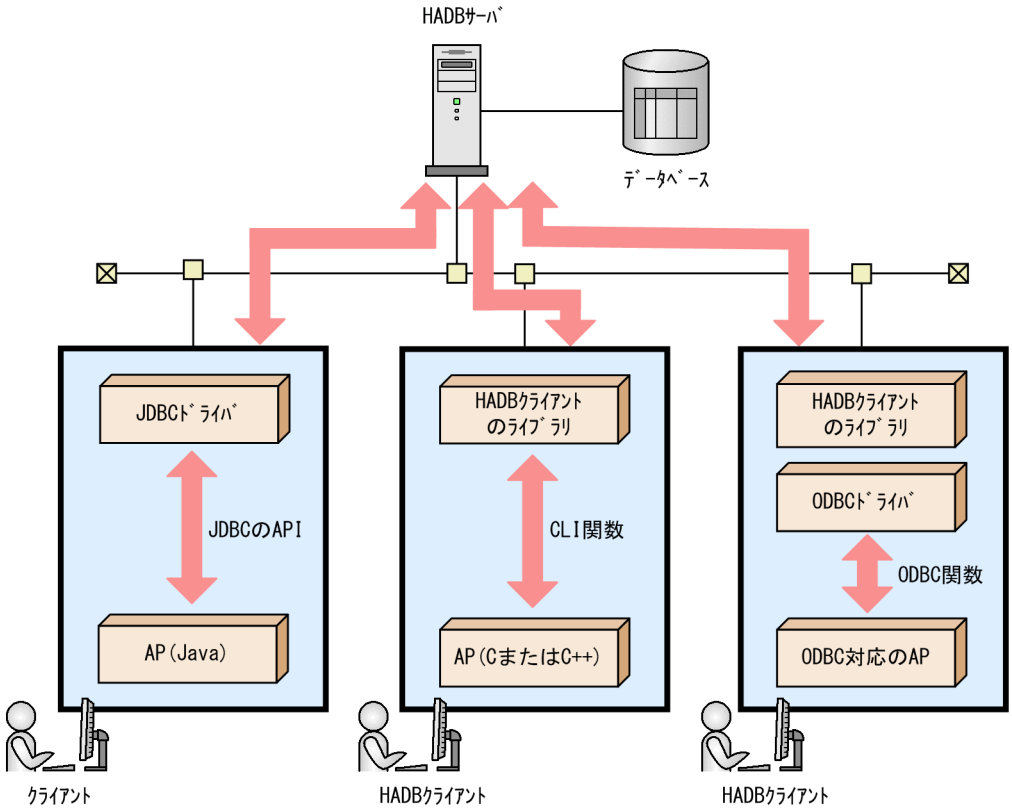
HADB サーバをインストールしたマシン，または HADB クライアントをインストールしたマシンのどちらかで AP の開発を行います。

1.2　AP の実行形態

AP は，HADB クライアントまたは HADB サーバのどちらでも実行できます。

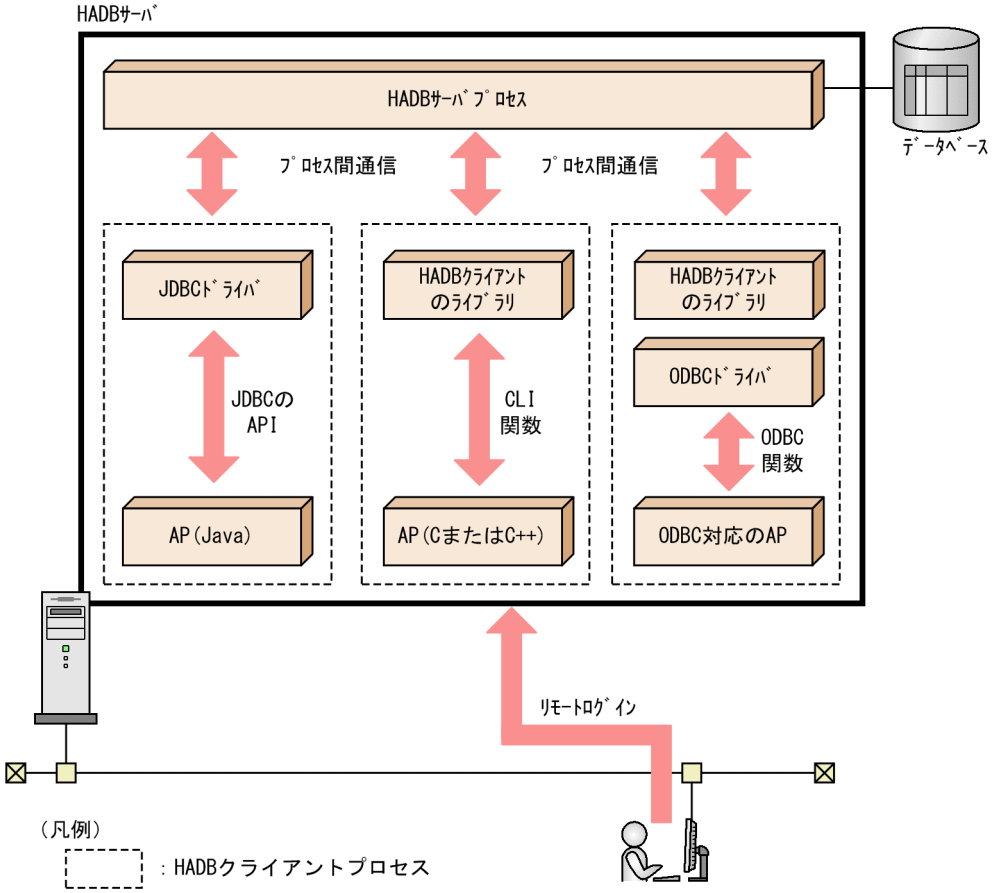
HADB クライアントで AP を実行する形態を図 1-2 に，HADB サーバで AP を実行する形態を図 1-3 に示します。

図 1-2　HADB クライアントで AP を実行する形態



注　HADB クライアントの適用 OS は，Linux または Windows です。

図 1-3　HADB サーバで AP を実行する形態



2 クライアント定義の設計

この章では，クライアント定義のオペランドの指定形式，内容，および文法規則について説明します。

2.1　クライアント定義のオペランドの指定形式

クライアント定義には，HADB クライアントの実行環境を指定します。クライアント定義の各オペランドの指定形式を次に示します。

システム構成に関するオペランド

```
    set adb_clt_rpc_srv_host = HADBサーバのホスト名
   〔set adb_clt_rpc_srv_port = HADBサーバのポート番号〕
   〔set adb_clt_group_name = クライアントグループ名〕
```

APの状態監視に関するオペランド

```
   〔set adb_clt_rpc_con_wait_time = HADBサーバへの接続処理の完了待ち時間〕
   〔set adb_clt_rpc_sql_wait_time = HADBサーバからの応答待ち時間〕
   〔set adb_clt_ap_name = AP識別子〕
```

性能に関するオペランド

```
   〔set adb_clt_fetch_size = FETCH処理時の一括送信行数〕
   〔set adb_dbbuff_wrktbl_clt_blk_num = ローカル作業表用バッファのページ数〕
   〔set adb_sql_exe_max_rthd_num = 最大SQL処理リアルスレッド数〕
   〔set adb_sql_exe_hashgrp_area_size = ハッシュグループ化領域サイズ〕
   〔set adb_sql_exe_hashtbl_area_size = ハッシュテーブル領域サイズ〕
   〔set adb_sql_exe_hashflt_area_size = ハッシュフィルタ領域サイズ〕
   〔set adb_clt_sql_parallel_exec =  {Y | N} 〕
```

SQLに関するオペランド

```
   〔set adb_sql_prep_delrsvd_use_srvdef =  {Y | N} 〕
   〔set adb_clt_trn_iso_lv =  {READ_COMMITTED | REPEATABLE_READ} 〕
   〔set adb_clt_trn_access_mode =  {READ_WRITE | READ_ONLY} 〕
   〔set adb_clt_sql_text_out =  {Y | N} 〕
   〔set adb_clt_sql_order_mode =  {BYTE | ISO} 〕
   〔set adb_sql_prep_dec_div_rs_prior =  {INTEGRAL_PART | FRACTIONAL_PART} 〕
```

PAM認証（外部認証）に関するオペランド

```
   〔set adb_clt_passwd_pubkey_path = パスワードの暗号化に使用する公開鍵ファイルのパス名〕
```

クライアント定義の各オペランドの説明については，「2.2　クライアント定義のオペランドの内容」を参照してください。

クライアント定義の作成方法および変更方法については，「4.4　クライアント定義の作成」を参照してください。

2.2　クライアント定義のオペランドの内容

ここでは，クライアント定義の各オペランドの内容について説明します。

重要

- クライアント定義の各オペランドの指定は，ODBC ドライバまたは CLI 関数を使用する場合に適用されます。JDBC ドライバを使用する場合は適用されません。
- JDBC ドライバを使用する場合は，システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティで，クライアント定義の各オペランドと同じ指定ができます。

  システムプロパティについては，「3.1.6　システムプロパティの設定」を参照してください。

  ユーザプロパティについては，「7.3.1　DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法」の「(2)　getConnection メソッドによる HADB サーバへの接続」の「(d)　引数 info の指定内容（ユーザプロパティの指定）」を参照してください。

  接続用の URL のプロパティについては，「7.3.1　DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法」の「(2)　getConnection メソッドによる HADB サーバへの接続」の「(a)　引数 url の指定内容（接続用の URL の指定）」を参照してください。
- `adbsql` コマンドで SQL 文を実行する場合は，クライアント定義の各オペランドの指定が適用されます。

2.2.1　システム構成に関するオペランド

●`adb_clt_rpc_srv_host` = *HADB サーバのホスト名*

～〈文字列〉((1～255 バイト))

接続先の HADB サーバのホスト名を指定してください。ここで指定したホスト名は，HADB クライアントと HADB サーバ間の通信で使用します。

ホスト名は，ホスト名，ドメイン名，IP アドレスのどれかの形式で指定してください。

このオペランドは必ず指定してください。

クライアント定義の集中管理機能を使用している場合であっても，このオペランドは必ず指定してください。クライアント定義の集中管理機能で使用するクライアント定義ファイルに，このオペランドを指定しても無視されます。

コールドスタンバイ構成の場合は，HADB サーバと HADB クライアント間の通信で使用するエイリアス IP アドレスを指定してください。

[マルチノード機能]

- HA モニタありのマルチノード構成の場合

  HADB サーバと HADB クライアント間の通信で使用するエイリアス IP アドレスを指定してください。

- HA モニタなしのマルチノード構成の場合

  HADB サーバと HADB クライアント間の通信で使用するプライマリノードのホスト名を指定してください。

●adb_clt_rpc_srv_port = *HADB サーバのポート番号*

〜〈整数〉((5,001〜65,535))《23,650》

HADB クライアントと HADB サーバ間の通信で使用する，HADB サーバのポート番号を指定してください。サーバ定義のadb_rpc_port オペランドで指定したポート番号を指定してください。

●adb_clt_group_name = *クライアントグループ名*

〜〈文字列〉((1〜30 バイト))

このクライアント定義を使用する AP が属するクライアントグループ名を指定します。サーバ定義の adbcltgrp オペランドに指定したクライアントグループ名を指定してください。

このオペランドを省略した場合，このクライアント定義を使用する AP はクライアントグループに属しません。

また，adbcltgrp オペランドで指定していないクライアントグループ名を指定した場合，クライアントグループに属していない AP と見なされます。

クライアントグループについては，マニュアル『HADB　システム構築・運用ガイド』の『クライアントグループ機能』を参照してください。

2.2.2　AP の状態監視に関するオペランド

●adb_clt_rpc_con_wait_time = *HADB サーバへの接続処理の完了待ち時間*

〜〈整数〉((1〜300))《300》(単位：秒)

HADB サーバへの接続処理の完了待ち時間を秒単位で指定します。ここで指定した待ち時間を超えても HADB サーバへの接続処理が完了しない場合，HADB サーバへの接続処理を中止し，AP にエラーリターンします。

通常は，このオペランドを指定する必要はありません。HADB サーバがビジー状態で接続に時間が掛かる場合に，タイムアウト時間を短くするときに指定してください。

■JDBC ドライバのプロパティにadb_clt_rpc_con_wait_time を指定している場合

JDBC ドライバのプロパティのadb_clt_rpc_con_wait_time には，0 を指定することができます。0 を指定した場合，デフォルト値が仮定されます。

●adb_clt_rpc_sql_wait_time = *HADB サーバからの応答待ち時間*

〜〈整数〉((0〜65,535))《0》(単位：秒)

HADB クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間を秒単位で指定します。ここで指定した待ち時間を超えても HADB サーバから応答がない場合，SQLCODE が-732（KFAA30732-E）のタイムアウトエラーが AP に返されます。このとき，SQL 文の処理はキャンセルされ，トランザクションはロールバックされます。そのあと，HADB サーバから AP が切り離されます。

ただし，処理要求をキャンセルしたタイミングや，通信障害などの要因によっては，トランザクションがロールバックされないことがあります。そのため，HADB サーバのメッセージログに出力されるメッセージなどを参照して，結果を確認することを推奨します。

処理時間が長い SQL を監視する場合などに，このオペランドを指定してください。「4.5　AP の無応答状態への対策」を参照して，このオペランドに指定する待ち時間を決めてください。

なお，このオペランドを省略した場合，または0 を指定した場合，待ち時間は設定されません。

■JDBC ドライバのプロパティにadb_clt_rpc_sql_wait_time を指定している場合

JDBC ドライバのプロパティにadb_clt_rpc_sql_wait_time を指定している場合は，さらに次の待ち時間も監視対象になります。

- 同一コネクションで複数のSELECT 文を同時実行した際に，処理リアルスレッド数不足が発生したときの処理リアルスレッドの確保処理の待ち時間

上記の待ち時間を超えた場合，SQLCODE が-1071570（KFAA71570-E）のタイムアウトエラーが AP に返されます。このとき，SQL 文の処理はキャンセルされますが，トランザクションはロールバックされません。また，AP は HADB サーバから切り離されません。

adb_clt_rpc_sql_wait_time を指定する目的については，「7.4.1　データの検索方法」の「(4)　同一コネクションで複数の SELECT 文を同時実行する際の注意事項」を参照してください。

●adb_clt_ap_name = *AP 識別子*

～〈文字列〉((1～30 バイト))《********》

HADB サーバに接続する AP の識別情報（AP 識別子）を指定します。

ここで指定した AP 識別子は，コネクションの状態を表示するコマンド（adbls -d cnct）の実行結果，メッセージ，および SQL トレース情報などに表示されます。どの AP が実行されているかを確認するときに必要な情報になります。

AP 識別子は環境変数ADBCLTLANG の指定に基づいて認識されるため，文字コードに依存しない半角英数字だけで構成される名称を指定することを推奨します。

2.2.3　性能に関するオペランド

●adb_clt_fetch_size = *FETCH 処理時の一括送信行数*

～〈整数〉((1～65,535))《1,024》

1 回のFETCH 処理で，HADB サーバから HADB クライアントに送信する検索結果の行数を指定します。

このオペランドの指定値を大きくすると，1 回のFETCH 処理で送信する結果行数が多くなるため性能向上が見込めますが，その分メモリが必要になります。

●adb_dbbuff_wrktbl_clt_blk_num = *ローカル作業表用バッファのページ数*

～〈整数〉((5～100,000,000))《サーバ定義のadb_dbbuff_wrktbl_clt_blk_num の値》

ローカル作業表用バッファのページ数を指定します。

通常，このオペランドは指定不要です。ローカル作業表を作成する SQL 文の実行時間を短縮したい場合に，このオペランドを指定してください。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『チューニング』の『バッファの見直しによるSQL文の実行時間の短縮に関するチューニング』を参照してください。

このオペランドの指定値の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』の『ローカル作業表用バッファのページ数の見積もり』を参照してください。

留意事項を次に示します。

- このオペランドと，サーバ定義adb_dbbuff_wrktbl_clt_blk_numオペランドを両方指定した場合は，このオペランドの指定値が優先されます。
- このオペランドで指定したバッファは，SQL処理リアルスレッドごとに，ローカル作業表を作成するSQL文を実行したときだけ使用されます。ローカル作業表とは，リアルスレッドごとに作成される，リアルスレッド固有の作業表です。そのため，HADBサーバはリアルスレッドごとにこのオペランドで指定したページ数分のローカル作業表バッファを確保します。ローカル作業表を作成するSQL文については，「5.10.2　SQLを実行した場合に作成される作業表について」を参照してください。
- このオペランドの指定を適用したコネクションについては，ローカル作業表用バッファのページ数をadbmodbuffコマンドで変更できません。
- コネクションごとに適用されているローカル作業表用バッファのページ数は，adbls -d lbufコマンドで確認できます。

●adb_sql_exe_max_rthd_num = *最大SQL処理リアルスレッド数*

～〈整数〉((0～4,096))《サーバ定義のadb_sql_exe_max_rthd_numの値，またはグループで使用できる処理リアルスレッドの最大数》

SQL文の実行時に使用する処理リアルスレッド数の最大値を指定します。

サーバ定義のadb_sql_exe_max_rthd_numオペランドで指定した，最大SQL処理リアルスレッド数を変更する場合にこのオペランドを指定します。

このオペランドを指定する場合は，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『性能に関するオペランド（set形式）』にあるadb_sql_exe_max_rthd_numオペランドの説明を参照してください。

■クライアントグループ機能を使用していない場合

- このオペランドを省略した場合，サーバ定義のadb_sql_exe_max_rthd_numオペランドの指定値が仮定されます。
- サーバ定義のadb_sql_exe_max_rthd_numオペランドの指定値より大きい値をこのオペランドに指定した場合，このオペランドの指定は無視されます。このとき，サーバ定義のadb_sql_exe_max_rthd_numオペランドの指定値が仮定されます。

■クライアントグループ機能を使用している場合

- このオペランドを省略した場合，グループで使用できる処理リアルスレッドの最大数が仮定されます。
- グループで使用できる処理リアルスレッドの最大数より大きい値をこのオペランドに指定した場合，このオペランドの指定は無視されます。このとき，グループで使用できる処理リアルスレッドの最大数が仮定されます。

クライアントグループ機能については，マニュアル『HADB　システム構築・運用ガイド』の『クライアントグループ機能』を参照してください。

■JDBC ドライバの setHADBSQLMaxRthdNum メソッドとの関係

setHADBSQLMaxRthdNum メソッドで最大 SQL 処理リアルスレッド数を指定できます。

setHADBSQLMaxRthdNum メソッドの指定と，このオペランドに適用される値の関係について次の表に示します。

setHADBSQLMaxRthdNum メソッドについては，「8.3.37　setHADBSQLMaxRthdNum(int rthdNum)」を参照してください。

setHADBSQLMaxRthdNum メソッドの指定	SV と CGV の関係	V，SV，および CGV の関係	このオペランドに適用される値
指定あり	*CGV* ＜ *SV*	*SV* ＜ *V*	*CGV*※1
		CGV ＜ *V* ≦ *SV*	
		V ≦ *CGV*	*V*
	SV ≦ *CGV*	*V* ≦ *SV*	*V*
		SV ＜ *V* ≦ *CGV*	*SV*※1
		CGV ＜ *V*	
	CGV なし※2	*V* ≦ *SV*	*V*
		SV ＜ *V*	*SV*※1
指定なし	－	－	*CNV*

（凡例）

V：setHADBSQLMaxRthdNum メソッドの指定値

SV：サーバ定義の値※3

CGV：クライアントグループ機能を使用している場合で，グループで使用できる処理リアルスレッド数の最大値

CNV：HADB サーバへのコネクト時に決定される最大 SQL 処理リアルスレッド数※4

－：条件なし

注※1

setHADBSQLMaxRthdNum メソッドの指定値を無効にします。このとき，無効にしたことを知らせるKFAA41106-W メッセージが出力されます。

注※2

クライアントグループ機能を使用していない場合を意味します。

注※3

HADB サーバの開始時に決定される最大 SQL 処理リアルスレッド数です。サーバ定義の adb_sql_exe_max_rthd_num オペランドの指定有無や，処理リアルスレッド数との大小関係などから決定される値です。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

『性能に関するオペランド（set 形式)』のadb_sql_exe_max_rthd_num オペランドの説明を参照してください。

注※4

クライアント定義のadb_sql_exe_max_rthd_num オペランドの指定有無，サーバ定義のadb_sql_exe_max_rthd_num オペランドの指定値，クライアントグループ機能のグループで使用できる処理リアルスレッドの最大値などとの大小関係から決定される値です。

> **重要**
>
> HADB サーバは，SQL 文の前処理時にこのオペランドの値を参照します。そのため，setHADBSQLMaxRthdNum メソッドで最大 SQL 処理リアルスレッド数を指定する場合は，SQL 文を実行するStatement オブジェクトまたはPreparedStatement オブジェクトを生成する前にsetHADBSQLMaxRthdNum メソッドを実行してください。

●adb_sql_exe_hashgrp_area_size =*ハッシュグループ化領域サイズ*

～〈整数〉((0，4～1,000,000))《サーバ定義のadb_sql_exe_hashgrp_area_size の値》(単位：キロバイト)

ハッシュグループ化領域の大きさをキロバイト単位で指定します。

サーバ定義のadb_sql_exe_hashgrp_area_size オペランドで指定した，ハッシュグループ化領域サイズを変更する場合にこのオペランドを指定します。

このオペランドを指定する場合は，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『性能に関するオペランド（set 形式)』にあるadb_sql_exe_hashgrp_area_size オペランドの説明を参照してください。

●adb_sql_exe_hashtbl_area_size =*ハッシュテーブル領域サイズ*

～〈整数〉((0～4,194,304))《サーバ定義のadb_sql_exe_hashtbl_area_size の値》(単位：メガバイト)

ハッシュテーブル領域サイズをメガバイト単位で指定します。

サーバ定義のadb_sql_exe_hashtbl_area_size オペランドで指定した，ハッシュテーブル領域サイズを変更する場合にこのオペランドを指定します。

このオペランドを指定する場合は，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『性能に関するオペランド（set 形式)』にあるadb_sql_exe_hashtbl_area_size オペランドの説明を参照してください。

なお，このオペランドの指定値がサーバ定義のadb_sql_exe_hashtbl_area_size オペランドの指定値より大きい場合は，サーバ定義の指定値が仮定されます。また，このオペランドを省略した場合も，サーバ定義の指定値が仮定されます。

■JDBC ドライバの setHADBSQLHashTblSize メソッドとの関係

setHADBSQLHashTblSize メソッドでハッシュテーブル領域サイズを指定できます。

setHADBSQLHashTblSize メソッドの指定と，このオペランドに適用される値の関係について次の表に示します。

setHADBSQLHashTblSize メソッドについては，「8.3.36　setHADBSQLHashTblSize(int areaSize)」を参照してください。

setHADBSQLHashTblSize メソッドの指定	V と SV の関係	このオペランドに適用される値（単位：メガバイト）
指定あり	*SV* ＜ *V*	*SV*※1
	V ≦ *SV*	*V*
指定なし	－	*CNV*

（凡例）

V：setHADBSQLHashTblSize メソッドの指定値

SV：サーバ定義の値※2

CNV：HADB サーバへのコネクト時に決定されるハッシュテーブル領域サイズ※3

－：条件なし

注※1

setHADBSQLHashTblSize メソッドの指定値を無効にします。このとき，無効にしたことを知らせるKFAA41106-W メッセージが出力されます。

注※2

HADB サーバの開始時に決定されるハッシュテーブル領域サイズです。サーバ定義の adb_sql_exe_hashtbl_area_size オペランドの指定値です。サーバ定義の adb_sql_exe_hashtbl_area_size オペランドを省略した場合は，その省略値です。

注※3

クライアント定義のadb_sql_exe_hashtbl_area_size オペランドの指定有無や，サーバ定義の adb_sql_exe_hashtbl_area_size オペランドとクライアント定義の adb_sql_exe_hashtbl_area_size オペランドとの大小関係から決定される値です。

重要

HADB サーバは，SQL 文の前処理時にこのオペランドの値を参照します。そのため，setHADBSQLHashTblSize メソッドでハッシュテーブル領域サイズを指定する場合は，SQL 文を実行するStatement オブジェクトまたはPreparedStatement オブジェクトを生成する前にsetHADBSQLHashTblSize メソッドを実行してください。

●adb_sql_exe_hashflt_area_size = *ハッシュフィルタ領域サイズ*

～〈整数〉((0～419,430))（単位：メガバイト）

ハッシュフィルタ領域サイズをメガバイト単位で指定します。

通常はこのオペランドを省略してください。ハッシュフィルタが適用される SQL 文の実行時間を短縮したい場合に，このオペランドを指定してください。

このオペランドを指定する場合は，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『性能に関するオペランド（set 形式）』にある adb_sql_exe_hashflt_area_size オペランドの説明を参照してください。

このオペランドの指定値のチューニング方法については，マニュアル『HADB　システム構築・運用ガイド』の『ハッシュフィルタ領域サイズの見直しによるSQL文の実行時間の短縮に関するチューニング』を参照してください。

このオペランドの規則を次に示します。

1. クライアント定義のadb_sql_exe_hashtbl_area_sizeオペランドの指定有無によって，このオペランドを省略したときの仮定値が変わります。

   - adb_sql_exe_hashtbl_area_sizeオペランドを指定している場合

     ↑クライアント定義のadb_sql_exe_hashtbl_area_sizeオペランドの値÷10↑

   - adb_sql_exe_hashtbl_area_sizeオペランドを指定していない場合

     サーバ定義のadb_sql_exe_hashflt_area_sizeオペランドの値

   なお，クライアント定義のadb_sql_exe_hashtbl_area_sizeオペランドの値が，サーバ定義の同オペランドの値より大きい場合は，サーバ定義の値が仮定されるため，ご注意ください。

2. 次の条件を満たす場合は，クライアント定義のadb_sql_exe_hashflt_area_sizeオペランドの値ではなく，サーバ定義の同オペランドの値が適用されます。

$A < B$またはC

   A：サーバ定義のadb_sql_exe_hashflt_area_sizeオペランドの指定値

   B：クライアント定義のadb_sql_exe_hashflt_area_sizeオペランドの指定値

   C：クライアント定義のadb_sql_exe_hashflt_area_sizeオペランドを省略したときの仮定値

■JDBCドライバのsetHADBSQLHashFltSizeメソッドとの関係

setHADBSQLHashFltSizeメソッドでハッシュフィルタ領域サイズを指定できます。
setHADBSQLHashFltSizeメソッドの指定と，このオペランドに適用される値の関係について次の表に示します。

setHADBSQLHashFltSizeメソッドについては，「8.3.35　setHADBSQLHashFltSize(int areaSize)」を参照してください。

setHADBSQLHashFltSizeメソッドの指定	Vと SVの関係	HTVと SVの関係	このオペランドに適用される値（単位：メガバイト）
指定あり	$SV < V$	−	SV※1
	$V \leqq SV$	−	V
指定なし	−	$SV <$ ↑$HTV \div 10$↑	SV
		↑$HTV \div 10$↑ $\leqq SV$	↑$HTV \div 10$↑
		HTVの指定なし※2	CNV

（凡例）

V：setHADBSQLHashFltSizeメソッドの指定値

SV：サーバ定義の値※3

HTV：ハッシュテーブル領域サイズ※4

CNV：HADB サーバへのコネクト時に決定されるハッシュフィルタ領域サイズ※5

－：条件なし

注※1

setHADBSQLHashFltSize メソッドの指定値を無効にします。このとき，無効にしたことを知らせるKFAA41106-W メッセージが出力されます。

注※2

setHADBSQLHashTblSize メソッドの指定がないことを意味しています。

注※3

HADB サーバの開始時に決定されるハッシュフィルタ領域サイズです。サーバ定義の adb_sql_exe_hashflt_area_size オペランドの指定有無や，処理リアルスレッド数との大小関係などから決定される値です。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『性能に関するオペランド（set 形式)』のadb_sql_exe_hashflt_area_size オペランドの説明を参照してください。

注※4

setHADBSQLHashTblSize メソッドを指定した場合に，最終的に適用されたハッシュテーブル領域サイズです。

注※5

クライアント定義のadb_sql_exe_hashflt_area_size オペランドの指定有無や，サーバ定義の adb_sql_exe_hashflt_area_size オペランドとクライアント定義の adb_sql_exe_hashflt_area_size オペランドとの大小関係などから決定される値です

重要

HADB サーバは，SQL 文の前処理時にこのオペランドの値を参照します。そのため，setHADBSQLHashFltSize メソッドでハッシュフィルタ領域サイズを指定する場合は，SQL 文を実行するStatement オブジェクトまたはPreparedStatement オブジェクトを生成する前にsetHADBSQLHashFltSize メソッドを実行してください。

●adb_clt_sql_parallel_exec = {Y | N}

当該 HADB クライアントからのトランザクションで実行される検索系 SQL に SQL パラレル実行機能を適用するかどうかを指定します。

Y：

SQL パラレル実行機能を適用して検索系 SQL を実行します。

N：

SQL パラレル実行機能を適用しません。

このオペランドの指定を省略した場合，N が仮定されます。

重要

SQL パラレル実行機能が適用されるのは，SQL パラレル実行機能が適用される条件が満たされた場合だけです。条件が満たされなかった場合，このオペランドにY を指定していてもSQL パラレル実行機能は適用されません。SQL パラレル実行機能が適用される条件については，「5.16　SQL パラレル実行機能が適用される条件」を参照してください。

SQL 文に SQL パラレル実行指定を指定すると，このオペランドの指定値に関係なく SQL パラレル実行機能を適用して検索系 SQL が実行できます。なお，この場合も SQL パラレル実行機能が適用されるのは，SQL パラレル実行機能が適用される条件が満たされたときだけです。

SQL パラレル実行指定とadb_clt_sql_parallel_exec オペランドの指定値の関係を次の表に示します。

表 2-1　SQL パラレル実行指定と adb_clt_sql_parallel_exec オペランドの指定値の関係

SQL パラレル実行指定	クライアント定義の adb_clt_sql_parallel_exec オペランドの指定値	
	Y	N
検索系 SQL 文に指定している	○	○
検索系 SQL 文に指定していない	○	×

（凡例）

○：SQL パラレル実行機能を適用します。

×：SQL パラレル実行機能を適用しません。

SQL パラレル実行指定の指定規則については，マニュアルの『HADB　SQL リファレンス』の『SELECT 文の指定形式および規則』を参照してください。

2.2.4　SQL に関するオペランド

●adb_sql_prep_delrsvd_use_srvdef =　{Y | N}

サーバ定義のadb_sql_prep_delrsvd_words オペランドの指定に従って予約語を削除するかどうかを指定します。

削除される予約語については，サーバ定義のadb_sql_prep_delrsvd_words オペランドの指定を確認してください。

Y：

サーバ定義のadb_sql_prep_delrsvd_words オペランドの指定を有効にします（サーバ定義のadb_sql_prep_delrsvd_words オペランドに指定した予約語を削除します）。

N：

サーバ定義のadb_sql_prep_delrsvd_words オペランドの指定を無効にします（サーバ定義のadb_sql_prep_delrsvd_words オペランドに指定した予約語は削除されません）。

このオペランドの指定を省略した場合，Y が仮定されます。

●adb_clt_trn_iso_lv = {READ_COMMITTED | REPEATABLE_READ}

適用するトランザクション隔離性水準を指定します。トランザクション隔離性水準については，マニュアル『HADB　システム構築・運用ガイド』の『HADB がサポートしているトランザクション隔離性水準』を参照してください。

READ_COMMITTED：

トランザクション隔離性水準にREAD COMMITTED を適用します。

REPEATABLE_READ：

トランザクション隔離性水準にREPEATABLE READ を適用します。

このオペランドを省略した場合，サーバ定義のadb_sys_trn_iso_lv オペランドに指定したトランザクション隔離性水準が適用されます。

[マルチノード機能]

- 次に示すすべての条件を満たす場合に，セカンダリノードおよびワーカーノードに対してもトランザクションの実行処理が割り当てられます。

  ・トランザクションアクセスモードが読み取り専用モードである

  ・トランザクション隔離性水準がREAD COMMITTED である

  そのため，セカンダリノードおよびワーカーノードのリソースを活用したい場合は，このオペランドにREAD_COMMITTED を指定することを推奨します。

- 読み書き可能モードのトランザクションや，一部のコマンドがプライマリノードで実行中の間は，すべてのトランザクション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トランザクションが実行されません。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『トランザクションおよびコマンドが実行されるノード』を参照してください。

  また，ここでいう一部のコマンドについては，マニュアル『HADB　システム構築・運用ガイド』の『トランザクションおよびコマンドが実行されるノード』の『トランザクションとコマンドの同時実行についての制限』を参照してください。

●adb_clt_trn_access_mode = {READ_WRITE | READ_ONLY}

トランザクションアクセスモードを指定します。トランザクションアクセスモードについては，マニュアル『HADB　システム構築・運用ガイド』の『トランザクションアクセスモード』を参照してください。

READ_WRITE：

トランザクションアクセスモードを読み書き可能モードにします。この場合，トランザクションは読み書き可能トランザクションになり，すべての SQL 文を実行できます。

READ_ONLY：

トランザクションアクセスモードを読み取り専用モードにします。この場合，トランザクションは読み取り専用トランザクションになるため，更新系 SQL および定義系 SQL は実行できません。

なお，ここで設定したトランザクションアクセスモードを，次のコネクション属性の設定で変更することができます。

- ODBC 関数の場合：SQLSetConnectAttr またはSQLSetConnectAttrW
- CLI 関数の場合：a_rdb_SQLSetConnectAttr()

このオペランドおよびコネクション属性の両方のトランザクションアクセスモードの指定を省略した場合，トランザクションアクセスモードは読み書き可能モードになります。

[マルチノード機能]

- 次に示すすべての条件を満たす場合に，セカンダリノードおよびワーカーノードに対してもトランザクションの実行処理が割り当てられます。

  ・トランザクションアクセスモードが読み取り専用モードである

  ・トランザクション隔離性水準がREAD COMMITTED である

  そのため，セカンダリノードおよびワーカーノードのリソースを活用したい場合は，このオペランドにREAD_ONLY を指定することを推奨します。

- 読み書き可能モードのトランザクションや，一部のコマンドがプライマリノードで実行中の間は，すべてのトランザクション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トランザクションが実行されません。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『トランザクションおよびコマンドが実行されるノード』を参照してください。

  また，ここでいう一部のコマンドについては，マニュアル『HADB　システム構築・運用ガイド』の『トランザクションおよびコマンドが実行されるノード』の『トランザクションとコマンドの同時実行についての制限』を参照してください。

●adb_clt_sql_text_out = {Y | N}

HADB クライアントが発行した SQL 文を，クライアントメッセージログファイルおよびサーバメッセージログファイルに出力するかどうかを指定します。

なお，出力される各 SQL 文の長さの上限は 2,048 バイトです。

Y：

SQL 文をクライアントメッセージログファイルおよびサーバメッセージログファイルに出力します。

N：

SQL 文をクライアントメッセージログファイルおよびサーバメッセージログファイルに出力しません。

サーバ定義のadb_sql_text_out オペランドとの関係を次の表に示します。

表 2-2　サーバ定義の adb_sql_text_out オペランドとの関係

サーバ定義の adb_sql_text_out オペランドの指定	クライアント定義の adb_clt_sql_text_out オペランドの指定	
	Y	N
Y	◎	○
N	◎	×

（凡例）

◎：SQL 文をサーバメッセージログファイルおよびクライアントメッセージログファイルの両方に出力します。

○：SQL 文をサーバメッセージログファイルにだけ出力します。

×：SQL 文をどちらのメッセージログファイルにも出力しません。

また，このオペランドを指定すると，トランザクションの正常終了を示すKFAA81002-I メッセージがサーバメッセージログファイルに出力されるようになります。クライアントメッセージログファイルには出力されません。

このオペランドの指定を省略した場合，N が仮定されます。

●adb_clt_sql_order_mode = {BYTE | ISO}

ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を指定します。

BYTE：

文字データをバイトコード順に並び替えます。

ISO：

文字データをソートコード順（ISO/IEC14651:2011 準拠）に並び替えます。

ただし，環境変数ADBCLTLANG にSJIS を指定した場合は，このオペランドにISO は指定できません。

なお，文字データの並び替え順序は，サーバ定義adb_sql_order_mode オペランド，およびコネクション属性でも指定できます。それぞれ指定した場合は，次の表に示す優先順位に従って指定が決まります（優先順位の番号が小さいほど，優先順位が高くなります）。

表 2-3 文字データの並び替え順序の優先順位

優先順位	文字データの並び替え順序の指定個所
1	コネクション属性
2	クライアント定義adb_clt_sql_order_mode オペランド
3	サーバ定義adb_sql_order_mode オペランド

[説明]

例えば，クライアント定義adb_clt_sql_order_mode オペランドにBYTE を指定し，サーバ定義adb_sql_order_mode オペランドにISO を指定した場合，AP から実行するSELECT 文（ORDER BY 句を指定）には，BYTE が適用されます。

なお，表中のすべての個所で指定を省略した場合は，BYTE が仮定されます。

●adb_sql_prep_dec_div_rs_prior = {INTEGRAL_PART | FRACTIONAL_PART}

SQL 文中に指定した除算（四則演算）の結果のデータ型がDECIMAL 型の場合，除算結果の位取りの最小値を指定します。

INTEGRAL_PART：

除算結果の位取りの最小値を 0 とします。INTEGRAL_PART を指定した場合，整数部の桁数が優先されます。そのため，除算結果が大きな数値となる可能性があり，オーバフローエラーの発生を極力回避したい場合は，INTEGRAL_PART を指定してください。

FRACTIONAL_PART：

除算の第 1 演算項（被除数）の位取りを，除算結果の位取りの最小値とします。FRACTIONAL_PART を指定した場合，第 1 演算項の小数部の桁数が，除算結果の小数部の桁数の最小値になります。

第 1 演算項をDECIMAL(*p1*,*s1*)，第 2 演算項をDECIMAL(*p2*,*s2*)とした場合，除算結果はDECIMAL(38,*s*)となります。

- INTEGRAL_PART を指定した場合

  $s = \mathrm{MAX}\ \{0,\ 38 - (p1 - s1 + s2)\}$

- FRACTIONAL_PART を指定した場合

  $s = \mathrm{MAX}\ \{s1,\ 38 - (p1 - s1 + s2)\}$

このオペランドの指定値による除算結果の違いの例を次に示します。

(例)

表T1

C1列 DECIMAL(38,4)	C2列 DECIMAL(10,5)
30.5256	0.05223

次のSELECT 文を実行したとします。

```
SELECT "C1"/"C2" AS "除算結果" FROM "T1"
```

- INTEGRAL_PART を指定した場合の除算結果

  584.

- FRACTIONAL_PART を指定した場合の除算結果

  584.4457

ビュー表を検索したときの除算結果の位取りの例については，マニュアル『HADB　SQL リファレンス』の『除算結果のデータ型が DECIMAL 型の場合の留意事項』を参照してください。

このオペランドの指定を省略した場合，サーバ定義のadb_sql_prep_dec_div_rs_prior オペランドの指定値が適用されます。

2.2.5　PAM 認証（外部認証）に関するオペランド

●adb_clt_passwd_pubkey_path = *パスワードの暗号化に使用する公開鍵ファイルのパス名*

～〈OS パス名〉((2～510 バイト))

パスワードの暗号化に使用する公開鍵ファイルの絶対パスを指定します。ユーザ認証方式に PAM 認証を使用する場合にこのオペランドを指定します。このオペランドを指定しないと，PAM 認証使用時にパスワードは暗号化されません。平文のまま，HADB クライアントから HADB サーバへ送信されます。

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。

2.3　オペランドの指定規則

クライアント定義のオペランドの指定規則は，サーバ定義のオペランドの指定規則と同じです。サーバ定義のオペランドの指定規則については，マニュアル『HADB　システム構築・運用ガイド』の『サーバ定義の文法規則』を参照してください。

2.4　クライアント定義の集中管理機能を使用する場合の留意事項

次に示すクライアント定義のオペランドは，クライアント定義の集中管理機能の適用対象外になります。そのため，次に示すオペランドは，各 HADB クライアントのクライアント定義に指定してください。

- adb_clt_rpc_srv_host
- adb_clt_rpc_srv_port
- adb_clt_rpc_con_wait_time
- adb_clt_rpc_sql_wait_time

クライアント定義の集中管理機能については，マニュアル『HADB　システム構築・運用ガイド』の『クライアント定義の集中管理機能』を参照してください。

3 JDBC ドライバの環境設定

この章では，JDBC ドライバのインストールや，環境変数の設定など，JDBC ドライバの環境設定方法について説明します。

3.1　JDBC ドライバの環境設定手順

JDBC ドライバの環境設定方法について説明します。「3.1.1　Java Runtime Environment または Java Development Kit のインストール」以降の説明に従って JDBC ドライバの環境設定をしてください。

3.1.1　Java Runtime Environment または Java Development Kit のインストール

AP を実行または開発するマシンに，次のどちらかの製品をインストールしてください。

- Java Runtime Environment（JRE）バージョン 8 以降
- Java Development Kit（JDK）バージョン 8 以降

JRE は，AP の実行をする場合に必要となる製品です。JDK は，AP の開発および実行をする場合に必要となる製品です。

> **メモ**
>
> HADB サーバで AP を実行する場合は，HADB サーバをインストールしたマシンに JRE をインストールしてください。HADB サーバで AP を開発および実行する場合は，HADB サーバをインストールしたマシンに JDK をインストールしてください。

■HADB が提供している JDBC ドライバ

HADB が提供している JDBC ドライバを次の表に示します。

表 3-1　HADB が提供している JDBC ドライバ

JRE または JDK のバージョン	HADB が提供している JDBC ドライバの種類	JAR ファイル名	対応する JDBC 規格
JRE8 以降または JDK8 以降	JRE8 版の JDBC ドライバ	adbjdbc8.jar	JDBC4.2

3.1.2　JDBC ドライバのインストール

JDBC ドライバのインストール手順を次に示します。

手順

1. HADB クライアントのインストール CD-ROM に格納されている圧縮ファイルを任意のフォルダにコピーする

   次のファイルをコピーします。

   - 64 ビット版の Windows の場合

hitachi_advanced_data_binder_client.zip

- 32 ビット版の Windows の場合
  hitachi_advanced_data_binder_client32.zip
- Linux の場合
  hitachi_advanced_data_binder_client-$VER.tar.gz
  $VER は，HADB のバージョンおよびリリース番号になります。

2. コピーした圧縮ファイルを解凍する
   圧縮ファイル解凍後の JAR ファイルの格納先を次の表に示します。

表 3-2　圧縮ファイル解凍後の JAR ファイルの格納先

解凍した圧縮ファイル	JAR ファイルの格納先
次のどちらかの圧縮ファイルを解凍した場合 • hitachi_advanced_data_binder_client.zip • hitachi_advanced_data_binder_client32.zip	%INSTCLTDIR%¥HADBCL¥client¥lib¥adbjdbc8.jar %INSTCLTDIR%は，圧縮ファイルの解凍先フォルダです。
hitachi_advanced_data_binder_client-$VER.tar.gz を解凍した場合	$INSTCLTDIR/ hitachi_advanced_data_binder_client_$VER/ client/lib/adbjdbc8.jar $INSTCLTDIR は，圧縮ファイルの解凍先ディレクトリです。

3. JAR ファイルを任意のフォルダにコピーする
4. 次のフォルダおよびファイルを削除する
   - 手順の 1.でコピーした圧縮ファイル
   - 手順の 2.で解凍したフォルダおよびファイル

3.1.3　環境変数 CLASSPATH の指定

環境変数CLASSPATH に JAR ファイルの絶対パスを指定してください。

重要

- OS が Windows の場合，CLASSPATH はシステム環境変数に設定してください。
- JAR ファイルの格納先を変更した場合は，CLASSPATH の指定も変更してください。

3.1.4　環境変数 TZ の指定値の確認

環境変数TZ に正しいタイムゾーンが設定されていることを確認してください。

重要

- うるう秒対応のタイムゾーンは設定しないでください。
- OS が Windows の場合，TZ はシステム環境変数に設定してください。

3.1.5　トレースファイルの出力先ディレクトリに対する書き込み権限の付与

JDBC ドライバのトラブルシュート機能である JDBC インタフェースメソッドトレース，または Exception トレースログを使用する場合，トレースファイルの出力先フォルダ（ディレクトリ）への書き込み権限を，JDBC ドライバを使用するユーザに与える必要があります。

これらのトラブルシュート機能の詳細については，「7.7　トラブルシュート」を参照してください。

3.1.6　システムプロパティの設定

AP の実行環境をシステムプロパティで設定します。次の表に示すプロパティをシステムプロパティに指定できます。

重要

- 次の表の項番 1～項番 20 までのプロパティで，クライアント定義のオペランドと同じ指定ができます。
- 項番 21 以降のプロパティには，Exception トレースログに関する設定を指定します。

表 3-3　システムプロパティに指定できるプロパティ

項番	プロパティ名	説明
1	adb_clt_rpc_srv_host	接続先の HADB サーバのホスト名を指定します。 このプロパティの機能は，クライアント定義のadb_clt_rpc_srv_host オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_srv_host オペランドを参照してください。
2	adb_clt_rpc_srv_port	HADB クライアントと HADB サーバ間の通信で使用する，HADB サーバのポート番号を指定します。 このプロパティの機能は，クライアント定義のadb_clt_rpc_srv_port オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_srv_port オペランドを参照してください。
3	adb_clt_rpc_con_wait_time	HADB サーバへの接続処理の完了待ち時間を指定します。 このプロパティの機能は，クライアント定義のadb_clt_rpc_con_wait_time オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_con_wait_time オペランドを参照してください。

項番	プロパティ名	説明
4	adb_clt_rpc_sql_wait_time	次に示す待ち時間を指定します。 • HADB クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間 • 同一コネクションで複数のSELECT 文を同時実行した際に，処理リアルスレッド数不足が発生したときの処理リアルスレッドの確保処理の待ち時間 このプロパティを指定したときの待ち時間の監視の詳細については，「3.2 AP の無応答状態への対策」を参照してください。 このプロパティの機能は，クライアント定義のadb_clt_rpc_sql_wait_time オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_sql_wait_time オペランドを参照してください。
5	adb_clt_ap_name	HADB サーバに接続する AP の識別情報（AP 識別子）を指定します。 このプロパティの機能は，クライアント定義のadb_clt_ap_name オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_ap_name オペランドを参照してください。
6	adb_clt_group_name	AP が属するクライアントグループ名を指定します。 このプロパティの機能は，クライアント定義のadb_clt_group_name オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_group_name オペランドを参照してください。
7	adb_clt_fetch_size	1 回のFETCH 処理で，HADB サーバから HADB クライアントに送信する検索結果の行数を指定します。 このプロパティの機能は，クライアント定義のadb_clt_fetch_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_fetch_size オペランドを参照してください。
8	adb_dbbuff_wrktbl_clt_blk_num	ローカル作業表用バッファのページ数を指定します。 このプロパティの機能は，クライアント定義の adb_dbbuff_wrktbl_clt_blk_num オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_dbbuff_wrktbl_clt_blk_num オペランドを参照してください。
9	adb_sql_exe_max_rthd_num	SQL 実行時に使用する処理リアルスレッド数の最大値を指定します。 このプロパティの機能は，クライアント定義のadb_sql_exe_max_rthd_num オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_sql_exe_max_rthd_num オペランドを参照してください。
10	adb_sql_exe_hashgrp_area_size	ハッシュグループ化領域の大きさをキロバイト単位で指定します。 このプロパティの機能は，クライアント定義の adb_sql_exe_hashgrp_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashgrp_area_size オペランドを参照してください。
11	adb_sql_exe_hashtbl_area_size	ハッシュテーブル領域サイズをメガバイト単位で指定します。 このプロパティの機能は，クライアント定義の adb_sql_exe_hashtbl_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashtbl_area_size オペランドを参照してください。
12	adb_sql_exe_hashflt_area_size	ハッシュフィルタ領域サイズをメガバイト単位で指定します。

項番	プロパティ名	説明
		このプロパティの機能は，クライアント定義の adb_sql_exe_hashflt_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashflt_area_size オペランドを参照してください。
13	adb_clt_sql_parallel_exec	SQL パラレル実行機能を適用して検索系 SQL を実行するかどうかを指定します。 このプロパティの機能は，クライアント定義のadb_clt_sql_parallel_exec オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_parallel_exec オペランドを参照してください。
14	adb_sql_prep_delrsvd_use_srvdef	サーバ定義のadb_sql_prep_delrsvd_words オペランドの指定に従って予約語を削除するかどうかを指定します。 このプロパティの機能は，クライアント定義の adb_sql_prep_delrsvd_use_srvdef オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_prep_delrsvd_use_srvdef オペランドを参照してください。
15	adb_clt_trn_iso_lv	トランザクション隔離性水準を指定します。 このプロパティの機能は，クライアント定義のadb_clt_trn_iso_lv オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_trn_iso_lv オペランドを参照してください。
16	adb_clt_trn_access_mode	トランザクションアクセスモードを指定します。 このプロパティの機能は，クライアント定義のadb_clt_trn_access_mode オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_trn_access_mode オペランドを参照してください。
17	adb_clt_sql_text_out	HADB クライアントが発行した SQL 文を，クライアントメッセージログファイルおよびサーバメッセージログファイルに出力するかどうかを指定します。 このプロパティの機能は，クライアント定義のadb_clt_sql_text_out オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_text_out オペランドを参照してください。
18	adb_clt_sql_order_mode	ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を指定します。 このプロパティの機能は，クライアント定義のadb_clt_sql_order_mode オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_order_mode オペランドを参照してください。
19	adb_sql_prep_dec_div_rs_prior	SQL 文中に指定した除算（四則演算）の結果のデータ型がDECIMAL 型の場合，除算結果の位取りの最小値を指定します。 このプロパティの機能は，クライアント定義の adb_sql_prep_dec_div_rs_prior オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_prep_dec_div_rs_prior オペランドを参照してください。
20	adb_clt_passwd_pubkey_path	ユーザ認証方式に PAM 認証を使用する場合，パスワードの暗号化に使用する公開鍵ファイルの絶対パスを指定します。

項番	プロパティ名	説明	
		このプロパティの機能は，クライアント定義のadb_clt_passwd_pubkey_pathオペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_passwd_pubkey_pathオペランドを参照してください。	
21	adb_jdbc_exc_trc_out_path	Exceptionトレースログを出力するディレクトリを，絶対パスで指定します。	これらのプロパティの機能詳細および指定できる値については，「7.7.2　Exceptionトレースログ」の「(1)　取得するメソッド，および取得するための設定」の「(b)　Exceptionトレースログを取得するための設定（プロパティの設定）」を参照してください。
22	adb_jdbc_info_max	1ファイルへ出力する情報数の上限を指定します。	
23	adb_jdbc_cache_info_max	メモリ内で蓄積する情報数の上限を指定します。	
24	adb_jdbc_trc_out_lv	トレース取得レベルを指定します。	

重要

ユーザプロパティまたは接続用のURLのプロパティでも，上記の表に示すプロパティの値を指定できます。

ユーザプロパティについては，「7.3.1　DriverManagerクラスのgetConnectionメソッドでHADBサーバに接続する方法」の「(2)　getConnectionメソッドによるHADBサーバへの接続」の「(d)　引数infoの指定内容（ユーザプロパティの指定）」を参照してください。

接続用のURLのプロパティについては，「7.3.1　DriverManagerクラスのgetConnectionメソッドでHADBサーバに接続する方法」の「(2)　getConnectionメソッドによるHADBサーバへの接続」の「(a)　引数urlの指定内容（接続用のURLの指定）」を参照してください。

各指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

メモ

HADB 03-00で，システムプロパティの各プロパティ名を次のように変更しました。変更前のプロパティ名も使用できますが，HADB 03-00以降にバージョンアップした場合は，プロパティ名を変更することを推奨します。

項番	変更前のプロパティ名（HADB 03-00より前のプロパティ名）	変更後のプロパティ名（HADB 03-00以降のプロパティ名）
1	adb_jdbc_ap_name	adb_clt_ap_name
2	adb_jdbc_dbbuff_wrktbl_blk_num	adb_dbbuff_wrktbl_clt_blk_num
3	adb_jdbc_fetch_size	adb_clt_fetch_size
4	adb_jdbc_rpc_sql_wait_time	adb_clt_rpc_sql_wait_time
5	adb_jdbc_rpc_srv_host	adb_clt_rpc_srv_host

項番	変更前のプロパティ名（HADB 03-00 より前のプロパティ名）	変更後のプロパティ名（HADB 03-00 以降のプロパティ名）
6	adb_jdbc_rpc_srv_port	adb_clt_rpc_srv_port
7	adb_jdbc_sql_delrsvd_use_srvdef	adb_sql_prep_delrsvd_use_srvdef
8	adb_jdbc_sql_hashgrp_area_size	adb_sql_exe_hashgrp_area_size
9	adb_jdbc_sql_hashtbl_area_size	adb_sql_exe_hashtbl_area_size
10	adb_jdbc_sql_max_rthd_num	adb_sql_exe_max_rthd_num
11	adb_jdbc_sql_order_mode	adb_clt_sql_order_mode
12	adb_jdbc_sql_text_out	adb_sql_text_out
13	adb_jdbc_trn_access_mode	adb_clt_trn_access_mode
14	adb_jdbc_trn_iso_lv	adb_clt_trn_iso_lv

3.1.7　ウイルス対策ソフトによるスキャン対象の見直し

JDBC ドライバをインストールしたマシンに，ウイルス対策ソフトがインストールされている場合は，スキャン対象を見直してください。ウイルス対策ソフトのスキャン対象に，JDBC ドライバが使用するディレクトリやファイルが含まれている場合，JDBC ドライバが正常に動作しないおそれがあります。そのため，ウイルス対策ソフトのスキャン対象から，JDBC ドライバが使用するディレクトリやファイルを除いてください。

3.2 AP の無応答状態への対策

通信障害，瞬断を含む一時的な障害，またはディスク障害などによって，AP が無応答状態になることがあります。無応答状態の AP が発生した場合，その影響によってほかの AP やコマンドの処理が停滞することがあります。

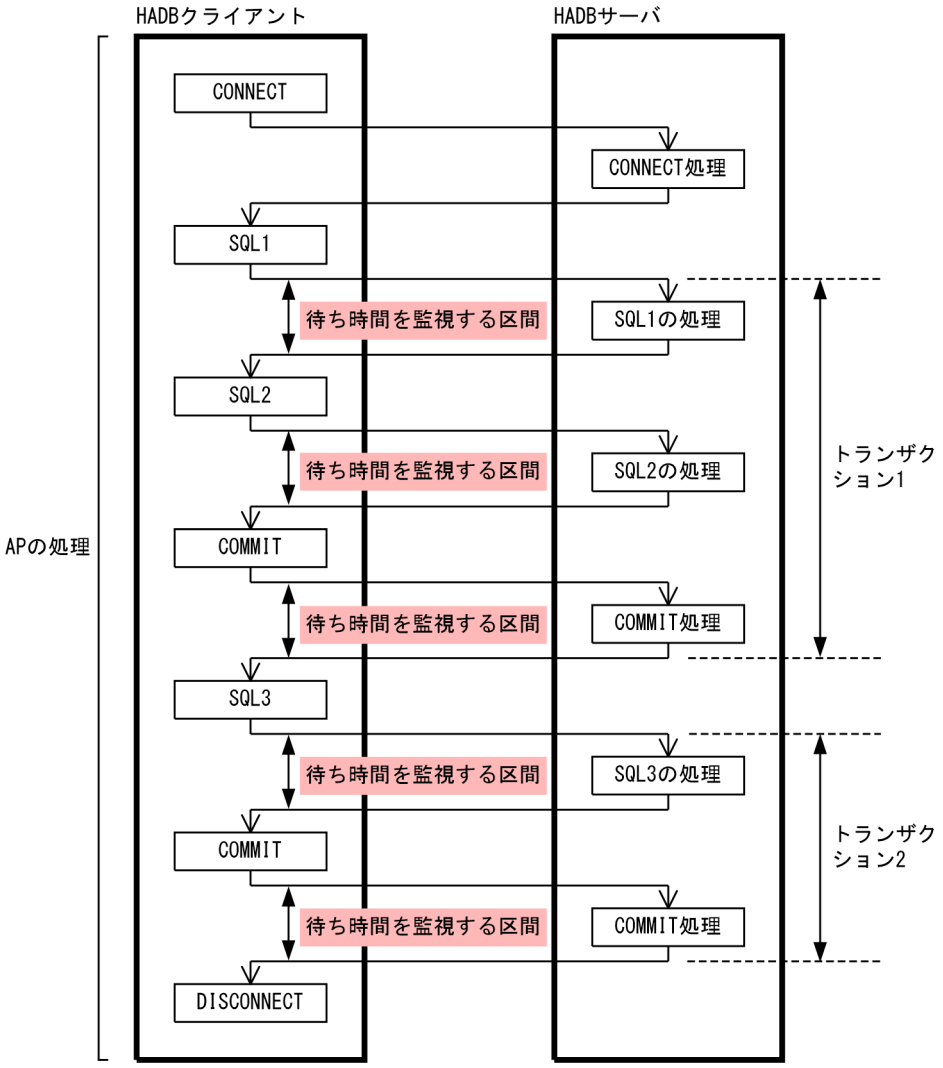
そこで，無応答状態の AP が発生したときに，その影響をなるべく小さくするために，システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティで次のプロパティを指定してください。

- adb_clt_rpc_sql_wait_time

このプロパティ値には，クライアント（JDBC ドライバをインストールしたマシン）から HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間を指定します。ここで指定した待ち時間を超えても HADB サーバから応答がない場合，SQLCODE が-732（KFAA30732-E）のタイムアウトエラーが AP に返されます。このとき，SQL 文の処理はキャンセルされ，トランザクションはロールバックされます。そのあと，HADB サーバから AP が切り離されます。

adb_clt_rpc_sql_wait_time による待ち時間の監視イメージを次の図に示します。

図 3-1　adb_clt_rpc_sql_wait_time による待ち時間の監視イメージ



[説明]

クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間を監視しています。例えば，adb_clt_rpc_sql_wait_time に 600 秒を指定した場合，各監視区間に対して 600 秒の待ち時間が設定されます。したがって，処理時間がいちばん長い SQL 文を目安にして，待ち時間を指定します。待ち時間には，AP が無応答状態になった可能性が高いと考えられる時間を指定してください。

なお，adb_clt_rpc_sql_wait_time を指定したときは，さらに次の待ち時間も監視対象になります。

- 同一コネクションで複数のSELECT 文を同時実行した際に，処理リアルスレッド数不足が発生したときの処理リアルスレッドの確保処理の待ち時間

上記の待ち時間を超えた場合，SQLCODE が-1071570（KFAA71570-E）のタイムアウトエラーが AP に返されます。このとき，SQL 文の処理はキャンセルされますが，トランザクションはロールバックされません。また，AP は HADB サーバから切り離されません。

adb_clt_rpc_sql_wait_time を指定する目的については，「7.4.1　データの検索方法」の「(4)　同一コネクションで複数の SELECT 文を同時実行する際の注意事項」を参照してください。

> **メモ**
>
> adb_clt_rpc_sql_wait_time については，「3.1.6　システムプロパティの設定」を参照してください。

3.3　JDBC ドライバのバージョンアップ（JAR ファイルの差し替え）

JDBC ドライバのバージョンアップ手順を次に示します。

手順

1. HADB クライアントのインストール CD-ROM に格納されている圧縮ファイルを任意のフォルダにコピーする

   次のファイルをコピーします。

   - 64 ビット版の Windows の場合

     `hitachi_advanced_data_binder_client.zip`

   - 32 ビット版の Windows の場合

     `hitachi_advanced_data_binder_client32.zip`

   - Linux の場合

     `hitachi_advanced_data_binder_client-$VER.tar.gz`

   `$VER` は，HADB のバージョンおよびリリース番号になります。

2. コピーした圧縮ファイルを解凍する

   圧縮ファイル解凍後の JAR ファイルの格納フォルダについては，「表 3-2　圧縮ファイル解凍後の JAR ファイルの格納先」を参照してください。

3. JAR ファイルを既存の JAR ファイルと差し替える

4. HADB サーバへの接続確認をする

5. 次のフォルダおよびファイルを削除する

   - 手順の 1.でコピーした圧縮ファイル
   - 手順の 2.で解凍したフォルダおよびファイル

> **重要**
>
> 上記の 1.～3.の作業を実行する OS ユーザを途中で変更しないでください。変更した場合，バージョンアップが正しくできないことがあります。

■バージョンアップ後の確認事項

JDBC ドライバをバージョンアップすると，プロパティ※のデフォルト値が変更されることがあります。各プロパティのデフォルト値が変更されているかどうかを確認してください。デフォルト値が変更されている場合は，必要に応じてプロパティを指定してください。 各プロパティのデフォルト値は，クライアント環境定義の各オペランドのデフォルト値と同じになります。クライアント定義の各オペランドのデフォルト値については，「2.2　クライアント定義のオペランドの内容」を参照してください。

注※

「表 3-3　システムプロパティに指定できるプロパティ」のプロパティが該当します。

■JDBC ドライバを旧バージョンに戻す場合

手順を次に示します。

手順

1. JAR ファイルを以前使用していた JAR ファイルに戻す
2. システムプロパティの指定値を戻す

   JDBC ドライバをバージョンアップした際，システムプロパティの値を変更している場合は，バージョンダウンするときに，旧バージョンの指定値に戻す必要があります。
3. HADB サーバへの接続確認をする

> **重要**
>
> 上記の 1.～2.の作業を実行する OS ユーザを途中で変更しないでください。変更した場合，JDBC ドライバが正しく動作しないことがあります。

3.4　修正版の JDBC ドライバとの入れ替え

修正版の JDBC ドライバとの入れ替え手順は，JDBC ドライバのバージョンアップ手順と同じになります。手順については，「3.3　JDBC ドライバのバージョンアップ（JAR ファイルの差し替え）」を参照してください。

3.5　JDBC ドライバをインストールしたマシンの OS の時刻変更

JDBC ドライバをインストールしたマシンの OS の時刻を変更する手順については，「4.9　クライアントマシンの OS の時刻変更」を参照してください。その際，クライアントマシンは，「**JDBC ドライバをインストールしたマシン**」と読み替えてください。

3.6　JDBC ドライバのアンインストール

JDBC ドライバのアンインストール手順を次に示します。

手順

1. JAR ファイルを削除する
2. 環境変数CLASSPATH の指定を削除する

4 HADB クライアントの環境設定（ODBC ドライバおよび CLI 関数を使用する場合）

この章では，HADB クライアントのインストールや環境変数の設定など，HADB クライアントの環境設定方法について説明します。

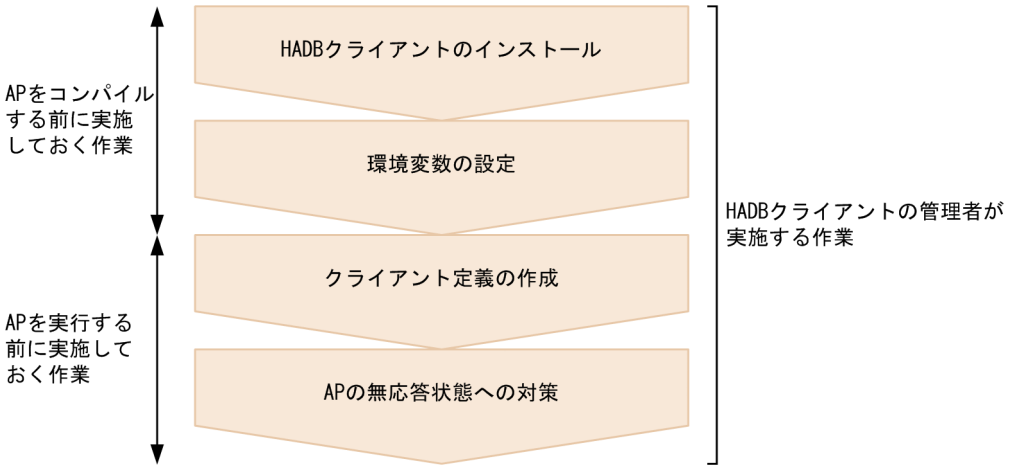
4.1　HADB クライアントの環境設定手順

HADB クライアントの環境設定手順について説明します。

4.1.1　Windows 版の HADB クライアントの場合

HADB サーバ以外のマシンで AP を作成または実行する場合，そのマシンを HADB クライアントとして運用します。Windows 版の HADB クライアントの環境設定手順を次に示します。

図 4-1　Windows 版の HADB クライアントの環境設定手順



各手順の参照先

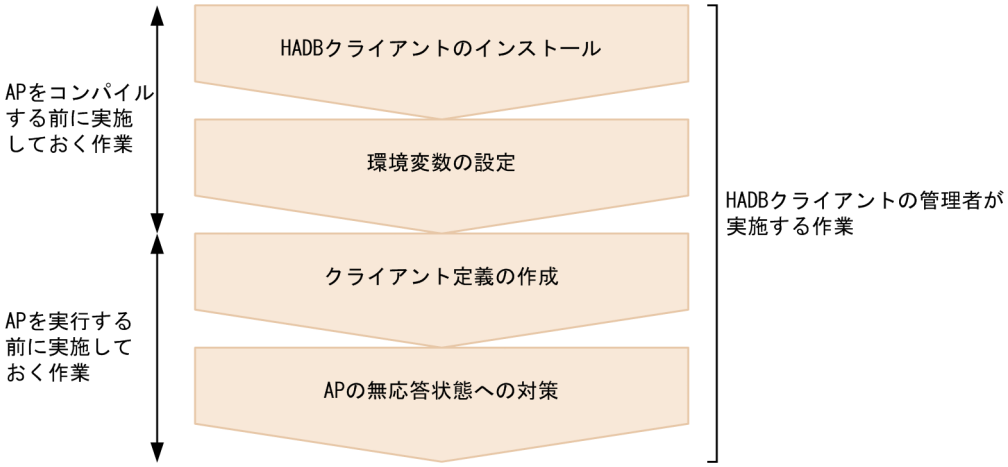
- HADB クライアントのインストールについては，「4.2.1　Windows 版の HADB クライアントの場合」を参照してください。
- 環境変数の設定については，「4.3.1　Windows 版の HADB クライアントの場合」を参照してください。
- クライアント定義の作成については，「4.4　クライアント定義の作成」を参照してください。
  クライアント定義の各オペランドの説明については，「2.　クライアント定義の設計」を参照してください。
- AP の無応答状態への対策については，「4.5　AP の無応答状態への対策」を参照してください。

なお，HADB クライアントをインストールする前に，HADB クライアントのメモリ所要量を計算してください。HADB クライアントのメモリ所要量の計算式については，「付録 C　HADB クライアントのメモリ所要量の見積もり」を参照してください。

4.1.2　Linux 版の HADB クライアントの場合

HADB サーバ以外のマシンで AP を作成または実行する場合，そのマシンを HADB クライアントとして運用します。Linux 版の HADB クライアントの環境設定手順を次に示します。

図 4-2　Linux 版の HADB クライアントの環境設定手順



各手順の参照先

- HADB クライアントのインストールについては，「4.2.2　Linux 版の HADB クライアントの場合」を参照してください。
- 環境変数の設定については，「4.3.2　Linux 版の HADB クライアントの場合」を参照してください。
- クライアント定義の作成については，「4.4　クライアント定義の作成」を参照してください。
  クライアント定義の各オペランドの説明については，「2.　クライアント定義の設計」を参照してください。
- AP の無応答状態への対策については，「4.5　AP の無応答状態への対策」を参照してください。

なお，HADB クライアントをインストールする前に，HADB クライアントのメモリ所要量を計算してください。HADB クライアントのメモリ所要量の計算式については，「付録 C　HADB クライアントのメモリ所要量の見積もり」を参照してください。

4.2　HADB クライアントのインストールおよびアンインストール

HADB クライアントのインストール方法およびアンインストール方法について説明します。

4.2.1　Windows 版の HADB クライアントの場合

ここでは，Windows 版の HADB クライアントのインストールおよびアンインストールについて説明します。

HADB クライアントをインストールまたはアンインストールするには，管理者特権が必要です。

(1)　HADB クライアントのインストール

HADB クライアントのインストール手順を次に示します。

メモ

- HADB サーバのバージョンに対応している HADB クライアントをインストールしてください。
- HADB クライアントのインストール CD-ROM に格納されているインストールデータを使用してください。別の CD-ROM に格納されている zip 形式のファイルなどを使用してインストールを実行しないでください。

手順

1. HADB クライアントを管理する OS ユーザでログインする
2. HADB クライアントのインストールデータをコピーする
   HADB クライアントのインストール CD-ROM に格納されている次の圧縮ファイル（zip 形式ファイル）を，クライアントマシンの任意のフォルダにコピーしてください。この手順の説明では，`D:¥hadb_clt` フォルダにコピーするものとします。
   - 64 ビット版の HADB クライアントを使用する場合：`hitachi_advanced_data_binder_client.zip`
   - 32 ビット版の HADB クライアントを使用する場合：`hitachi_advanced_data_binder_client32.zip`

   コピー先のフォルダは，パス長が 200 バイト以内のフォルダにしてください。
   コピー先のフォルダのパス名に使用できる文字については，『はじめに』の「■このマニュアルで使用する構文要素記号」の『〈パス名〉』を参照してください。
3. コピーしたファイルのハッシュ値を取得する
   コマンドプロンプトを起動して，次のコマンドを実行してください。
   - 64 ビット版の HADB クライアントを使用する場合

```
cd /D D:¥hadb_clt
certutil -hashfile hitachi_advanced_data_binder_client.zip SHA256
```

- 32 ビット版の HADB クライアントを使用する場合

```
cd /D D:¥hadb_clt
certutil -hashfile hitachi_advanced_data_binder_client32.zip SHA256
```

下線部分は，手順 2.でインストールデータをコピーしたフォルダです。

4. ハッシュ値を比較する

手順 3.で取得したハッシュ値と，インストール CD-ROM に格納されている次のファイルに記載されているハッシュ値を比較してください。

- 64 ビット版の HADB クライアントを使用する場合：adbhashfile_winclt64_sha256.txt
- 32 ビット版の HADB クライアントを使用する場合：adbhashfile_winclt32_sha256.txt

ハッシュ値が同じ場合は，問題ありません。次の手順に進んでください。

ハッシュ値が異なる場合は，ファイルのコピーが失敗しているおそれがあります。手順 2.から作業をし直してください。

5. 圧縮ファイルを解凍する

（例）hitachi_advanced_data_binder_client.zip を解凍した場合

D:¥hadb_clt¥hitachi_advanced_data_binder_client¥HADBCL フォルダ下に，HADB クライアントのフォルダおよびファイルが展開されます。

D:¥hadb_clt¥hitachi_advanced_data_binder_client¥HADBCL フォルダは，圧縮ファイルを解凍したときのデフォルトの解凍先です。

6. HADB クライアントのバージョン情報を確認する

解凍先フォルダ下に格納されているreadme.txt ファイルを開いてください。

1 行目に記載されている HADB クライアントのバージョン情報を見て，次のどちらかの情報が記載されていることを確認してください。

- P-2462-C114 *vv-rr*
- P-2462-C114 *vv-rr-/s*

vv-rr および *vv-rr-/s* は，HADB クライアントのバージョンを示しています。

重要

ODBC ドライバを使用する場合，HADB クライアントと ODBC ドライバのバージョン情報が一致している必要があります。そのため，クライアントディレクトリ下のclient¥bin フォルダ内に格納されている dll の一部差し替えはしないでください。

■インストール後の作業

インストール後，次の作業を実施してください。

- 環境変数の設定

  設定する必要がある環境変数については，「4.3.1　Windows 版の HADB クライアントの場合」を参照してください。

  なお，環境変数ADBCLTDIR には，クライアントディレクトリの絶対パスを指定します。上記のインストールの手順で説明した例の場合，D:¥hadb_clt¥hitachi_advanced_data_binder_client¥HADBCL がクライアントディレクトリになります。

  クライアントディレクトリとは，1 つのクライアントプロセスに関するファイル群を格納したフォルダのことです。

- レジストリキーの登録

  環境変数の設定後，次のレジストリ登録コマンドを実行してレジストリキーを登録してください。

  ・64 ビット版の HADB クライアントを使用する場合：%ADBCLTDIR%¥adbreg.reg

  ・32 ビット版の HADB クライアントを使用する場合：%ADBCLTDIR%¥adbreg32.reg

- Microsoft Visual C++ライブラリのランタイムコンポーネントのインストール

  HADB クライアント（ODBC ドライバを含む）を使用する可能性がある OS ユーザは，次に示す Microsoft Visual Studio 2019 の Microsoft Visual C++再頒布可能パッケージを実行して，Microsoft Visual C++ライブラリのランタイムコンポーネントをインストールしてください。

  ・64 ビット版の HADB クライアントを使用する場合：%ADBCLTDIR%¥vclib¥VC_redist.x64.exe

  ・32 ビット版の HADB クライアントを使用する場合：%ADBCLTDIR%¥vclib32¥VC_redist.x86.exe

- フォルダへの書き込み権限の付与

  HADB クライアント（ODBC ドライバを含む）を使用する可能性がある OS ユーザに対して，次に示すフォルダへの書き込み権限を与えてください。

  ・%ADBCLTDIR%¥spool

  ・%ADBODBTRCPATH%

  このフォルダは，64 ビット版の HADB クライアントと 32 ビット版の HADB クライアントで共通です。

- ウイルス対策ソフトによるスキャン対象の見直し

  HADB クライアントをインストールしたクライアントマシンに，ウイルス対策ソフトがインストールされている場合は，スキャン対象を見直してください。

  - ウイルス対策ソフトのスキャン対象に，HADB クライアントが使用するディレクトリやファイルを設定した場合，HADB クライアントが正常に動作しないおそれがあります。そのため，ウイルス対策ソフトのスキャン対象から，クライアントディレクトリを除いてください。
  - PAM 認証を使用している場合は，ウイルス対策ソフトのスキャン対象から，公開鍵ファイルの保存先ディレクトリを除いてください。

(2)　HADB クライアントのアンインストール

HADB クライアントをアンインストールする前に次のことをしてください。

- HADB クライアントから実行しているコマンドまたは AP がないか確認してください。コマンドまたは AP の実行中に HADB クライアントのアンインストールを実行すると，アンインストールが失敗するおそれがあります。
- クライアントディレクトリ下に必要なファイルがある場合は，バックアップを取得してください。

HADB クライアントのアンインストール手順を次に示します。

手順

1. レジストリキーを削除する
   インストール時に登録したレジストリキーを，次のレジストリ削除コマンドを実行して削除してください。
   - 64 ビット版の HADB クライアントを使用している場合：`%ADBCLTDIR%¥adbunreg.reg`
   - 32 ビット版の HADB クライアントを使用している場合：`%ADBCLTDIR%¥adbunreg32.reg`
2. HADB クライアントのインストール時にコピーしたすべてのフォルダおよびファイルを削除する
   インストール後に作成したフォルダおよびファイルも削除してください。

4.2.2 Linux 版の HADB クライアントの場合

ここでは，Linux 版の HADB クライアントのインストールおよびアンインストールについて説明します。

重要

- HADB クライアントのインストールは，HADB クライアントを管理する OS ユーザが実行します。
- インストールを実行する OS ユーザのユーザ名は 32 バイト以内にしてください。

なお，以降の説明で出てくるクライアントディレクトリとは，1 つのクライアントプロセスに関するファイル群を格納したディレクトリのことです。

(1) HADB クライアントをインストールする前の確認事項

HADB クライアントをインストールする前に，HADB クライアントが正しく動作するために必要なパッケージが OS にインストールされていることを確認してください。必要なパッケージを次の表に示します。

表 4-1 必要なパッケージ

項番	パッケージ名	Linux のバージョン		
		RHEL 7	RHEL 8	RHEL 9
1	compat-openssl10	−	○	−

項番	パッケージ名	Linux のバージョン		
		RHEL 7	RHEL 8	RHEL 9
2	glibc	○	○	○
3	libaio	○	○	○
4	libuuid	○	○	○
5	openssl-libs	○	－	○
6	zlib	○	○	○

（凡例）

○：この Linux のバージョンでは必要なパッケージです。

－：この Linux のバージョンでは不要なパッケージです。

■インストール済みのパッケージを確認する方法

次のコマンドを実行すると，OS にインストール済みのパッケージの一覧が表示されます。

```
yum list installed
```

インストールされていないパッケージがある場合は，そのパッケージを OS にインストールしてください。パッケージのインストール方法については，OS のマニュアルを参照してください。

■特定のパッケージがインストール済みかどうかを確認する方法

（例）

パッケージlibaio がインストール済みかどうかを確認します。

```
yum list installed | grep libaio
```

実行結果にパッケージlibaio が表示された場合は，パッケージlibaio はインストール済みです。実行結果にパッケージlibaio が表示されなかった場合は，パッケージlibaio はインストールされていません。

(2) HADB クライアントのインストール

HADB クライアントのインストール手順を次に示します。

メモ

- HADB サーバのバージョンに対応している HADB クライアントをインストールしてください。
- HADB クライアントのインストールデータには，RHEL 7 および RHEL 8 用のインストールデータと，RHEL9 用のインストールデータの 2 種類があります。使用している Linux のバージョンに対応するインストールデータを使用して，HADB クライアントのインストールを実行してください。特に，RHEL 7 および RHEL 8 用のインストールデータと，RHEL9

用のインストールデータの両方を持っている場合は，インストールデータを間違えないようにご注意ください。

- HADB クライアントのインストール CD-ROM に格納されているインストールデータを使用してください。別の CD-ROM に格納されている tar.gz 形式のファイルなどを使用してインストールを実行しないでください。

手順

1. HADB クライアントを管理する OS ユーザでログインする
2. インストールデータを格納するディレクトリを作成する

```
mkdir /home/osuser01/client
```

上記の例では，HADB クライアントのインストールデータを格納するディレクトリとして/home/osuser01/client を作成しています。

3. インストールデータを格納するディレクトリに対して書き込み権限を付与する

```
chmod 755 /home/osuser01/client
```

インストールデータを格納するディレクトリに対して，HADB クライアントを管理する OS ユーザが書き込みできるように，ディレクトリ（この例では/home/osuser01/client）に対して書き込み権限を付与してください。

4. CD-ROM ファイルシステムをマウントする

HADB クライアントのインストールデータが格納されている CD-ROM ファイルシステムを自動マウントしてください。

CD-ROM ファイルシステムが自動マウントされない場合，次のコマンドを実行して CD-ROM ファイルシステムをマウントしてください。

```
mount /dev/cdrom /media
```

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

重要

CD-ROM のディレクトリ名やファイル名は，ご使用のマシンによっては，ここの説明内容と表示が異なることがあります。OS の ls コマンドを実行して，表示されたディレクトリ名をそのまま入力してください。

5. インストールデータをコピーする

マウントした CD-ROM ファイルシステムに格納されている次のファイルを，手順 2.で作成したインストールデータを格納するディレクトリにコピーしてください。

- tar.gz 形式のファイル
- adbinstall コマンドの実行形式ファイル

```
cp /media/hitachi_advanced_data_binder_client-$VER.tar.gz /home/osuser01/client   ...a
cp /media/adbinstall /home/osuser01/client                                        ...b
```

[説明]

a. tar.gz 形式のファイル（HADB クライアントのプログラムの圧縮ファイル）をコピーします。

b. adbinstall コマンドの実行形式ファイル（HADB クライアントのインストールコマンド）をコピーします。

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

$VER は，HADB クライアントのバージョンおよびリリース番号です。

> **重要**
>
> 上記の 2 つのファイルは，必ず同じディレクトリにコピーしてください。異なるディレクトリにコピーすると，HADB クライアントがインストールできません。

6. コピーしたファイルに破損がないことを確認する

手順 5.でコピーしたファイルに破損がないことを確認します。次のコマンドを実行してください。

```
cd /home/osuser01/client
sha256sum -c /media/adbhashfile_client_sha256.txt
```

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

ファイルが破損しているおそれがある（ファイルのハッシュ値が異なる）場合は，エラーを示すメッセージが出力されます。この場合，手順 5.から作業をし直してください。

出力されるメッセージの詳細については，OS のマニュアルを参照してください。

7. 権限を付与する

HADB クライアントを管理する OS ユーザがadbinstall コマンド（HADB クライアントのインストールコマンド）を実行できるように，adbinstall コマンドに対する実行権限を付与してください。

また，tar.gz 形式のファイルに対する読み取り権限も付与してください。

```
chmod 555 /home/osuser01/client/adbinstall
chmod 444 /home/osuser01/client/hitachi_advanced_data_binder_client-$VER.tar.gz
```

$VER は，HADB クライアントのバージョンおよびリリース番号です。

8. HADB クライアントをインストールする

adbinstall コマンドを実行して，HADB クライアントをインストールしてください。

```
/home/osuser01/client/adbinstall -c /home/osuser01/clientdir
```

下線部分には，手順 2.で作成したインストールデータを格納するディレクトリを指定します。

-c オプションに指定したディレクトリ下（上記の場合，/home/osuser01/clientdir ディレクトリ下）に，HADB クライアントがインストールされます。このディレクトリが，クライアントディレクトリになります。

> **メモ**
>
> -c オプションに指定したディレクトリが存在しないときは，adbinstall コマンドを実行すると，自動的にディレクトリが作成されます。

クライアントディレクトリに関する規則を次に示します。

- クライアントディレクトリのパス長は，118 バイト以内になるようにしてください。
- クライアントディレクトリのパス名に使用できる文字については，『はじめに』の「■このマニュアルで使用する構文要素記号」の『〈パス名〉』を参照してください。
- adbinstall コマンドの-c オプションで指定するディレクトリには，必ず HADB クライアントを管理する OS ユーザが書き込みできるディレクトリを指定してください。

9. HADB クライアントが正常にインストールされたことを確認する

クライアントディレクトリ下のadbinstcl.log ファイルを開いて内容を確認してください。HADB クライアントが正常にインストールされている場合は，次のどちらかの情報が記載されています。

- Hitachi Advanced Data Binder Client *vv-rr* (*yyyymmddhhmmss*)
- Hitachi Advanced Data Binder Client *vv-rr-/s* (*yyyymmddhhmmss*)

vv-rr および *vv-rr-/s* は，HADB クライアントのバージョンを示しています。

yyyymmddhhmmss は，HADB クライアントのインストールを実行した日付と時刻を示しています。

> **メモ**
>
> - adbinstcl.log ファイルがすでにある状態のときにadbinstall コマンドを実行すると，adbinstcl.log ファイルの内容が更新されます。また，adbinstall コマンドを複数回実行した場合，最後に実行した日付と時刻が記載されます。
> - adbinstall コマンドのリターンコード（KFAA91552-I メッセージに出力されるリターンコード）が0 または4 以外の場合，adbinstcl.log ファイルの内容は更新されません。
> - adbinstall コマンドの実行時にKFAA91555-I メッセージが出力された場合，adbinstcl.log ファイルの内容は更新されません。

> **重要**
>
> ODBC ドライバを使用する場合，HADB クライアントと ODBC ドライバのバージョン情報が一致している必要があります。そのため，クライアントディレクトリ下のclient¥bin フォルダ内に格納されている dll の一部差し替えはしないでください。

■KFAA91553-E メッセージが出力された場合の対処方法

HADB クライアントを管理する OS ユーザが書き込みできないディレクトリを，adbinstall コマンドの-c オプションに指定した場合，KFAA91553-E メッセージが出力されます。

また，インストールデータを格納するディレクトリに，HADB クライアントを管理する OS ユーザが書き込みできない場合も，KFAA91553-E メッセージが出力されます。

KFAA91553-E メッセージが出力された場合は，対象のディレクトリに書き込み権限を付与してください。

■KFAA91558-W メッセージが出力された場合の対処方法

HADB クライアントを管理する OS ユーザではなく，root でadbinstall コマンドを実行した場合，KFAA91558-W メッセージが出力されます。

通常は，HADB クライアントを管理する OS ユーザで，adbinstall コマンドを実行します。そのため，KFAA91558-W メッセージが出力された場合は，root でadbinstall コマンドを実行しても問題がないかどうかを確認してください。

問題がある場合は，KFAA91558-W メッセージが出力されたあとに出力されるKFAA91559-Q メッセージの入力要求に対して，n（またはN）を入力してください。そのあとで，HADB クライアントを管理する OS ユーザでadbinstall コマンドを実行してください。

> **メモ**
>
> - root 以外のスーパユーザでadbinstall コマンドを実行した場合は，KFAA91558-W メッセージは出力されません。
> - root は，OS のid -u コマンドを実行して表示される値が0 のユーザを指します。また，OS のsu コマンドを使用して，ほかの OS ユーザからroot に切り替えたあとで，OS の id -u コマンドを実行して表示される値が0 のときも含みます。

■インストール後の作業

- 環境変数の設定

  インストール後，環境変数を設定してください。設定する必要がある環境変数については，「4.3.2 Linux 版の HADB クライアントの場合」を参照してください。

  なお，環境変数ADBCLTDIR には，クライアントディレクトリの絶対パスを指定します。上記のインストールの手順で説明した例の場合，/home/osuser01/clientdir がクライアントディレクトリになります。

- ウイルス対策ソフトによるスキャン対象の見直し

  HADB クライアントをインストールしたクライアントマシンに，ウイルス対策ソフトがインストールされている場合は，スキャン対象を見直してください。

  - ウイルス対策ソフトのスキャン対象に，HADB クライアントが使用するディレクトリやファイルを設定した場合，HADB クライアントが正常に動作しないおそれがあります。そのため，ウイルス対策ソフトのスキャン対象から，クライアントディレクトリを除いてください。
  - PAM 認証を使用している場合は，ウイルス対策ソフトのスキャン対象から，公開鍵ファイルの保存先ディレクトリを除いてください。

メモ

adbinstall コマンドで HADB クライアントをインストールしたときのクライアントディレクトリの構成については，「付録 B.2　Linux 版の HADB クライアントの場合」の「(1)　クライアントディレクトリ（インストール時）の構成」を参照してください。

(3)　HADB クライアントのアンインストール

HADB クライアントのアンインストールは，インストールを実施した OS ユーザが実行してください。

アンインストール前に次のことをしてください。

- HADB クライアントから実行しているコマンドまたは AP がないか確認してください。コマンドまたは AP の実行中に HADB クライアントのアンインストールを実行すると，アンインストールが失敗するおそれがあります。
- クライアントディレクトリ下に必要なファイルがある場合は，バックアップを取得してください。

HADB クライアントのアンインストール手順を次に示します。

手順

1. クライアントディレクトリを削除する
   インストールを実施した OS ユーザで，クライアントディレクトリを削除します。

```
rm -rf /home/osuser01/clientdir
```

2. インストールデータを格納するディレクトリを削除する
   インストールを実施した OS ユーザで，インストールデータを格納するディレクトリを削除します。

```
rm -rf /home/osuser01/client
```

3. インストール時に設定した環境変数の指定を削除する

4.3　環境変数の設定

HADB クライアントで設定する環境変数について説明します。

4.3.1　Windows 版の HADB クライアントの場合

Windows 版の HADB クライアントで次の表に示す環境変数を設定してください。

表 4-2　環境変数に設定する値

項番	環境変数	設定する値
1	PATH	この環境変数には，次に示すフォルダを追加してください。 64 ビット版の HADB クライアントを使用する場合： • %ADBCLTDIR%¥client¥bin 32 ビット版の HADB クライアントを使用する場合： • %ADBCLTDIR%¥client¥bin この環境変数はシステム環境変数に設定してください。
2	TZ	この環境変数には，HADB クライアントをインストールするマシンのタイムゾーンを設定します。 この環境変数はシステム環境変数に設定してください。 なお，うるう秒対応のタイムゾーンは設定しないでください。
3	ADBCLTDIR	クライアントディレクトリの絶対パスを指定してください。レジストリ登録または削除コマンドを実施する前に，この環境変数を必ず設定しておいてください。 パスのうち，最初の 3 文字は次の規則に従って設定してください。 • 1 文字目 ドライブを表す英字 • 2 文字目 :（コロン） • 3 文字目 /（スラッシュ）または¥（バックスラッシュ）
4	ADBCLTLANG	この環境変数には，HADB クライアントで使用する文字コードを指定します。HADB サーバで使用する文字コードと同じ文字コードを指定してください。 HADB サーバの環境変数ADBLANG で指定した文字コードと同じ文字コードを指定します。 • HADB サーバで Unicode（UTF-8）を使用する場合 この環境変数にUTF8 を指定してください。 • HADB サーバで Shift-JIS を使用する場合 この環境変数にSJIS を指定してください。
5	ADBMSGLOGSIZE	この環境変数には，1 ファイル当たりのクライアントメッセージログファイルの最大容量をメガバイト単位で指定します。指定できる範囲は1～2,000 です。

項番	環境変数	設定する値
		HADB クライアントには，4 つのクライアントメッセージログファイルが作成されます。この環境変数を省略した場合，1 つ当たりのクライアントメッセージログファイルの最大容量は 16 メガバイトになります。 なお，1 つのクライアントディレクトリから複数のクライアントプロセスを起動する場合，この環境変数の指定値は必ず同じ値を指定してください。異なる値で複数のクライアントプロセスを動作させた場合，クライアントメッセージログファイルが壊れるおそれがあります。
6	ADBODBTRC	HADB ODBC ドライバトレース情報を出力するかどうかを指定します。この環境変数には，次のどちらかを指定してください。 • YES：HADB ODBC ドライバトレース情報を出力する場合に指定します。 • NO：HADB ODBC ドライバトレース情報を出力しない場合に指定します。 この環境変数を省略した場合，または不正な値を指定した場合は，NO が仮定されます。 この環境変数は，ODBC ドライバを使用する場合に設定を検討してください。
7	ADBODBTRCSIZE	この環境変数には，HADB ODBC ドライバトレースファイルの 1 ファイル当たりのサイズの上限値（単位：メガバイト）を指定します。32～1,024 の値を指定できます。 この環境変数を省略した場合，または不正な値を指定した場合は，256 が仮定されます。 この環境変数は，ODBC ドライバを使用する場合に設定を検討してください。
8	ADBODBTRCPATH	この環境変数には，HADB ODBC ドライバトレースファイルを格納するフォルダを絶対パスで指定します。 210 バイト以下のパス名を指定してください。 次に示す場合は，環境変数ADBODBTRC にYES を指定していても，HADB ODBC ドライバトレース情報は出力されません。 • この環境変数を省略した場合 • 不正なフォルダを指定した場合 • 211 バイト以上のパス名を指定した場合 この環境変数は，ODBC ドライバを使用する場合に設定を検討してください。
9	ADBODBTRCLV	HADB ODBC ドライバトレース情報を出力する際のトレースレベルを指定します。この環境変数には，次のどちらかを指定してください。 • 1：トレースレベル 1 とする場合に指定します。 • 2：トレースレベル 2 とする場合に指定します。 トレースレベルについては，「17.4.1　トレースレベルとは」を参照してください。 この環境変数を省略した場合，または不正な値を指定した場合は，1 が仮定されます。 この環境変数は，ODBC ドライバを使用する場合に設定を検討してください。
10	ADBODBAPMODE	この環境変数には，HADB ODBC ドライバのアプリケーションモードを指定します。通常，この環境変数を指定する必要はありません。 この環境変数の指定値によって HADB ODBC ドライバの処理が変わります。 • ACCESS：ODBC3.5 の規格ではなく，Microsoft Access 互換モードで動作します。

項番	環境変数	設定する値
		Microsoft Access を使用して HADB サーバにアクセスする場合に指定します。この環境変数を指定すると，検索結果が#Deleted となる，エラーが発生するなどの現象を回避できることがあります。 • NORMAL：通常どおり動作します。 この環境変数を省略した場合，または不正な値を指定した場合は，NORMAL が仮定されます。 この環境変数は，ODBC ドライバを使用する場合に設定を検討してください。
11	ADBODBSGDST	この環境変数には，HADB ODBC ドライバのSQLGetData 関数で，分割取得機能を有効にするかどうかを指定します。通常，この環境変数を指定する必要はありません。 この環境変数の指定値によって HADB ODBC ドライバの処理が変わります。 • USE：分割取得機能が有効になります。 • NOUSE：分割取得機能が無効になります。 この環境変数を省略した場合，または不正な値を指定した場合は，USE が仮定されます。 SQLGetData 関数を使用して取得した返却データの長さがユーザバッファのサイズより大きくて返却データの一部しか取得できなかった場合に，十分なサイズのユーザバッファを用意できるとき（再度SQLGetData 関数を発行して完全な返却データを取得できるとき）は，NOUSE を指定してください。 この環境変数は，ODBC ドライバを使用する場合に設定を検討してください。

4.3.2 Linux 版の HADB クライアントの場合

Linux 版の HADB クライアントで次の表に示す環境変数を設定してください。環境変数の設定は HADB クライアントを管理する OS ユーザが実行します。

環境変数に設定した値が HADB クライアント使用時のシェルで有効になるようにしてください。設定方法については，シェルのマニュアルを参照してください。

表 4-3 環境変数に設定する値

項番	環境変数	設定する値
1	LANG	この環境変数には，OS の文字コードを指定します。HADB サーバの環境変数 LANG の設定値と同じにしてください。
2	LD_LIBRARY_PATH	この環境変数には，次に示すディレクトリを追加してください。 • $ADBCLTDIR/client/lib この環境変数は，次のどちらかの条件を満たす場合に設定してください。 • HADB クライアントでadbsql コマンドを実行する場合 • HADB クライアントで CLI 関数を使用した AP を開発または実行する場合
3	PATH	この環境変数には，次に示すディレクトリを追加してください。 • $ADBCLTDIR/client/bin

項番	環境変数	設定する値
4	TZ	この環境変数には，HADB クライアントをインストールするマシンのタイムゾーンを設定します。 なお，うるう秒対応のタイムゾーンは設定しないでください。
5	ADBCLTDIR	この環境変数には，クライアントディレクトリの絶対パスを指定します。 パスの最初の 1 文字目は必ず/（スラッシュ）で始めてください。
6	ADBCLTLANG	この環境変数には，HADB クライアントで使用する文字コードを指定します。HADB サーバで使用する文字コードと同じ文字コードを指定してください。HADB サーバの環境変数ADBLANG で指定した文字コードと同じ文字コードを指定します。 • HADB サーバで Unicode（UTF-8）を使用する場合 この環境変数にUTF8 を指定してください。 • HADB サーバで Shift-JIS を使用する場合 この環境変数にSJIS を指定してください。
7	ADBMSGLOGSIZE	この環境変数には，1 ファイル当たりのクライアントメッセージログファイルの最大容量をメガバイト単位で指定します。指定できる範囲は1～2,000 です。 HADB クライアントには，4 つのクライアントメッセージログファイルが作成されます。この環境変数を省略した場合，1 つ当たりのクライアントメッセージログファイルの最大容量は 16 メガバイトになります。 なお，1 つのクライアントディレクトリから複数のクライアントプロセスを起動する場合，この環境変数の指定値は必ず同じ値を指定してください。異なる値で複数のクライアントプロセスを動作させた場合，クライアントメッセージログファイルが壊れるおそれがあります。
8	ADBSQLNULLCHAR	この環境変数には，adbsql コマンドの検索結果にナル値がある場合に，ナル値を表示する文字列（ナル値表示文字列）を指定します。0～32 バイトの文字列を指定できます。0 バイトの文字列を指定した場合は，ナル値の表示は空白になります。 この環境変数を省略した場合，ナル値は*（アスタリスク）で表示されます。 検索データ中に，アスタリスクがある場合や，ナル値を任意の文字で出力したい場合に，この環境変数を指定してください。 なお，マルチバイト文字を指定すると，検索結果の表示が乱れることがあります。 この環境変数は，HADB クライアントでadbsql コマンドを実行する場合に設定を検討してください。
9	ADBODBSGDST	この環境変数には，HADB ODBC ドライバのSQLGetData 関数で，分割取得機能を有効にするかどうかを指定します。通常，この環境変数を指定する必要はありません。 この環境変数の指定値によって HADB ODBC ドライバの処理が変わります。 • USE：分割取得機能が有効になります。 • NOUSE：分割取得機能が無効になります。 この環境変数を省略した場合，または不正な値を指定した場合は，USE が仮定されます。 SQLGetData 関数を使用して取得した返却データの長さがユーザバッファのサイズより大きくて返却データの一部しか取得できなかった場合に，十分なサイズのユーザバッファを用意できるとき（再度SQLGetData 関数を発行して完全な返却データを取得できるとき）は，NOUSE を指定してください。

項番	環境変数	設定する値
		この環境変数は，ODBC ドライバを使用する場合に設定を検討してください。

4.4　クライアント定義の作成

クライアント定義を指定して HADB クライアントの稼働環境を設定します。クライアント定義の作成は HADB クライアントを管理する OS ユーザが実行します。

4.4.1　クライアント定義の作成方法

クライアント定義の作成手順を次に示します。

手順

1. %ADBCLTDIR%¥sample¥conf¥に格納されているクライアント定義のひな形ファイル（client.def※）を，%ADBCLTDIR%¥conf¥にコピーします。

   注※　Linux 版の HADB クライアントの場合は，$ADBCLTDIR/sample/conf/client.def になります。

2. コピーしたクライアント定義のファイル（%ADBCLTDIR%¥conf¥client.def※）をテキストエディタで開いてください。

   注※　Linux 版の HADB クライアントの場合は，$ADBCLTDIR/conf/client.def になります。

3. クライアント定義の各オペランドを指定してください。クライアント定義の各オペランドの説明については，「2.　クライアント定義の設計」を参照してください。

4. クライアント定義を作成後，ファイルを上書きしてください。ファイル名はclient.def のままにしてください。

クライアント定義ファイルのパーミッションは，HADB クライアントを管理する OS ユーザに対してだけ，読み取り権限と書き込み権限を設定するようにしてください。

クライアント定義に指定したオペランドの指定値は，コネクションハンドル割り当て時にKFAA50027-I メッセージとしてクライアントメッセージログファイルに出力されます。

4.4.2　クライアント定義変更時の注意事項

クライアント定義の指定を変更する場合は，該当するクライアント定義ファイルを使用している AP を HADB サーバから切り離したあとに行ってください。

4.4.3　クライアント定義の選択

CLI 関数のa_rdb_SQLAllocConnect()の引数ClientDefPath に，クライアント定義ファイルのパスを指定できます。クライアント定義を複数作成しておき，使用するクライアント定義ファイルをコネクションハンドルの割り当て時に選択できます。

なお，引数ClientDefPath の指定を省略した場合，%ADBCLTDIR%¥conf¥client.def※が指定されたと仮定されます。a_rdb_SQLAllocConnect()については，「19.2.1　a_rdb_SQLAllocConnect()（コネクションハンドルの割り当て）」を参照してください。

注※　Linux 版の HADB クライアントの場合は，$ADBCLTDIR/conf/client.def になります。

4.5 AP の無応答状態への対策

通信障害，瞬断を含む一時的な障害，またはディスク障害などによって，AP が無応答状態になることがあります。無応答状態の AP が発生した場合，その影響によってほかの AP やコマンドの処理が停滞することがあります。

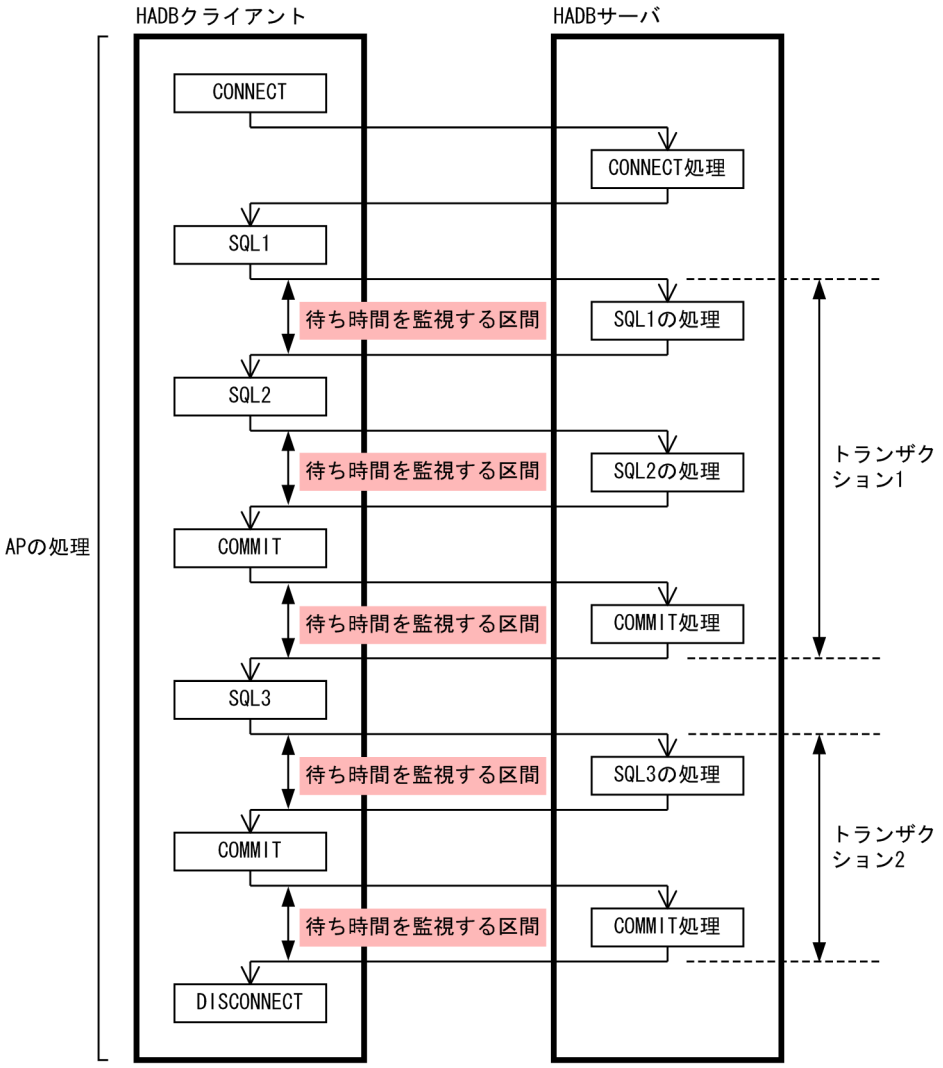
そこで，無応答状態の AP が発生したときに，その影響をなるべく小さくするために，クライアント定義で次に示すオペランドを指定してください。

- adb_clt_rpc_sql_wait_time オペランド

このオペランドには，HADB クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間を指定します。ここで指定した待ち時間を超えても HADB サーバから応答がない場合，SQLCODE が-732（KFAA30732-E）のタイムアウトエラーが AP に返されます。このとき，SQL 文の処理はキャンセルされ，トランザクションはロールバックされます。そのあと，HADB サーバから AP が切り離されます。

adb_clt_rpc_sql_wait_time オペランドによる待ち時間の監視イメージを次の図に示します。

図 4-3　adb_clt_rpc_sql_wait_time オペランドによる待ち時間の監視イメージ



[説明]

HADB クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間を監視しています。例えば，adb_clt_rpc_sql_wait_time オペランドに 600 秒を指定した場合，各監視区間に対して 600 秒の待ち時間が設定されます。したがって，処理時間がいちばん長い SQL の処理時間を目安にして，待ち時間を指定します。待ち時間には，AP が無応答状態になった可能性が高いと考えられる時間を指定してください。

adb_clt_rpc_sql_wait_time オペランドについては，「2.2.2　AP の状態監視に関するオペランド」にある，adb_clt_rpc_sql_wait_time オペランドの説明を参照してください。

4.6　HADB クライアントのバージョンアップ

HADB クライアントのバージョンアップ方法について説明します。

なお，HADB クライアントのバージョンアップではなく，修正版 HADB クライアントとの入れ替えの場合は，「4.8　修正版 HADB クライアントとの入れ替え」を参照してください。修正版 HADB クライアントについても，「4.8　修正版 HADB クライアントとの入れ替え」で説明しています。

4.6.1　HADB クライアントのバージョンアップ前に実施すること

バージョンアップ前に，次の内容を必ず実行してください。

(1)　環境変数の確認

HADB クライアントをバージョンアップすると，環境変数の値が変更になることがあります。詳細は，「4.3　環境変数の設定」を参照してください。

(2)　クライアント定義の確認

HADB クライアントをバージョンアップすると，クライアント定義のデフォルト値が変更されることがあります。クライアント定義の各オペランドをデフォルト値で設定している場合は，「2.2　クライアント定義のオペランドの内容」を参照して，変更がないかどうかを確認してください。

また，バージョン 03-00 以降では，クライアント定義の次のオペランドは指定できません。次のオペランドを指定している場合は削除してください。

- adb_clt_rpc_open_wait_time

(3)　クライアントディレクトリのバックアップ

HADB クライアントをバージョンアップする前に，クライアントディレクトリのバックアップを取得してください。

取得したバックアップは，新バージョンの動作確認後に削除してください。

4.6.2　バージョンアップ時の注意事項

HADB クライアントのバージョンアップ時の注意事項について説明します。

- HADB クライアントをバージョンアップする際，HADB サーバのバージョンに対応している HADB クライアントをインストールしてください。

- インストールデータに関する注意事項があります。HADB クライアントをバージョンアップする前に，Windows 版の HADB クライアントの場合は，「4.2.1　Windows 版の HADB クライアントの場合」の「(1)　HADB クライアントのインストール」のメモを参照してください。Linux 版の HADB クライアントの場合は，「4.2.2　Linux 版の HADB クライアントの場合」の「(2)　HADB クライアントのインストール」のメモを参照してください。
- バージョンアップをしているときに，OS ユーザを変更しないでください。正しくバージョンアップできないことがあります。
- バージョンアップをしているときに，環境変数ADBCLTLANG に設定した値は変更しないでください。正しくバージョンアップできないことがあります。
- HADB クライアントと ODBC ドライバのバージョンは必ず一致させてください。バージョンが一致していない場合，HADB サーバに接続する前にエラーとなります。

4.6.3　バージョンアップ手順

HADB クライアントをバージョンアップする手順を次に示します。

(1)　HADB クライアント（Windows 版）のバージョンアップ

HADB クライアントのバージョンアップは，HADB クライアントを管理する OS ユーザが実行します。

次の手順に従って，HADB クライアントをバージョンアップしてください。

手順

1. HADB クライアントを管理する OS ユーザでログインする
2. HADB クライアントのインストールデータをコピーする

   HADB クライアントのインストール CD-ROM に格納されている次の圧縮ファイル（zip 形式ファイル）を，クライアントマシンの任意のフォルダにコピーしてください。この手順の説明では，D:¥hadbclt_install フォルダにコピーするものとします。

   - 64 ビット版の HADB クライアントを使用する場合：hitachi_advanced_data_binder_client.zip
   - 32 ビット版の HADB クライアントを使用する場合：hitachi_advanced_data_binder_client32.zip

   コピー先のフォルダは，パス長が 200 バイト以内のフォルダにしてください。

   コピー先のフォルダのパス名に使用できる文字については，『はじめに』の「■このマニュアルで使用する構文要素記号」の『〈パス名〉』を参照してください。
3. コピーしたファイルのハッシュ値を取得する

   コマンドプロンプトを起動して，次のコマンドを実行してください。

   - 64 ビット版の HADB クライアントを使用する場合

   ```
   cd /D D:¥hadbclt_install
   certutil -hashfile hitachi_advanced_data_binder_client.zip SHA256
   ```

- 32 ビット版の HADB クライアントを使用する場合

```
cd /D D:¥hadbclt_install
certutil -hashfile hitachi_advanced_data_binder_client32.zip SHA256
```

下線部分は，手順 2.でインストールデータをコピーしたフォルダです。

4. ハッシュ値を比較する

手順 3.で取得したハッシュ値と，インストール CD-ROM に格納されている次のファイルに記載されているハッシュ値を比較してください。

- 64 ビット版の HADB クライアントを使用する場合：adbhashfile_winclt64_sha256.txt
- 32 ビット版の HADB クライアントを使用する場合：adbhashfile_winclt32_sha256.txt

ハッシュ値が同じ場合は，問題ありません。次の手順に進んでください。

ハッシュ値が異なる場合は，ファイルのコピーが失敗しているおそれがあります。手順 2.から作業をし直してください。

5. 圧縮ファイルを解凍する

（例）hitachi_advanced_data_binder_client.zip を解凍した場合

D:¥hadbclt_install¥hitachi_advanced_data_binder_client¥HADBCL フォルダ下に，HADB クライアントのフォルダおよびファイルが展開されます。

D:¥hadbclt_install¥hitachi_advanced_data_binder_client¥HADBCL フォルダは，圧縮ファイルを解凍したときのデフォルトの解凍先です。

6. HADB クライアントのバージョン情報を確認する

解凍先フォルダ下に格納されているreadme.txt ファイルを開いてください。

1 行目に記載されている HADB クライアントのバージョン情報を見て，次のどちらかの情報が記載されていることを確認してください。

- P-2462-C114 *vv-rr*
- P-2462-C114 *vv-rr-/s*

vv-rr および *vv-rr-/s* は，HADB クライアントのバージョンを示しています。

7. クライアントディレクトリを上書きする

（例）

D:¥hadbclt_install¥hitachi_advanced_data_binder_client¥HADBCL フォルダを，環境変数 ADBCLTDIR に指定しているクライアントディレクトリに上書きコピーしてください。

クライアントディレクトリをD:¥hadbclt_install¥hitachi_advanced_data_binder_client¥HADBCL フォルダに変更する場合は，環境変数ADBCLTDIR に指定しているクライアントディレクトリのパスを変更してください。

(2) HADB クライアント（Linux 版）のバージョンアップ

次の手順に従って，HADB クライアントをバージョンアップしてください。

手順

1. HADB クライアントを管理する OS ユーザでログインする
2. インストールデータを格納するディレクトリを作成する

```
mkdir /home/osuser01/client
```

上記の例では，HADB クライアントのインストールデータを格納するディレクトリとして/home/osuser01/client を作成しています。

3. インストールデータを格納するディレクトリに対して書き込み権限を付与する

```
chmod 755 /home/osuser01/client
```

インストールデータを格納するディレクトリに対して，HADB クライアントを管理する OS ユーザが書き込みできるように，ディレクトリ（この例では/home/osuser01/client）に対して書き込み権限を付与してください。

4. CD-ROM ファイルシステムをマウントする

HADB クライアントのインストールデータが格納されている CD-ROM ファイルシステムを自動マウントしてください。

CD-ROM ファイルシステムが自動マウントされない場合，次のコマンドを実行して CD-ROM ファイルシステムをマウントしてください。

```
mount /dev/cdrom /media
```

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

> **重要**
>
> CD-ROM のディレクトリ名やファイル名は，ご使用のマシンによっては，ここの説明内容と表示が異なることがあります。OS の ls コマンドを実行して，表示されたディレクトリ名をそのまま入力してください。

5. インストールデータをコピーする

マウントした CD-ROM ファイルシステムに格納されている次のファイルを，手順 2.で作成したインストールデータを格納するディレクトリにコピーしてください。

- tar.gz 形式のファイル
- adbinstall コマンドの実行形式ファイル

```
cp /media/hitachi_advanced_data_binder_client-$VER.tar.gz /home/osuser01/client   ...a
cp /media/adbinstall /home/osuser01/client                                        ...b
```

[説明]

a. tar.gz 形式のファイル（HADB クライアントのプログラムの圧縮ファイル）をコピーします。

b. adbinstall コマンドの実行形式ファイル（HADB クライアントのインストールコマンド）をコピーします。

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

$VER は HADB クライアントのバージョンおよびリリース番号です。

> **重要**
>
> 上記の 2 つのファイルは，必ず同じディレクトリにコピーしてください。異なるディレクトリにコピーすると，HADB クライアントがインストールできません。

6. コピーしたファイルに破損がないことを確認する

手順 5.でコピーしたファイルに破損がないことを確認します。次のコマンドを実行してください。

```
cd /home/osuser01/client
sha256sum -c /media/adbhashfile_client_sha256.txt
```

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

ファイルが破損しているおそれがある（ファイルのハッシュ値が異なる）場合は，エラーを示すメッセージが出力されます。この場合，手順 5.から作業をし直してください。

出力されるメッセージの詳細については，OS のマニュアルを参照してください。

7. 権限を付与する

HADB クライアントを管理する OS ユーザが**adbinstall** コマンド（HADB クライアントのインストールコマンド）を実行できるように，**adbinstall** コマンドに対する実行権限を付与してください。

また，tar.gz 形式のファイルに対する読み取り権限も付与してください。

```
chmod 555 /home/osuser01/client/adbinstall
chmod 444 /home/osuser01/client/hitachi_advanced_data_binder_client-$VER.tar.gz
```

$VER は，HADB クライアントのバージョンおよびリリース番号です。

8. HADB クライアントをインストールする

adbinstall コマンドを実行して，HADB クライアントをインストールしてください。

```
/home/osuser01/client/adbinstall -c /home/osuser01/clientdir
```

下線部分には，手順 2.で作成したインストールデータを格納するディレクトリを指定します。

-c オプションには，クライアントディレクトリのパスを指定します。次のどちらかのパスを指定してください。

- 環境変数**ADBCLTDIR** に指定しているパス
- 任意のディレクトリのパス

  任意のディレクトリのパスを指定した場合は，環境変数**ADBCLTDIR** にそのパスを指定してください。

-c オプションには，HADB クライアントを管理する OS ユーザが書き込みできるディレクトリを指定してください。

> **メモ**
>
> adbinstall コマンドの実行方法および規則については，マニュアル『HADB　コマンドリファレンス』の『adbinstall（HADB サーバおよび HADB クライアントのインストール）』を参照してください。

9. HADB クライアントが正常にインストールされたことを確認する

クライアントディレクトリ下のadbinstcl.log ファイルを開いて内容を確認してください。HADB クライアントが正常にインストールされている場合は，次のどちらかの情報が記載されています。

- Hitachi Advanced Data Binder Client *vv-rr* (*yyyymmddhhmmss*)
- Hitachi Advanced Data Binder Client *vv-rr-/s* (*yyyymmddhhmmss*)

vv-rr および *vv-rr-/s* は，HADB クライアントのバージョンを示しています。

yyyymmddhhmmss は，HADB クライアントのインストールを実行した日付と時刻を示しています。

> **メモ**
>
> - adbinstcl.log ファイルがすでにある状態のときにadbinstall コマンドを実行すると，adbinstcl.log ファイルの内容が更新されます。また，adbinstall コマンドを複数回実行した場合，最後に実行した日付と時刻が記載されます。
> - adbinstall コマンドのリターンコード（KFAA91552-I メッセージに出力されるリターンコード）が0 または4 以外の場合，adbinstcl.log ファイルの内容は更新されません。
> - adbinstall コマンドの実行時にKFAA91555-I メッセージが出力された場合，adbinstcl.log ファイルの内容は更新されません。

■KFAA91553-E メッセージが出力された場合の対処方法

対処方法については，「4.2.2　Linux 版の HADB クライアントの場合」の「(2)　HADB クライアントのインストール」の「■KFAA91553-E メッセージが出力された場合の対処方法」を参照してください。

■KFAA91558-W メッセージが出力された場合の対処方法

対処方法については，「4.2.2　Linux 版の HADB クライアントの場合」の「(2)　HADB クライアントのインストール」の「■KFAA91558-W メッセージが出力された場合の対処方法」を参照してください。

4.6.4　バージョンアップ後に実施すること

HADB クライアントのバージョンアップが正しく実施できたか次の手順で確認してください。

(1) HADB クライアント（Windows 版）

次に示すファイルのプロパティで，ファイルバージョンが新しいバージョンになっていることを確認してください。

- 64 ビット版の HADB クライアントを使用する場合：%ADBCLTDIR%client¥bin¥adbclt.dll
- 32 ビット版の HADB クライアントを使用する場合：%ADBCLTDIR%client¥bin¥adbclt32.dll

なお，ODBC ドライバを使用している場合は，adbodbc で始まるすべての dll のバージョンについても確認してください。

(2) HADB クライアント（Linux 版）

次に示す手順で正しくバージョンアップが実施できたか確認してください。

手順

1. adbsql コマンドを実行して HADB サーバに接続する
2. KFAA70003-I メッセージを確認する
   クライアントのメッセージログファイルに出力されるKFAA70003-I メッセージに，新しいバージョンが出力されていることを確認してください。

4.7　HADB クライアントのバージョンダウン（旧バージョンに戻す方法）

ここでは，HADB クライアントをバージョンダウンする方法について説明します。

HADB サーバをバージョンダウンした場合は，HADB クライアントもバージョンダウンしてください（HADB サーバと HADB クライアントのバージョンを合わせてください）。

> **メモ**
>
> HADB サーバのバージョンダウンについては，マニュアル『HADB　システム構築・運用ガイド』の『システム構築』の『HADB サーバのバージョンダウン（旧バージョンに戻す方法）』を参照してください。

4.7.1　バージョンダウン前に実施すること

HADB クライアントをバージョンダウンする前に，次に示す指定値を確認してください。

- 環境変数
- クライアント定義

バージョンアップした際に各指定値を変更している場合は，バージョンダウンするときに，旧バージョンの指定値に戻す必要があります。

4.7.2　バージョンダウン時の注意事項

HADB クライアントをバージョンダウンするときの注意事項について説明します。

- HADB クライアントをバージョンダウンする際，HADB サーバのバージョンに対応している HADB クライアントをインストールしてください。
- インストールデータに関する注意事項があります。HADB クライアントをバージョンダウンする前に，Windows 版の HADB クライアントの場合は，「4.2.1　Windows 版の HADB クライアントの場合」の「(1)　HADB クライアントのインストール」のメモを参照してください。Linux 版の HADB クライアントの場合は，「4.2.2　Linux 版の HADB クライアントの場合」の「(2)　HADB クライアントのインストール」のメモを参照してください。
- バージョンダウンをしているときに，OS ユーザを変更しないでください。正しくバージョンダウンできないことがあります。
- バージョンダウンをしているときに，環境変数ADBCLTLANG に設定した値は変更しないでください。正しくバージョンダウンできないことがあります。
- HADB クライアントと ODBC ドライバのバージョンは必ず一致させてください。バージョンが一致していない場合，HADB サーバに接続する前にエラーとなります。

4.7.3　バージョンダウン手順

HADB クライアントをバージョンダウンする手順を次に示します。

HADB クライアントのバージョンダウンは，HADB クライアントを管理する OS ユーザが実行します。

(1)　HADB クライアント（Windows 版）のバージョンダウン

次の手順に従って，HADB クライアントをバージョンダウンしてください。

手順

1. HADB クライアントを管理する OS ユーザでログインする
2. HADB クライアントのインストールデータをコピーする

   HADB クライアントのインストール CD-ROM に格納されている次の圧縮ファイル（zip 形式ファイル）を，クライアントマシンの任意のフォルダにコピーしてください。この手順の説明では，`D:¥hadbclt_install` フォルダにコピーするものとします。

   - 64 ビット版の HADB クライアントを使用する場合：`hitachi_advanced_data_binder_client.zip`
   - 32 ビット版の HADB クライアントを使用する場合：`hitachi_advanced_data_binder_client32.zip`

   コピー先のフォルダは，パス長が 200 バイト以内のフォルダにしてください。

   コピー先のフォルダのパス名に使用できる文字については，『はじめに』の「■このマニュアルで使用する構文要素記号」の『〈パス名〉』を参照してください。
3. コピーしたファイルのハッシュ値を取得する

   > **重要**
   >
   > 手順 3.～手順 4.の作業は，バージョン 05-02 以降の HADB クライアントにバージョンダウンする場合に限り実施してください。

   コマンドプロンプトを起動して，次のコマンドを実行してください。

   - 64 ビット版の HADB クライアントを使用する場合

     ```
     cd /D D:¥hadbclt_install
     certutil -hashfile hitachi_advanced_data_binder_client.zip SHA256
     ```

   - 32 ビット版の HADB クライアントを使用する場合

     ```
     cd /D D:¥hadbclt_install
     certutil -hashfile hitachi_advanced_data_binder_client32.zip SHA256
     ```

   下線部分は，手順 2.でインストールデータをコピーしたフォルダです。
4. ハッシュ値を比較する

   手順 3.で取得したハッシュ値と，インストール CD-ROM に格納されている次のファイルに記載されているハッシュ値を比較してください。

- 64 ビット版の HADB クライアントを使用する場合：adbhashfile_winclt64_sha256.txt
- 32 ビット版の HADB クライアントを使用する場合：adbhashfile_winclt32_sha256.txt

ハッシュ値が同じ場合は，問題ありません。次の手順に進んでください。

ハッシュ値が異なる場合は，ファイルのコピーが失敗しているおそれがあります。手順 2.から作業をし直してください。

5. 圧縮ファイルを解凍する

（例）hitachi_advanced_data_binder_client.zip を解凍した場合

D:¥hadbclt_install¥hitachi_advanced_data_binder_client¥HADBCL フォルダ下に，HADB クライアントのフォルダおよびファイルが展開されます。

D:¥hadbclt_install¥hitachi_advanced_data_binder_client¥HADBCL フォルダは，圧縮ファイルを解凍したときのデフォルトの解凍先です。

6. HADB クライアントのバージョン情報を確認する

解凍先フォルダ下に格納されているreadme.txt ファイルを開いてください。

1 行目に記載されている HADB クライアントのバージョン情報を見て，次のどちらかの情報が記載されていることを確認してください。

- P-2462-C114 *vv-rr*
- P-2462-C114 *vv-rr-/s*

vv-rr および *vv-rr-/s* は，HADB クライアントのバージョンを示しています。

7. クライアントディレクトリを上書きする

（例）

D:¥hadbclt_install¥hitachi_advanced_data_binder_client¥HADBCL フォルダを，環境変数 ADBCLTDIR に指定しているクライアントディレクトリに上書きコピーしてください。

クライアントディレクトリをD:¥hadbclt_install¥hitachi_advanced_data_binder_client¥HADBCL フォルダに変更する場合は，環境変数ADBCLTDIR に指定しているクライアントディレクトリのパスを変更してください。

(2) HADB クライアント（Linux 版）のバージョンダウン

次の手順に従って，HADB クライアントをバージョンダウンしてください。

手順

1. HADB クライアントを管理する OS ユーザでログインする
2. インストールデータを格納するディレクトリを作成する

```
mkdir /home/osuser01/client
```

上記の例では，HADB クライアントのインストールデータを格納するディレクトリとして/home/osuser01/client を作成しています。

3. インストールデータを格納するディレクトリに対して書き込み権限を付与する

```
chmod 755 /home/osuser01/client
```

インストールデータを格納するディレクトリに対して，HADB クライアントを管理する OS ユーザが書き込みできるように，ディレクトリ（この例では/home/osuser01/client）に対して書き込み権限を付与してください。

4. CD-ROM ファイルシステムをマウントする

HADB クライアントのインストールデータが格納されている CD-ROM ファイルシステムを自動マウントしてください。

CD-ROM ファイルシステムが自動マウントされない場合，次のコマンドを実行して CD-ROM ファイルシステムをマウントしてください。

```
mount /dev/cdrom /media
```

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

重要

CD-ROM のディレクトリ名やファイル名は，ご使用のマシンによっては，ここの説明内容と表示が異なることがあります。OS の ls コマンドを実行して，表示されたディレクトリ名をそのまま入力してください。

5. インストールデータをコピーする

マウントした CD-ROM ファイルシステムに格納されている次のファイルを，手順 2.で作成したインストールデータを格納するディレクトリにコピーしてください。

- tar.gz 形式のファイル
- adbinstall コマンドの実行形式ファイル

```
cp /media/hitachi_advanced_data_binder_client-$VER.tar.gz /home/osuser01/client   ...a
cp /media/adbinstall /home/osuser01/client                                        ...b
```

[説明]

a. tar.gz 形式のファイル（HADB クライアントのプログラムの圧縮ファイル）をコピーします。

b. adbinstall コマンドの実行形式ファイル（HADB クライアントのインストールコマンド）をコピーします。

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

$VER は，HADB クライアントのバージョンおよびリリース番号です。

重要

上記の 2 つのファイルは，必ず同じディレクトリにコピーしてください。異なるディレクトリにコピーすると，HADB クライアントがインストールできません。

6. コピーしたファイルに破損がないことを確認する

重要

手順 6.の作業は，バージョン 05-02 以降の HADB クライアントにバージョンダウンする場合に限り実施してください。

手順 5.でコピーしたファイルに破損がないことを確認します。次のコマンドを実行してください。

```
cd /home/osuser01/client
sha256sum -c /media/adbhashfile_client_sha256.txt
```

下線部分は，CD-ROM ファイルシステムのマウントディレクトリ名です。環境によって異なります。

ファイルが破損しているおそれがある（ファイルのハッシュ値が異なる）場合は，エラーを示すメッセージが出力されます。この場合，手順 5.から作業をし直してください。

出力されるメッセージの詳細については，OS のマニュアルを参照してください。

7. 権限を付与する

HADB クライアントを管理する OS ユーザがadbinstall コマンド（HADB クライアントのインストールコマンド）を実行できるように，adbinstall コマンドに対する実行権限を付与してください。

また，tar.gz 形式のファイルに対する読み取り権限も付与してください。

```
chmod 555 /home/osuser01/client/adbinstall
chmod 444 /home/osuser01/client/hitachi_advanced_data_binder_client-$VER.tar.gz
```

$VER は，HADB クライアントのバージョンおよびリリース番号です。

8. HADB クライアントをインストールする

adbinstall コマンドを実行して，HADB クライアントをインストールしてください。

```
/home/osuser01/client/adbinstall -c /home/osuser01/clientdir
```

下線部分には，手順 2.で作成したインストールデータを格納するディレクトリを指定します。

-c オプションには，クライアントディレクトリのパスを指定します。次のどちらかのパスを指定してください。

- 環境変数ADBCLTDIR に指定しているパス
- 任意のディレクトリのパス

  任意のディレクトリのパスを指定した場合は，環境変数ADBCLTDIR にそのパスを指定してください。

-c オプションには，HADB クライアントを管理する OS ユーザが書き込みできるディレクトリを指定してください。

メモ

adbinstall コマンドの実行方法および規則については，マニュアル『HADB　コマンドリファレンス』の『adbinstall（HADB サーバおよび HADB クライアントのインストール)』を参照してください。

9. HADB クライアントが正常にインストールされたことを確認する

重要

手順 9.の作業は，バージョン 05-02 以降の HADB クライアントにバージョンダウンする場合に限り実施してください。

クライアントディレクトリ下のadbinstcl.log ファイルを開いて内容を確認してください。HADB クライアントが正常にインストールされている場合は，次のどちらかの情報が記載されています。

- Hitachi Advanced Data Binder Client *vv-rr* (*yyyymmddhhmmss*)
- Hitachi Advanced Data Binder Client *vv-rr-/s* (*yyyymmddhhmmss*)

vv-rr および *vv-rr-/s* は，HADB クライアントのバージョンを示しています。

yyyymmddhhmmss は，HADB クライアントのインストールを実行した日付と時刻を示しています。

メモ

- adbinstcl.log ファイルがすでにある状態のときにadbinstall コマンドを実行すると，adbinstcl.log ファイルの内容が更新されます。また，adbinstall コマンドを複数回実行した場合，最後に実行した日付と時刻が記載されます。
- adbinstall コマンドのリターンコード（KFAA91552-I メッセージに出力されるリターンコード）が0 または4 以外の場合，adbinstcl.log ファイルの内容は更新されません。
- adbinstall コマンドの実行時にKFAA91555-I メッセージが出力された場合，adbinstcl.log ファイルの内容は更新されません。

■KFAA91553-E メッセージが出力された場合の対処方法

対処方法については，「4.2.2　Linux 版の HADB クライアントの場合」の「(2)　HADB クライアントのインストール」の「■KFAA91553-E メッセージが出力された場合の対処方法」を参照してください。

■KFAA91558-W メッセージが出力された場合の対処方法

対処方法については，「4.2.2　Linux 版の HADB クライアントの場合」の「(2)　HADB クライアントのインストール」の「■KFAA91558-W メッセージが出力された場合の対処方法」を参照してください。

4.7.4　バージョンダウン後に実施すること

HADB クライアントのバージョンダウンが正しく実施できたかについて，次の手順で確認してください。

(1)　HADB クライアント（Windows 版）

次に示すファイルのプロパティで，ファイルバージョンが旧バージョンになっていることを確認してください。

- 64 ビット版の HADB クライアントを使用する場合：%ADBCLTDIR%client¥bin¥adbclt.dll
- 32 ビット版の HADB クライアントを使用する場合：%ADBCLTDIR%client¥bin¥adbclt32.dll

なお，ODBC ドライバを使用している場合は，adbodbc で始まるすべての dll のバージョンについても確認してください。

(2) HADB クライアント（Linux 版）

次に示す手順で正しくバージョンダウンが実施できたか確認してください。

手順

1. adbsql コマンドを実行して HADB サーバに接続する
2. KFAA70003-I メッセージを確認する

   クライアントメッセージログファイルに出力されるKFAA70003-I メッセージに，旧バージョンが出力されていることを確認してください。

4.8　修正版 HADB クライアントとの入れ替え

ここでは，修正版 HADB クライアントとの入れ替え方法について説明します。

次に示す条件を満たす場合は，HADB クライアントのバージョンアップではなく，修正版 HADB クライアントとの入れ替えとなります。

- バージョン番号およびリビジョン番号が同じで，コードだけが異なる HADB クライアントを上書きインストールする場合

> **メモ**
>
> バージョン番号，リビジョン番号，コードについて説明します。
>
> （例）HADB クライアントのバージョン情報が，03-02-/A の場合
>
> - 03 の部分がバージョン番号になります。
> - 02 の部分がリビジョン番号になります。
> - 下線部分の-/A の部分が，コードになります。

修正版 HADB クライアントとの入れ替えになる場合の例と，入れ替えにならない場合の例を次に示します。

■修正版 HADB クライアントとの入れ替えになる場合

例えば，次に示すケースは，バージョン番号およびリビジョン番号が同じため，修正版 HADB クライアントとの入れ替えになります。

03-02 → 03-02-/A

03-02-/B → 03-02-/D

■修正版 HADB クライアントとの入れ替えにならない場合

例えば，次に示すケースは，バージョン番号またはリビジョン番号が異なるため，修正版 HADB クライアントとの入れ替えになりません。

03-02 → 03-03-/A

03-02-/B → 03-03-/D

この場合，HADB クライアントのバージョンアップになります。HADB クライアントのバージョンアップ方法については，「4.6　HADB クライアントのバージョンアップ」を参照してください。

> **メモ**
>
> - 修正版 HADB サーバのバージョンに対応している修正版 HADB クライアントと入れ替えてください。
> - インストールデータに関する注意事項があります。修正版 HADB クライアントとの入れ替え作業をする前に，Windows 版の HADB クライアントの場合は，「4.2.1　Windows 版の HADB クライアントの場合」の「(1)　HADB クライアントのインストール」のメモを

参照してください。Linux 版の HADB クライアントの場合は，「4.2.2　Linux 版の HADB クライアントの場合」の「(2)　HADB クライアントのインストール」のメモを参照してください。

4.8.1　修正版 HADB クライアントとの入れ替え手順

修正版 HADB クライアントとの入れ替え手順を次に示します。

手順

1. 入れ替え前に必要な作業を実施する

   入れ替え前に必要な作業については，HADB クライアントのバージョンアップ前に必要な作業と同じになります。詳細については，「4.6.1　HADB クライアントのバージョンアップ前に実施すること」を参照してください。

2. 入れ替え時の注意事項を確認する

   入れ替え時の注意事項については，HADB クライアントのバージョンアップ時の注意事項と同じになります。詳細については，「4.6.2　バージョンアップ時の注意事項」を参照してください。

3. 修正版 HADB クライアントとの入れ替えを実施する

   入れ替え手順については，HADB クライアントのバージョンアップ時の手順と同じになります。詳細については，「4.6.3　バージョンアップ手順」を参照してください。

4. 入れ替えが完了したことを確認する

   HADB クライアントのバージョン情報のコードが変わっていることを確認してください。確認方法については，「4.6.4　バージョンアップ後に実施すること」を参照してください。

4.9　クライアントマシンの OS の時刻変更

ここでは，HADB クライアントがインストールされているクライアントマシンの OS の時刻を変更する場合の手順について説明します。

クライアントマシンの OS の時刻を変更する場合，必ず「4.9.1　注意事項（OS の時刻変更）」を読んだあとで，次のどちらかに従って OS の時刻を変更してください。

- 「4.9.2　クライアントマシンの OS の時刻を進める方法」
- 「4.9.3　クライアントマシンの OS の時刻を戻す方法」

4.9.1　注意事項（OS の時刻変更）

クライアントマシンの OS の時刻を変更する場合の注意事項を，次に示します。

- クライアントマシンの OS の時刻を変更する場合，必ず HADB サーバに接続するすべての AP が停止しているときに実施してください。HADB サーバに接続中の AP がある場合に，クライアントマシンの OS の時刻を変更してしまうと，HADB サーバおよび HADB クライアントが予期しない動作をするおそれがあります。
- クライアントマシンの OS の時刻を変更した場合，次に示す時刻表示などは，変更後の OS の時刻になります。そのため，時刻を意識する AP がある場合は，影響を受けるおそれがあります。
  - コマンドを実行した時刻
  - クライアントメッセージログファイル内のメッセージログのタイムスタンプ

4.9.2　クライアントマシンの OS の時刻を進める方法

「4.9.1　注意事項（OS の時刻変更）」を読んだあとで，次に示す手順に従って，HADB クライアントがインストールされているクライアントマシンの OS の時刻を進めてください。

手順

1. HADB サーバに接続する AP を停止する
   HADB クライアントから，HADB サーバに接続するすべての AP を停止してください。
2. クライアントマシンの OS の時刻を進める
   すべての AP を停止したら，HADB クライアントがインストールされているクライアントマシンの OS の時刻を進めてください。
3. HADB サーバに接続する AP を開始する
   クライアントマシンの OS の時刻を進めたら，停止した AP を開始してください。

これで，クライアントマシンの OS の時刻を進められます。

4.9.3　クライアントマシンの OS の時刻を戻す方法

「4.9.1　注意事項（OS の時刻変更）」を読んだあとで，次に示す手順に従って，HADB クライアントがインストールされているクライアントマシンの OS の時刻を戻してください。

手順

1. HADB サーバに接続する AP を停止する
   HADB クライアントから，HADB サーバに接続するすべての AP を停止してください。
2. クライアントマシンの OS の時刻を戻す
   すべての AP を停止したら，HADB クライアントがインストールされているクライアントマシンの OS の時刻を戻してください。
3. HADB サーバに接続する AP を開始する
   クライアントマシンの OS の時刻を戻したら，停止した AP を開始してください。

これで，クライアントマシンの OS の時刻を戻せます。

5 AP の性能向上に関する設計

この章では，AP の性能向上に関する設計について説明します。

5.1　表の検索方式

表の検索方式には，次の 3 つがあります。

- テーブルスキャン
- インデクススキャン
- キースキャン

表の検索方式は，HADB が自動的に決定します。SQL 文を実行した結果，どの検索方式が適用されたかを，アクセスパスで確認することができます。アクセスパスについては，次に示す個所を参照してください。

- アクセスパスの確認方法
  「6.1.2　アクセスパスを確認するには」を参照してください。
- アクセスパスに表示される内容
  「6.1.4　ツリー表示に出力される情報」の「(21)　表の検索方式」を参照してください。

> **メモ**
>
> インデクス指定によって，表の検索に使用するインデクスを指定したり，またはインデクスの使用を抑止したりすることができます。インデクス指定については，マニュアル『HADB　SQL リファレンス』の『インデクス指定の指定形式および規則』を参照してください。

なお，ここでいう表とは，実表を意味しています。

5.1.1　テーブルスキャンとは

テーブルスキャンとは，B-tree インデクスおよびテキストインデクスを使用しないで表を検索する方式のことです。次の場合にテーブルスキャンが実行されます。

- B-tree インデクスおよびテキストインデクスが表に定義されていない場合
- B-tree インデクスおよびテキストインデクスを有効に利用できる探索条件が指定されていない場合
- インデクス指定に`WITHOUT INDEX` が指定されている場合

テーブルスキャンの例を次の図に示します。

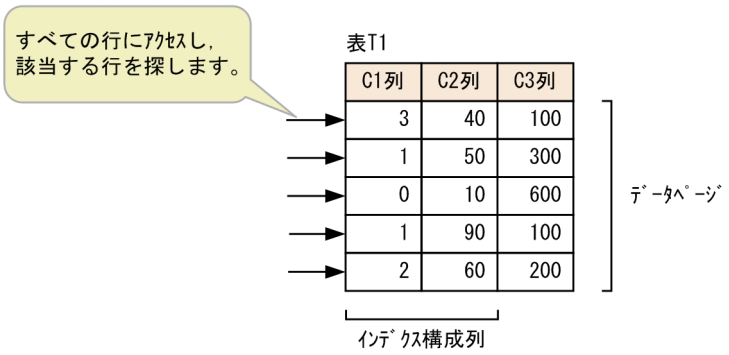
図 5-1　テーブルスキャンの例

■表およびB-treeｲﾝﾃﾞｸｽの定義

```
CREATE TABLE "T1"
            ("C1" INTEGER,
             "C2" INTEGER,
             "C3" INTEGER) IN "DBAREA01"

CREATE INDEX "IDX_C1C2" ON "T1"("C1","C2")
              IN "DBAREA01" EMPTY
```

■実行したSQL文

```
SELECT * FROM "T1" WHERE "C3"<300
```



[説明]

探索条件中に指定されているC3 列はインデクス構成列ではないため，上記のSELECT 文を実行した場合，B-tree インデクスIDX_C1C2 は使用されません。そのため，テーブルスキャンが実行されて，データページ内のすべての行にアクセスします。

なお，表にレンジインデクスが定義されている場合，レンジインデクスが使用されることがあります。

メモ

インデクス指定に，B-tree インデクスまたはテキストインデクスを使用しない指定をした場合，テーブルスキャンが実行されます。

重要

テーブルスキャンが実行された場合は，表にレンジインデクスを定義することを推奨します。レンジインデクスが使用されることで性能が向上することがあります。レンジインデクスが使用される条件については，「5.3　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。

5.1.2　インデクススキャンとは

インデクススキャンとは，B-tree インデクスまたはテキストインデクスを使用して探索条件の評価を行い，探索条件を満たす行をデータページから取り出す方式のことです。次の場合にインデクススキャンが実行されます。

- B-tree インデクスまたはテキストインデクスが表に定義されていて，これらのインデクスを有効に利用できる探索条件が指定されている場合
- 検索に使用する B-tree インデクスまたはテキストインデクスが，インデクス指定で指定されている場合

インデクススキャンの例を次の図に示します。

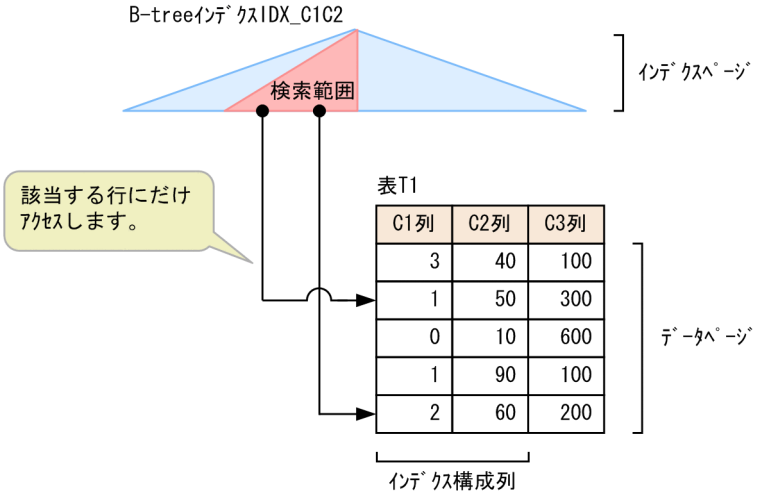
図 5-2　インデクススキャンの例

■表およびB-treeｲﾝﾃﾞｸｽの定義

```
CREATE TABLE "T1"
            ("C1" INTEGER,
             "C2" INTEGER,
             "C3" INTEGER) IN "DBAREA01"

CREATE INDEX "IDX_C1C2" ON "T1"("C1","C2")
              IN "DBAREA01" EMPTY
```

■実行したSQL文

```
SELECT "C3" FROM "T1"
    WHERE "C1" IN(1,2)
      AND "C2" BETWEEN 40 AND 60
```



[説明]

B-tree インデクスIDX_C1C2 を使用して探索条件の評価を行い，C3 列の値を取り出すためにデータページにアクセスします。

なお，表にレンジインデクスが定義されている場合，レンジインデクスも使用されることがあります。

5.1.3 キースキャンとは

キースキャンとは，B-tree インデクスを使用して探索条件の評価を行い，探索条件を満たす行の列値をインデクスページから取り出す方式のことです。B-tree インデクスのインデクス構成列（キー）から列値を直接取り出すため，参照するページ数を削減できます。

SQL 文中に指定されているすべての列が，インデクス構成列である B-tree インデクスが定義されていて，かつ次のどれかの条件を満たす場合に，キースキャンが実行されます。

- B-tree インデクスが表に定義されていて，B-tree インデクスを有効に利用できる探索条件が指定されている場合
- 検索に使用する B-tree インデクスが，インデクス指定で指定されている場合
- 集合関数MIN またはMAX が指定されている場合※
- SELECT DISTINCT が指定されている場合※
- UNION またはUNION DISTINCT が指定されている場合※
- EXCEPT またはEXCEPT DISTINCT が指定されている場合※
- INTERSECT またはINTERSECT DISTINCT が指定されている場合※
- =ANY 指定の限定述語が指定されている場合※
- 表副問合せを指定したIN 述語が指定されている場合※

注※

HADB が B-tree インデクスを有効に利用できると判断した場合に限り，B-tree インデクスが使用されます。

ただし，次のどれかの条件を満たす場合は，データページを参照することがあります。

- 行を追加，更新，または削除したことがある表を検索する場合
- 文字列の末尾が半角空白であるデータを含むVARCHAR 型の列を検索する場合
- バイナリデータの末尾がX'00'であるVARBINARY 列のデータを参照する場合

キースキャンの例を次の図に示します。

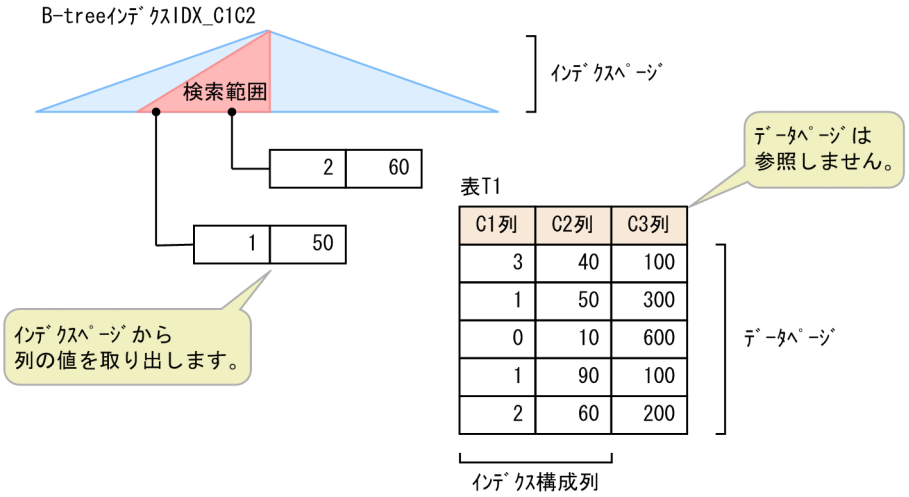
図 5-3　キースキャンの例

■表およびB-treeｲﾝﾃﾞｸｽの定義

```
CREATE TABLE "T1"
            ("C1" INTEGER,
             "C2" INTEGER,
             "C3" INTEGER) IN "DBAREA01"

CREATE INDEX "IDX_C1C2" ON "T1"("C1","C2")
              IN "DBAREA01" EMPTY
```

■実行したSQL文

```
SELECT "C1","C2" FROM "T1"
    WHERE "C1" IN(1,2)
      AND "C2" BETWEEN 40 AND 60
```



[説明]

B-tree インデクスIDX_C1C2 を使用して探索条件の評価を行います。C1 列およびC2 列の値をインデクスページから直接取り出すため，データページにはアクセスしません。

メモ

テキストインデクスの場合は，キースキャンが実行されることはありません。

5.2 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

B-tree インデクスおよびテキストインデクスは性能に大きな影響を及ぼすため，探索条件に合った B-tree インデクスおよびテキストインデクスを定義する必要があります。

ここでは，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の決定方法と，SQL 文の実行時に使用されるインデクスを確認する方法について説明します。

なお，この節では，インデクスと表記されている場合，B-tree インデクスおよびテキストインデクスの両方を意味しています。

留意事項

- この節で説明しているインデクスの決定方法は，内部導出表の展開後の問合せ式，または探索条件の等価変換によって変換された探索条件に対して適用されます。内部導出表の展開については，マニュアル『HADB　SQL リファレンス』の『内部導出表』を参照してください。探索条件の等価変換機能については，「5.11　探索条件の等価変換」を参照してください。
- 探索条件中に指定した値式で，スカラ演算中に定数だけを指定している場合，そのスカラ演算を定数と見なすことがあります。定数と等価なスカラ演算については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。
- インデクス指定がある場合，ここで説明するインデクスの優先順位や選択規則とは関係なく，インデクス指定に従って使用されるインデクスが決定されます。インデクス指定については，マニュアル『HADB　SQL リファレンス』の『インデクス指定の指定形式および規則』を参照してください。
- 表を結合する際，表の結合方式によっては結合条件の評価時にインデクスを使用しないことがあります。表の結合方式については，「5.5　表の結合方式」を参照してください。
- カラムストア表にはテキストインデクスを定義できません。

重要

カラムストア表に定義した B-tree インデクスは，次の場合に使用されます。

- インデクス指定を指定した場合

  インデクス指定については，マニュアル『HADB　SQL リファレンス』の『インデクス指定の指定形式および規則』を参照してください。
- 集合関数MIN またはMAX を指定した場合※
- SELECT DISTINCT を指定した場合※
- UNION またはUNION DISTINCT を指定した場合※
- EXCEPT またはEXCEPT DISTINCT を指定した場合※

- INTERSECT またはINTERSECT DISTINCT を指定した場合※
- =ANY 指定の限定述語を指定した場合※
- 表副問合せを指定したIN 述語を指定した場合※
- UPDATE 文の更新対象表にカラムストア表を指定した場合
- DELETE 文の削除対象表にカラムストア表を指定した場合
- カラムストア表のコスト情報を収集している場合

注※

HADB サーバが B-tree インデクスを有効に利用できると判断した場合に限り，B-tree インデクスが使用されます。

使用される B-tree インデクスを確認する方法については，「5.2.5　SQL 文の実行時に使用されるインデクスを確認する方法」を参照してください。

5.2.1　インデクスの優先順位と選択規則

表に複数のインデクスが定義されている場合，WHERE 句の探索条件，更新系 SQL のWHERE 探索条件，または結合表のON 探索条件に指定された条件に従って，使用されるインデクスが決定されます。

ここでは，SQL 文の実行時に使用されるインデクスの優先順位と選択規則について説明します。

(1)　インデクスの優先順位

探索条件に指定した述語が次の表に示す形式で指定されている場合に，SQL 文の実行時にインデクスが使用されます。また，SQL 文の実行時に使用されるインデクスは 1 つだけです。そのため，表に複数のインデクスが定義されている場合，次の表に示す優先順位に従って，使用されるインデクスが決定されます。

表 5-1　表に複数のインデクスが定義されている場合のインデクスの優先順位

優先順位	インデクスの条件	この優先順位に該当する指定例（C1 はインデクスが定義されている列）
1	= 条件中にすべてのインデクス構成列が指定されているユニークインデクス（B-tree インデクス）	"C1"=100
		"C1"=100+?
		"T1"."C1"=CAST("T2"."C1" AS INTEGER)
2	= 条件中にインデクス構成列が指定されている B-tree インデクス	"C1"=100
		"C1"=100+?
		"T1"."C1"=CAST("T2"."C1" AS INTEGER)

優先順位	インデクスの条件	この優先順位に該当する指定例（C1 はインデクスが定義されている列）
3	IS NULL 条件中にインデクス構成列が指定されている B-tree インデクス	"C1" IS NULL
4	次のすべての条件を満たすLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス • パターン文字列に定数を指定している • 特殊文字%を指定した前方一致検索である エスケープ文字を指定する場合は，エスケープ文字を定数で指定している必要があります。	"C1" LIKE 'ABC%'
		"C1" LIKE 'AB¥%C%' ESCAPE '¥'
5	次のすべての条件を満たすLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス • パターン文字列に定数を指定している • 優先順位 4 の指定方法以外の前方一致検索である エスケープ文字を指定する場合は，エスケープ文字を定数で指定している必要があります。	"C1" LIKE 'ABC__'
		"C1" LIKE 'ABC%E'
		"C1" LIKE 'AB¥_C%E' ESCAPE '¥'
	パターン文字列に定数を指定した完全一致検索のLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス エスケープ文字を指定する場合は，エスケープ文字を定数で指定している必要があります。	"C1" LIKE 'ABCDE'
		"C1" LIKE 'AB¥_CDE' ESCAPE '¥'
	パターン文字列にユーザ情報取得関数を指定したLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス エスケープ文字を指定する場合は，エスケープ文字を定数で指定している必要があります。	"C1" LIKE CURRENT_USER
		"C1" LIKE CURRENT_USER ESCAPE '¥'
	パターン文字列に？パラメタを指定したLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス エスケープ文字を指定する場合は，エスケープ文字を定数で指定している必要があります。	"C1" LIKE ?
		"C1" LIKE ? ESCAPE '¥'
	エスケープ文字に？パラメタを指定したLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス パターン文字列は，定数，ユーザ情報取得関数，または？パラメタのどれかを指定している必要があります。	"C1" LIKE 'AB¥%C%' ESCAPE ?
		"C1" LIKE CURRENT_USER ESCAPE ?
		"C1" LIKE ? ESCAPE ?
6	ワード検索指定（表記ゆれ補正検索指定または同義語検索指定を指定していない場合に限る）を指定したスカラ関数CONTAINS 中に，インデクス構成列が指定されているテキストインデクス※2，※8	CONTAINS("C1",'WORDCONTEXT("ABC")')>0
7	次のどちらかの条件を満たすスカラ関数CONTAINS 中に，インデクス構成列が指定されているテキストインデクス※2，※4，※8 • ワード検索指定と表記ゆれ補正検索指定を指定している • ワード検索指定と同義語検索指定を指定している	CONTAINS("C1",'WORDCONTEXT(IGNORECASE("ABC"))')>0
		CONTAINS("C1",'WORDCONTEXT(SORTCODE("ABC"))')>0
		CONTAINS("C1",'WORDCONTEXT(SYNONYM("DIC1","ABC"))')>0

優先順位	インデクスの条件	この優先順位に該当する指定例（C1 はインデクスが定義されている列）
8	LIKE 述語中にインデクス構成列が指定されているテキストインデクス※1	”C1” LIKE ’ABC’
		”C1” LIKE ’ABC%’
		”C1” LIKE ’%ABC%’
		”C1” LIKE ’%ABC’
		”C1” LIKE ?
	スカラ関数CONTAINS 中（表記ゆれ補正検索指定，同義語検索指定，またはワード検索指定を指定していない場合に限る）に，インデクス構成列が指定されているテキストインデクス※2	CONTAINS(”C1”,’”ABC”’)>0
9	次のすべての条件を満たすテキストインデクス※1，※3 • LIKE 述語を指定した条件が，論理演算子OR を使って複数指定されている • LIKE 述語中の一致値にインデクス構成列が指定されている	”C1” LIKE ’ABC’ OR ”C1” LIKE ’DEF’
		”C1” LIKE ’ABC%’ OR ”C1” LIKE ’DEF%’
		”C1” LIKE ’%ABC%’ OR ”C1” LIKE ’%DEF%’
		”C1” LIKE ? OR ”C1” LIKE ?
	次のどちらかの指定をしたスカラ関数CONTAINS 中に，インデクス構成列が指定されているテキストインデクス※2，※4 • 表記ゆれ補正検索指定（ワード検索指定を指定していない場合に限る） • 同義語検索指定（ワード検索指定を指定していない場合に限る）	CONTAINS(”C1”,’IGNORECASE(”ABC”)’)>0
		CONTAINS(”C1”,’SORTCODE(”ABC”)’)>0
		CONTAINS(”C1”,’SYNONYM(”DIC1”,”ABC”)’)>0
	次のすべての条件を満たすテキストインデクス※1，※2，※3，※4 • LIKE 述語を指定した条件，およびスカラ関数CONTAINS を指定した条件が，論理演算子OR を使って複数指定されている • LIKE 述語中の一致値にインデクス構成列が指定されている • スカラ関数CONTAINS 中にインデクス構成列が指定されている	”C1” LIKE ’%ABC%’ OR CONTAINS(”C1”,’”DEF”’)>0
		”C1” LIKE ’%ABC%’ OR CONTAINS(”C1”,’”DEF”’)>0 OR CONTAINS(”C1”,’IGNORECASE(”GHI”)’)>0
	次のすべての条件を満たすテキストインデクス※2，※3，※4 • スカラ関数CONTAINS を指定した条件が，論理演算子OR を使って複数指定されている • スカラ関数CONTAINS 中にインデクス構成列が指定されている	CONTAINS(”C1”,’”ABC”’)>0 OR CONTAINS(”C1”,’”DEF”’)>0
		CONTAINS(”C1”,’”ABC”’)>0 OR CONTAINS(”C1”,’IGNORECASE(”ABC”)’)>0 OR CONTAINS(”C1”,’SORTCODE(”ABC”)’)>0
		CONTAINS(”C1”,’”ABC”’)>0 OR

優先順位	インデクスの条件	この優先順位に該当する指定例（C1 はインデクスが定義されている列）
		CONTAINS(”C1”,’IGNORECASE(”DEF”)’)>0 OR CONTAINS(”C1”,’SORTCODE(”GHI”)’)>0 OR CONTAINS(”C1”,’SYNONYM(”DIC1”,”JKL”)’)>0
		CONTAINS(”C1”,’WORDCONTEXT(”ABC”)’)>0 OR CONTAINS(”C1”,’WORDCONTEXT(”DEF”)’)>0
10	LIKE_REGEX 述語中の一致値に，インデクス構成列が指定されているテキストインデクス※4，※5	”C1” LIKE_REGEX ’^ABC’
		”C1” LIKE_REGEX ’^ABC’ FLAG IGNORECASE
11	次のすべての条件を満たすテキストインデクス※3，※4，※5 • LIKE_REGEX 述語を指定した条件が，論理演算子OR を使って複数指定されている • LIKE_REGEX 述語中の一致値にインデクス構成列が指定されている	”C1” LIKE_REGEX ’^ABC’ OR ”C1” LIKE_REGEX ’^DEF’
	次のすべての条件を満たすテキストインデクス※1，※3，※4，※5 • LIKE 述語およびLIKE_REGEX 述語を指定した条件が，論理演算子OR を使って複数指定されている • LIKE 述語中の一致値にインデクス構成列が指定されている • LIKE_REGEX 述語中の一致値にインデクス構成列が指定されている	”C1” LIKE ’%ABC%’ OR ”C1” LIKE_REGEX ’^DEF’
	次のすべての条件を満たすテキストインデクス※2，※3，※4，※5 • LIKE_REGEX 述語を指定した条件，およびスカラ関数CONTAINS を指定した条件が，論理演算子OR を使って複数指定されている • LIKE_REGEX 述語中の一致値にインデクス構成列が指定されている • スカラ関数CONTAINS 中にインデクス構成列が指定されている	”C1” LIKE_REGEX ’^ABC’ OR CONTAINS(”C1”,’”XYZ”’)>0
	次のすべての条件を満たすテキストインデクス※1，※2，※3，※4，※5 • LIKE 述語を指定した条件，LIKE_REGEX 述語を指定した条件，およびスカラ関数CONTAINS を指定した条件が，論理演算子OR を使って複数指定されている • LIKE 述語中の一致値にインデクス構成列が指定されている • LIKE_REGEX 述語中の一致値にインデクス構成列が指定されている • スカラ関数CONTAINS 中にインデクス構成列が指定されている	”C1” LIKE_REGEX ’^ABC’ OR ”C1” LIKE ’%DEF%’ OR CONTAINS(”C1”,’”XYZ”’)>0

優先順位	インデクスの条件	この優先順位に該当する指定例（C1 はインデクスが定義されている列）
12	比較値に値指定だけを指定したIN 述語中に，インデクス構成列が指定されている B-tree インデクス	”C1” IN (10,20,30)
		(”C1”,”C2”,”C3”) IN ((10,20,30), (40,50,60))
	比較値にスカラ演算を含む値指定だけを指定したIN 述語中に，インデクス構成列が指定されている B-tree インデクス	”C1” IN (10,20,30+?)
		”T1”.”C1” IN (CASE WHEN 100=? THEN 10 ELSE 20 END,30,40)
13	BETWEEN 述語中にインデクス構成列が指定されている B-tree インデクス	”C1” BETWEEN 20 AND 40
		”C1” BETWEEN 20 AND 40+?
		”T1”.”C1” BETWEEN ”T2”.”C1”-6 MONTH AND ”T2”.”C1”
	2 つの比較述語を組み合わせた範囲条件に，インデクス構成列が指定されている B-tree インデクス	”C1”>=20 AND ”C1”<=40
14	次の条件を満たすIN 述語中にインデクス構成列が指定されている B-tree インデクス • 比較値に外への参照列を含まない副問合せを指定している	”C1” IN (SELECT ”C1” FROM ”T2”)
	外への参照列を含まない副問合せを指定した=ANY 条件に，インデクス構成列が指定されている B-tree インデクス	”C1”=ANY(SELECT ”C1” FROM ”T2”)
	外への参照列を含まない副問合せを指定した=SOME 条件に，インデクス構成列が指定されている B-tree インデクス	”C1”=SOME(SELECT ”C1” FROM ”T2”)
15	>，>=，く，または<=の条件に，インデクス構成列が指定されている B-tree インデクス	”C1”>50
		”C1”<=200
		”C1”>=50+?
		”T1”.”C1” <”T2”.”C1”||’X’
16	比較値に列指定を含むIN 述語中に，インデクス構成列が指定されている B-tree インデクス	”T1”.”C1” IN (10,”T2”.”C1”)
		”T1”.”C1” IN (10,”T2”.”C1”+?,50)
17	論理演算子OR を指定した条件に，インデクス構成列が指定されている B-tree インデクス※6	”C1”<20 OR ”C1”>40
18	次のすべての条件を満たすLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス※7 • パターン文字列に定数を指定している • 前方一致以外の検索である	”C1” LIKE ’%BCD%’
		”C1” LIKE ’%B¥_CD%’ ESCAPE ’¥’
	次の条件を満たすLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス • パターン文字列に列指定を含む値式を指定している	”T1”.”C1” LIKE ”T2”.”C2” || ’%’

優先順位	インデクスの条件	この優先順位に該当する指定例（C1 はインデクスが定義されている列）
	次の条件を満たすLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス • パターン文字列にスカラ演算を含む値式を指定している	"C1" LIKE CURRENT_USER || '%'
	次の条件を満たすLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス • エスケープ文字に列指定を含む値式を指定している	"T1"."C1" LIKE 'A¥%B@_C%' ESCAPE "T2"."C1"
	次の条件を満たすLIKE 述語中に，インデクス構成列が指定されている B-tree インデクス • エスケープ文字にスカラ演算を含む値式を指定している	"C1" LIKE 'A¥%B@_C%' ESCAPE CASE WHEN 10=? THEN '¥' ELSE '@' END

注

- 複数列インデクスの場合，B-tree インデクスの第 1 構成列から順に，上記の表に示す優先順位が適用されます。
- 基本的には，上記の表に示す優先順位に従って，使用されるインデクスが決まります。ただし，探索条件の指定内容によっては，上記の表に示す優先順位では，有効な評価ができないことがあります。この場合，上記の表の優先順位に従わないインデクスが使用されることがあります。検索時に実際に使用されたインデクスを確認したい場合は，「5.2.5　SQL 文の実行時に使用されるインデクスを確認する方法」を参照してください。

注※1

LIKE 述語に，次に示す形式のパターン文字列を指定した場合，この条件は該当しません。

- パターン文字列に特殊文字の%だけを指定している場合
  （例）"C1" LIKE '%'
- パターン文字列に空文字だけを指定している場合
  （例）"C1" LIKE ''
- パターン文字列に特殊文字以外の文字を 2 文字以上連続して指定していない場合
  （例）"C1" LIKE '%A%'，"C1" LIKE '%A%B%'
- 一致値に指定した列と同じ表の列をパターン文字列に指定している場合
  （例）"T1"."C1" LIKE "T1"."C2"

注※2

スカラ関数CONTAINS（同義語検索指定を指定していない場合，または同義語検索指定を指定しているが，検索文字列に対して同義語が存在しない場合が該当）に，次に示す形式の検索文字列を指定したときは，この条件は該当しません。

- 検索文字列に空文字だけを指定している場合
  （例 1）CONTAINS("C1",'""')>0
  （例 2）CONTAINS("C1",'IGNORECASE("")')>0
  （例 3）CONTAINS("C1",'SORTCODE("")')>0

（例 4）CONTAINS(”C1”,’WORDCONTEXT(””)’)>0

（例 5）CONTAINS(”C1”,’WORDCONTEXT_PREFIX(””)’)>0

- 検索文字列に 1 文字だけを指定している場合

（例 1）CONTAINS(”C1”,’”A”’)>0

（例 2）CONTAINS(”C1”,’IGNORECASE(”A”)’)>0

（例 3）CONTAINS(”C1”,’SORTCODE(”A”)’)>0

（例 4）CONTAINS(”C1”,’WORDCONTEXT(”A”)’)>0

（例 5）CONTAINS(”C1”,’WORDCONTEXT_PREFIX(”A”)’)>0

スカラ関数CONTAINS に同義語検索指定を指定し，検索文字列に対して同義語が存在する場合でも，「A」のような 1 文字の同義語が存在するときは，上記の条件には該当しません。

- スカラ関数CONTAINS にワード検索指定が指定されていて，次の記号を取り除くと 1 文字以下になる検索文字列が指定されている場合
  - 半角空白(0x20)，タブ(0x09)，改行(0x0A)，復帰(0x0D)，ピリオド，疑問符，感嘆符を含む 1 バイトの記号(0x21～0x2F，0x3A～0x40，0x5B～0x60，0x7B～0x7E)

（例 1）CONTAINS(”C1”,’WORDCONTEXT(”###A”)’)>0

（例 2）CONTAINS(”C1”,’WORDCONTEXT_PREFIX(”###A”)’)>0

注※3

論理演算子OR に含まれるすべての条件が，LIKE 述語，LIKE_REGEX 述語，またはスカラ関数CONTAINS を使用した条件である必要があります。また，論理演算子OR に含まれるすべての列は，選択対象となるテキストインデクスのインデクス構成列に含まれている必要があります。

注※4

次に示すどちらかの場合は，CREATE INDEX 文でCORRECTIONRULE（テキストインデクス表記ゆれ補正指定）を指定して定義したテキストインデクスだけが対象になります。

- FLAG にIGNORECASE（またはI）を指定したLIKE_REGEX 述語を指定した場合
- 表記ゆれ補正検索指定を指定したスカラ関数CONTAINS を指定した場合

注※5

LIKE_REGEX 述語の正規表現文字列に指定した文字が 1 文字以下の場合，この条件は該当しません。

注※6

論理演算子OR に含まれるすべての列は，選択対象となる B-tree インデクスのインデクス構成列に含まれている必要があります。また，論理演算子OR に指定される述語の数や条件の形式によっては，インデクスの優先順位が上下することがあります。

注※7

LIKE 条件を指定するインデクス構成列のデータ型が可変長文字列型で，かつ次に示す形式のパターン文字列の場合はこの条件に該当しません。

- 末尾に特殊文字%（パーセント）が指定されていない場合

（例 1）”C1” LIKE ’%BCD’

（例 2）”C1” LIKE ’%BCD_’

- 末尾の特殊文字%（パーセント）の前に半角空白，または特殊文字_（下線）が指定されている場合

（例 1）”C1” LIKE ’%BCD△%’

（例 2）”C1” LIKE ’%BCD_%’

（凡例）△：半角空白

注※8

ワード検索用のテキストインデクスだけが，検索時に使用されるインデクスの選択対象になります。

(2) インデクスの選択規則

SQL 文の実行時に使用されるインデクスは，「(1) インデクスの優先順位」で説明したインデクスの優先順位だけでは決まりません。ここで説明する条件なども加えて，使用されるインデクスが総合的に決定されます。

(a) B-tree インデクスの選択規則

B-tree インデクスの選択規則を次の表に示します。

なお，選択規則は，1 番目の規則で判定できなければ 2 番目の規則を適用するというように，1 番目から順に判定されます。

表 5-2 B-tree インデクスの選択規則

選択規則	条件指定方法
1	ユニークインデクスのすべての列に=条件を指定している
2	サーチ条件の先頭から連続する=条件の中に，=結合条件が含まれている方を優先する
3	B-tree インデクスの第 1 構成列にサーチ条件が指定されているインデクス同士の場合は，そのサーチ条件の優先順位に従う
4	サーチ条件の数が多い方を優先する
5	行値構成子でサーチ条件の数が多い方を優先する
6	キー条件の数が多い方を優先する
7	インデクス構成列数が少ない方を優先する（ただし，選択規則 1 に該当するインデクス同士の場合は，インデクス構成列数が多い方を優先する）
8	SQL 文上で前に指定された条件をサーチ条件に使用している方を優先する
9	作業表を作らない方を優先する
10	ユニークインデクスを優先する（ただし，サーチ条件が指定されていないインデクス同士の場合は，非ユニークインデクスを優先する）
11	インデクスのキー長が短い方を優先する
12	「選択規則 1 から 10」以外は内部処理に依存して B-tree インデクスが選択される

サーチ条件とキー条件については，「5.4.1　B-tree インデクスによる評価方式」を参照してください。

(b)　テキストインデクスの選択規則

テキストインデクスの選択規則を次の表に示します。

なお，選択規則は，1 番目の規則で判定できなければ 2 番目の規則を適用するというように，1 番目から順に判定されます。

表 5-3　テキストインデクスの選択規則

選択規則	条件指定方法
1	• テキストインデクスで評価可能なLIKE 述語を指定した条件の指定数が多い方を優先する • テキストインデクスで評価可能なLIKE_REGEX 述語を指定した条件の指定数が多い方を優先する • テキストインデクスで評価可能なスカラ関数CONTAINS を指定した条件の指定数が多い方を優先する
2	• LIKE 述語のパターン文字列が長い方を優先する • LIKE_REGEX 述語の正規表現文字列が長い方を優先する • スカラ関数CONTAINS の検索文字列が長い方を優先する
3	インデクス構成列の定義長が短い方を優先する
4	• テキストインデクスで評価可能なOR 条件に指定されているLIKE 述語，LIKE_REGEX 述語，およびスカラ関数CONTAINS のうち，指定数が少ない方を優先する • 同義語検索指定を指定したスカラ関数CONTAINS が指定されている場合は，指定数に検索文字列に対する同義語数を加算して比較する
5	「選択規則 1 から 4」以外は，内部処理に依存してインデクスが選択される

(c)　留意事項

- 取得したコスト情報などから，HADB がインデクスを有効に利用できないと判断したときは，ここで説明した選択規則とは異なる扱いをすることがあります。
- WHERE 句の探索条件，更新系 SQL のWHERE 探索条件，または結合表のON 探索条件に，否定の述語だけしか指定されていない場合など，HADB がインデクスを有効に利用できないと判断したときは，インデクスが使用されないことがあります。

5.2.2　表の検索時に使用されるインデクスの例

表の検索時に使用されるインデクスの例を説明します。

(1)　例 1（B-tree インデクス（単一列インデクス）の場合）

B-tree インデクスの定義例：

```
CREATE INDEX "IDX_C1"
       ON "T1" ("C1")
```

```
      IN "DBAREA01"
      EMPTY
```

実行するSELECT文の例：

```
SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1"=100

SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1">100 AND "C2"='U0100'
```

上記のSELECT文を実行した場合，B-treeインデクスIDX_C1が使用されます。

■B-treeインデクスが使用されないケース

次に示す場合は，B-treeインデクスは使用されません。

B-treeインデクスの定義例：

```
CREATE INDEX "IDX_C1"
       ON "T1" ("C1")
     IN "DBAREA01"
     EMPTY
```

実行するSELECT文の例：

```
SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1">100 OR "C2"='U0100'
```

OR条件の場合，C1列だけ（またはC2列だけ）にB-treeインデクスが定義されていても，そのB-treeインデクスは使用されません。そのため，上記のSELECT文を実行した場合，B-treeインデクスIDX_C1は使用されません。

また，C1列とC2列の両方にそれぞれB-treeインデクスが定義されていても，それらのB-treeインデクスは使用されません。

(2) 例2（B-treeインデクス（複数列インデクス）の場合）

B-treeインデクスの定義例：

```
CREATE INDEX "IDX_C2C1"
      ON "T1" ("C2","C1")
    IN "DBAREA01"
    EMPTY
```

実行するSELECT文の例：

```
SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1"=100

SELECT "C1","C2","C3" FROM "T1"
    WHERE "C2"='U0100'
```

```
SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1">100 AND "C2"='U0100'

SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1">100 OR "C2"='U0100'
```

上記のSELECT 文を実行した場合，B-tree インデクスIDX_C2C1 が使用されます。

(3) 例 3（テキストインデクスの場合）

テキストインデクスの定義例：

```
CREATE INDEX "IDX_TXT_C1"
       ON "T1" ("C1")
     IN "DBAREA01"
     EMPTY
     INDEXTYPE TEXT
```

実行する SELECT 文の例：

```
SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1" LIKE '%XYZ%'

SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1" LIKE '%XYZ%' AND "C2"='U0100'
```

上記のSELECT 文を実行した場合，テキストインデクスIDX_TXT_C1 が使用されます。

■テキストインデクスが使用されないケース

次に示す場合は，テキストインデクスは使用されません。

テキストインデクスの定義例：

```
CREATE INDEX "IDX_TXT_C1"
       ON "T1" ("C1")
     IN "DBAREA01"
     EMPTY
     INDEXTYPE TEXT
```

実行する SELECT 文の例：

```
SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1" LIKE '%XYZ%' OR "C2"='U0100'
```

OR 条件の場合，C1 列だけ（またはC2 列だけ）にテキストインデクスが定義されていても，そのテキストインデクスは使用されません。そのため，上記のSELECT 文を実行した場合，テキストインデクスIDX_TXT_C1 は使用されません。

なお，(1)～(3)で説明しているインデクスが使用される例と使用されない例は代表的な例です。また，上記で説明しているインデクスが使用される例であっても，探索条件の書き方によってはインデクスが使用

されないことがあります。検索時に実際に使用されるインデクスを確認したい場合は，「5.2.5　SQL 文の実行時に使用されるインデクスを確認する方法」を参照してください。

5.2.3　表の検索時に使用されるインデクスの例（インデクスの優先順位の例）

表に複数のインデクスが定義されている場合，「表 5-1　表に複数のインデクスが定義されている場合のインデクスの優先順位」に示す優先順位に従って，使用されるインデクスが決定されます。

ここでは，表の検索時に使用されるインデクスの優先順位の例（代表的な例）を説明します。

検索時に実際に使用されるインデクスを確認したい場合は，「5.2.5　SQL 文の実行時に使用されるインデクスを確認する方法」を参照してください。

(1)　例 1（単一列インデクス同士の優先順位）

B-tree インデクス（単一列インデクス）同士の優先順位の例を説明します。

B-tree インデクスの定義例：

```
CREATE INDEX "IDX_C1"
      ON "T1" ("C1")
    IN "DBAREA01"
    EMPTY

CREATE INDEX "IDX_C2"
      ON "T1" ("C2")
    IN "DBAREA01"
    EMPTY
```

実行する SELECT 文の例：

```
SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1">100 AND "C2"='U0100'
```

この場合，B-tree インデクス**IDX_C1** の優先順位は 15 になり，B-tree インデクス**IDX_C2** の優先順位は 2 になります。したがって，B-tree インデクス**IDX_C2** が検索時に使用されます。

> **重要**
>
> 上記のケースで，優先順位に従ってどちらかの B-tree インデクスだけが使用される場合，絞り込みがより効く B-tree インデクスが使用されるように，可能ならば探索条件を変更してください。絞り込みが効く B-tree インデクスを使用した方が性能向上が見込めます。

(2)　例 2（単一列インデクス同士の優先順位）

B-tree インデクス（単一列インデクス）同士の優先順位の例を説明します。

B-tree インデクスの定義例：

```
CREATE INDEX ”IDX_C1”
      ON ”T1” (”C1”)
    IN ”DBAREA01”
    EMPTY

CREATE INDEX ”IDX_C2”
      ON ”T1” (”C2”)
    IN ”DBAREA01”
    EMPTY
```

実行する SELECT 文の例：

```
SELECT * FROM ”T1”
    WHERE ”C2”=’U0100’ AND ”C1”=100
```

この場合，B-tree インデクスIDX_C1 と，B-tree インデクスIDX_C2 の優先順位はどちらも 2 になります。この場合，探索条件で先に書いた列の B-tree インデクスが優先されるため，B-tree インデクスIDX_C2 が検索時に使用されます。

(3) 例 3（複数列インデクス同士の優先順位）

B-tree インデクス（複数列インデクス）同士の優先順位の例を説明します。

B-tree インデクスの定義例：

```
CREATE INDEX ”IDX_C1C2”
      ON ”T1” (”C1”,”C2”)
    IN ”DBAREA01”
    EMPTY

CREATE INDEX ”IDX_C2C3”
      ON ”T1” (”C2”,”C3”)
    IN ”DBAREA01”
    EMPTY
```

実行する SELECT 文の例：

```
SELECT ”C1”,”C2”,”C3” FROM ”T1”
    WHERE ”C2”=’U0100’
```

この場合，B-tree インデクスIDX_C2C3 が使用されます。

IDX_C1C2 はC2 列がインデクス第 2 構成列のため，C2 列がインデクス第 1 構成列のIDX_C2C3 が使用されます。

(4) 例 4（単一列インデクスと複数列インデクスの優先順位）

B-tree インデクスの単一列インデクスと複数列インデクスの優先順位の例を説明します。

B-tree インデクスの定義例：

```
CREATE INDEX ”IDX_C1”
      ON ”T1” (”C1”)
    IN ”DBAREA01”
    EMPTY

CREATE INDEX ”IDX_C3C2”
      ON ”T1” (”C3”,”C2”)
    IN ”DBAREA01”
    EMPTY
```

実行する SELECT 文の例：

```
SELECT ”C1”,”C2”,”C3” FROM ”T1”
    WHERE ”C1”>100 AND ”C2”='U0100'
```

この場合，B-tree インデクスIDX_C1 が使用されます。

IDX_C3C2 はC2 列がインデクス第 2 構成列のため，C1 列をインデクス構成列とするIDX_C1 が使用されます。

(5) 例 5（単一列インデクスと複数列インデクスの優先順位）

B-tree インデクスの単一列インデクスと複数列インデクスの優先順位の例を説明します。

B-tree インデクスの定義例：

```
CREATE INDEX ”IDX_C1”
      ON ”T1” (”C1”)
    IN ”DBAREA01”
    EMPTY

CREATE INDEX ”IDX_C2C3”
      ON ”T1” (”C2”,”C3”)
    IN ”DBAREA01”
    EMPTY
```

実行する SELECT 文の例：

```
SELECT ”C1”,”C2”,”C3” FROM ”T1”
    WHERE ”C1”>100 AND ”C2”='U0100'
```

この場合，B-tree インデクスIDX_C1 の優先順位は 15 になり，B-tree インデクスIDX_C2C3 の優先順位は 2 になります。したがって，B-tree インデクスIDX_C2C3 が使用されます。

C1 列をインデクス構成列とするIDX_C1 と，C2 列をインデクス第 1 構成列とするIDX_C2C3 は，「表 5-1　表に複数のインデクスが定義されている場合のインデクスの優先順位」に示す優先順位の比較対象になります。

(6)　例 6（テキストインデクスと B-tree インデクスの優先順位）

テキストインデクスと B-tree インデクス（単一列インデクス）の優先順位の例を説明します。

インデクスの定義例：

```
CREATE INDEX "IDX_TXT_C1"       ←テキストインデクスの定義
      ON "T1" ("C1")
    IN "DBAREA01"
    EMPTY
    INDEXTYPE TEXT

CREATE INDEX "IDX_C2"           ←B-treeインデクスの定義
      ON "T1" ("C2")
    IN "DBAREA01"
    EMPTY
```

実行する SELECT 文の例：

```
SELECT "C1","C2","C3" FROM "T1"
    WHERE "C1" LIKE '%XYZ%' AND "C2" LIKE 'ABC%'
```

この場合，テキストインデクスIDX_TXT_C1 の優先順位は 8 になり，B-tree インデクスIDX_C2 の優先順位は 4 になります。したがって，B-tree インデクスIDX_C2 が検索時に使用されます。

> **重要**
>
> 上記のケースで，優先順位に従ってどちらかのインデクスだけが使用される場合，絞り込みがより効くインデクスが使用されるように，可能ならば探索条件を変更してください。絞り込みが効くインデクスを使用した方が性能向上が見込めます。

(7)　検索時に使用されるインデクスを変更したい場合

検索時に使用されるインデクスを変更する方法について説明します。

(a)　優先順位が同じ場合は，使用したいインデクス構成列を先に書く

「(2)　例 2（単一列インデクス同士の優先順位）」で説明したとおり，優先順位が同じ場合，探索条件で先に書いた列の B-tree インデクスが優先されます。これを利用して，使用する B-tree インデクスを変更できます。例えば，次のように探索条件の指定を変更したとします。

＜変更前＞

```
SELECT * FROM "T1"
    WHERE "C3"='A001' AND "C2"='U0100'
```

＜変更後＞

```
SELECT * FROM "T1"
    WHERE "C2"='U0100' AND "C3"='A001'
```

変更前はC3 列に定義した B-tree インデクスIDX_C3 が使用されていましたが，変更後はC2 列に定義した B-tree インデクスIDX_C2 が使用されます。

(b)　現在使用されているインデクスの優先順位を下げる

現在使用されているインデクスの優先順位を下げて，もう一方のインデクスを使用するようにします。例えば，次のように探索条件の指定を変更したとします。

＜変更前＞

```
SELECT * FROM "T1"
    WHERE "C1"=100 AND "C2" LIKE 'ABC%'
```

＜変更後＞

```
SELECT * FROM "T1"
    WHERE "C1" BETWEEN 100 AND 100 AND "C2" LIKE 'ABC%'
```

変更前はIDX_C1 の優先順位は 2，IDX_C2 の優先順位は 4 でしたが，変更後はIDX_C1 の優先順位が 13 になります。そのため，変更前は B-tree インデクスIDX_C1 が使用されていましたが，変更後は B-tree インデクスIDX_C2 が使用されます。

(c)　インデクス指定を使用する

検索時に使用するインデクスをインデクス指定で指定できます。例を次に示します。

＜変更前＞

```
SELECT * FROM "T1"
    WHERE "C1"=100 AND "C2" LIKE 'ABC%'
```

C1 列に定義した B-tree インデクスIDX_C1 の優先順位は 2，C2 列に定義した B-tree インデクスIDX_C2 の優先順位は 4 のため，IDX_C1 が使用されます。

＜変更後＞

```
SELECT * FROM "T1" /*>> WITH INDEX (IDX_C2) <<*/
    WHERE "C1"=100 AND "C2" LIKE 'ABC%'
```

下線部分のインデクス指定で，検索時に使用するインデクスを指定できます。上記のSELECT 文を実行した場合，IDX_C2 が使用されます。

インデクス指定の詳細については，マニュアル『HADB　SQL リファレンス』の『インデクス指定』を参照してください。

5.2.4　インデクスが使用されないケース

探索条件に次の条件を指定している場合，インデクスが使用されません。なお，例に示すC1，C2 は表の列名を意味しています。

- 否定の条件を指定している場合

  次に示す例のように，否定の条件を探索条件に指定している場合，C1 列をインデクス構成列とするインデクスが定義されていても，そのインデクスは使用されません。

  （例）

```
WHERE "C1"<>100
WHERE "C1" IS NOT NULL
WHERE "C1" NOT LIKE 'ABC%'
WHERE "C1" NOT IN (10,20,30)
WHERE "C1" NOT BETWEEN 20 AND 40
```

- 論理演算子NOT を指定している場合

  （例）

```
WHERE NOT ("C1"=100)
```

- 四則演算やCASE 式などのスカラ演算を含む条件を指定している場合

  次に示す例のように，四則演算やCASE 式などのスカラ演算を含む条件を探索条件に指定している場合，C1 列をインデクス構成列とするインデクスが定義されていても，そのインデクスは使用されません。

  （例）

```
WHERE C1*10=200
```

  ただし，スカラ演算の指定方法によっては，探索条件が自動的に等価変換され，インデクスが使用されることがあります。スカラ演算に関する等価変換については，「5.11.5　スカラ演算に関する等価変換」を参照してください。

  なお，スカラ演算の中に定数だけを指定している場合，そのスカラ演算は定数として扱われることがあります。定数と等価なスカラ演算については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。

- 外への参照列を含む副問合せが指定されているIN 副問合せを指定している場合

  （例）

```
WHERE "T1"."C1" IN (SELECT "C1" FROM "T2" WHERE "C2"="T1"."C2")
```

- 外への参照列を含む副問合せが指定されている=ANY の限定述語を指定している場合

  （例）

```
WHERE "T1"."C1"=ANY(SELECT "C1" FROM "T2" WHERE "C2"="T1"."C2")
```

- 外への参照列を含む副問合せが指定されている=SOME の限定述語を指定している場合

  （例）

```
WHERE "T1"."C1"=SOME(SELECT "C1" FROM "T2" WHERE "C2"="T1"."C2")
```

- =ANY または=SOME を除く限定述語を指定している場合

  （例）

```
WHERE "C1"<>ANY(SELECT "C1" FROM "T2")
WHERE "C1"<>SOME(SELECT "C1" FROM "T2")
WHERE "C1"=ALL(SELECT "C1" FROM "T2")
```

- EXISTS 述語を指定している場合

  （例）

```
WHERE EXISTS(SELECT * FROM "T2")
```

- LIKE 述語のパターン文字列に次の指定をしている場合
  - パターン文字列に特殊文字の%だけを指定している場合

    （例）

```
WHERE "C1" LIKE '%'
```

  - パターン文字列に空文字だけを指定している場合

    （例）

```
WHERE "C1" LIKE ''
```

  - パターン文字列に，特殊文字以外の文字を 2 文字以上連続して指定していない場合

    （例）

```
WHERE "C1" LIKE '%A%'
WHERE "C1" LIKE '%A%B%'
```

  - 一致値に指定した列と同じ表の列をパターン文字列に指定している場合

    （例）

```
WHERE "T1"."C1" LIKE "T1"."C2"
```

  上記の例のような指定をしている場合，C1 列をインデクス構成列とするテキストインデクスが定義されていても，そのテキストインデクスは使用されません。

5.2.5　SQL 文の実行時に使用されるインデクスを確認する方法

SQL 文の実行時に使用されるインデクスは，アクセスパスで確認できます。アクセスパスについては，次に示す個所を参照してください。

- アクセスパスの確認方法

  「6.1.2　アクセスパスを確認するには」を参照してください。

- アクセスパスに表示される内容

  「6.1.5　詳細表示に出力される情報」の「(1)　表の検索方式，インデクス，および集まり導出表に関する情報」の「(b)　インデクスに関する情報」を参照してください。

アクセスパスを確認して，意図したとおりにインデクスが使用されているかどうかを確認してください。

検索対象表にインデクスが定義されている場合，指定した探索条件に基づいてインデクスを使用した検索が行われます。ただし，指定した探索条件によっては，インデクスが使用されなかったり，期待したインデクスとは異なるインデクスが使用されたりすることがあります。上記の方法で確認した結果，意図したとおりにインデクスが使用されていない場合，探索条件とインデクスの定義が合っていない可能性があります。この場合，インデクスの定義の変更を検討するか，または探索条件の変更を検討してみてください。

なお，インデクスの定義を変更する必要がある場合，AP 開発者は HADB のシステム設計者またはシステム管理者に，インデクスの定義を変更するよう依頼をしてください。

5.2.6 テキストインデクスを使用して検索する際の留意事項

論理演算子OR を使用して，テキストインデクスの構成列を指定したLIKE 述語，LIKE_REGEX 述語，およびスカラ関数CONTAINS を複数指定した場合，すべての条件に指定した検索文字列を使用して検索処理が行われます。

（例）C1 はテキストインデクスの構成列とします。

```
SELECT * FROM "T1" WHERE "C1" LIKE 'ABCDEFG%'
                      OR "C1" LIKE_REGEX 'XYZ[0-9]+'
```

上記のSELECT 文を実行した場合，LIKE 述語およびLIKE_REGEX 述語の条件に指定した検索文字列（ABCDEFG%とXYZ[0-9]+）を使用して検索処理が行われます。

LIKE 述語に指定したパターン文字列，LIKE_REGEX 述語に指定した正規表現文字列，スカラ関数CONTAINS に指定した検索条件式指定の文字列の合計数が 1,001 文字以上になると，検索処理時間が急激に増加するおそれがあります。そのため，合計文字数が 1,001 文字以上の場合は，次に示すどちらかの対処をすることを検討してください。

- UNION を使用して，それぞれの検索結果を統合するなど SQL 文を変更する
- インデクス指定などを指定し，テキストインデクスを使用しない検索を行う

UNION については，マニュアル『HADB　SQL リファレンス』の『問合せ式』を参照してください。インデクス指定については，マニュアル『HADB　SQL リファレンス』の『インデクス指定』を参照してください。

5.3　SQL 文の実行時に使用されるレンジインデクス

ここでは，SQL 文の実行時にレンジインデクスが使用される条件と，SQL 文の実行時に使用されたレンジインデクスを確認する方法について説明します。

5.3.1　SQL 文の実行時にレンジインデクスが使用される条件

表にレンジインデクスが定義されている場合，WHERE 句の探索条件，更新系 SQL のWHERE 探索条件，結合表のON 探索条件，および SQL 文の実行時に使用される B-tree インデクスまたはテキストインデクスによって，SQL 文の実行時にレンジインデクスが使用されるかどうかが決まります。

ここでは，チャンクのスキップに使用するレンジインデクスの決定方法と，セグメントのスキップに使用するレンジインデクスの決定方法について説明します。

留意事項

- ここで説明しているレンジインデクスが使用される条件は，内部導出表の展開後の問合せ式，または探索条件の等価変換によって変換された探索条件に対して適用されます。内部導出表の展開については，マニュアル『HADB　SQL リファレンス』の『内部導出表』を参照してください。探索条件の等価変換機能については，「5.11　探索条件の等価変換」を参照してください。
- 探索条件中に指定した値式で，スカラ演算中に定数だけを指定している場合，そのスカラ演算を定数と見なすことがあります。定数と等価なスカラ演算については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。
- 表を結合する際，表の結合方式によっては結合条件の評価時にレンジインデクスを使用しないことがあります。表の結合方式については，「5.5　表の結合方式」を参照してください。
- 副問合せを指定した場合，適用される副問合せの処理方式によってはレンジインデクスを使用しないことがあります。副問合せの処理方式については，「5.6　副問合せの処理方式」を参照してください。

(1)　チャンクのスキップに使用されるレンジインデクス

次の表に示す条件をすべて満たす場合，SQL 文の実行時にレンジインデクスが使用されて，チャンクのスキップが行われます。

表 5-4　レンジインデクスが使用される条件

項番	レンジインデクスが使用される条件	対応する例
1	WHERE 句の探索条件，更新系 SQL のWHERE 探索条件，または結合表のON 探索条件で，次に示すどれかの述語に，レンジインデクスの構成列※1 が指定されている必要があります。	－
	• 比較述語	(例 1) (例 2)

項番	レンジインデクスが使用される条件	対応する例
	ただし，比較演算子の左側と右側に同じ表の列※2 が指定されている場合は，レンジインデクスは使用されません。	(例 3) (例 14) (例 15)
	• BETWEEN 述語 ただし，値式 1 と同じ表の列※2 が，値式 2 または値式 3 に指定されている場合は，レンジインデクスは使用されません。	(例 4) (例 5)
	• IN 述語（値式） ただし，IN 述語の左側の値式と同じ表の列※2 が，IN 述語の右側の値式に指定されている場合は，レンジインデクスは使用されません。	(例 6) (例 7)
	• IN 述語（表副問合せ） 副問合せの処理方式にハッシュ実行が適用された場合に限り，レンジインデクスが使用されます。	(例 17)
	• LIKE 述語 LIKE 指定の場合は，次に示すどれかの条件に該当している必要があります。なお，LIKE 述語中にESCAPE を指定している場合は，エスケープ文字が，定数または？パラメタで指定されている必要があります。 ・パターン文字列にユーザ情報取得関数だけが指定されている ・パターン文字列に？パラメタだけが指定されている ・パターン文字列に，'文字列%'（%は特殊文字）を先頭から含む定数だけが指定されている ・パターン文字列に，'文字列_'（_は特殊文字）を先頭から含む定数だけが指定されている ・パターン文字列に特殊文字（%，_ ）を含まない定数だけが指定されている（完全一致） NOT LIKE 指定の場合は，次に示すどれかの条件に該当している必要があります。 ・パターン文字列に？パラメタだけが指定されている ・パターン文字列に，'文字列%'（%は特殊文字）を先頭から含む定数だけが指定されている	(例 8) (例 9)
	• 限定述語 副問合せの処理方式にハッシュ実行が適用された場合に限り，レンジインデクスが使用されます。	(例 17)
2	項番 1 で説明している述語が，論理演算子OR またはNOT を使用した条件中に指定されている場合は，レンジインデクスは使用されません。	(例 10)
3	項番 1 で説明している述語中にスカラ演算がある場合は，レンジインデクスは使用されません。	(例 11)
4	項番 1 で説明している述語中に，3 つ以上の異なる表の列※2 を指定している場合は，レンジインデクスは使用されません。	(例 12)
5	• 外への参照列を含む副問合せ中に項番 1 で説明している述語が指定されていて，かつ副問合せの処理方式にネストループ実行が適用された場合 ただし，項番 1 で説明している述語中に，外への参照列がある場合は，外への参照列をインデクス構成列とするレンジインデクスは使用されません。 • 外への参照列を含む副問合せ中に項番 1 で説明している述語が指定されていて，かつ副問合せの処理方式にハッシュ実行が適用された場合 項番 1 で説明している述語中に，「*外への参照列＝列指定*」または「*列指定＝外への参照列*」のどちらかの形式で指定された外への参照列がある場合に限り，レンジインデクスが使用されます。	(例 13) (例 16)

注※1

配列要素参照の配列値式に指定した列（次の例の下線部分の列）がレンジインデクスの構成列の場合，レンジインデクスが使用される条件に該当します。

（例）

”C1”[ANY]=10

”C1”[ANY(1)]=1

”C1”[2]=1

注※2

配列要素参照の配列値式に指定された列も該当します。

「表 5-4　レンジインデクスが使用される条件」に示す条件をすべて満たすレンジインデクスが表に複数定義されている場合，条件を満たすレンジインデクスがすべて（複数個）使用されます。

チャンクのスキップに使用できるレンジインデクスかどうかを確認する場合は，マニュアル『HADB　システム構築・運用ガイド』の『レンジインデクスの確認（チャンクのスキップの可否)』を参照してください。

(2)　セグメントのスキップに使用されるレンジインデクス

「表 5-4　レンジインデクスが使用される条件」に示す条件をすべて満たす場合，SQL 文の実行時にレンジインデクスが使用されて，セグメントのスキップが行われます。「表 5-4　レンジインデクスが使用される条件」に示す条件をすべて満たすレンジインデクスが表に複数定義されている場合，条件を満たすレンジインデクスがすべて（複数個）使用されます。

ただし，SQL 文の実行時に B-tree インデクスまたはテキストインデクスが使用された場合，レンジインデクスを使用したセグメントのスキップは行われません。

5.3.2　検索時に使用されるレンジインデクスの例

(1)　例 1

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1”(”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” WHERE ”C1”>10
```

[説明]

この例の場合，条件に合致するデータを含まないチャンクおよびセグメントをスキップするために，レンジインデクスRIDX1 が使用されます。

(2) 例 2

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1”(”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
CREATE INDEX ”RIDX2” ON ”T1”(”C2”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
CREATE INDEX ”RIDX3” ON ”T1”(”C3”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” WHERE ”C1”=10 AND ”C2”>20 AND ”C3”>30
```

[説明]

この例の場合，条件に合致するデータを含まないチャンクおよびセグメントをスキップするために，レンジインデクスRIDX1，RIDX2 およびRIDX3 が使用されます。

(3) 例 3（レンジインデクスが使用されない例）

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1”(”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” WHERE ”C1”>”C2”
```

[説明]

比較演算子の左側と右側に同じ表の列が指定されている場合は，レンジインデクスは使用されません（条件に合致するデータを含まないチャンクおよびセグメントはスキップされません）。この例の場合，C1，C2 は表T1 の列のため，レンジインデクスRIDX1 は使用されません。

(4) 例 4

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1”(”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” WHERE ”C1” BETWEEN 10 AND 30
```

[説明]

この例の場合，条件に合致するデータを含まないチャンクおよびセグメントをスキップするために，レンジインデクスRIDX1 が使用されます。

(5) 例５（レンジインデクスが使用されない例）

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1”(”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” WHERE ”C1” BETWEEN ”C2” AND ”C3”
```

[説明]

値式 1 と同じ表の列が，値式 2 または値式 3 に指定されている場合は，レンジインデクスは使用されません（条件に合致するデータを含まないチャンクおよびセグメントはスキップされません）。この例の場合，C1，C2，C3 は表T1 の列のため，レンジインデクスRIDX1 は使用されません。

(6) 例６

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1”(”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” WHERE ”C1” IN (10,20,30)
```

[説明]

この例の場合，条件に合致するデータを含まないチャンクおよびセグメントをスキップするために，レンジインデクスRIDX1 が使用されます。

(7) 例７（レンジインデクスが使用されない例）

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1”(”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” WHERE ”C1” IN (”C2”,20,30)
```

[説明]

IN 述語の左側の値式と同じ表の列が，IN 述語の右側の値式に指定されている場合は，レンジインデクスは使用されません（条件に合致するデータを含まないチャンクおよびセグメントはスキップされません）。この例の場合，C1，C2 は表T1 の列のため，レンジインデクスRIDX1 は使用されません。

(8) 例８

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1”(”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM "T1" WHERE "C1" LIKE CURRENT_USER
SELECT * FROM "T1" WHERE "C1" LIKE ?
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC%'
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC%E'
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC%E%'
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC%E%G'
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC_'
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC_E'
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC_E_'
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC_E_G'
SELECT * FROM "T1" WHERE "C1" LIKE 'ABC'
SELECT * FROM "T1" WHERE "C1" NOT LIKE ?
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC%'
SELECT * FROM "T1" WHERE "C1" LIKE 'AB¥_C%' ESCAPE '¥'
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'AB¥_C%' ESCAPE '¥'
```

[説明]

上記の例の場合，どのSELECT 文を実行しても，条件に合致するデータを含まないチャンクおよびセグメントをスキップするために，レンジインデクスRIDX1 が使用されます。

LIKE 述語に？パラメタを指定した場合はレンジインデクスが使用されますが，次に示す条件に該当しないパターン文字列を指定した場合は，レンジインデクスの効果はありません。

- LIKE の場合
  - '文字列%'（%は特殊文字）を先頭から含むパターン文字列
  - '文字列_'（_は特殊文字）を先頭から含むパターン文字列
  - %または_の特殊文字を含まないパターン文字列（完全一致）
- NOT LIKE の場合
  - '文字列%'（%は特殊文字）のパターン文字列

(9) 例９（レンジインデクスが使用されない例）

レンジインデクスの定義

```
CREATE INDEX "RIDX1" ON "T1"("C1") IN "DBAREA01" EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM "T1" WHERE "C1" LIKE '%ABC%'
SELECT * FROM "T1" WHERE "C1" LIKE '%ABC'
SELECT * FROM "T1" WHERE "C1" LIKE '_ABC_'
SELECT * FROM "T1" WHERE "C1" LIKE '_ABC'
SELECT * FROM "T1" WHERE "C1" NOT LIKE CURRENT_USER
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC%E'
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC%E%'
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC%E%G'
SELECT * FROM "T1" WHERE "C1" NOT LIKE '%ABC%'
SELECT * FROM "T1" WHERE "C1" NOT LIKE '%ABC'
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC_'
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC_E'
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC_E_'
```

```
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC_E_G'
SELECT * FROM "T1" WHERE "C1" NOT LIKE '_ABC_'
SELECT * FROM "T1" WHERE "C1" NOT LIKE '_ABC'
SELECT * FROM "T1" WHERE "C1" NOT LIKE 'ABC'
SELECT * FROM "T1" WHERE "C1" LIKE '%AB¥_C%' ESCAPE '¥'
SELECT * FROM "T1" WHERE "C1" LIKE 'A¥%B@_C%'
                    ESCAPE CASE WHEN 10=? THEN '¥' ELSE '@' END
SELECT * FROM "T1" WHERE "C1" NOT LIKE '%AB¥_C%' ESCAPE '¥'
```

[説明]

上記のすべてのSELECT 文は，「表 5-4　レンジインデクスが使用される条件」の項番 1 の『LIKE 述語』の条件を満たしません。したがって，レンジインデクスRIDX1 は使用されません（条件に合致するデータを含まないチャンクおよびセグメントはスキップされません）。

(10)　例 10（レンジインデクスが使用されない例）

レンジインデクスの定義

```
CREATE INDEX "RIDX1" ON "T1"("C1") IN "DBAREA01" EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM "T1" WHERE "C1"=10 OR "C2"=20
SELECT * FROM "T1" WHERE NOT("C1"=10)
```

[説明]

論理演算子OR またはNOT を使用した条件中に指定されている場合は，レンジインデクスは使用されません（条件に合致するデータを含まないチャンクおよびセグメントはスキップされません）。

(11)　例 11（レンジインデクスが使用されない例）

レンジインデクスの定義

```
CREATE INDEX "RIDX1" ON "T1"("C1") IN "DBAREA01" EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM "T1" WHERE "T1"."C1"+10=20
```

[説明]

レンジインデクスの構成列を含むスカラ演算を使用している場合は，レンジインデクスは使用されません（条件に合致するデータを含まないチャンクおよびセグメントはスキップされません）。

(12)　例 12（レンジインデクスが使用されない例）

レンジインデクスの定義

```
CREATE INDEX "RIDX1" ON "T1"("C1") IN "DBAREA01" EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM "T1","T2","T3" WHERE "T1"."C1" BETWEEN "T2"."C2" AND "T3"."C3"
SELECT * FROM "T1","T2","T3" WHERE "T1"."C1" IN ("T2"."C2","T3"."C3")
```

[説明]

述語中に 3 つ以上の異なる表の列を指定している場合は，レンジインデクスは使用されません（条件に合致するデータを含まないチャンクおよびセグメントはスキップされません）。

(13) 例 13

レンジインデクスの定義

```
CREATE INDEX "RIDX1" ON "T1"("C2") IN "DBAREA01" EMPTY INDEXTYPE RANGE
CREATE INDEX "RIDX2" ON "T1"("C3") IN "DBAREA01" EMPTY INDEXTYPE RANGE
CREATE INDEX "RIDX3" ON "T2"("C1") IN "DBAREA01" EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM "T1" "X"
    WHERE "X"."C1"=ANY(SELECT "T2"."C1" FROM "T2"
                       WHERE "X"."C2"=10 AND "X"."C3"="T2"."C1")
```

[説明]

レンジインデクスRIDX1 およびRIDX2 のインデクス構成列（表T1 のC2 列，C3 列）は，外への参照列のため，レンジインデクスRIDX1 およびRIDX2 は使用されません。

レンジインデクスRIDX3 は，表T2 のC1 列（外への参照列ではない列）をインデクス構成列とし，条件に合致するデータを含まないチャンクおよびセグメントをスキップするために使用されます。

(14) 例 14

レンジインデクスの定義

```
CREATE INDEX "RIDX1" ON "T2" ("C1") IN "DBAREA01" EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM "T1","T2" WHERE "T1"."C1"<"T2"."C1"
SELECT * FROM "T1" INNER JOIN "T2" ON "T1"."C1"<"T2"."C1"
```

[説明]

表T1 を外表，表T2 を内表とするネストループジョインが適用される場合，チャンクおよびセグメントをスキップするために，レンジインデクスRIDX1 が使用されます。ネストループジョインの詳細については，「5.5.1 ネストループジョインとは」を参照してください。

(15) 例 15

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T2” (”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1”,”T2” WHERE ”T1”.”C1”=”T2”.”C1”
SELECT * FROM ”T1” INNER JOIN ”T2” ON ”T1”.”C1”=”T2”.”C1”
```

[説明]

表の結合方式に，表T1 を外表，表T2 を内表とするハッシュジョインが適用される場合，チャンクおよびセグメントをスキップするために，レンジインデクスRIDX1 が使用されます。ハッシュジョインの詳細については，「5.5.2 ハッシュジョインとは」を参照してください。

(16) 例 16

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1” (”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” ”X”
    WHERE EXISTS(SELECT * FROM ”T2” WHERE ”X”.”C1”=”T2”.”C1”)
```

[説明]

外への参照列を含む副問合せの処理方式にハッシュ実行が適用される場合，チャンクおよびセグメントをスキップするために，レンジインデクスRIDX1 が使用されます。外への参照列を含む副問合せの処理方式のハッシュ実行の詳細については，「5.6.3 外への参照列を含む副問合せの処理方式とは」の「(3) ハッシュ実行」を参照してください。

(17) 例 17

レンジインデクスの定義

```
CREATE INDEX ”RIDX1” ON ”T1” (”C1”) IN ”DBAREA01” EMPTY INDEXTYPE RANGE
```

SQL 文の例

```
SELECT * FROM ”T1” WHERE ”C1” IN (SELECT ”T2”.”C1” FROM ”T2”)
SELECT * FROM ”T1” WHERE ”C1”=ANY(SELECT ”T2”.”C1” FROM ”T2”)
```

[説明]

外への参照列を含まない副問合せの処理方式にハッシュ実行が適用される場合，チャンクおよびセグメントをスキップするために，レンジインデクスRIDX1 が使用されます。外への参照列を含まない副問合せの処理方式のハッシュ実行の詳細については，「5.6.1 外への参照列を含まない副問合せの処理方式とは」の「(4) ハッシュ実行」を参照してください。

5.3.3　SQL 文の実行時に使用されるレンジインデクスを確認する方法

SQL 文の実行時に使用されるレンジインデクスは，アクセスパスで確認できます。アクセスパスについては，次に示す個所を参照してください。

- アクセスパスの確認方法
  「6.1.2　アクセスパスを確認するには」を参照してください。
- アクセスパスに表示される内容
  「6.1.5　詳細表示に出力される情報」の「(1)　表の検索方式，インデクス，および集まり導出表に関する情報」の「(b)　インデクスに関する情報」を参照してください。

5.4　インデクスを使用した探索条件の評価方式

インデクスを使用した探索条件の評価方式には，次の 2 つがあります。

- B-tree インデクスによる評価方式
- レンジインデクスによる評価方式

WHERE 句の探索条件，更新系 SQL のWHERE 探索条件，またはON 探索条件を，インデクスを使用して評価する際の評価方式について説明します。

5.4.1　B-tree インデクスによる評価方式

B-tree インデクスを使用して探索条件を評価する場合，サーチ条件とキー条件に従って評価されます。

(1)　サーチ条件とは

B-tree インデクスによる検索範囲を特定する条件をサーチ条件といいます。主に次の条件が該当します。

- =，不等号，IS NULL，LIKE 述語の前方一致指定（定数指定），LIKE 述語の？パラメタ指定，IN 述語※，BETWEEN 述語，限定述語（=ANY，=SOME）

  注※

  ネストループジョインの内表に指定されたIN 述語（表副問合せ指定ではない場合）は，サーチ条件にならないことがあります。

サーチ条件による評価方式の例を次の図に示します。

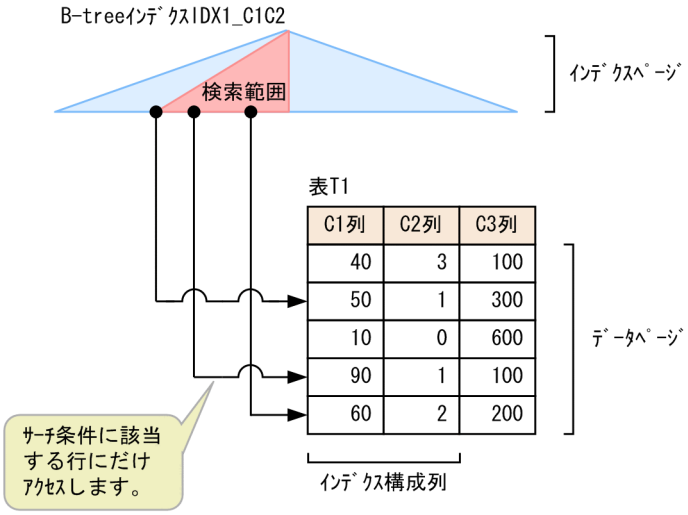
図 5-4　サーチ条件による評価方式の例

■表およびB-treeｲﾝﾃﾞｸｽの定義

```
CREATE TABLE "T1"
            ("C1" INTEGER,
             "C2" INTEGER,
             "C3" INTEGER) IN "DBAREA01"

CREATE INDEX "IDX_C1C2" ON "T1"("C1","C2")
              IN "DBAREA01" EMPTY
```

■実行したSQL文

```
SELECT * FROM "T1" WHERE "C1" BETWEEN 50 AND 100
```



[説明]

この場合，サーチ条件は 50≦C1≦100 になります。サーチ条件に該当する行のデータページにアクセスします。

(2)　キー条件とは

B-tree インデクスのインデクス構成列だけで評価できる条件をキー条件といいます。サーチ条件のように B-tree インデクスによる検索範囲を絞ることはできませんが，B-tree インデクスのインデクスページだけで条件を評価できるため，データページを参照する回数を減らすことができ，その分検索性能が向上します。

キー条件による評価方式の例を次の図に示します。

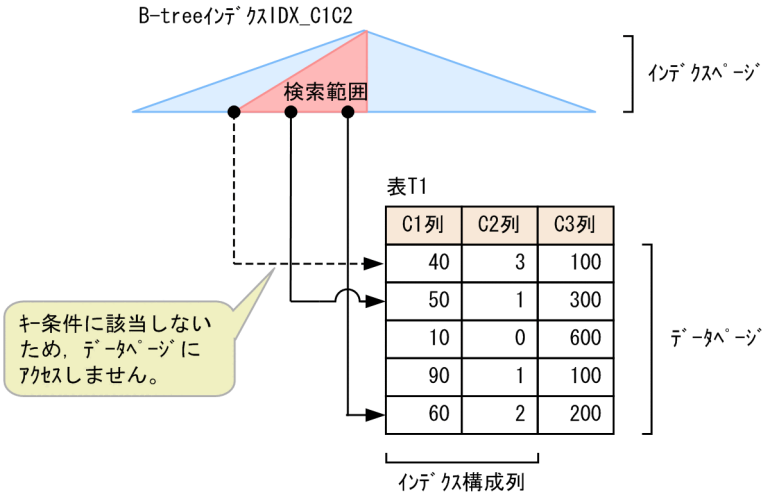
図 5-5　キー条件による評価方式の例

■表およびB-treeｲﾝﾃﾞｸｽの定義

```
CREATE TABLE "T1"
            ("C1" INTEGER,
             "C2" INTEGER,
             "C3" INTEGER) IN "DBAREA01"

CREATE INDEX "IDX_C1C2" ON "T1"("C1","C2")
              IN "DBAREA01" EMPTY
```

■実行したSQL文

```
SELECT * FROM "T1"
    WHERE "C1" BETWEEN 40 AND 60
      AND "C2" IN(1,2)
```



[説明]

この場合，各条件は次のようになります。

- サーチ条件：40≦C1≦60
- キー条件：C2 IN(1, 2)

サーチ条件で検索範囲を絞り込み，キー条件に該当する行のデータページにだけアクセスします。

5.4.2　レンジインデクスによる評価方式

レンジインデクスを使用して探索条件を評価する場合，次の 2 つのレンジインデクス条件に従って評価されます。

- チャンクのスキップにレンジインデクスを使用する条件

  レンジインデクスを使用して，探索条件に該当するデータを含まないチャンクをスキップするための条件です。アクセスするチャンクが減ることで，参照するデータページの量を減らすことができ，その分検索性能が向上します。

  この条件は，チャンクのスキップが使用できるレンジインデクスの場合だけ使用されます。

チャンクのスキップに使用できるレンジインデクスかどうかを確認する場合は，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『チャンクのスキップができるレンジインデクスかどうかを調べる場合』を参照してください。

- セグメントのスキップにレンジインデクスを使用する条件

  レンジインデクスを使用して，探索条件に該当するデータを含まないセグメントをスキップするための条件です。アクセスするセグメントが減ることで，参照するデータページの量を減らすことができ，その分検索性能が向上します。

> **メモ**
>
> チャンクのスキップ，およびセグメントのスキップについては，マニュアル『HADB　システム構築・運用ガイド』の『レンジインデクス』を参照してください。

(1)　レンジインデクスを使用したチャンクのスキップ例

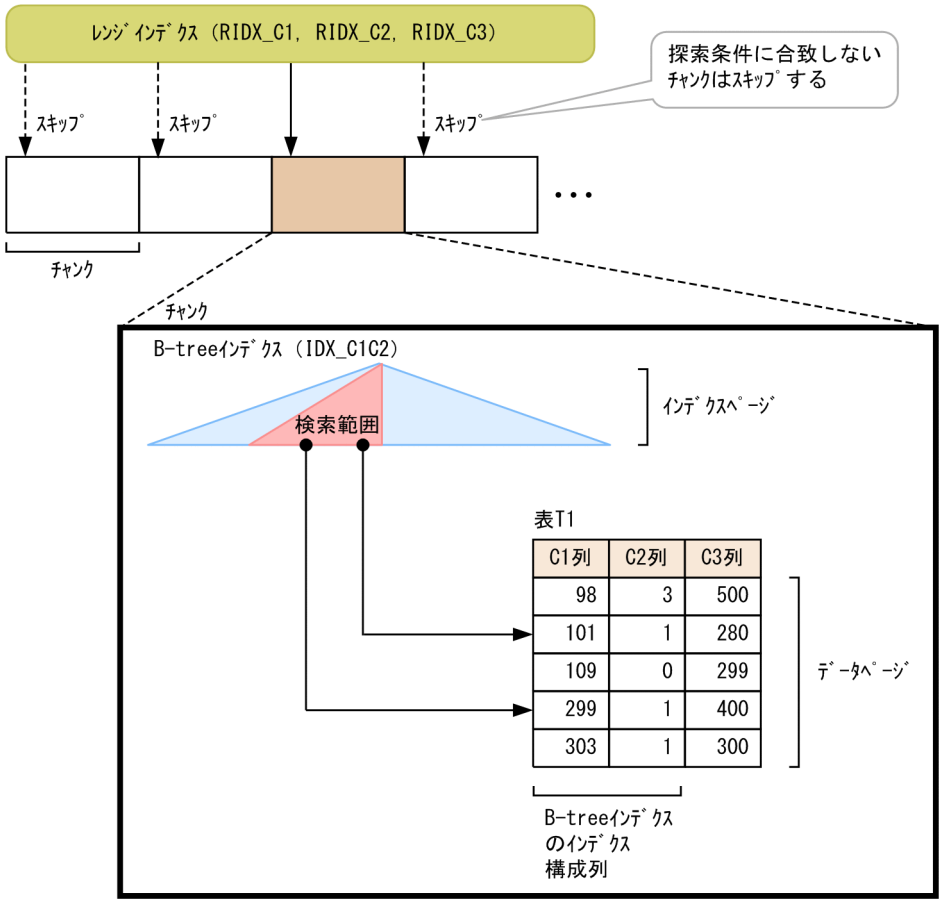
レンジインデクスを使用したチャンクのスキップ例を説明します。

■表，B-treeｲﾝﾃﾞｸｽおよびﾚﾝｼﾞｲﾝﾃﾞｸｽの定義

```
CREATE TABLE "T1"
            ("C1" INTEGER,
             "C2" INTEGER,
             "C3" INTEGER) IN "DBAREA01" CHUNK

CREATE INDEX "IDX_C1C2" ON "T1"("C1","C2")        <-- B-treeｲﾝﾃﾞｸｽの定義
              IN "DBAREA01" EMPTY

CREATE INDEX "RIDX_C1" ON "T1"("C1")              ┐
              IN "DBAREA01" EMPTY INDEXTYPE RANGE |
                                                  |
CREATE INDEX "RIDX_C2" ON "T1"("C2")              | ﾚﾝｼﾞｲﾝﾃﾞｸｽの定義
              IN "DBAREA01" EMPTY INDEXTYPE RANGE |
                                                  |
CREATE INDEX "RIDX_C3" ON "T1"("C3")              |
              IN "DBAREA01" EMPTY INDEXTYPE RANGE ┘
```

■実行したSQL文

```
SELECT * FROM "T1"
    WHERE "C1" BETWEEN 100 AND 300
      AND "C2" IN(1,2)
      AND "C3"<300
```



[説明]

検索処理の流れを次に示します。

1. チャンクにアクセスする前に，チャンクのスキップに使用するレンジインデクス条件を評価します。
   - チャンクのスキップに使用するレンジインデクス条件：100≦C1≦300，C2 IN(1，2)，C3<300
2. 検索対象となるチャンクの B-tree インデクスでサーチ条件とキー条件を評価します。
   - サーチ条件：100≦C1≦300
   - キー条件：C2 IN(1，2)
3. データページにアクセスして，残りの探索条件の評価を行います。
   - データページにアクセスしたあとに評価する探索条件：C3<300

なお，C1 列，C2 列，C3 列に定義されているレンジインデクスは，セグメントのスキップには使用しません。

(2)　レンジインデクスを使用したセグメントのスキップ例

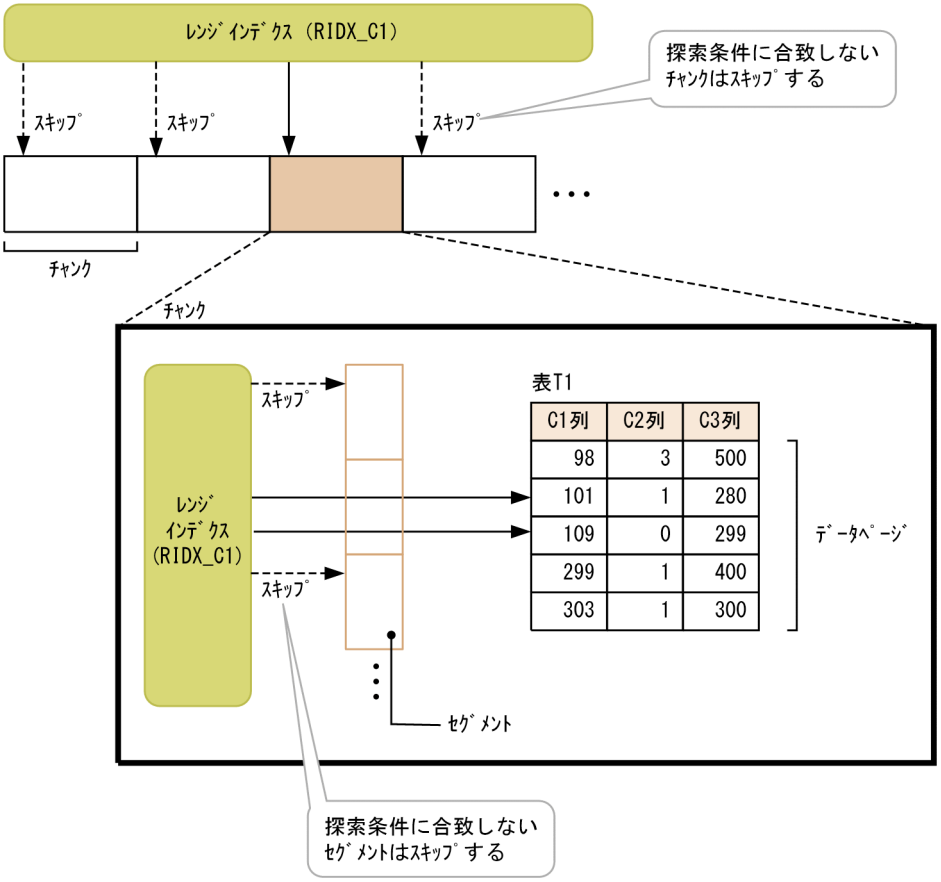
レンジインデクスを使用したセグメントのスキップ例を説明します。

■表およびﾚﾝｼﾞｲﾝﾃﾞｸｽの定義

```
CREATE TABLE "T1"
            ("C1" INTEGER,
             "C2" INTEGER,
             "C3" INTEGER) IN "DBAREA01" CHUNK

CREATE INDEX "RIDX_C1" ON "T1"("C1")                    ←― ﾚﾝｼﾞｲﾝﾃﾞｸｽの定義
              IN "DBAREA01" EMPTY INDEXTYPE RANGE
```

■実行したSQL文

```
SELECT * FROM "T1"
    WHERE "C1" BETWEEN 100 AND 110
      AND "C2" IN(1,2)
```



[説明]

検索処理の流れを次に示します。

1. チャンクにアクセスする前に，チャンクのスキップに使用するレンジインデクス条件を評価します。
   - チャンクのスキップに使用するレンジインデクス条件：100≦C1≦110
2. 検索対象となるチャンクのセグメントのスキップに使用するレンジインデクス条件を評価します。
   - セグメントのスキップに利用するレンジインデクス条件：100≦C1≦110
3. データページにアクセスして探索条件の評価を行います。
   - データページにアクセスしたあとに評価する探索条件：C2 IN(1,2)

5.5　表の結合方式

表の結合方式には次の 2 種類があります。

- ネストループジョイン
- ハッシュジョイン

この節では，各結合方式とその特徴について説明します。

表の結合方式は，HADB が自動的に決定します。SQL 文を実行した結果，どの結合方式が適用されたかを，アクセスパスで確認することができます。アクセスパスについては，次に示す個所を参照してください。

- アクセスパスの確認方法
  「6.1.2　アクセスパスを確認するには」を参照してください。
- アクセスパスに表示される内容
  「6.1.4　ツリー表示に出力される情報」の「(29)　表の結合方式」を参照してください。

> **メモ**
>
> 結合表の場合は，結合方式指定で表の結合方式を指定することができます。結合方式指定については，マニュアル『HADB　SQL リファレンス』の『結合方式指定の指定形式および規則』を参照してください。

5.5.1　ネストループジョインとは

外表の結合列の値を使用して，内表の結合列をサーチして突き合わせます。この処理を外表の行数分繰り返して表を結合します。この結合方式を**ネストループジョイン**といいます。

外表と内表を結合するための結合条件に指定した列に，インデクスが定義されている場合，結合条件の評価時にインデクスを使用して，内表の検索範囲を絞り込めることがあります。

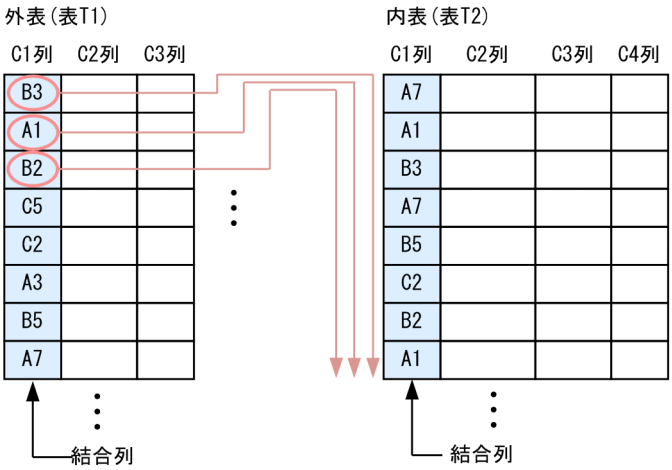
次のSELECT 文を実行した際にネストループジョインが適用された場合を例にして，ネストループジョインによる結合方式を説明します。

(例)

```
SELECT * FROM "T1","T2" WHERE "T1"."C2">10 AND "T1"."C1">"T2"."C1"
```

結合対象の表のうち，どちらを外表，内表にするかは，HADB が決定します。この例では，表T1 を外表，表T2 を内表として説明します。また，下線部分の結合条件に指定しているT1.C1 列とT2.C1 列が結合列となります。

図 5-6　ネストループジョインによる結合方式



[説明]

外表の結合列の値（B3）を取り出し，内表の結合列の値と突き合わせます。次に，外表の結合列の値（A1）を取り出し，内表の結合列の値と突き合わせます。この処理を外表の行数分繰り返します。

表T1 を外表，表T2 を内表とするネストループジョインの場合，結合条件に指定した列（"T1"."C1"および"T2"."C1"）にインデクスが定義されていると，結合条件（"T1"."C1">"T2"."C1"）の評価時にインデクスを使用して，内表の検索範囲を絞り込めることがあります。

メモ

結合表の場合は，結合方式指定によって外表を指定することができます。結合方式指定については，マニュアル『HADB　SQL リファレンス』の『結合方式指定の指定形式および規則』を参照してください。

5.5.2　ハッシュジョインとは

外表の結合列を基に作成したハッシュテーブルと，内表の結合列をハッシングした結果を突き合わせて表を結合します。この結合方式を**ハッシュジョイン**といいます。

外表と内表を結合するための結合条件に指定した列に，インデクスが定義されている場合でも，結合条件の評価時にインデクスは使用されません（ハッシュを使用して結合条件を評価します）。ただし，結合条件に指定した内表の列にレンジインデクスが定義されていて，かつレンジインデクスが使用される条件を満たしている場合は，そのレンジインデクスが使用されることがあります。

なお，コスト情報を取得している場合，ハッシュジョインをチャンク単位に分割実行することがあります。この分割実行では，マルチチャンク表に対してチャンクごとに外表のデータを分割し，分割したデータごとにハッシュジョインを繰り返し実行します。

(1) ハッシュジョインによる結合方式

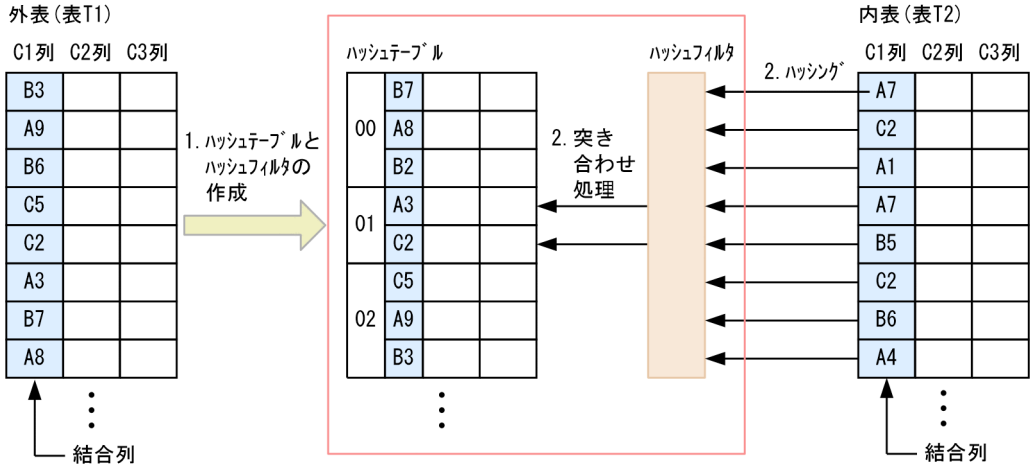
次のSELECT 文を実行した際にハッシュジョインが適用された場合を例にして，ハッシュジョインによる結合方式を説明します。

(例)

```
SELECT * FROM "T1","T2" WHERE "T1"."C2">10 AND "T1"."C1"="T2"."C1"
```

結合対象の表のうち，どちらを外表，内表にするかは，HADB が決定します。この例では，表T1 を外表，表T2 を内表として説明します。また，下線部分の結合条件に指定しているT1.C1 列とT2.C1 列が結合列となります。

図 5-7　ハッシュジョインによる結合方式



[説明]

1. 外表（表T1）の結合列の値を基に，ハッシュテーブルとハッシュフィルタを作成します。
2. 内表（表T2）の結合列の値をハッシングした結果と，ハッシュテーブルを突き合わせて，表の結合を行います。内表の結合列をハッシュテーブルと突き合わせる前に，ハッシュフィルタを使用して事前にフィルタリングします。これによって，内表の結合列とハッシュテーブルの突き合わせ回数を削減できます。

表T1 を外表，表T2 を内表とするハッシュジョインの場合，結合条件に指定した列（"T1"."C1"および"T2"."C1"）にインデクスが定義されている場合でも，結合条件（"T1"."C1"="T2"."C1"）の評価時にインデクスは使用されません。

"T1"."C2"にインデクスが定義されている場合は，「"T1"."C2">10」の評価時にインデクスが使用されることがあります。

また，次の 2 つの条件を満たす場合，ハッシュジョインの処理の際にレンジインデクスが使用されることがあります。

- ハッシュジョインの内表の結合列（上記の例の表T2 のC1 列）にレンジインデクスが定義されている
- レンジインデクスが使用される条件を満たしている

レンジインデクスが使用される条件については，「5.3.1　SQL 文の実行時にレンジインデクスが使用される条件」を参照してください。

上記の 2 つの条件を満たす場合，ハッシュジョインの処理で，外表の結合列からハッシュテーブルを作成するときに，外表の結合列の最大値と最小値を求めます。そして，内表を検索するときにレンジインデクスを使用して，先に求めた結合列の最大値と最小値の範囲外となる内表のチャンクまたはセグメントをスキップします。

ハッシュテーブルは，ハッシュテーブル領域に作成されます。ハッシュテーブル領域サイズは，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。

ハッシュフィルタは，ハッシュフィルタ領域に作成されます。ハッシュフィルタ領域サイズは，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。

(2)　ハッシュジョインが適用される例

(例 1)

```
SELECT * FROM "T1","T2" WHERE "T1"."C1"="T2"."C1"
```

＝指定の結合条件の両側に単独の列指定を指定した場合，ハッシュジョインが適用されます。

(例 2)

```
SELECT * FROM "T1","T2" WHERE "T1"."C1"="T2"."C1"+10
```

次の条件を満たす場合，ハッシュジョインが適用されます。

- ＝指定の結合条件の片側に単独の列指定を指定していて，もう片方には列指定を含むスカラ演算を指定している

(例 3)

```
SELECT * FROM "T1" INNER JOIN "T2"
    ON "T1"."C1"=CAST("T2"."C1" AS INTEGER)
```

次の条件を満たす場合，ハッシュジョインが適用されます。

- ＝指定の結合条件の片側に単独の列指定を指定していて，もう片方には列指定を含むスカラ関数を指定している

(例 4)

```
SELECT * FROM "T1" LEFT JOIN "T2"
    ON "T1"."C1"="T2"."C1"||"T2"."C2"
```

次の条件を満たす場合，ハッシュジョインが適用されます。

- ＝指定の結合条件の片側に単独の列指定を指定していて，もう片方には列指定を含む連結演算を指定している

(3) ハッシュジョインを適用する場合の留意事項

＝指定の結合条件の左右に指定した値式のデータ型とデータ長は，できる限り同じにしてください。＝指定の結合条件の左右に指定した値式のデータ型とデータ長が異なる場合，データ型とデータ長が同じ形式に変換されたあとにハッシュテーブルが作成されてハッシングが行われます。そのため，変換の分だけオーバヘッドが掛かります。

変換後のデータ型については，マニュアル『HADB　SQL リファレンス』の『変換，代入，比較できるデータ型』を参照してください。

なお，変換後のデータ型がDECIMAL 型の場合，精度と位取りは次の計算式を基に決定されます。

計算式

```
精度＝Pmax＋Smax
位取り＝Smax
Pmax＝MAX（p1-s1，p2-s2）
Smax＝MAX（s1，s2）
```

p1，*s1*：＝指定の結合条件の左側に指定した値式の精度と位取り

p2，*s2*：＝指定の結合条件の右側に指定した値式の精度と位取り

なお，変換前のデータ型がINTEGER 型の場合は，DECIMAL(20,0)として計算されます。変換前のデータ型がSMALLINT 型の場合は，DECIMAL(10,0)として計算されます。

（例）

変換後のデータ型がDECIMAL 型の場合の精度と位取りの決定例を次に示します。

表の定義

```
CREATE TABLE ”T1”(”C1” INTEGER,”C2” CHAR(3),”C3” DATE) IN ”DBAREA01”
CREATE TABLE ”T2”(”C1” DECIMAL(7,3),”C2” CHAR(3),”C3” DATE) IN ”DBAREA01”
```

SQL 文の例

```
SELECT * FROM ”T1”,”T2” WHERE ”T1”.”C1”=”T2”.”C1”
```

下線部の＝指定の結合条件の左側に指定された列（”T1”.”C1”）と，右側に指定された列（”T2”.”C1”）は，データ型とデータ長が異なります。そのため，データ型とデータ長が変換されます。”T1”.”C1”列はINTEGER 型，”T2”.”C1”列はDECIMAL 型です。

この場合，＝指定の結合条件の左右に指定された列のデータ型はDECIMAL 型に変換されます。”T1”.”C1”列はDECIMAL(20,0)と仮定されます。変換後のDECIMAL 型の精度と位取りは以下のように計算されます。

```
Pmax＝MAX（p1-s1，p2-s2）＝MAX（20-0，7-3）＝20
Smax＝MAX（s1，s2）＝MAX（0，3）＝3
精度＝Pmax＋Smax＝23
位取り＝Smax＝3
```

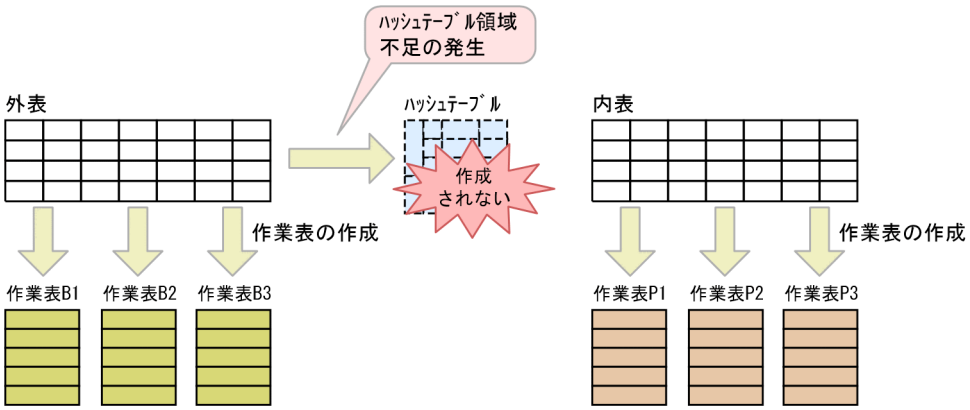
＝指定の結合条件の左右に指定された列のデータ型とデータ長をDECIMAL(23,3)に変換してハッシュテーブルが作成され，ハッシングが行われます。

(4) ハッシュテーブル領域が不足した場合の対処方法

■ハッシュテーブル領域が不足した場合の処理の流れ

ハッシュテーブルの作成中に，ハッシュテーブル領域が不足した場合の処理の流れを説明します。

1. ハッシュテーブルの作成中にハッシュテーブル領域が不足した場合，外表のデータを複数の作業表に分割して格納します。また，内表のデータも，外表のデータと同様に複数の作業表に分割して格納します。

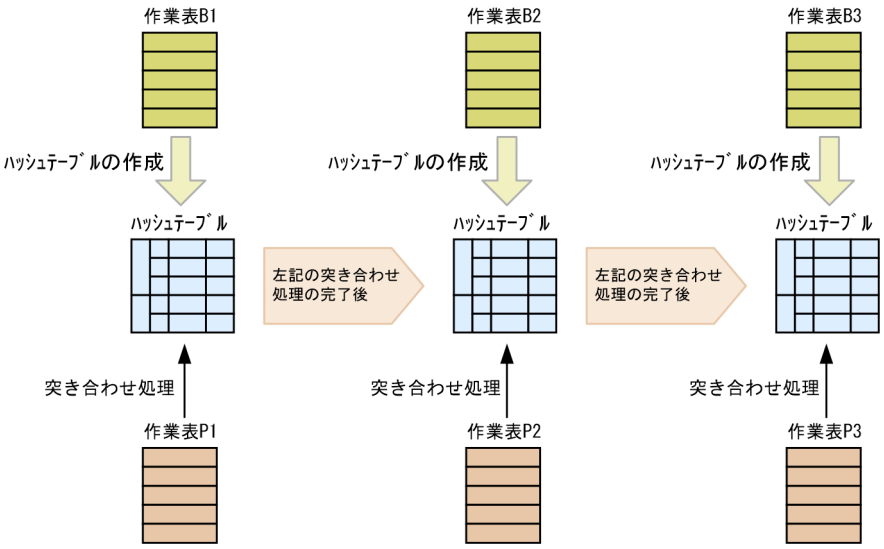


> **メモ**
>
> 上記の例では，外表の作業表が３つ，内表の作業表が３つ作成されていますが，SQL 文の指定内容などによっては作成される作業表の個数が変わります。

2. 外表の作業表（作業表B1）からハッシュテーブルを作成し，そのハッシュテーブルと内表の作業表（作業表P1）の突き合わせ処理を実行します。

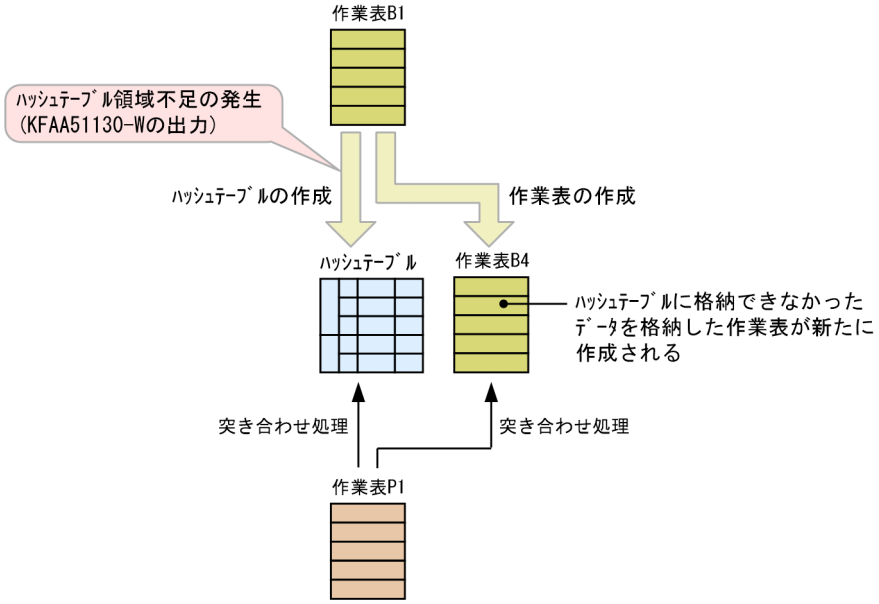
   ハッシュテーブルと作業表P1 の突き合わせ処理の完了後，作業表B2 からハッシュテーブルを作成し，そのハッシュテーブルと作業表P2 の突き合わせ処理を実行します。

   ハッシュテーブルと作業表P2 の突き合わせ処理の完了後，作業表B3 からハッシュテーブルを作成し，そのハッシュテーブルと作業表P3 の突き合わせ処理を実行します。



上記の 2.の処理中にハッシュテーブル領域が不足した場合

上記の 2.の処理中にハッシュテーブル領域が不足した場合，ハッシュテーブルに格納できなかったデータを別の作業表に格納します。この場合，内表の作業表とハッシュテーブルの突き合わせ処理のほかに，内表の作業表と新たに作成された作業表（作業表B4）との突き合わせ処理も発生します。



> **メモ**
>
> 上記の 2.の処理中にハッシュテーブル領域が不足して作業表（作業表B4）が新たに作成された場合，サーバメッセージログファイルにKFAA51130-W メッセージが出力されます。

■ハッシュテーブル領域が不足した場合の対処方法

ハッシュテーブル領域が不足した場合，作業表を作成する時間や，突き合わせ処理に掛かる時間によって，SQL 文の処理時間が長くなることがあります。ハッシュテーブル領域の不足を解消するには，サーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドの指定値（ハッシュテーブル領域サイズの指定値）を大きくしてください。

(5) ハッシュジョインが適用されない条件

次に示すどれかの条件に該当する場合は，結合方式にハッシュジョインが適用されることはありません。

- サーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドに0 が指定されている場合
- サーバ定義のadb_sys_uthd_num オペランドに0 が指定されている場合
- サーバ定義またはクライアント定義のadb_sql_exe_max_rthd_num オペランドに0 が指定されている場合
- 更新系 SQL に指定された問合せの場合
- 結合する表の間に，＝指定の結合条件がない場合
- 結合する表の間に，＝指定の結合条件があるが，次のどれかの条件に該当する場合

- ＝指定の結合条件の左右に列指定を含んだスカラ演算が指定されている
- ＝指定の結合条件の片側に列指定を含んだスカラ演算が指定され，もう片方には単独の列指定が指定され，かつ次のどれかの条件に該当する場合

  ・列指定を含んだスカラ演算中に，外への参照列を含む副問合せが指定されている

  ・列指定を含んだスカラ演算中に，＝指定の結合条件のもう片方に指定された単独の列指定と同じ表の列が指定されている
- ＝指定の結合条件の左右に指定された値式の変換後のデータ型がDECIMAL 型であり，精度が 38 より大きい
- 次のすべての条件を満たしている場合

  ・＝指定の結合条件の左右のどちらかに指定された値式のデータ型がVARCHAR 型であり，データ長が 32,000 バイトを超えている

  ・＝指定の結合条件の左右に指定された値式のデータ型またはデータ長が異なる
- ＝指定の結合条件の左右のどちらかに配列要素参照が指定されている
- 外への参照列を含む副問合せに指定されている

- 結合する表の間に，＝指定の結合条件が指定されているが，どちらかの表が表値構成子によって導出された導出表の場合
- 選択式にROW を指定した問合せの場合
- 結合方式指定で，結合方式にネストループジョインを指定している場合

> **メモ**
>
> 結合表の場合は，結合方式指定によって外表を指定することができます。結合方式指定については，マニュアル『HADB　SQL リファレンス』の『結合方式指定の指定形式および規則』を参照してください。

(6)　ハッシュフィルタが適用される条件

1. 次の条件をすべて満たす場合に，ハッシュジョインの際にハッシュフィルタが適用されます。
   - サーバ定義またはクライアント定義のadb_sql_exe_hashflt_area_size オペランドに0 が指定されていない
   - コンマ結合またはINNER JOIN の指定で表が結合されている
2. サーバ定義またはクライアント定義のadb_sql_exe_hashflt_area_size オペランドに指定したハッシュフィルタ領域サイズが小さい場合，ハッシュ検索ごとに割り当てられるハッシュフィルタ領域が不足することがあります。その結果，ハッシュフィルタのサイズが不足したハッシュ検索がある場合，そのハッシュ検索にハッシュフィルタは適用されません。すべてのハッシュ検索に対してハッシュフィルタを適用したい場合は，次の条件式を満たすようにadb_sql_exe_hashflt_area_size オペランドの指定値を変更してください。

adb_sql_exe_hashflt_area_sizeの指定値＞↑*A*×*B*×*SQL文の処理リアルスレッド数*÷1024↑

A：

ハッシュフィルタが適用されなかったハッシュ検索で使用するハッシュフィルタ数

ハッシュフィルタが適用されなかったハッシュ検索が 2 つ以上ある場合は，ハッシュ検索ごとに数を求め，その中の最大値を代入してください。1 つのハッシュ検索で使用するハッシュフィルタの数を次に示します。

- ハッシュジョインの場合：ハッシュジョインの＝結合条件数
- ハッシュ実行が適用される副問合せのうち，外への参照を含まない副問合せの場合：1
- ハッシュ実行が適用される副問合せのうち，外への参照を含む副問合せの場合：外への参照列を含む＝条件の数

B：

SQL 文中に指定した次の数の合計値

- ハッシュフィルタが適用されるハッシュジョインの数
- ハッシュフィルタを使用したハッシュ実行が適用される副問合せの数

5.5.3 各結合方式の特徴

各結合方式の特徴を次の表に示します。

表 5-5 表の結合方式の特徴

結合方式	初回のデータの取り出し速度	長所	短所
ネストループジョイン	速い	内表の結合列を B-tree インデクスまたはテキストインデクスで絞り込める場合は，高速に検索できます。	外表のヒット行数が多い場合は，処理性能が低下します。
ハッシュジョイン	遅い	外表のヒット行数が少なく，内表のヒット行数が多い場合は，高速に検索できます。	外表のヒット行数が多い場合は，使用するハッシュテーブル領域サイズが大きくなります。 ハッシュテーブル領域が不足した場合，いったん作業表に退避するため，処理性能が低下します。 ただし，外表がマルチチャンク表であり，かつチャンク単位のハッシュジョインの分割実行を HADB が適用可能と判断した場合は，作業表への退避処理が発生する可能性が低くなります（処理性能の低下する可能性が低くなります）。

5.6　副問合せの処理方式

副問合せの処理方式には次の 2 種類があります。

- 外への参照列を含まない副問合せの処理方式
- 外への参照列を含む副問合せの処理方式

この節では，各副問合せの処理方式とその特徴について説明します。

なお，SQL 文を実行した結果，どの処理方式が適用されたかを，アクセスパスで確認することができます。アクセスパスについては，次に示す個所を参照してください。

- アクセスパスの確認方法
  「6.1.2　アクセスパスを確認するには」を参照してください。
- アクセスパスに表示される内容
  「6.1.4　ツリー表示に出力される情報」の「(2)　副問合せの処理方式」を参照してください。

外への参照列については，マニュアル『HADB　SQL リファレンス』の『副問合せの指定形式および規則』を参照してください。

5.6.1　外への参照列を含まない副問合せの処理方式とは

外への参照列を含まない副問合せの処理方式には，次の 4 種類があります。

- 作業表実行
- 行値実行
- 作業表行値実行
- ハッシュ実行

各処理方式について説明します。

(1)　作業表実行

次に示す場合，作業表実行が適用されて副問合せの処理が実行されることがあります。

- 限定述語を指定した場合
- `IN` 述語中に表副問合せを指定した場合

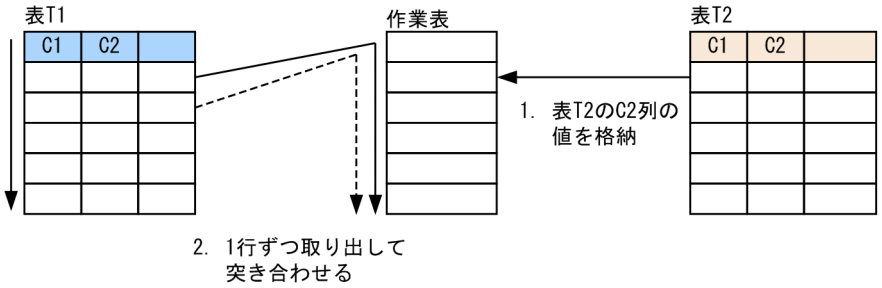
作業表実行の適用例を次に示します。

■実行するSELECT 文

```
SELECT ”T1”.”C1” FROM ”T1”
    WHERE ABS(”T1”.”C2”)=ANY(SELECT ”T2”.”C2” FROM ”T2”)
```

表T1 のC2 列には，B-tree インデクスおよびテキストインデクスが定義されていないものとします。

図 5-8　作業表実行の処理方式



[説明]

1. 副問合せの結果を作業表に格納します。

   この例の場合，副問合せ中に指定した表T2 を検索して，表T2 のC2 列の値を作業表に格納します。

2. 副問合せの外側の問合せが実行されます。このとき，副問合せの外側の問合せを 1 行検索するごとに副問合せの結果（作業表）と突き合わせて探索条件を評価します。

   この例の場合，表T1 から 1 行ずつ取り出し，表T1 のC2 列の絶対値と，作業表に格納された表T2 のC2 列の値を突き合わせて探索条件を評価します。

(2)　行値実行

次に示す場合，行値実行が適用されて副問合せの処理が実行されることがあります。

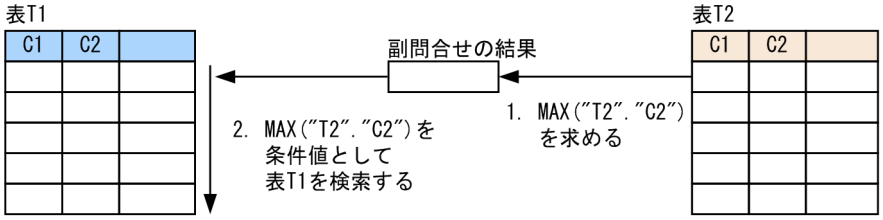
- スカラ副問合せを指定した場合
- EXISTS 述語を指定した場合

行値実行の適用例を次に示します。

■実行するSELECT 文

```
SELECT ”T1”.”C1” FROM ”T1”
    WHERE ”T1”.”C2”< (SELECT MAX(”T2”.”C2”) FROM ”T2”)
```

図 5-9　行値実行の処理方式



[説明]

1. 副問合せの結果を求めます。

   この例の場合，副問合せ中に指定した表T2 を検索して，MAX(”T2”.”C2”)を求めます。

2. 副問合せの結果を使用して，副問合せの外側の問合せの副問合せを含む条件を評価します。比較述語の場合は，副問合せの外側の問合せを実行する際に，B-tree インデクスまたはテキストインデクスを使用することがあります。

   この例の場合，1.で求めたMAX(”T2”.”C2”)を条件値として，表T1 を検索します。条件によっては，B-tree インデクスまたはテキストインデクスを使用して検索します。

(3) 作業表行値実行

次に示す場合，作業表行値実行が適用されて副問合せの処理が実行されることがあります。

- 限定述語を指定した場合
- IN 述語中に表副問合せを指定した場合

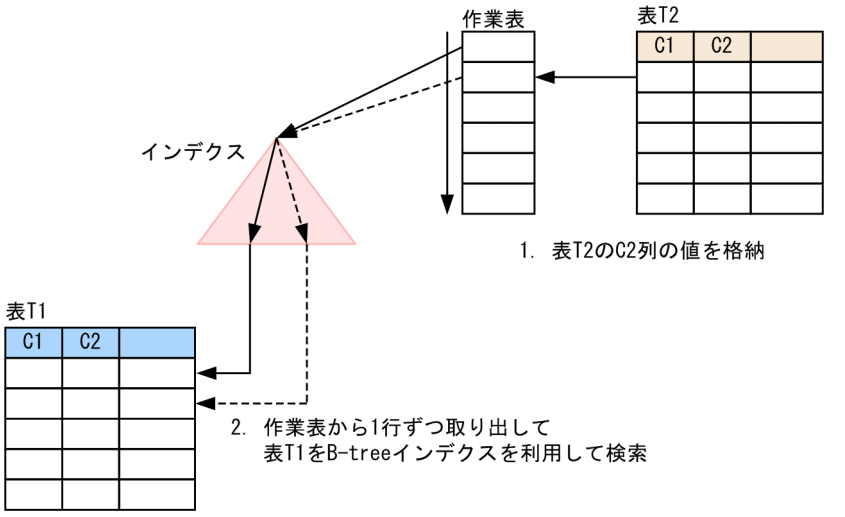
作業表行値実行の適用例を次に示します。

■実行するSELECT 文

```
SELECT ”T1”.”C1” FROM ”T1”
    WHERE ”T1”.”C2”=ANY(SELECT ”T2”.”C2” FROM ”T2”)
```

表T1 のC2 列には，B-tree インデクスが定義されているものとします。

図 5-10 作業表行値実行の処理方式



[説明]

1. 副問合せの結果を作業表に格納します。

   この例の場合，副問合せ中に指定した表T2 を検索して，表T2 のC2 列の値を作業表に格納します。

2. 作業表から行値を 1 行ずつ取り出して副問合せの外側の問合せを実行し，探索条件を評価します。このとき，B-tree インデクスを使用します。テキストインデクスが定義されている場合は，テキストインデクスを使用します。

   この例の場合，作業表から表T2 のC2 列の値を 1 行ずつ取り出して，表T1 のC2 列に定義した B-tree インデクスを使用して表T1 を検索します。

(4) ハッシュ実行

ハッシュテーブルを使用した副問合せの処理方式を**ハッシュ実行**といいます。次の場合に，ハッシュ実行が適用されることがあります。

- 限定述語を指定した場合
- IN 述語中に表副問合せを指定した場合

副問合せの処理方式にハッシュ実行が適用された場合，最初に，副問合せの結果を基にハッシュテーブルを作成します。次に，副問合せの外側の問合せを実行し，限定述語の左側に指定した列（またはIN 述語の左側に指定した列）の値からハッシュ値を生成します。最後に，ハッシュ値とハッシュテーブルの突き合わせ処理を行います。

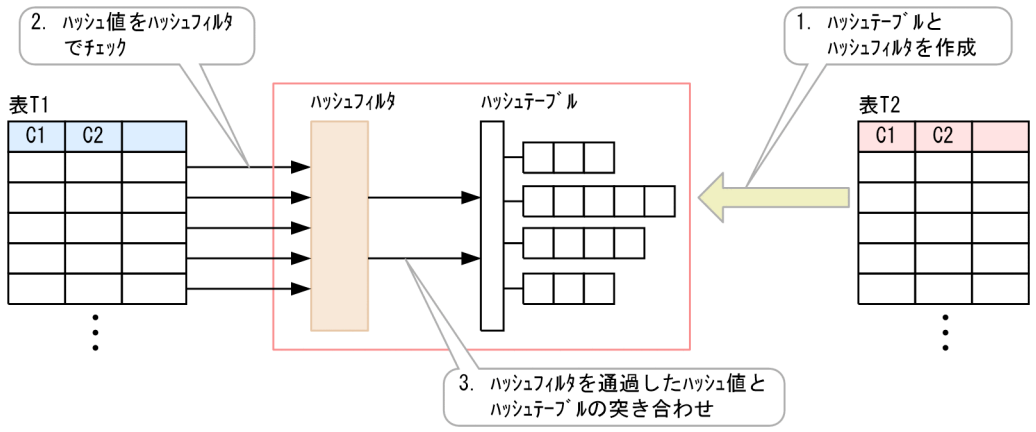
また，ハッシュテーブルを作成する際，同時にハッシュフィルタを作成します。ハッシュ値とハッシュテーブルの突き合わせ処理をする前に，ハッシュフィルタを使用してハッシュ値のフィルタリングを行います。これによって，ハッシュ値とハッシュテーブルの突き合わせ回数を削減することができます。

ハッシュ実行の適用例を次に示します。

■実行する SELECT 文

```
SELECT "T1"."C1" FROM "T1"
    WHERE "T1"."C2"=ANY(SELECT "T2"."C2" FROM "T2")
```

図 5-11 ハッシュ実行の処理方式



［説明］

1. 副問合せ（上記の SQL 文の例の下線部分）の結果を基に，ハッシュテーブルとハッシュフィルタを作成します。上記の例の場合，副問合せ中に指定した表T2 を検索し，表T2 のC2 列の値からハッシュテーブルとハッシュフィルタを作成します。
2. 副問合せの外側の問合せを実行し，限定述語の左側に指定した列（上記の SQL 文の例の場合は表T1のC2 列）の値からハッシュ値を生成します。そのハッシュ値をハッシュフィルタでチェックします。上記の例の場合，表T1 から 1 行ずつ取り出して，表T1 のC2 列の値からハッシュ値を生成し，ハッシュフィルタでそのハッシュ値をチェックします。
3. ハッシュフィルタを通過したハッシュ値と，ハッシュテーブルの突き合わせ処理を行います。

なお，次の 2 つの条件を満たす場合，ハッシュ実行の処理の際にレンジインデクスが使用されることがあります。

- 副問合せの外側の問合せに指定された表（上記の例の表T1）に対して，限定述語またはIN 述語の左側に指定した列（上記の例の列T1.C2）にレンジインデクスが定義されている
- レンジインデクスが使用される条件を満たしている

  レンジインデクスが使用される条件については，「5.3.1　SQL 文の実行時にレンジインデクスが使用される条件」を参照してください。

上記の 2 つの条件を満たす場合，ハッシュ実行の処理の際に，副問合せの結果を基にハッシュテーブルを作成するときに，その副問合せの結果の最大値と最小値を求めます。そして，副問合せの外側の問合せに指定された表を検索するときにレンジインデクスを使用して，先に求めた副問合せの結果の最大値と最小値の範囲外となる当該表（上記の例の表T1）のチャンクまたはセグメントをスキップします。

ハッシュテーブルはハッシュテーブル領域に作成されます。ハッシュテーブル領域サイズは，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。なお，adb_sql_exe_hashtbl_area_size オペランドに0 を指定した場合，ハッシュ実行は適用されません。

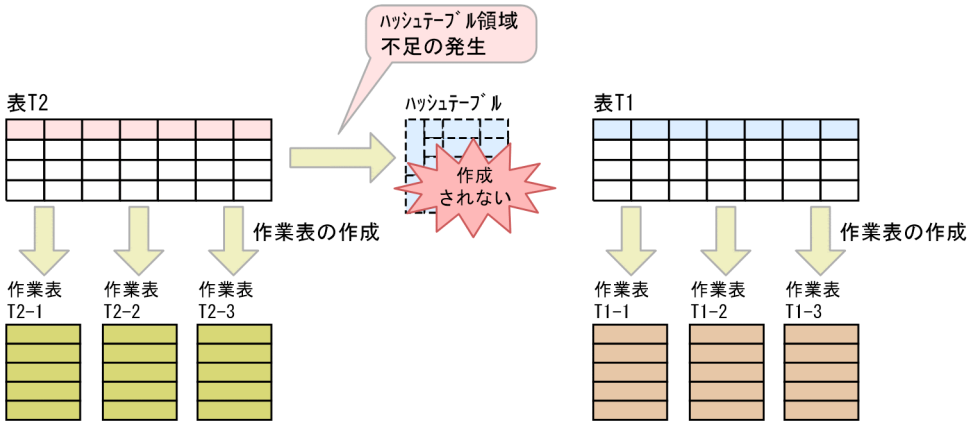
ハッシュフィルタは，ハッシュフィルタ領域に作成されます。ハッシュフィルタ領域サイズは，サーバ定義またはクライアント定義のadb_sql_exe_hashflt_area_size オペランドで指定します。なお，adb_sql_exe_hashflt_area_size オペランドに0 を指定した場合，ハッシュ実行の際にハッシュフィルタは適用されません。

■ハッシュテーブル領域が不足した場合の処理の流れ

ハッシュテーブルの作成中に，ハッシュテーブル領域が不足した場合の処理の流れを説明します。

なお，説明文中の表T1，T2 は，「図 5-11　ハッシュ実行の処理方式」の表T1，T2 と対応しています。

1. ハッシュテーブルの作成中にハッシュテーブル領域が不足した場合，表T2 のデータを複数の作業表に分割して格納します。また，表T1 のデータも，表T2 のデータと同様に複数の作業表に分割して格納します。

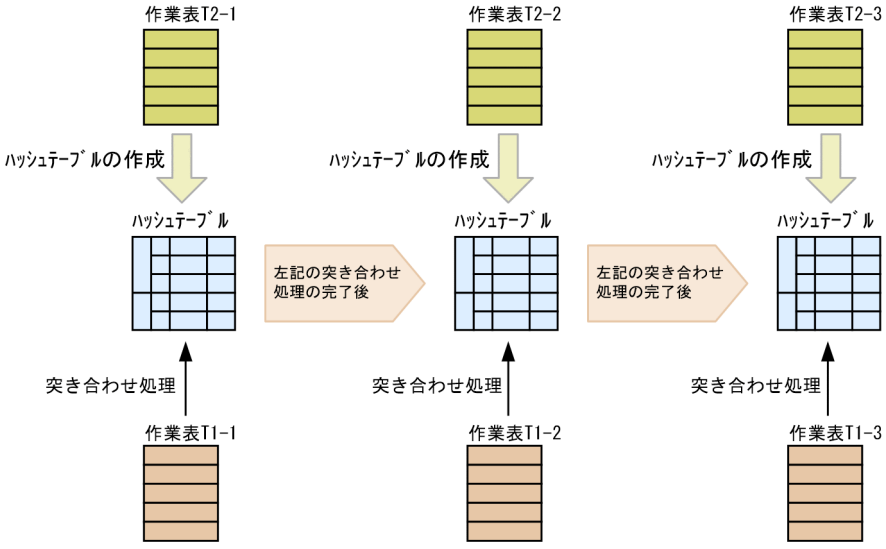


> **メモ**
>
> 上記の例では，表T2 の作業表が 3 つ，表T1 の作業表が 3 つ作成されていますが，SQL 文の指定内容などによっては作成される作業表の個数が変わります。

2. 表T2 の作業表（作業表T2-1）からハッシュテーブルを作成し，そのハッシュテーブルと表T1 の作業表（作業表T1-1）の突き合わせ処理を実行します。

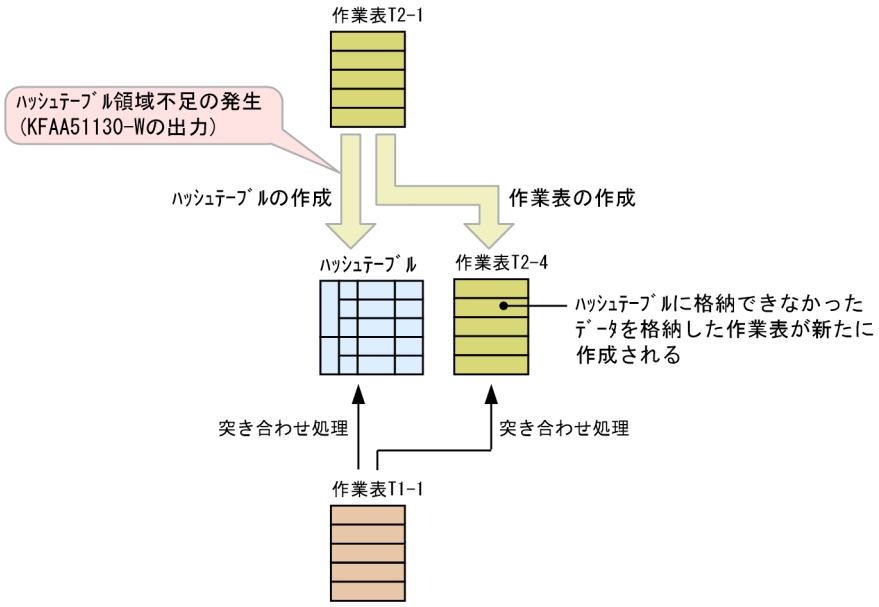
   ハッシュテーブルと作業表T1-1 の突き合わせ処理の完了後，作業表T2-2 からハッシュテーブルを作成し，そのハッシュテーブルと作業表T1-2 の突き合わせ処理を実行します。

   ハッシュテーブルと作業表T1-2 の突き合わせ処理の完了後，作業表T2-3 からハッシュテーブルを作成し，そのハッシュテーブルと作業表T1-3 の突き合わせ処理を実行します。



上記の 2.の処理中にハッシュテーブル領域が不足した場合

上記の 2.の処理中にハッシュテーブル領域が不足した場合，ハッシュテーブルに格納できなかったデータを別の作業表に格納します。この場合，表T1 の作業表とハッシュテーブルの突き合わせ処理のほかに，表T1 の作業表と新たに作成された作業表（作業表T2-4）との突き合わせ処理も発生します。



メモ

上記の 2.の処理中にハッシュテーブル領域が不足して作業表（作業表T2-4）が新たに作成された場合，サーバメッセージログファイルにKFAA51130-W メッセージが出力されます。

■ハッシュテーブル領域が不足した場合の対処方法

ハッシュテーブル領域が不足した場合，作業表を作成する時間や，突き合わせ処理に掛かる時間によって，SQL 文の処理時間が長くなることがあります。ハッシュテーブル領域の不足を解消するには，サーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドの指定値（ハッシュテーブル領域サイズの指定値）を大きくしてください。

■ハッシュフィルタ領域が不足した場合の対処方法

サーバ定義またはクライアント定義のadb_sql_exe_hashflt_area_size オペランドに指定したハッシュフィルタ領域サイズが小さい場合，ハッシュ検索ごとに割り当てられるハッシュフィルタ領域が不足することがあります。その結果，ハッシュフィルタのサイズが不足したハッシュ検索がある場合，そのハッシュ検索にハッシュフィルタは適用されません。すべてのハッシュ検索に対してハッシュフィルタを適用したい場合は，adb_sql_exe_hashflt_area_size オペランドの指定値を大きくしてください。指定値の見積もり式については，「5.5.2　ハッシュジョインとは」の「(6)　ハッシュフィルタが適用される条件」を参照してください。

5.6.2　外への参照列を含まない副問合せの各処理方式の特徴

外への参照列を含まない副問合せの，各処理方式の特徴を次の表に示します。

表 5-6　外への参照列を含まない副問合せの各処理方式

項番	処理方式	長所	短所
1	作業表実行	作業表を必要とするすべての副問合せの条件に対して適用できます。	副問合せの外側の問合せの件数が多い場合は，処理性能が低下します。
2	行値実行	副問合せの外側の問合せに対して，B-tree インデクスまたはテキストインデクスを使用できます。したがって，副問合せの外側の問合せの件数が多い場合，B-tree インデクスまたはテキストインデクスを使用して検索範囲を絞り込めるときは高速に検索できます。	副問合せの外側の問合せの件数が多く，B-tree インデクスまたはテキストインデクスを使用して副問合せを含む述語を絞り込めない場合は，処理性能が低下します。
3	作業表行値実行	副問合せの外側の問合せに対して，B-tree インデクスまたはテキストインデクスを使用できます。したがって，副問合せのヒット件数が少なく，かつ副問合せの外側の問合せの件数が多い場合に，B-tree インデクスまたはテキストインデクスを使用して検索範囲を絞り込めるときは高速に検索できます。	副問合せの結果の行数分，副問合せの外側の問合せに対して，B-tree インデクスまたはテキストインデクスを使用して検索するため，副問合せのヒット件数が多い場合は処理性能が低下します。
4	ハッシュ実行	処理を実行するために必要なすべてのデータがハッシュテーブル領域に格納できる場合は，高速に検索できます。	ハッシュテーブルに格納するデータが多い場合は，使用するハッシュテーブル領域サイズが大きくなります。ハッシュテーブル領域が不足した場合は，いったん作業表にすべてのデータを退避するため，処理性能が低下します。

5.6.3　外への参照列を含む副問合せの処理方式とは

外への参照列を含む副問合せの処理方式には，次の 3 種類があります。

- ネストループ作業表実行
- ネストループ行値実行
- ハッシュ実行

ネストループ行値実行の場合，副問合せの実行回数を減らすために，副問合せの結果を格納するキャッシュを作成することがあります。

各処理方式について説明します。

(1)　ネストループ作業表実行

次に示す場合，ネストループ作業表実行が適用されて副問合せの処理が実行されることがあります。

- 限定述語を指定した場合

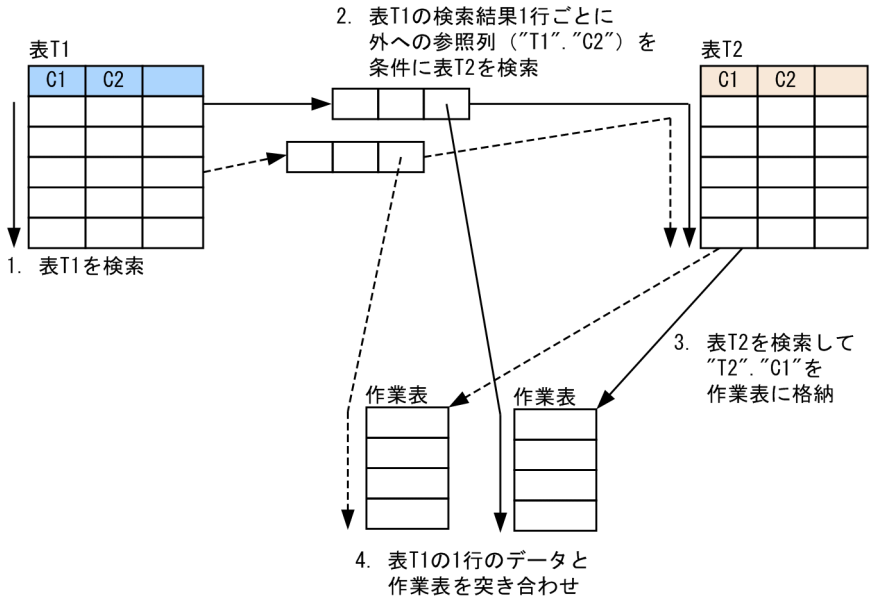
- IN 述語中に表副問合せを指定した場合

ネストループ作業表実行の適用例を次に示します。

■実行するSELECT 文

```
SELECT "T1"."C1" FROM "T1"
    WHERE "T1"."C1"=ANY(SELECT "T2"."C1" FROM "T2"
                              WHERE "T2"."C2"="T1"."C2")
```

図 5-12　ネストループ作業表実行の処理方式



[説明]

1. 副問合せの外側の問合せを実行します。

   この例の場合，表T1 を検索します。

2. 副問合せの外側の問合せを 1 行取り出すごとに，外への参照列の値を使用して副問合せを実行します。

   この例の場合，表T1 の検索結果 1 行ごとに，外への参照列（"T1"."C2"）の値を条件値に使用して表T2 を検索します。

3. 実行した副問合せの結果を基に作業表を作成します。

   この例の場合，表T2 を検索して"T2"."C1"の値を作業表に格納します。

4. 作成した作業表を使用して，副問合せの外側の副問合せを含む条件を評価します。

   この例の場合，表T1 の検索結果 1 行ごとに，対応する作業表の"T2"."C1"の値と突き合わせて副問合せを含む条件を評価します。

(2)　ネストループ行値実行

次に示す場合，ネストループ行値実行が適用されて副問合せの処理が実行されることがあります。

- スカラ副問合せを指定した場合

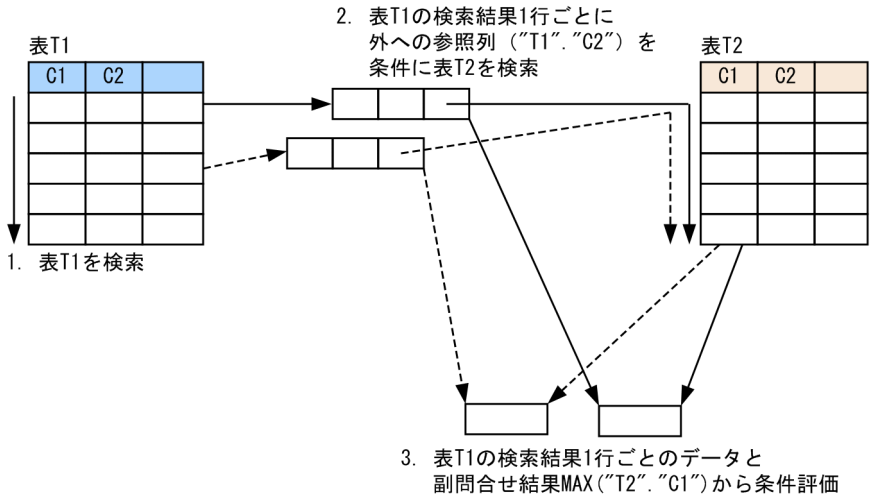
- EXISTS 述語を指定した場合

ネストループ行値実行の適用例を次に示します。

■実行するSELECT 文

```
SELECT "T1"."C1" FROM "T1"
    WHERE "T1"."C1"=(SELECT MAX("T2"."C1") FROM "T2"
                          WHERE "T2"."C2"="T1"."C2")
```

図 5-13　ネストループ行値実行の処理方式



[説明]

1. 副問合せの外側の問合せを実行します。

   この例の場合，表T1 を検索します。

2. 副問合せの外側の問合せを 1 行取り出すごとに，外への参照列の値を使用して副問合せを実行します。

   この例の場合，表T1 の検索結果 1 行ごとに外への参照列（"T1"."C2"）の値を使用して，副問合せの結果MAX("T2"."C1")を求めます。

3. 実行した副問合せの結果を求めます（作業表は作成しません）。副問合せの結果を使用して，副問合せの外側の副問合せを含む条件を評価します。

   この例の場合，表T1 の検索結果 1 行ごとに，対応するMAX("T2"."C1")の値を使用して条件を評価します。

(3)　ハッシュ実行

ハッシュテーブルを使用した副問合せの処理方式を**ハッシュ実行**といいます。次の場合に，ハッシュ実行が適用されることがあります。

- EXISTS 述語を指定した場合
- スカラ副問合せを指定した場合

副問合せの処理方式にハッシュ実行が適用された場合，最初に，外への参照列を含む条件を除外して副問合せを実行し，その結果からハッシュテーブルを作成します。次に，副問合せの外側の問合せを実行し，外への参照列の値からハッシュ値を生成します。最後に，ハッシュ値とハッシュテーブルの突き合わせ処理を行います。

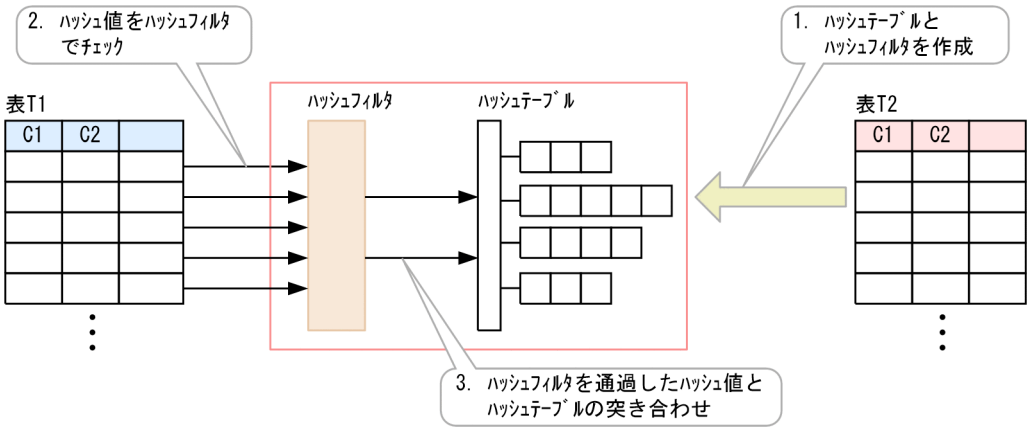
また，ハッシュテーブルを作成する際，同時にハッシュフィルタを作成します。ハッシュ値とハッシュテーブルの突き合わせ処理をする前に，ハッシュフィルタを使用してハッシュ値のフィルタリングを行います。これによって，ハッシュ値とハッシュテーブルの突き合わせ回数を削減することができます。

ハッシュ実行の適用例を次に示します。

■実行するSELECT 文

```
SELECT "T1"."C1" FROM "T1"
   WHERE "T1"."C3"<(SELECT "T2"."C3" FROM "T2"
                        WHERE "T2"."C1"='A' AND "T2"."C2"="T1"."C2")
```

図 5-14　ハッシュ実行の処理方式



[説明]

1. 外への参照列を含む条件（上記の SQL 文の例の下線部分）を除外して副問合せを実行し，その結果を基にハッシュテーブルとハッシュフィルタを作成します。
2. 副問合せの外側の問合せを実行し，外への参照列（"T1"."C2"）の値からハッシュ値を生成します。そのハッシュ値をハッシュフィルタでチェックします。上記の例の場合，表T1 を検索し，外への参照列（"T1"."C2"）からハッシュ値を生成し，ハッシュフィルタでそのハッシュ値をチェックします。
3. ハッシュフィルタを通過したハッシュ値と，ハッシュテーブルの突き合わせ処理を行います。

なお，次の 2 つの条件を満たす場合，ハッシュ実行の処理の際にレンジインデクスが使用されることがあります。

- 副問合せの外側の問合せに指定された表（上記の例の表T1）に対して，外への参照列（上記の例の列T1.C2）にレンジインデクスが定義されている
- レンジインデクスが使用される条件を満たしている

レンジインデクスが使用される条件については，「5.3.1　SQL 文の実行時にレンジインデクスが使用される条件」を参照してください。

上記の 2 つの条件を満たす場合，ハッシュ実行の処理の際に，外への参照列を含む条件を除外して副問合せを実行した結果からハッシュテーブルを作成するときに，外への参照列と比較する列の最大値と最小値を求めます。そして，副問合せの外側の問合せに指定された表を検索するときにレンジインデクスを使用して，先に求めた外への参照列と比較する列の最大値と最小値の範囲外となる当該表（上記の例の表T1）のチャンクまたはセグメントをスキップします。

ハッシュテーブルはハッシュテーブル領域に作成されます。ハッシュテーブル領域サイズは，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。なお，adb_sql_exe_hashtbl_area_size オペランドに0 を指定した場合，ハッシュ実行は適用されません。

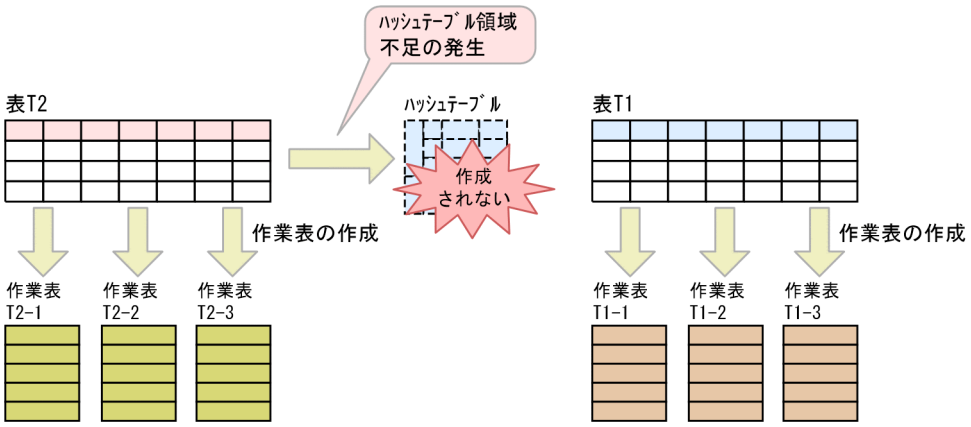
ハッシュフィルタは，ハッシュフィルタ領域に作成されます。ハッシュフィルタ領域サイズは，サーバ定義またはクライアント定義のadb_sql_exe_hashflt_area_size オペランドで指定します。なお，adb_sql_exe_hashflt_area_size オペランドに0 を指定した場合，ハッシュ実行の際にハッシュフィルタは適用されません。

■ハッシュテーブル領域が不足した場合の処理の流れ

ハッシュテーブルの作成中に，ハッシュテーブル領域が不足した場合の処理の流れを説明します。

なお，説明文中の表T1，T2 は，「図 5-14　ハッシュ実行の処理方式」の表T1，T2 と対応しています。

1. ハッシュテーブルの作成中にハッシュテーブル領域が不足した場合，表T2 のデータを複数の作業表に分割して格納します。また，表T1 のデータも，表T2 のデータと同様に複数の作業表に分割して格納します。

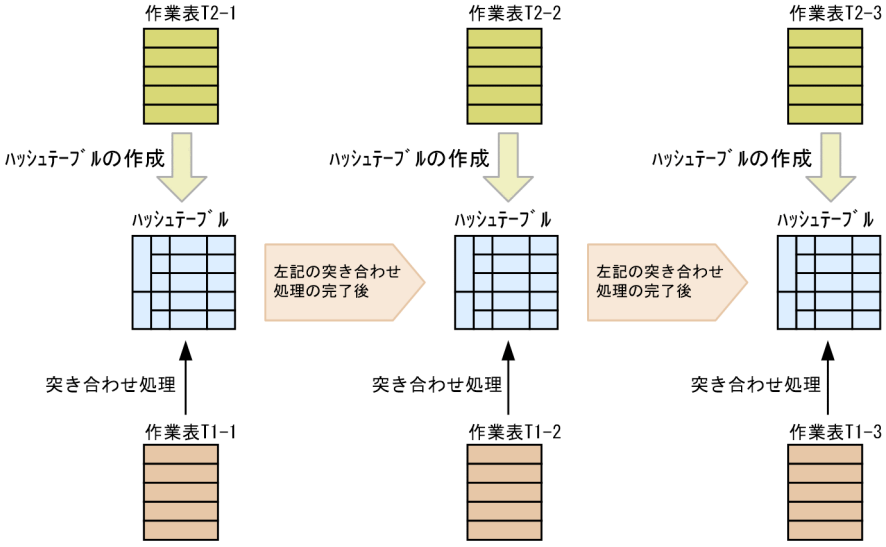


> **メモ**
>
> 上記の例では，表T2 の作業表が 3 つ，表T1 の作業表が 3 つ作成されていますが，SQL 文の指定内容などによっては作成される作業表の個数が変わります。

2. 表T2 の作業表（作業表T2-1）からハッシュテーブルを作成し，そのハッシュテーブルと表T1 の作業表（作業表T1-1）の突き合わせ処理を実行します。

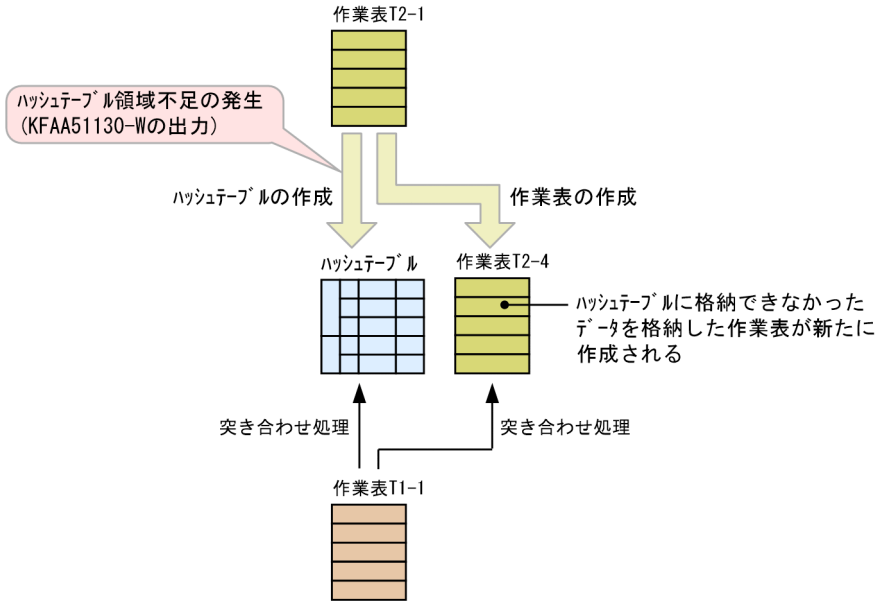
   ハッシュテーブルと作業表T1-1 の突き合わせ処理の完了後，作業表T2-2 からハッシュテーブルを作成し，そのハッシュテーブルと作業表T1-2 の突き合わせ処理を実行します。

ハッシュテーブルと作業表T1-2 の突き合わせ処理の完了後，作業表T2-3 からハッシュテーブルを作成し，そのハッシュテーブルと作業表T1-3 の突き合わせ処理を実行します。



上記の 2.の処理中にハッシュテーブル領域が不足した場合

上記の 2.の処理中にハッシュテーブル領域が不足した場合，ハッシュテーブルに格納できなかったデータを別の作業表に格納します。この場合，表T1 の作業表とハッシュテーブルの突き合わせ処理のほかに，表T1 の作業表と新たに作成された作業表（作業表T2-4）との突き合わせ処理も発生します。



> **メモ**
>
> 上記の 2.の処理中にハッシュテーブル領域が不足して作業表（作業表T2-4）が新たに作成された場合，サーバメッセージログファイルにKFAA51130-W メッセージが出力されます。

■ハッシュテーブル領域が不足した場合の対処方法

ハッシュテーブル領域が不足した場合，作業表を作成する時間や，突き合わせ処理に掛かる時間によって，SQL 文の処理時間が長くなることがあります。ハッシュテーブル領域の不足を解消するには，サーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドの指定値（ハッシュテーブル領域サイズの指定値）を大きくしてください。

■ハッシュフィルタ領域が不足した場合の対処方法

サーバ定義またはクライアント定義のadb_sql_exe_hashflt_area_size オペランドに指定したハッシュフィルタ領域サイズが小さい場合，ハッシュ検索ごとに割り当てられるハッシュフィルタ領域が不足することがあります。その結果，ハッシュフィルタのサイズが不足したハッシュ検索がある場合，そのハッシュ検索にハッシュフィルタは適用されません。すべてのハッシュ検索に対してハッシュフィルタを適用したい場合は，adb_sql_exe_hashflt_area_size オペランドの指定値を大きくしてください。指定値の見積もり式については，「5.5.2　ハッシュジョインとは」の「(6)　ハッシュフィルタが適用される条件」を参照してください。

5.6.4　外への参照列を含む副問合せの各処理方式の特徴

外への参照列を含む副問合せの，各処理方式の特徴を次の表に示します。

表 5-7　外への参照列を含む副問合せの各処理方式の特徴

項番	処理方式	長所	短所
1	ネストループ作業表実行	副問合せの探索条件のうち，外への参照列を含む条件に対して，B-tree インデクスまたはテキストインデクスを使用できます。したがって，副問合せの探索条件が，B-tree インデクスまたはテキストインデクスを使用して検索範囲を絞り込める場合は，高速に検索できます。	副問合せの外側の問合せのヒット件数が多い場合は，処理性能が低下します。
2	ネストループ行値実行		
3	ハッシュ実行	処理を実行するために必要なすべてのデータがハッシュテーブル領域に格納できる場合は，高速に検索できます。	ハッシュテーブルに格納するデータが多い場合は，使用するハッシュテーブル領域サイズが大きくなります。ハッシュテーブル領域が不足した場合は，いったん作業表にすべてのデータを退避するため，処理性能が低下します。

5.7　グループ化の処理方式

グループ化の処理方式には次の 2 種類があります。

- ハッシュグループ化
- ソートグループ化

GROUP BY 句またはDISTINCT 集合関数を指定した場合，グループ化の処理が実行されます。ここでは，各処理方式について説明します。

グループ化の処理方式は，HADB が自動的に決定します。SQL 文を実行した結果，どの処理方式が適用されたかを，アクセスパスで確認することができます。アクセスパスについては，次に示す個所を参照してください。

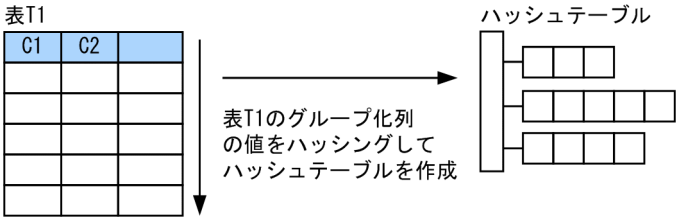
- アクセスパスの確認方法
  「6.1.2　アクセスパスを確認するには」を参照してください。
- アクセスパスに表示される内容
  「6.1.4　ツリー表示に出力される情報」の「(14)　グループ化の処理方式」を参照してください。

> **メモ**
>
> グループ化方式指定を指定することによって，グローバルハッシュグループ化の適用を抑止することができます。グループ化方式指定については，マニュアル『HADB　SQL リファレンス』の『GROUP BY 句の指定形式および規則』を参照してください。

5.7.1　ハッシュグループ化とは

ハッシュグループ化とは，グループ化列の値をハッシングしてハッシュテーブルを作成しながらグループ化する処理方式のことです。ハッシュグループ化の処理方式を次の図に示します。

図 5-15　ハッシュグループ化の処理方式



ハッシュグループ化には次の 2 種類があります。

(1) ローカルハッシュグループ化

最初に SQL 処理リアルスレッドごとにハッシュテーブルを作成し，グループ化します。次に，各 SQL 処理リアルスレッドでグループ化した結果を集めて，全体をグループ化します。この処理方式をローカルハッシュグループ化といいます。

ハッシュテーブルは，SQL 処理リアルスレッドごとにハッシュグループ化領域に作成されます。ハッシュテーブル 1 つ当たりのハッシュグループ化領域サイズは，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。

(2) グローバルハッシュグループ化

SQL 処理リアルスレッド間で共有するハッシュテーブルを作成し，グループ化します。この処理方式をグローバルハッシュグループ化といいます。

ハッシュテーブルは，ハッシュテーブル領域に作成されます。ハッシュテーブル領域サイズは，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。なお，adb_sql_exe_hashtbl_area_size オペランドに0 を指定した場合，グローバルハッシュグループ化は適用されません。

DISTINCT 集合関数を含む SQL 文を実行した場合に，検索結果の重複を排除するためにグローバルハッシュグループ化が適用されることがあります。

■ハッシュテーブル領域が不足した場合の対処方法

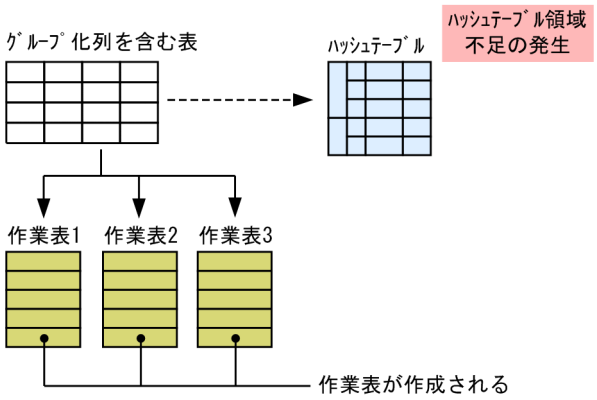
ハッシュテーブル領域が不足した場合，ハッシュテーブルに格納されるデータが複数の作業表に分割されて格納されます。そのため，SQL 文の処理時間が長くなることがあります。ハッシュテーブル領域の不足を解消するには，サーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドの指定値（ハッシュテーブル領域サイズの指定値）を大きくしてください。

ハッシュテーブル領域が不足して作業表が作成された場合，サーバメッセージログファイルにKFAA51130-W メッセージが出力されます。

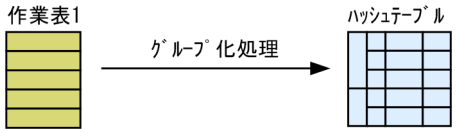
■ハッシュテーブル領域が不足した場合の処理の流れ

ハッシュテーブルの作成中に，ハッシュテーブル領域が不足した場合の処理の流れを説明します。

1. ハッシュテーブルの作成中にハッシュテーブル領域が不足した場合，複数の作業表が作成されます。
   ハッシュテーブルに格納されるデータが，各作業表に分割されて格納されます。

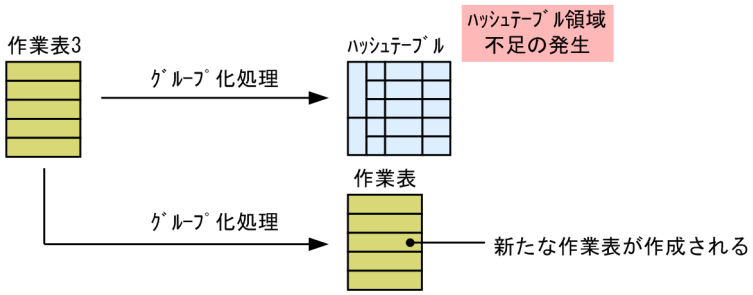


2. 作業表 1 に対するハッシュテーブルを作成しながらグループ化処理が行われます。



上記のグループ化処理が完了したあとに，作業表 2 に対するハッシュテーブルを作成しながらグループ化処理が行われます。そのあとに，作業表 3 についても同様の処理が行われます。

なお，作業表に対するハッシュテーブルが作成される際に，ハッシュテーブル領域不足が発生した場合，新たな作業表が作成されます。ハッシュテーブル領域に格納できなかったデータが，その作業表に格納されます。この場合，新たに作成された作業表を使用したグループ化処理が追加で発生します。



5.7.2 ソートグループ化とは

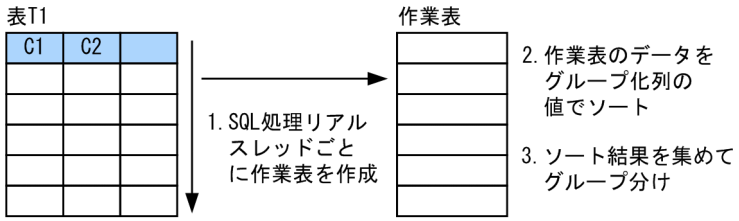
ソートグループ化とは，ソートしたあとにグループ化する処理方式のことです。

最初に SQL 処理リアルスレッドごとに作業表を作成します。次に，作業表のデータをグループ化列の値でソートし，最後に各 SQL 処理リアルスレッドのソートされたデータを集めて，グループ化します。

DISTINCT 集合関数を含む SQL 文を実行した場合に，検索結果の重複を排除するためにソートグループ化が適用されることがあります。

ソートグループ化の処理方式を次の図に示します。

図 5-16　ソートグループ化の処理方式



5.7.3　各グループ化の特徴

各グループ化の特徴を次の表に示します。

表 5-8　各グループ化の特徴

項番	グループ化の種類		長所	短所
1	ハッシュグループ化	ローカルハッシュグループ化	グループの数が少ない場合など，グループ化に必要なデータが SQL 処理リアルスレッドごとのハッシュグループ化領域に収まるときは，高速にグループ化されます。	グループの数が多い場合は，処理性能が低下します。
2		グローバルハッシュグループ化	グループ化に必要なデータがハッシュテーブル領域に収まる場合に，高速にグループ化されます。	グループ化に必要なデータ量が多く，ハッシュテーブル領域が不足した場合は，いったんデータを作業表に格納するため，処理性能が低下します。
3	ソートグループ化		ハッシュグループ化領域，およびハッシュテーブル領域を確保しない場合でも，グループ化できます。	検索結果の件数が多い場合は，作業表に格納されるデータ量が多くなります。作業表に格納したデータをソートしたあとにグループ化するため，処理性能が低下します。

5.8　集合演算の処理方式

集合演算（UNION ALL を除く）を指定した場合，次のどちらかの処理方式で集合演算が実行されます。

- ハッシュ実行
- 作業表実行

上記のうち，どちらの処理方式を適用するかは HADB が自動的に決定します。適用される処理方式については，アクセスパスで確認することができます。アクセスパスについては，次に示す個所を参照してください。

- アクセスパスの確認方法
  「6.1.2　アクセスパスを確認するには」を参照してください。
- アクセスパスに表示される内容
  「6.1.4　ツリー表示に出力される情報」の「(13)　重複排除の処理方式」を参照してください。

メモ

- 集合演算方式指定を指定することによって，ハッシュ実行の適用を抑止することができます。集合演算方式指定については，マニュアル『HADB　SQL リファレンス』の『問合せ式の指定形式および規則』を参照してください。
- UNION ALL を指定した場合，集合演算項に指定した問合せ式本体のデータがそのまま返却されます。

5.8.1　ハッシュ実行

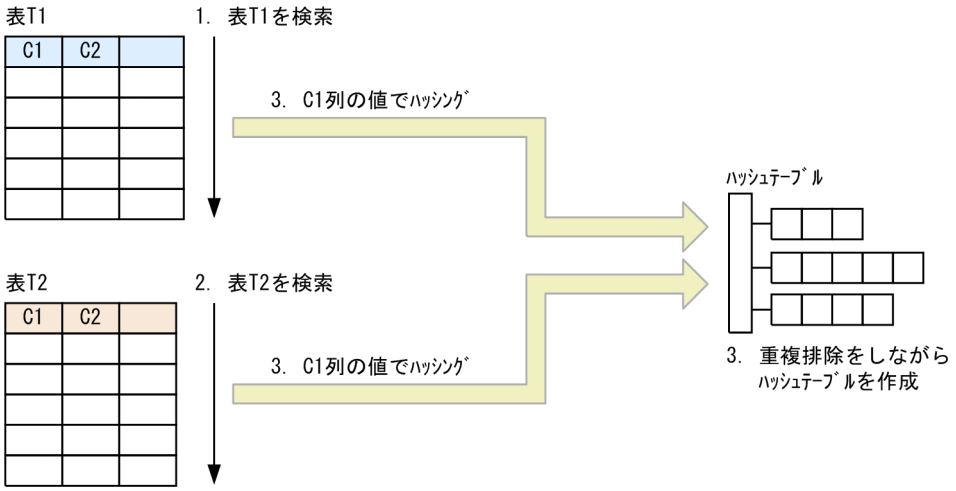
集合演算の処理方式にハッシュ実行が適用された場合，検索結果をハッシングしてハッシュテーブルを作成しながら重複排除を行います。

ハッシュ実行の適用例を次に示します。

■実行する SELECT 文

```
SELECT ”T1”.”C1” FROM ”T1”
UNION
SELECT ”T2”.”C1” FROM ”T2”
```

図 5-17　ハッシュ実行の処理方式



[説明]

1. 表T1 を検索して，表T1 のC1 列の値を取り出します。
2. 表T2 を検索して，表T2 のC1 列の値を取り出します。
3. 1.と 2.の結果をハッシングしてハッシュテーブルを作成しながら重複排除を行います。

ハッシュテーブルはハッシュテーブル領域に作成されます。ハッシュテーブル領域サイズは，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。なお，adb_sql_exe_hashtbl_area_size オペランドに0 を指定した場合，ハッシュ実行は適用されません。

■ハッシュテーブル領域が不足した場合の対処方法

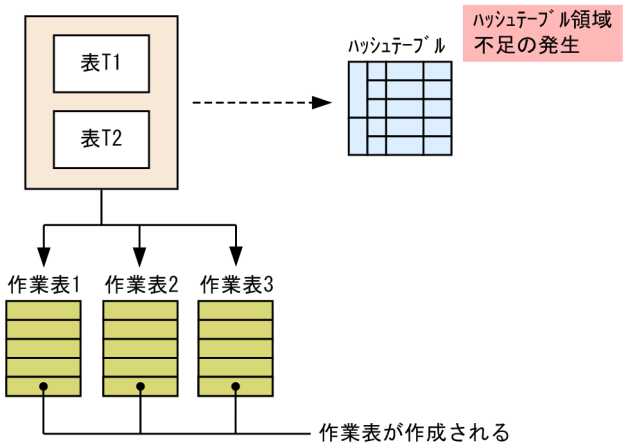
ハッシュテーブル領域が不足した場合，ハッシュテーブルに格納されるデータが複数の作業表に分割されて格納されます。そのため，SQL 文の処理時間が長くなることがあります。ハッシュテーブル領域の不足を解消するには，サーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドの指定値（ハッシュテーブル領域サイズの指定値）を大きくしてください。

ハッシュテーブル領域が不足して作業表が作成された場合，サーバメッセージログファイルにKFAA51130-W メッセージが出力されます。

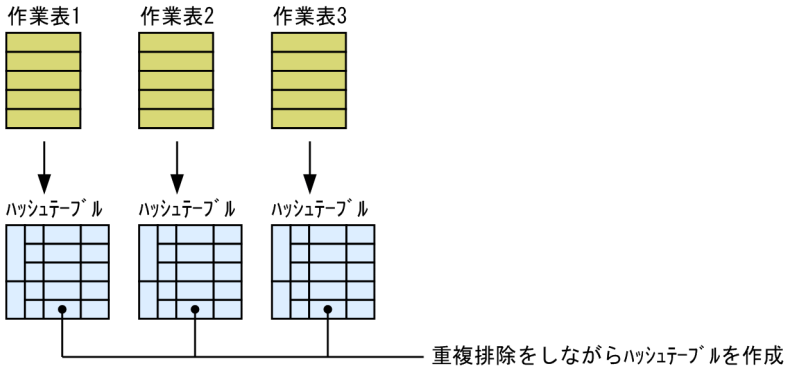
■ハッシュテーブル領域が不足した場合の処理の流れ

ハッシュテーブルの作成中に，ハッシュテーブル領域が不足した場合の処理の流れを説明します。

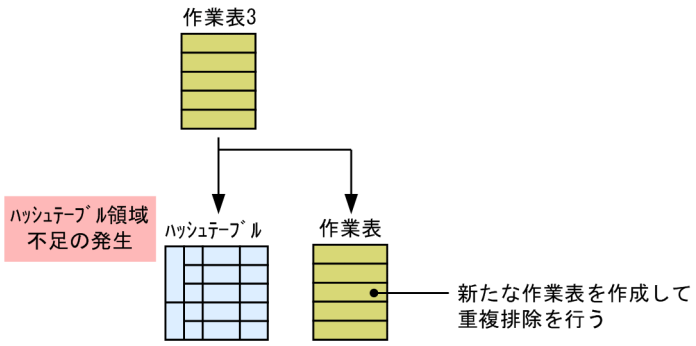
1. ハッシュテーブルの作成中にハッシュテーブル領域が不足した場合，複数の作業表を作成します。
   ハッシュテーブルに格納するデータを，各作業表に分割して格納します。



2. 作業表ごとにハッシュテーブルを作成します。



作業表に対するハッシュテーブルを作成する際に，ハッシュテーブル領域不足が発生した場合，新たな作業表をさらに作成します。ハッシュテーブル領域に格納できなかったデータをその作業表に格納して重複排除を行います。



5.8.2 作業表実行

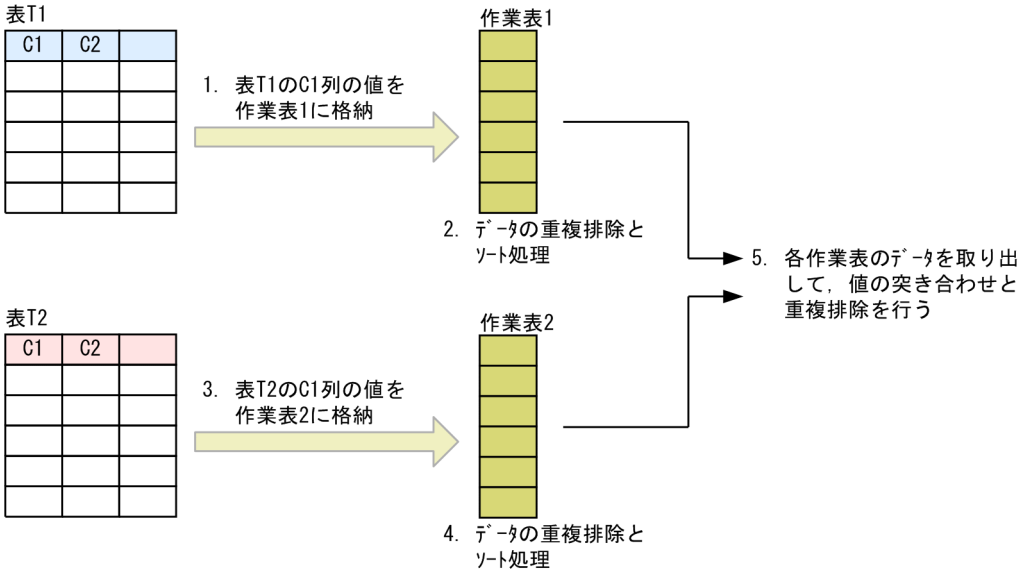
集合演算の処理方式に作業表実行が適用された場合，問合せ指定ごとに作業表を作成してデータをソートします。集合演算にALL 指定がない場合は，作業表内のデータの重複排除を行います。そのあと，各作業表のデータを突き合わせて集合演算を評価します。集合演算にALL 指定がない場合は，集合演算の評価結果の重複排除を行います。

作業表実行の適用例を次に示します。

■実行する SELECT 文

```
SELECT ”T1”.”C1” FROM ”T1”
UNION
SELECT ”T2”.”C1” FROM ”T2”
```

図 5-18　作業表実行の処理方式



[説明]

1. 問合せ指定「SELECT ”T1”.”C1” FROM ”T1”」の結果の値を格納する作業表 1 を作成します（表T1 を検索して，表T1 のC1 列の値を作業表 1 に格納します）。
2. 1.で作成した作業表 1 のデータの重複排除とソート処理を行います。
3. 問合せ指定「SELECT ”T2”.”C1” FROM ”T2”」の結果の値を格納する作業表 2 を作成します（表T2 を検索して，表T2 のC1 列の値を作業表 2 に格納します）。
4. 3.で作成した作業表 2 のデータの重複排除とソート処理を行います。
5. 各作業表から 1 行ずつデータを取り出し，値の突き合せと重複排除を行います。

5.8.3　集合演算の各処理方式の特徴

集合演算の各処理方式の特徴を次の表に示します。

表 5-9　集合演算の各処理方式の特徴

集合演算の処理方式	長所	短所
ハッシュ実行	集合演算に必要なデータがハッシュテーブル領域に格納できる場合は，高速に結果を求めることができます。	集合演算に必要なデータ量が多く，ハッシュテーブル領域が不足した場合は，いったんデータを作業表に格納するため，処理性能が低下します。

集合演算の処理方式	長所	短所
作業表実行	すべての集合演算に対して適用できます。	検索結果の件数が多い場合，作業表に格納するデータ量が多くなります。このとき，作業表に格納したデータをソートしたあとに重複排除をするため，処理性能が低下します。

5.9　SELECT DISTINCT の処理方式

SELECT 文にDISTINCT を指定した場合，次のどちらかの処理方式で重複排除処理が実行されます。

- ハッシュ実行
- 作業表実行

上記のうち，どちらの処理方式を適用するかは HADB が自動的に決定します。適用される処理方式については，アクセスパスで確認することができます。アクセスパスについては，次に示す個所を参照してください。

- アクセスパスの確認方法
  「6.1.2　アクセスパスを確認するには」を参照してください。
- アクセスパスに表示される内容
  「6.1.4　ツリー表示に出力される情報」の「(13)　重複排除の処理方式」を参照してください。

メモ

- SELECT 重複排除方式指定を指定することによって，ハッシュ実行の適用を抑止することができます。SELECT 重複排除方式指定については，マニュアル『HADB　SQL リファレンス』の『問合せ指定の指定形式および規則』を参照してください。
- HADB がSELECT DISTINCT の重複排除処理を不要と判断した場合，重複排除処理は行われません。
- SELECT DISTINCT の処理方式は，内部導出表の展開によって書き換えられた問合せ式と，探索条件の等価変換によって書き換えられた探索条件を基にして決定されます。内部導出表の展開については，「5.13　内部導出表の展開」を参照してください。探索条件の等価変換については，「5.11　探索条件の等価変換」を参照してください。
- スカラ演算の中に定数だけを指定している場合，そのスカラ演算は定数として扱われることがあります。定数と等価なスカラ演算については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。

5.9.1　ハッシュ実行

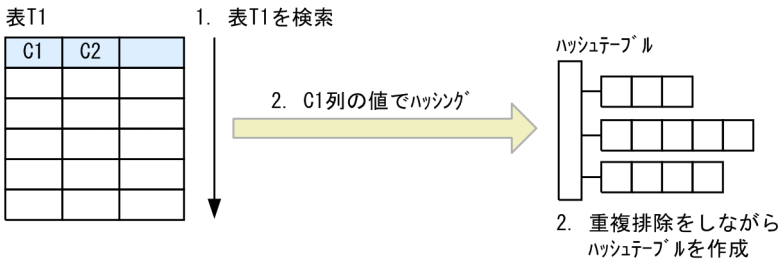
SELECT DISTINCT の処理方式にハッシュ実行が適用された場合，検索結果をハッシングしてハッシュテーブルを作成しながら重複排除を行います。

ハッシュ実行の適用例を次に示します。

■実行する SELECT 文

```
SELECT DISTINCT ”C1” FROM ”T1”
```

図 5-19　ハッシュ実行の処理方式



[説明]

1. 表T1 を検索して，表T1 のC1 列の値を取り出します。
2. 1.の結果をハッシングしてハッシュテーブルを作成しながら重複排除を行います。

ハッシュテーブルはハッシュテーブル領域に作成されます。ハッシュテーブル領域サイズは，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。なお，adb_sql_exe_hashtbl_area_size オペランドに0 を指定した場合，ハッシュ実行は適用されません。

■ハッシュテーブル領域が不足した場合の対処方法

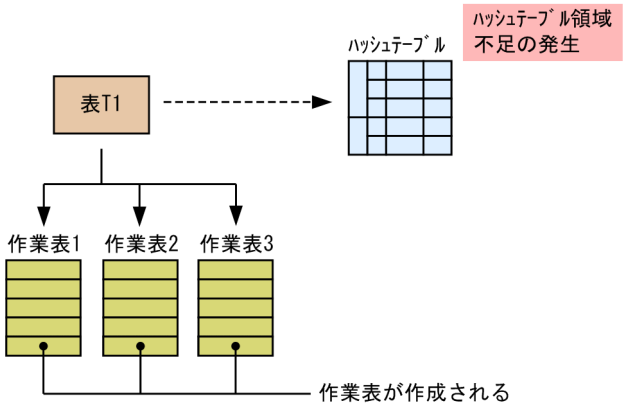
ハッシュテーブル領域が不足した場合，ハッシュテーブルに格納されるデータが複数の作業表に分割されて格納されます。そのため，SQL 文の処理時間が長くなることがあります。ハッシュテーブル領域の不足を解消するには，サーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドの指定値（ハッシュテーブル領域サイズの指定値）を大きくしてください。

ハッシュテーブル領域が不足して作業表が作成された場合，サーバメッセージログファイルにKFAA51130-W メッセージが出力されます。

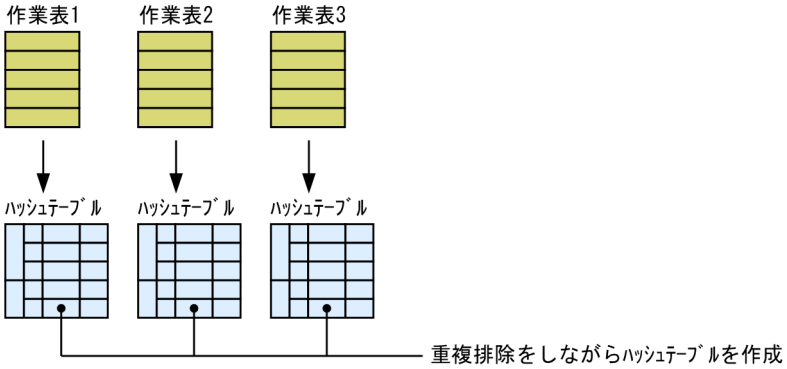
■ハッシュテーブル領域が不足した場合の処理の流れ

ハッシュテーブルの作成中に，ハッシュテーブル領域が不足した場合の処理の流れを説明します。

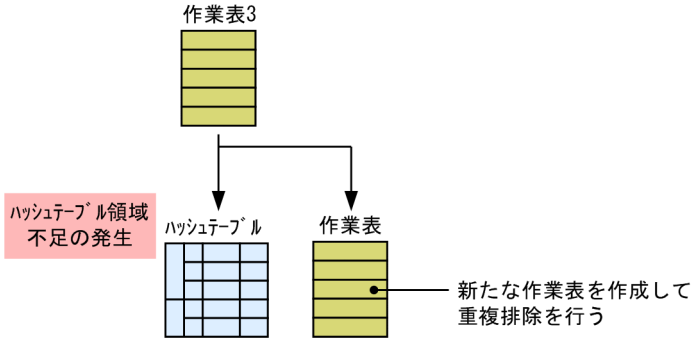
1. ハッシュテーブルの作成中にハッシュテーブル領域が不足した場合，複数の作業表を作成します。
   ハッシュテーブルに格納するデータを，各作業表に分割して格納します。



2. 作業表ごとにハッシュテーブルを作成します。



作業表に対するハッシュテーブルを作成する際に，ハッシュテーブル領域不足が発生した場合，新たな作業表をさらに作成します。ハッシュテーブル領域に格納できなかったデータをその作業表に格納して重複排除を行います。



5.9.2　作業表実行

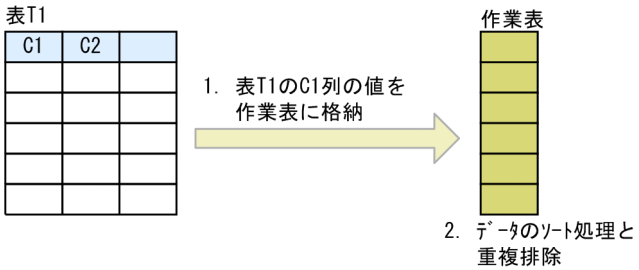
SELECT DISTINCT の処理方式に作業表実行が適用された場合，検索結果を格納した作業表を作成します。そのあと，作業表内のデータをソートし，重複排除を行います。

作業表実行の適用例を次に示します。

■実行する SELECT 文

```
SELECT DISTINCT "C1" FROM "T1"
```

図 5-20　作業表実行の処理方式



[説明]

1. 表T1 を検索して，表T1 のC1 列の値を作業表に格納します。

2.1.で作成した作業表のデータのソート処理と重複排除を行います。

5.9.3 SELECT DISTINCT の各処理方式の特徴

SELECT DISTINCT の各処理方式の特徴を次の表に示します。

表 5-10 SELECT DISTINCT の各処理方式の特徴

SELECT DISTINCT の処理方式	長所	短所
ハッシュ実行	SELECT DISTINCT の実行に必要なデータがハッシュテーブル領域に格納できる場合は，高速に結果を求めることができます。	SELECT DISTINCT の実行に必要なデータ量が多く，ハッシュテーブル領域が不足した場合は，いったんデータを作業表に格納するため，処理性能が低下します。
作業表実行	すべてのSELECT DISTINCT に対して適用できます。	検索結果の件数が多い場合，作業表に格納するデータ量が多くなります。このとき，作業表に格納したデータをソートしたあとに重複排除をするため，処理性能が低下します。

5.10　作業表が作成される SQL を実行する際の考慮点

SQL 文の実行時にデータのソート処理や重複排除処理などが実行される場合，作業表用 DB エリアに作業表が作成されることがあります。作業表には，ソートまたは重複排除したデータなどが一時的に格納されます。このため，大量のデータをソートまたは重複排除すると，作業表を作成する処理による負荷によって，十分な性能が得られないことがあります。

> **重要**
>
> 性能低下を抑えるためには，作業表用 DB エリアの容量見積もりを正しく行う必要があります。作業表が作成される SQL が当初の見積もり時より多く実行される場合，AP 開発者は HADB のシステム設計者またはシステム管理者に，作業表用 DB エリアの容量を見積もり直すよう依頼をしてください。

5.10.1　作業表の種類

作業表には，グローバル作業表とローカル作業表があります。

(1)　グローバル作業表

グローバル作業表は，1 つの SQL を複数のリアルスレッドで処理するときに，作業表のデータを複数のリアルスレッド間で共有するための作業表です。表を結合するための作業表や副問合せのための作業表などに利用されます。

また，グローバル作業表が使用するグローバルバッファのページ数を，サーバ定義の adb_dbbuff_wrktbl_glb_blk_num オペランドで指定します。adb_dbbuff_wrktbl_glb_blk_num 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『性能に関するオペランド（set 形式)』を参照してください。

(2)　ローカル作業表

ローカル作業表は，SQL を処理するリアルスレッドごとに作成される作業表です。ORDER BY 句を指定した場合に作成される作業表や，グループ化の処理方式としてローカルハッシュグループ化が選択されたときのグループ分けの作業表などに利用されます。

また，ローカル作業表が使用するローカル作業表用バッファのページ数は，サーバ定義の adb_dbbuff_wrktbl_clt_blk_num オペランド，クライアント定義のadb_dbbuff_wrktbl_clt_blk_num オペランド，またはエクスポートオプションadb_export_wrktbl_blk_num（adbexport コマンド実行時）で指定します。なお，対象のオペランドおよびオプションについては，次を参照してください。

- サーバ定義のadb_dbbuff_wrktbl_clt_blk_num オペランドの場合：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『性能に関するオペランド（set 形式）』
- クライアント定義のadb_dbbuff_wrktbl_clt_blk_num オペランドの場合：「2.2.3　性能に関するオペランド」にある，adb_dbbuff_wrktbl_clt_blk_num オペランドの説明
- エクスポートオプションadb_export_wrktbl_blk_num の場合：マニュアル『HADB　コマンドリファレンス』の『adbexport（データのエクスポート）』の『adbexport コマンドの指定形式』

5.10.2　SQL を実行した場合に作成される作業表について

次の表に示す SQL を実行した場合に作業表が作成されます。

> **重要**
>
> 作業表の行長が最大行長を超えた場合，SQL 文がエラーになります。作業表の最大行長については，マニュアル『HADB　システム構築・運用ガイド』の『データベースに関する最大値と最小値』を参照してください。作業表の行長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『作業表を格納するために必要な基本行用ページ数の求め方』の変数 *ROWSZ* を参照してください。

表 5-11　作業表が作成される SQL

項番	作業表が作成される SQL		作業表の用途	作業表の構成列	作業表の種類
1	ORDER BY 句を指定した場合	集合演算の結果をソートキーとする場合	検索結果のソート処理に使われます。	• 問合せ式から導出される結果の列の値を格納するための列	ローカル作業表
		問合せ指定の結果をソートキーとする場合		• ORDER BY 句のソートキーに指定した値式の結果を格納するための列 • 実表の検索結果行に対する行 ID を格納するための列（FROM 句に実表を指定した場合）※1 • 検索対象列の値を格納するための列（表の検索方式にキースキャンが適用される実表を，FROM 句に指定した場合） • カラムストア表の検索対象列※11 の値を格納するための列（FROM 句にカラムストア表を指定した場合）	

項番	作業表が作成される SQL		作業表の用途	作業表の構成列	作業表の種類
				• 導出表の問合せ式本体から導出された結果の列の値を格納するための列（FROM 句に導出表を指定した場合）※6 • ビュー定義の問合せ式から導出される結果の列の値を格納するための列（FROM 句にビュー表を指定した場合）※6 • WITH 句の問合せ式本体から導出される結果の列の値を格納するための列（FROM 句に問合せ名を指定した場合）※6 • 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結果の列の値を格納するための列（FROM 句に表関数導出表を指定した場合） • 集まり導出表の結果の列の値を格納するための列（FROM 句に集まり導出表を指定した場合） • スカラ関数RANDOMROW の結果の値を格納するための列（選択式にスカラ関数 RANDOMROW を指定した場合）	
2	GROUP BY 句を指定した場合	グループ化の処理方式がグローバルハッシュグループ化の場合※2	グループ分けの結果を保持するために使われます。なお，この作業表は，ハッシュテーブル領域が不足した場合に使われます。 ハッシュテーブル領域は，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。	• GROUP BY 句のグループ化列に指定した値式の結果の値を格納するための列 • 集合関数の結果を格納するための列 • HADB サーバがハッシュ処理で使用する情報を格納するための列※9※10	• ローカル作業表※7 • グローバル作業表※8
		グループ化の処理方式がローカルハッシュグループ化の場合※2	グループ分けのソート処理に使います。なお，この作業表は，ハッシュグループ化領域が不足した場合に使われます。ハッシュグループ化領域は，サーバ定義またはクライアント定義の	• GROUP BY 句のグループ化列に指定した値式の結果の値を格納するための列 • 集合関数の引数に指定した列を格納するための列 • 集合関数の結果を格納するための列	ローカル作業表

項番	作業表が作成される SQL		作業表の用途	作業表の構成列	作業表の種類
			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がソートグループ化の場合※2	グループ分けのソート処理に使われます。		
3	SELECT DISTINCT を指定した場合	SELECT DISTINCT の処理方式がハッシュ実行の場合※12	検索結果を保持するための処理に使われます。なお，この作業表は，ハッシュテーブル領域が不足した場合に使われます。ハッシュテーブル領域は，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。	• 選択式の結果の列の値を格納するための列 • HADB がハッシュ処理で使用する情報を格納するための列※9※10	• ローカル作業表※7 • グローバル作業表※8
		SELECT DISTINCT の処理方式が作業表実行の場合※12	検索結果をソート，重複排除するための処理に使われます。	• 選択式の結果の列の値を格納するための列	ローカル作業表
4	次のどれかの関数を指定した場合 • DISTINCT 集合関数 • 逆分布関数 • ORDER BY 句を指定した ARRAY_AGG 集合関数 • LISTAGG 集合関数	グループ化の処理方式がグローバルハッシュグループ化の場合※2	重複排除された集合関数の入力値を保持するために使われます。なお，この作業表は，ハッシュテーブル領域が不足した場合に使われます。ハッシュテーブル領域は，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。	• GROUP BY 句のグループ化列に指定した値式の結果の値を格納するための列 • ALL 集合関数の引数に指定した列の値を格納するための列 • DISTINCT 集合関数の引数に指定した値式の結果を格納するための列 • 逆分布関数のWITHIN グループ指定のソートキーに指定した値式の結果を格納するための列 • 逆分布関数MEDIAN の引数に指定した値式の結果を格納するための列 • LISTAGG 集合関数の引数に指定した値式の結果を格納するための列 • LISTAGG 集合関数のWITHIN グループ指定のソートキーに指定した値式の結果を格納するための列 • ARRAY_AGG 集合関数の引数に指定した値式の結果を格	• ローカル作業表※7 • グローバル作業表※8
		グループ化の処理方式がグローバルハッシュグループ化以外の場合※2	集合関数の入力値を重複排除するための処理，または集合関数の入力値をソートするための処理に使われます。		

項番	作業表が作成される SQL		作業表の用途	作業表の構成列	作業表の種類
				納するための列（ARRAY_AGG 集合関数に ORDER BY 句を指定した場合） • ARRAY_AGG 集合関数の ORDER BY 句のソートキーに指定した値式の結果を格納するための列 • HADB サーバがハッシュ処理で使用する情報を格納するための列※9※10	
5	ウィンドウ関数を指定した場合		ウィンドウ関数の結果を求めるためのソート処理に使われます。	• FROM 句に指定した表の列の値を格納するための列 • 選択式に指定したウィンドウ関数の結果の値を格納するための列	ローカル作業表
6	FROM 句に複数の表参照を指定した場合	表の結合方式がハッシュジョインの場合※3	表の結合処理で，結合対象となる表参照の結果を保持するために使われます。なお，この作業表は，ハッシュテーブル領域が不足した場合に使われます。ハッシュテーブル領域は，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。	• FROM 句に指定した表の列の値を格納するための列 • HADB サーバがハッシュ処理で使用する情報を格納するための列※9※10	• ローカル作業表※7 • グローバル作業表※8
		表の結合方式がネストループジョインの場合※3	表の結合処理で，結合対象となる表参照の結果を保持するために使われます。		グローバル作業表
7	導出表を指定した場合		導出表に対応する問合せ式本体の結果を保持するために使われます。	• 導出表の問合せ式本体から導出される結果の列の値を格納するための列※6	グローバル作業表
8	ビュー表を指定した場合		ビュー表に対応する問合せ式の結果を保持するために使われます。	• ビュー定義の問合せ式から導出される結果の列の値を格納するための列※6	グローバル作業表
9	WITH 句を指定した場合		問合せ名に対応する問合せ式本体の結果を保持するために使われます。	• WITH 句中の問合せ式本体から導出される結果の列の値を格納するための列※6	グローバル作業表
10	表関数導出表を指定した場合		表関数導出表を導出するシステム定義関数の結果を保持するために使われます。	• システム定義関数の結果の列の値を格納するための列	グローバル作業表
11	結合表を指定した場合		結合表の結果を保持するために使われます。	• 結合表の列の値を格納するための列	グローバル作業表

項番	作業表が作成される SQL		作業表の用途	作業表の構成列	作業表の種類
			なお，結合表の結果を求めるための作業表については，項番 6 の『FROM 句に複数の表参照を指定した場合』を参照してください。		
12	副問合せを指定した場合	副問合せの処理方式がハッシュ実行の場合※4	副問合せの結果を保持するために使われます。なお，この作業表は，ハッシュテーブル領域が不足した場合に使われます。ハッシュテーブル領域は，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。	• 副問合せの選択式の結果の列の値を格納するための列 • 副問合せに含まれる外への参照列の値を格納するための列※5 • 副問合せに含まれる集合関数の結果を格納するための列 • HADB サーバがハッシュ処理で使用する情報を格納するための列※9※10	• ローカル作業表※7 • グローバル作業表※8
		副問合せの処理方式がハッシュ実行以外の場合※4	副問合せの結果を保持するために使われます。	• 副問合せの選択式の結果の列の値を格納するための列	グローバル作業表
13	集合演算を指定した場合	集合演算の処理方式がハッシュ実行の場合※13	重複排除の結果を保持するために使われます。なお，この作業表は，ハッシュテーブル領域が不足した場合に使われます。ハッシュテーブル領域は，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。	• 問合せ指定に指定した選択式の結果の列の値を格納するための列 • HADB がハッシュ処理で使用する情報を格納するための列※9※10	• ローカル作業表※7 • グローバル作業表※8
		集合演算の処理方式が作業表実行の場合※13	検索結果をソート，重複排除するための処理に使われます。	• 問合せ指定に指定した選択式の結果の列の値を格納するための列	ローカル作業表
14	再帰的問合せの指定がある場合		アンカーメンバの結果，および再帰的メンバの結果を保持するために使われます。	• 再帰的問合せの結果の列の値を格納するための列	グローバル作業表

注※1

行 ID とは，行の格納位置を示す値のことをいいます。行 ID のデータ型はCHAR(16)です。

注※2

グループ化の処理方式については，「5.7　グループ化の処理方式」を参照してください。

注※3

表の結合方式については，「5.5　表の結合方式」を参照してください。

注※4

副問合せの処理方式については，「5.6　副問合せの処理方式」を参照してください。

注※5

外への参照列については，マニュアル『HADB　SQL リファレンス』の『副問合せの指定形式および規則』を参照してください。

注※6

内部導出表の展開によって実表として扱われることがあります。内部導出表の展開については，マニュアル『HADB　SQL リファレンス』の『内部導出表』を参照してください。

注※7

ハッシュテーブル領域が不足した場合に，ハッシュテーブル領域に格納するデータを分割して格納する際に使用する作業表です。

注※8

ハッシュテーブル領域に格納するデータを作業表に格納したあと，さらにハッシュテーブル領域が不足した場合に，処理できなかったデータを格納する際に使用する作業表です。

注※9

HADB サーバがハッシュ処理で使用する情報を格納するための列のデータ型はINTEGER 型になります。

注※10

ハッシュテーブル領域に格納するデータを作業表に格納したあと，さらにハッシュテーブル領域が不足した場合に，処理できなかったデータを格納する際に使用する作業表で作成される列です。

注※11

定義長が 128 バイト以上の列は，対象外になります。定義長が 128 バイト以上の列とは，次の列のことです。

- 定義長が 128 バイト以上のCHARACTER 型の列
- 定義長が 128 バイト以上のVARCHAR 型の列
- 定義長が 128 バイト以上のBINARY 型の列
- 定義長が 128 バイト以上のVARBINARY 型の列

注※12

SELECT DISTINCT の処理方式については，「5.9　SELECT DISTINCT の処理方式」を参照してください。

注※13

集合演算の処理方式については，「5.8　集合演算の処理方式」を参照してください。

SQL を実行した結果，作業表が作成されたかどうかを，アクセスパスで確認することができます。アクセスパスについては，次に示す個所を参照してください。

- アクセスパスの確認方法

  「6.1.2　アクセスパスを確認するには」を参照してください。

- アクセスパスに表示される内容
  「6.1.4　ツリー表示に出力される情報」の「(8)　作業表の作成情報」を参照してください。

5.10.3　作成される作業表の個数

SQL 文の実行時に作成される作業表の個数を例を使って説明します。なお，ここで説明する作業表の数は，SQL 文から想定される値です。実際に SQL 文実行時に作成される作業表の数は，サーバ定義またはクライアント定義の最大 SQL 処理リアルスレッド数（adb_sql_exe_max_rthd_num オペランド），操作対象のデータ件数などに影響されます。

(1)　例 1（ORDER BY 句を指定している場合）

SQL 文の例

```
SELECT ”C1”,”C2”,”C3” FROM ”T1” ORDER BY ”C1” ASC
```

[説明]

ORDER BY 句のソート処理で使う作業表が 1 つ作成されます。

(2)　例 2（GROUP BY 句を指定している場合）

SQL 文の例

```
SELECT ”C1”,”C2” FROM ”T1” GROUP BY ”C1”,”C2”
```

[説明]

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るかどうかによって，作成される作業表の個数が異なります。

- グローバルハッシュグループ化が適用される場合
  - SQL パラレル実行機能が適用されない場合
    GROUP BY 句の処理に使用する作業表が 2 つ作成されます。
  - SQL パラレル実行機能が適用される場合
    GROUP BY 句の処理に使用する作業表が 4 つ作成されます。
- グローバルハッシュグループ化が適用されない場合
  GROUP BY 句の処理に使用する作業表が 1 つ作成されます。

グループ化の処理方式については，「5.7　グループ化の処理方式」を参照してください。

(3) 例 3（SELECT DISTINCT を指定している場合）

SQL 文の例

```
SELECT DISTINCT ”C1”,”C2”,”C3” FROM ”T1”
```

[説明]

適用されるSELECT DISTINCT の処理方式によって，作成される作業表の個数が異なります。

- ハッシュ実行が適用される場合

  SELECT DISTINCT の処理に使用する作業表が 2 つ作成されます。

- 作業表実行が適用される場合

  SELECT DISTINCT の処理に使用する作業表が 1 つ作成されます。

SELECT DISTINCT の処理方式については，「5.9 SELECT DISTINCT の処理方式」を参照してください。

(4) 例 4（DISTINCT 集合関数を指定している場合）

SQL 文の例

```
SELECT COUNT(DISTINCT ”C1”) FROM ”T1”
```

[説明]

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るかどうかによって，作成される作業表の個数が異なります。

- グローバルハッシュグループ化が適用される場合

  集合関数の処理に使用する作業表が 2 つ作成されます。

- グローバルハッシュグループ化が適用されない場合

  集合関数の処理に使用する作業表が 1 つ作成されます。

グループ化の処理方式については，「5.7 グループ化の処理方式」を参照してください。

(5) 例 5（ウィンドウ関数（RANK）を指定している場合）

SQL 文の例

```
SELECT ”C1”,RANK() OVER (PARTITION BY ”C2” ORDER BY ”C3” ASC) FROM ”T1”
```

[説明]

ウィンドウ関数（RANK）の処理で使う作業表が 1 つ作成されます。

(6) 例 6（GROUP BY 句と ORDER BY 句を指定している場合）

SQL 文の例

```
SELECT ”C1”,”C2” FROM ”T1” GROUP BY ”C1”,”C2” ORDER BY ”C1” ASC
```

[説明]

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るかどうかによって，作成される作業表の個数が異なります。

- グローバルハッシュグループ化が適用される場合

  GROUP BY 句の処理に使用する作業表が 2 つ，ORDER BY 句の処理に使用する作業表が 1 つ，合計 3 つの作業表が作成されます。

- グローバルハッシュグループ化が適用されない場合

  GROUP BY 句の処理に使用する作業表が 1 つ作成されます。この例の場合，GROUP BY 句とORDER BY 句で必要なソート処理が 1 回で済むため，作成される作業表は 1 つになります。

グループ化の処理方式については，「5.7　グループ化の処理方式」を参照してください。

(7) 例 7（GROUP BY 句と ORDER BY 句を指定している場合）

SQL 文の例

```
SELECT ”C1”,”C2”,COUNT(*) ”DC1” FROM ”T1”
     GROUP BY ”C1”,”C2” ORDER BY ”DC1” ASC
```

[説明]

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るかどうかによって，作成される作業表の個数が異なります。

- グローバルハッシュグループ化が適用される場合

  GROUP BY 句の処理に使用する作業表が 2 つ，ORDER BY 句の処理に使用する作業表が 1 つ，合計 3 つの作業表が作成されます。

- グローバルハッシュグループ化が適用されない場合

  GROUP BY 句の処理に使用する作業表が 1 つ，ORDER BY 句の処理に使用する作業表が 1 つ，合計 2 つの作業表が作成されます。

グループ化の処理方式については，「5.7　グループ化の処理方式」を参照してください。

(8) 例 8（SELECT DISTINCT と ORDER BY 句を指定している場合）

SQL 文の例

```
SELECT DISTINCT ”C1”,”C2”,”C3” FROM ”T1” ORDER BY ”C1” ASC
```

[説明]

SELECT DISTINCT のソート処理で使う作業表が 1 つ作成されます。

この例の場合，SELECT DISTINCT とORDER BY 句に必要なソート処理が 1 回で済むため，作成される作業表は 1 つになります。

(9) 例 9（GROUP BY 句と DISTINCT 集合関数を指定している場合）

SQL 文の例

```
SELECT ”C1”,COUNT(DISTINCT ”C2”) FROM ”T1” GROUP BY ”C1”
```

[説明]

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るかどうかによって，作成される作業表の個数が異なります。

- グローバルハッシュグループ化が適用される場合

  GROUP BY 句の処理に使用する作業表が 2 つ，集合関数の処理に使用する作業表が 1 つ，合計 3 つの作業表が作成されます。

- グローバルハッシュグループ化が適用されない場合

  GROUP BY 句の処理に使用する作業表が 1 つ作成されます。この例の場合，GROUP BY 句と集合関数で必要なソート処理が 1 回で済むため，作成される作業表は 1 つになります。

グループ化の処理方式については，「5.7　グループ化の処理方式」を参照してください。

(10) 例 10（表を結合している場合）

SQL 文の例

```
SELECT ”T1”.”C1”,”T1”.”C2”,”T2”.”C1”,”T2”.”C2” FROM ”T1”,”T2”
    WHERE ”T1”.”C1”=”T2”.”C1”
```

[説明]

結合処理で使用する作業表が 3 つ作成されます。

(11) 例 11（FROM 句に複数の表参照を指定している場合）

SQL 文の例

```
SELECT ”DT1”.”C1”,”DT2”.”C1”
  FROM (SELECT COUNT(”T1”.”C1”) FROM ”T1”) AS ”DT1”(”C1”),
       (SELECT COUNT(”T2”.”C1”) FROM ”T2”) AS ”DT2”(”C1”)
  WHERE ”DT1”.”C1”>”DT2”.”C1”
```

[説明]

表の結合処理で使う作業表が 1 つ作成されます。

(12) 例 12（副問合せを指定している場合）

SQL 文の例

```
SELECT "T1"."C1","T1"."C2","T1"."C3" FROM "T1"
    WHERE "T1"."C1"=(SELECT "T2"."C1" FROM "T2"
                          WHERE "T2"."C2"="T1"."C2")
```

[説明]

副問合せの処理方式にハッシュ実行が適用されるかどうかによって，作成される作業表の個数が異なります。

- ハッシュ実行が適用される場合

  外への参照列を含む副問合せの処理で使う作業表が 3 つ作成されます。

- ハッシュ実行が適用されない場合

  外への参照列を含む副問合せの処理で使う作業表は作成されません。

副問合せの処理方式については，「5.6 副問合せの処理方式」を参照してください。

(13) 例 13（IN 副問合せを指定している場合）

SQL 文の例

```
SELECT "C1","C2","C3" FROM "T1" WHERE "C1" IN (SELECT "C1" FROM "T2")
```

[説明]

副問合せの処理方式にハッシュ実行が適用されるかどうかによって，作成される作業表の個数が異なります。

- ハッシュ実行が適用される場合

  IN 述語に指定した副問合せの処理で使う作業表が 3 つ作成されます。

- ハッシュ実行が適用されない場合

  IN 述語に指定した副問合せの処理で使う作業表が 1 つ作成されます。

副問合せの処理方式については，「5.6 副問合せの処理方式」を参照してください。

(14) 例 14（限定述語を指定している場合）

SQL 文の例

```
SELECT "C1","C2","C3" FROM "T1" WHERE "C1"=ANY(SELECT "C1" FROM "T2")
```

[説明]

副問合せの処理方式にハッシュ実行が適用されるかどうかによって，作成される作業表の個数が異なります。

- ハッシュ実行が適用される場合

限定述語に指定した副問合せの処理で使う作業表が３つ作成されます。

- ハッシュ実行が適用されない場合

  限定述語に指定した副問合せの処理で使う作業表が１つ作成されます。

副問合せの処理方式については，「5.6　副問合せの処理方式」を参照してください。

(15)　例 15（EXISTS 述語を指定している場合）

SQL 文の例

```
SELECT "T1"."C1","T1"."C2","T1"."C3" FROM "T1"
    WHERE EXISTS(SELECT * FROM "T2" WHERE "T2"."C2"="T1"."C2")
```

[説明]

副問合せの処理方式にハッシュ実行が適用されるかどうかによって，作成される作業表の個数が異なります。

- ハッシュ実行が適用される場合

  EXISTS 述語に指定した副問合せの処理で使う作業表が３つ作成されます。

- ハッシュ実行が適用されない場合

  EXISTS 述語に指定した副問合せの処理で使う作業表は作成されません。

副問合せの処理方式については，「5.6　副問合せの処理方式」を参照してください。

(16)　例 16（表関数導出表を指定している場合）

SQL 文の例

```
SELECT "C1","C2","C3"
    FROM TABLE(ADB_CSVREAD(MULTISET (SELECT "FNAME" FROM "TFILE"),
                           'COMPRESSION_FORMAT=GZIP;
                            FIELD_NUM=1,2,3;'))
          AS "T1"("C1" INTEGER,"C2" CHAR(10),"C3" DATE)
```

[説明]

マルチ集合値式に指定した表副問合せの結果を保持するための作業表が１つ作成されます。マルチ集合値式については，マニュアル『HADB　SQL リファレンス』の『マルチ集合値式の指定形式および規則』を参照してください。

(17)　例 17（ビュー表を指定している場合）

SQL 文の例

```
CREATE VIEW "VT1"("C1","C2")
    AS SELECT "T1"."C1","T2"."C1" FROM "T1","T2"
          WHERE "T1"."C2"<="T2"."C2"
SELECT * FROM "VT1" AS "XT1","VT1" AS "XT2"
    WHERE "XT1"."C1">"XT2"."C1"
```

[説明]

ビュー表の処理で使う作業表が 1 つ作成されます。

(18) 例 18（WITH 句を指定している場合）

SQL 文の例

```
WITH ”QT1”(”C1”,”C2”) AS (SELECT ”T1”.”C1”,”T2”.”C1” FROM ”T1”,”T2”
                           WHERE ”T1”.”C2”<=”T2”.”C2”)
SELECT * FROM ”QT1” WHERE ”C1”=(SELECT MAX(”C1”) FROM ”QT1”)
```

[説明]

WITH 句の処理で使う作業表が 1 つ作成されます。

(19) 例 19（集合演算を指定している場合）

SQL 文の例

```
SELECT ”C1”,”C2” FROM ”T1” UNION SELECT ”C1”,”C2” FROM ”T2”
```

[説明]

集合演算の処理で使う作業表が 2 つ作成されます。

(20) 例 20（再帰的問合せを指定している場合）

SQL 文の例

```
WITH ”QT1”(”C1”,”C2”)
  AS (SELECT ”C1”,”C2” FROM ”T1” WHERE ”C2” BETWEEN ’AA’ AND ’EE’
      UNION ALL
      SELECT ”C1”+1,”C2” FROM ”QT1” WHERE ”C1”<=10)
SELECT * FROM ”QT1”
```

[説明]

再帰的問合せでは，アンカーメンバの結果と再帰的メンバの結果を保持するため，再帰的問合せの処理で使う作業表が 2 つ作成されます。再帰的問合せについては，マニュアル『HADB　SQL リファレンス』の『問合せ式』を参照してください。

(21) 例 21（DISTINCT 集合関数と GROUP BY 句を指定している場合）

SQL 文の例

```
SELECT ”C1”,COUNT(DISTINCT ”C2”),SUM(”C3”) FROM ”T1”
    GROUP BY ”C1”
```

[説明]

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るかどうかによって，作成される作業表の個数が異なります。

- グローバルハッシュグループ化が適用される場合

  GROUP BY 句の処理に使用する作業表が 2 つ，集合関数の処理に使用する作業表が 2 つ，合計 4 つの作業表が作成されます。

- グローバルハッシュグループ化が適用されない場合

  GROUP BY 句の処理に使用する作業表が 1 つ作成されます。この例の場合，GROUP BY 句のソート処理と集合関数のソート処理が同時に実行されるため，集合関数の処理に使用する作業表は作成されません。

グループ化の処理方式については，「5.7　グループ化の処理方式」を参照してください。

(22)　例 22（複数の DISTINCT 集合関数を指定している場合）

SQL 文の例

```
SELECT COUNT(DISTINCT ”C1”),COUNT(DISTINCT ”C2”) FROM ”T1”
```

[説明]

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るかどうかによって，作成される作業表の個数が異なります。

- グローバルハッシュグループ化が適用される場合

  集合関数の処理に使用する作業表が 3 つ作成されます。

- グローバルハッシュグループ化が適用されない場合

  集合関数の処理に使用する作業表が 2 つ作成されます。

グループ化の処理方式については，「5.7　グループ化の処理方式」を参照してください。

5.11　探索条件の等価変換

探索条件に指定された条件を効率的に評価するために，指定された探索条件を HADB が変換して評価することがあります。これを**探索条件の等価変換**といいます。探索条件が等価変換された場合，検索時に使用されるインデクスは等価変換後の探索条件を基に決定されます。検索時に使用されるインデクスの決定方法については，「5.2　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」および「5.3　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。

HADB は，探索条件に指定された条件を次の順序で等価変換します。

1. 行値構成子を指定した比較述語に関する等価変換（行値構成子を含まない条件への変換）
   行値構成子を指定した比較述語を，行値構成子を含まない条件に等価変換します。
2. OR 条件に関する等価変換（OR 条件の外側への抜き出し）
   OR 条件中の条件を，OR 条件の外側に抜き出す等価変換を行います。
3. OR 条件に関する等価変換（IN 条件への変換）
   OR 条件中の = 条件から作成したIN 条件を，OR 条件の外側に追加する等価変換を行います。
4. 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）
   OR 条件中の探索条件を，集合演算UNION ALL を指定した導出表に等価変換します。
5. スカラ演算に関する等価変換
   スカラ演算を移項して等価変換します。
6. 行値構成子を指定したIN 述語に関する等価変換（同じ表の列指定の抽出）
   行値構成子を指定したIN 述語から，同じ表の列指定だけを抽出して条件を追加する等価変換を行います。
7. 行値構成子を指定したIN 述語に関する等価変換（行値構成子要素ごとの条件の追加）
   行値構成子を指定したIN 述語から，行値構成子要素ごとの条件を追加する等価変換を行います。
8. IN 述語に関する等価変換
   IN 述語を = 条件または <> 条件に等価変換します。
9. HAVING 句に関する等価変換（WHERE 句への変換）
   HAVING 句の探索条件をWHERE 句の探索条件に等価変換します。
10. 導出問合せを指定した SQL 文の探索条件に関する等価変換（導出問合せのWHERE 句への移動）
    導出問合せを指定した SQL 文のWHERE 句に指定された探索条件を，導出問合せのWHERE 句に移動する等価変換を行います。

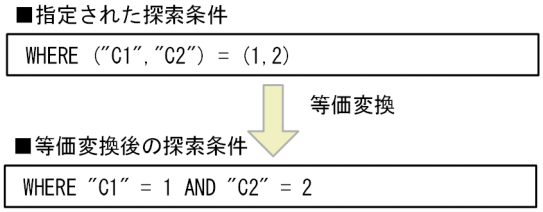
ここでは，上記の各等価変換について説明します。

5.11.1　行値構成子を指定した比較述語に関する等価変換（行値構成子を含まない条件への変換）

探索条件に行値構成子を指定した比較述語が指定されている場合，行値構成子を含まない条件に等価変換されます。探索条件が等価変換されると，検索範囲を絞り込むための条件として有効に利用できることがあります。

等価変換の例を次に示します。例中のC1，C2 は列名を意味しています。

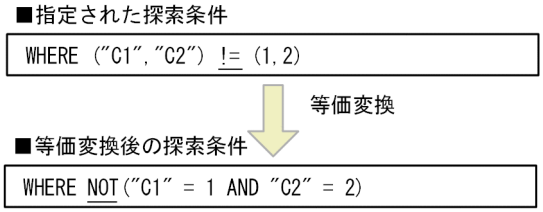
(1)　例 1



[説明]

行値構成子が指定された条件が，行値構成子を含まない条件に等価変換されます。

(2)　例 2



[説明]

!=条件がNOT 条件に等価変換されます。

(3)　等価変換される条件の形式

等価変換される条件の形式を次に示します。

- 比較述語

  行値構成子 比較演算子 行値構成子

  比較演算子に!=，^=，または<>が指定されている場合，論理演算子NOT を使用して等価変換されます。

5.11.2　OR 条件に関する等価変換（OR 条件の外側への抜き出し）

OR 条件中に同じ条件が指定されている場合※，同じ条件をOR 条件の外側に抜き出す等価変換が実行されます。同じ条件をOR 条件の外側に抜き出すと，検索範囲を絞り込むための条件として有効に利用できることがあります。また，OR 条件中の同じ条件が 1 つに集約されるため，条件評価の負荷が軽減されることがあります。

注※

ANY を指定した配列要素参照（識別番号は省略）が探索条件に指定されている場合は，探索条件が同じであっても等価変換は実行されません。具体例については，「(2)　等価変換されない例」の「(d)　例 4」を参照してください。

重要

- WHERE 句の探索条件，更新系 SQL のWHERE 探索条件，結合表のON 探索条件，およびHAVING 句の探索条件に指定したOR 条件が等価変換の対象になります。
- 探索条件が等価変換された場合，検索時に使用されるインデクスは等価変換後の探索条件を基に決定されます。

等価変換の例を次に示します。例中のC1，C2，C3 は列名を意味しています。

(1)　等価変換される例

(a)　例 1

■指定された探索条件

```
WHERE (“C1” = 100 AND “C2” = 'A')
   OR (“C1” = 100 AND “C3” > 20)
```

等価変換

■等価変換後の探索条件

```
WHERE “C1” = 100
   AND (“C2” = 'A' OR “C3” > 20)
```

[説明]

OR 条件の両側に「”C1” = 100」という同じ条件があるため，「”C1” = 100」がOR 条件の外側に抜き出されます。

(b) 例 2

■指定された探索条件

```
WHERE ("C1" < CURRENT_DATE AND "C2" = 'A')
   OR ("C1" < CURRENT_DATE AND "C3" > 20)
```

■等価変換後の探索条件

```
WHERE "C1" < CURRENT_DATE
   AND ("C2" = 'A' OR "C3" > 20)
```

[説明]

OR 条件の両側に「"C1" < CURRENT_DATE」という同じ条件があるため，「"C1" < CURRENT_DATE」がOR 条件の外側に抜き出されます。

(c) 例 3

■指定された探索条件

```
WHERE ("T1"."C1" = "T2"."C1" AND "T1"."C2" = 'A')
   OR ("T1"."C1" = "T2"."C1" AND "T1"."C3" > 20)
```

■等価変換後の探索条件

```
WHERE "T1"."C1" = "T2"."C1"
   AND ("T1"."C2" = 'A' OR "T1"."C3" > 20)
```

[説明]

OR 条件の両側に「"T1"."C1" = "T2"."C1"」という同じ条件があるため，「"T1"."C1" = "T2"."C1"」がOR 条件の外側に抜き出されます。

(d) 例 4

■指定された探索条件

```
WHERE ("C1" IS NULL AND "C2" = 'A')
   OR ("C1" IS NULL AND "C3" > 20)
```

■等価変換後の探索条件

```
WHERE "C1" IS NULL
   AND ("C2" = 'A' OR "C3" > 20)
```

[説明]

OR 条件の両側に「"C1" IS NULL」という同じ条件があるため，「"C1" IS NULL」がOR 条件の外側に抜き出されます。

(e) 例5

■指定された探索条件

```
WHERE ("C1" IN (100,200,300) AND "C2" = 'A')
   OR ("C1" IN (100,200,300) AND "C3" > 20)
```

■等価変換後の探索条件

```
WHERE "C1" IN (100,200,300)
   AND ("C2" = 'A' OR "C3" > 20)
```

[説明]

OR 条件の両側に「"C1" IN (100,200,300)」という同じ条件があるため,「"C1" IN (100,200,300)」がOR 条件の外側に抜き出されます。

(f) 例6

■指定された探索条件

```
WHERE ("C1" BETWEEN 100 AND 300 AND "C2" = 'A')
   OR ("C1" BETWEEN 100 AND 300 AND "C3" > 20)
```

■等価変換後の探索条件

```
WHERE "C1" BETWEEN 100 AND 300
   AND ("C2" = 'A' OR "C3" > 20)
```

[説明]

OR 条件の両側に「"C1" BETWEEN 100 AND 300」という同じ条件があるため,「"C1" BETWEEN 100 AND 300」がOR 条件の外側に抜き出されます。

(g) 例7

■指定された探索条件

```
WHERE "C1" = 100 OR "C1" = 100 OR "C1" = 100
```

■等価変換後の探索条件

```
WHERE "C1" = 100
```

[説明]

OR 条件中に指定した条件がすべて「"C1" = 100」のため, =条件に等価変換されます。

(h) 例 8

■指定された探索条件

```
WHERE "C1" <> 100 OR "C1" <> 100 OR "C1" <> 100
```

■等価変換後の探索条件

```
WHERE "C1" <> 100
```

[説明]

OR 条件中に指定した条件がすべて「"C1" <> 100」のため，<>条件に等価変換されます。

(i) 例 9

■指定された探索条件

```
WHERE ("C1_ARRAY"[ANY(1)] = 100 AND "C2" = 'A')
   OR ("C1_ARRAY"[ANY(1)] = 100 AND "C3" > 20)
```

■等価変換後の探索条件

```
WHERE "C1_ARRAY"[ANY(1)] = 100
   AND ("C2" = 'A' OR "C3" > 20)
```

[説明]

OR 条件の両側に「"C1_ARRAY"[ANY(1)] = 100」という同じ条件があるため，「"C1_ARRAY"[ANY(1)] = 100」がOR 条件の外側に抜き出されます。

(j) 例 10

■指定された探索条件

```
WHERE (("C1","C2") IN ((1,2),(3,4)) AND "C2" = 'A')
   OR (("C1","C2") IN ((1,2),(3,4)) AND "C3" > 20)
```

■等価変換後の探索条件

```
WHERE ("C1","C2") IN ((1,2),(3,4))
  AND ("C2" = 'A' OR "C3" > 20)
```

[説明]

OR 条件の両側に「("C1","C2") IN ((1,2),(3,4))」という同じ条件があるため，「("C1","C2") IN ((1,2),(3,4))」がOR 条件の外側に抜き出されます。

(2)　等価変換されない例

(a)　例 1

■指定された探索条件

```
WHERE ("T1"."C1" = "T2"."C1" AND "T1"."C2" = 'A')
   OR ("T2"."C1" = "T1"."C1" AND "T1"."C3" > 20)
```

[説明]

「"T1"."C1" = "T2"."C1"」と「"T2"."C1" = "T1"."C1"」は，異なる条件と見なされるため，等価変換されません。

(b)　例 2

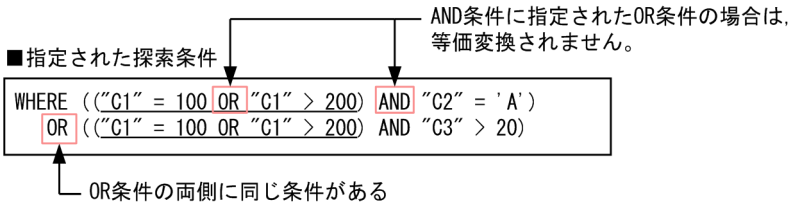
■指定された探索条件

```
WHERE NOT(("C1" = 100 AND "C2" = 'A')
       OR ("C1" = 100 AND "C3" > 20))
```

[説明]

OR 条件の両側に「"C1" = 100」という同じ条件がありますが，NOT 条件中に指定されたOR 条件の場合は，等価変換されません。

(c)　例 3

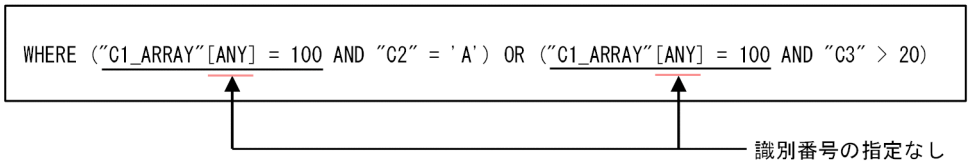


[説明]

OR 条件の両側に「"C1" = 100 OR "C1" > 200」という同じ条件がありますが，AND 条件に指定されたOR 条件の場合は，等価変換されません。

(d)　例 4

■指定された探索条件



［説明］

OR 条件の両側に「"C1_ARRAY"[ANY] = 100」という同じ条件がありますが，配列要素参照に識別番号が指定されていないため，HADB が各配列要素参照に異なる識別番号を割り振ります。そのため，指定された探索条件は同じではないと判定されて，等価変換されません。

(3)　等価変換の規則

1. 比較述語で次に示す形式の場合は，OR 条件中の条件をOR 条件の外側に抜き出します。

```
{列指定 | 列指定[ANY〔(識別番号)〕]}
   比較演算子　{定数 | 日時情報取得関数 | ユーザ情報取得関数}
```

```
{定数 | 日時情報取得関数 | ユーザ情報取得関数}
   比較演算子　{列指定 | 列指定[ANY〔(識別番号)〕]}
```

```
{列指定 | 列指定[ANY〔(識別番号)〕]} ※
   比較演算子　{列指定 | 列指定[ANY〔(識別番号)〕]} ※
```

注※

列指定，または配列値式が列指定のANY を指定した配列要素参照が左右逆に指定されている場合は，異なる条件と見なされて等価変換されません（「(2)　等価変換されない例」の「(a)　例 1」を参照）。

2. NULL 述語で次に示す形式の場合は，OR 条件中の条件をOR 条件の外側に抜き出します。

```
列指定※ IS 〔NOT〕 NULL
```

注※

列指定に配列型の列が指定されている場合は，OR 条件の外側への抜き出しは行われません。

3. 行値構成子を指定していないIN 述語で次に示す形式の場合は，OR 条件中の条件をOR 条件の外側に抜き出します。

```
{列指定 | 列指定[ANY〔(識別番号)〕]}
   〔NOT〕 IN ( {定数 | 日時情報取得関数 | ユーザ情報取得関数}
            〔, {定数 | 日時情報取得関数 | ユーザ情報取得関数}〕…)
```

4. 行値構成子を指定しているIN 述語で次に示す形式の場合は，OR 条件中の条件をOR 条件の外側に抜き出します。

```
行値構成子1 〔NOT〕 IN (行値構成子2 〔,行値構成子2〕…)
  行値構成子1::=(列指定 〔,列指定〕…)
  行値構成子2::=( {定数 | 日時情報取得関数 | ユーザ情報取得関数}
               〔, {定数 | 日時情報取得関数 | ユーザ情報取得関数}〕…)
```

5. BETWEEN 述語で次に示す形式の場合は，OR 条件中の条件をOR 条件の外側に抜き出します。

```
{列指定 | 列指定[ANY〔(識別番号)〕]}
   〔NOT〕 BETWEEN {定数 | 日時情報取得関数 | ユーザ情報取得関数}
              AND {定数 | 日時情報取得関数 | ユーザ情報取得関数}
```

6. 次に示す場合は等価変換されません。

- NOT 条件中に指定されたOR 条件の場合（「(2)　等価変換されない例」の「(b)　例 2」を参照）
- AND 条件に指定されたOR 条件の場合（「(2)　等価変換されない例」の「(c)　例 3」を参照）

5.11.3　OR 条件に関する等価変換（IN 条件への変換）

OR 条件中に同じ列に対する=条件が指定されている場合※，次に示す等価変換が実行されます。

- 同じ列に対する=条件をIN 条件に変換する
- 同じ列に対する=条件をIN 条件に変換して，OR 条件の外側に追加する

IN 条件に変換してOR 条件の外側に追加すると，検索範囲を絞り込むための条件として有効に利用できることがありますが，IN 条件を追加すれば条件評価の負荷が増えることもあります。

注※

ANY を指定した配列要素参照（識別番号は省略）が探索条件に指定されている場合は，探索条件が同じであっても等価変換は実行されません。具体例については，「(2)　等価変換されない例」の「(c)　例 3」を参照してください。

重要

- WHERE 句の探索条件，更新系 SQL のWHERE 探索条件，結合表のON 探索条件，およびHAVING 句の探索条件に指定したOR 条件が等価変換の対象になります。
- 探索条件が等価変換された場合，検索時に使用されるインデクスは等価変換後の探索条件を基に決定されます。

等価変換の例を次に示します。例中のC1，C2，C3 は列名を意味しています。

(1)　等価変換される例

(a)　例 1

■指定された探索条件

```
WHERE "C1" = 100 OR "C1" = 200 OR "C1" =300
```

等価変換

■等価変換後の探索条件

```
WHERE "C1" IN (100,200,300)
```

[説明]

OR 条件中に指定されているすべての条件が，C1 列に対する=条件のため，IN 条件に等価変換されます。

(b) 例 2

■指定された探索条件

```
WHERE ("C1" = CURRENT_DATE AND "C2" = 'A')
   OR ("C1" = ? AND "C3" > 20)
```

■等価変換後の探索条件

```
WHERE "C1" IN (CURRENT_DATE,?)
   AND (("C1" = CURRENT_DATE AND "C2" = 'A')
   OR ("C1" = ? AND "C3" > 20))
```

[説明]

OR 条件中に指定されているC1 列に対する=条件を，IN 条件に変換してOR 条件の外側に追加します。

(c) 例 3

■指定された探索条件

```
WHERE "C1_ARRAY"[ANY(1)] = 100
   OR "C1_ARRAY"[ANY(1)] = 200
   OR "C1_ARRAY"[ANY(1)] = 300
```

■等価変換後の探索条件

```
WHERE "C1_ARRAY"[ANY(1)] IN (100,200,300)
```

[説明]

OR 条件中に指定されている条件が，同じ配列要素参照に対する=条件（"C1_ARRAY"[ANY(1)] = *XXX*）のため，OR 条件がIN 条件に等価変換されます。

(2) 等価変換されない例

(a) 例 1

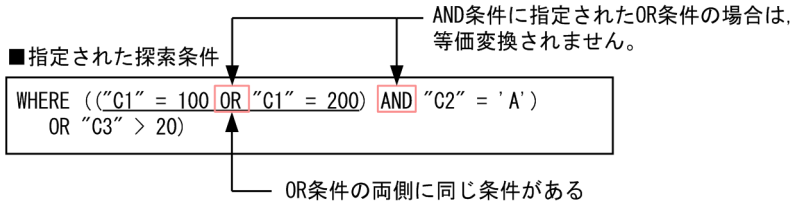
■指定された探索条件

```
WHERE NOT("C1" = 100 OR "C1" = 200 OR "C1" = 300)
```

[説明]

OR 条件中に指定されているすべての条件が，C1 列に対する=条件ですが，NOT 条件中に指定されているため，等価変換されません。

(b)　例 2



[説明]

OR 条件の両側に「"C1" = 100」と「"C1" = 200」というIN 条件に等価変換される条件がありますが，AND 条件に指定されたOR 条件の場合は等価変換されません。

(c)　例 3

■指定された探索条件

```
WHERE "C1_ARRAY"[ANY] = 100
   OR "C1_ARRAY"[ANY] = 200
   OR "C1_ARRAY"[ANY] = 300
```

[説明]

OR 条件に指定されている配列要素参照には識別番号が指定されていないため，HADB が各配列要素参照に異なる識別番号を割り振ります。そのため，指定された探索条件は同じではないと判定されて，等価変換されません。

(3)　等価変換の規則

1. 比較述語で次に示す形式の場合は，OR 条件の外側にIN 条件を追加します。ただし，列指定に指定する列が外への参照列の場合は，等価変換されません。外への参照列については，マニュアル『HADB SQL リファレンス』の『副問合せの指定形式および規則』を参照してください。

```
{列指定 | 列指定[ANY〔(識別番号)〕]}
   =  {定数 | 日時情報取得関数 | ユーザ情報取得関数 | ?パラメタ}
```

```
{定数 | 日時情報取得関数 | ユーザ情報取得関数 | ?パラメタ}
   =  {列指定 | 列指定[ANY〔(識別番号)〕]}
```

2. 次に示す場合は等価変換されません。
   - NOT 条件中に指定されたOR 条件の場合（「(2)　等価変換されない例」の「(a)　例 1」を参照）
   - AND 条件に指定されたOR 条件の場合（「(2)　等価変換されない例」の「(b)　例 2」を参照）

5.11.4　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）

指定した SQL 文が次の 2 つの条件を満たす場合，SQL 文が等価変換されることがあります。

- FROM 句にコンマ結合または結合表を指定している
- WHERE 句にOR 条件を指定している

OR 条件に指定した探索条件を，集合演算UNION ALL を指定した導出表に等価変換します。この等価変換の詳細を例を使って説明します。例中のT1，T2 は表名を意味し，C1 は列名を意味しています。

■等価変換前の SQL 文（指定された SQL 文）

```
SELECT "T1"."C1" FROM "T1","T2"
  WHERE "T1"."C1" = "T2"."C1"
    AND ("T1"."C1" = 1 OR "T2"."C1" = 2)     ←1
```

上記の SQL 文では，FROM 句にコンマ結合を指定し，かつWHERE 句にOR 条件を指定しているため（上記の例の下線部分），SQL 文が等価変換されます。

■等価変換後の SQL 文

```
SELECT "DRVTBL"."C1"
  FROM (
        SELECT "T1"."C1" FROM "T1","T2"
          WHERE "T1"."C1" = "T2"."C1"
            AND "T1"."C1" = 1                ←2
        UNION ALL
        SELECT "T1"."C1" FROM "T1","T2"
          WHERE "T1"."C1" = "T2"."C1"
            AND "T2"."C1" = 2                ←3
            AND (CASE WHEN ("T1"."C1" = 1) THEN 1 ELSE 0 END) <> 1
        )AS DRVTBL("C1")
```

集合演算項

上記の SQL 文のように，集合演算UNION ALL を指定した導出表への等価変換が行われます。等価変換前の SQL 文のOR 条件に指定した探索条件（等価変換前の SQL 文の 1 の個所）が，等価変換後の各集合演算項の問合せ指定の探索条件（等価変換後の SQL 文の 2 と 3 の個所）にそれぞれ指定されます。

■等価変換のメリット

等価変換前の SQL 文の場合，WHERE 句にOR 条件があるため，表の結合処理が行われたあとに探索条件の評価が行われます。

一方，等価変換後の SQL 文の場合，各集合演算項のWHERE 句にOR 条件がないため，探索条件の評価が行われたあとに表の結合処理が行われます。これによって，表の結合処理を行う際に必要となる入力行数を削減できることがあります（検索性能が向上することがあります）。

また，"T1"."C1"列，"T2"."C1"列にインデクスが定義されている場合は，検索時にインデクスが使用されます。検索時に使用されるインデクスは，等価変換後の探索条件を基に決定されます。

ただし，等価変換によって問合せ指定や探索条件が増えるため，場合によっては検索時間が逆に長くなるケースもあります。

(1) 等価変換の例

等価変換の例を次に示します。例中のT1，T2，T3 は表名を意味し，C1 は列名を意味しています。

（例 1）探索条件にOR 条件を 2 つ指定している場合

■等価変換前のSQL文（指定されたSQL文）

```
SELECT "T1"."C1",SUM("T2"."C1"),MIN("T3"."C1")
  FROM "T1" INNER JOIN ("T2" INNER JOIN "T3" ON "T2"."C1" = "T3"."C1") ON "T1"."C1" = "T2"."C1"
    WHERE "T1"."C1" = 1 OR "T2"."C1" = 10 OR "T3"."C1" = 100
    GROUP BY "T1"."C1"
```

等価変換

■等価変換後のSQL文

```
SELECT "DRVTBL"."C1",SUM("DRVTBL"."C2"),MIN("DRVTBL"."C3")
  FROM(
      SELECT "T1"."C1","T2"."C1","T3"."C1"
        FROM "T1" INNER JOIN ("T2" INNER JOIN "T3" ON "T2"."C1" = "T3"."C1") ON "T1"."C1" = "T2"."C1"
          WHERE "T1"."C1" = 1
      UNION ALL
      SELECT "T1"."C1","T2"."C1","T3"."C1"
        FROM "T1" INNER JOIN ("T2" INNER JOIN "T3" ON "T2"."C1" = "T3"."C1") ON "T1"."C1" = "T2"."C1"
          WHERE "T2"."C1" = 10
            AND (CASE WHEN ("T1"."C1" = 1) THEN 1 ELSE 0 END) <> 1
      UNION ALL
      SELECT "T1"."C1","T2"."C1","T3"."C1"
        FROM "T1" INNER JOIN ("T2" INNER JOIN "T3" ON "T2"."C1" = "T3"."C1") ON "T1"."C1" = "T2"."C1"
          WHERE "T3"."C1" = 100
            AND (CASE WHEN ("T1"."C1" = 1 OR "T2"."C1" = 10) THEN 1 ELSE 0 END) <> 1
      )AS DRVTBL("C1","C2","C3")
  GROUP BY "DRVTBL"."C1"
```

集合演算項

集合演算項

集合演算項

[説明]

- 等価変換前の SQL 文では，FROM 句に結合表を指定し，かつWHERE 句にOR 条件を指定しているため（上記の例の下線部分），SQL 文が等価変換されます。
- 等価変換前の SQL 文のOR 条件に指定した各探索条件が，等価変換後の各集合演算項の問合せ指定の探索条件にそれぞれ指定されます。等価変換前の SQL 文のようにOR 条件を 2 つ指定している場合は，等価変換後の集合演算項が 3 つになります。

（例 2）探索条件にAND 条件とOR 条件を指定している場合

■等価変換前のSQL文（指定されたSQL文）

```
SELECT "T1"."C1",SUM("T2"."C2") FROM "T1","T2"
  WHERE "T1"."C1" = "T2"."C1" AND ("T1"."C2" = 1 OR "T2"."C2" = 10)
  GROUP BY "T1"."C1"
```

等価変換

■等価変換後のSQL文

```
SELECT "DRVTBL"."C1",SUM("DRVTBL"."C2")
  FROM (
      SELECT "T1"."C1","T2"."C2" FROM "T1","T2"
        WHERE "T1"."C1" = "T2"."C1"
          AND "T1"."C2" = 1
      UNION ALL
      SELECT "T1"."C1","T2"."C2" FROM "T1","T2"
        WHERE "T1"."C1" = "T2"."C1"
          AND "T2"."C2" = 10
          AND (CASE WHEN ("T1"."C2" = 1) THEN 1 ELSE 0 END) <> 1
      )AS DRVTBL("C1","C2")
  GROUP BY "DRVTBL"."C1"
```

集合演算項

集合演算項

［説明］

- 等価変換前の SQL 文では，FROM 句にコンマ結合を指定し，かつWHERE 句にOR 条件を指定しているため（上記の例の下線部分），SQL 文が等価変換されます。
- 等価変換前の SQL 文のOR 条件に指定した各探索条件が，等価変換後の各集合演算項の問合せ指定の探索条件にそれぞれ指定されます。等価変換前の SQL 文のAND 条件の左側に指定した探索条件は，等価変換後のすべての集合演算項の問合せ指定の探索条件に指定されます。

(2) 等価変換の適用条件

次の条件をすべて満たす場合に，等価変換が行われます。

■サーバ定義またはクライアント定義についての条件

次の条件をすべて満たす必要があります。

- サーバ定義またはクライアント定義のadb_sql_exe_hashtbl_area_size オペランドに0 を指定していないこと。
- サーバ定義のadb_sys_uthd_num オペランドに0 を指定していないこと。
- サーバ定義またはクライアント定義のadb_sql_exe_max_rthd_num オペランドに0 を指定していないこと。

■SQL 文についての条件

次の条件をすべて満たす必要があります。

- 検索系 SQL であること。
- SQL 文中に指定しているどれかの表に対して，バージョン 04-03 以降の HADB サーバでコスト情報を収集していること。
- SQL 文中に外への参照列を含む問合せ指定がないこと。

■問合せ指定についての条件

次の条件をすべて満たす必要があります。

1. 選択式についての条件
   - 選択式にROW が指定されていないこと。
2. FROM 句に指定する表参照についての条件
   - 選択式に指定した表数が 16 以下であること（ただし，集まり導出表は表数に含めない）。
   - FROM 句にコンマ結合が指定されていること。

     コンマ結合だけを指定している場合，またはコンマ結合と結合表の両方を混在して指定している場合に適用条件を満たします。

     また，指定できる表の種別を次に示します。次の表の種別が混在して指定されていても適用条件を満たします。

     ・実表（ただし，アーカイブマルチチャンク表が指定されている場合は，等価変換されません）

     ・結合表（ただし，FULL OUTER JOIN が指定されている場合は，等価変換されません）

・導出表（表副問合せ，問合せ名，およびビュー表が該当します）

・集まり導出表

■結合条件についての条件

次の条件をすべて満たす必要があります。

- 結合表の結合指定中にOR 条件が指定されていないこと。
- OR 条件中に指定した列が属する表，またはその表に定義されている配列型の列を指定した集まり導出表の結合条件が，次の条件を満たしていること。
  - OR 条件中に指定した列が属する表，またはその表に定義されている配列型の列を指定した集まり導出表の間に，「*列指定＝列指定*」の形式の結合条件が 1 つ以上存在すること。
- 集合演算UNION ALL を指定した導出表への等価変換を行う際，各集合演算項の問合せ指定の探索条件に，結合対象のすべての表に対する結合条件が存在すること。

■WHERE 句に指定する探索条件についての条件

次の条件をすべて満たす必要があります。

1. 論理演算子AND に指定する条件に，OR 条件を指定した条件が複数個指定されていないこと。

   （例）

   - 適用条件を満たす例

     ```
     条件1 AND 条件2 AND（条件3 OR 条件4 OR 条件5）
     ```

     OR 条件を指定した条件が 1 つだけ指定されているため，適用条件を満たします。

   - 適用条件を満たさない例

     ```
     （条件1 OR 条件2）AND 条件3 AND（条件4 OR 条件5）
     ```

     OR 条件を指定した条件が 2 つ指定されているため，適用条件を満たしません。

   - OR 条件がネストして複数個指定されている場合も適用条件を満たします。
   - OR 条件以外の連続した探索条件中に指定する表の数は考慮しません（結合指定でなくてもよい）。
   - NOT が指定されたOR 条件は適用条件を満たしません。

2. OR 条件中に結合表の列が指定されていないこと。

   ただし，結合表がコンマ結合に変換された場合は，適用条件を満たします。

3. OR 条件中に指定されている列の表に配列型の列が定義されている場合，その配列型の列が次のどれかに指定されていないこと。

   - 選択式の値式
   - ソート指定リストのソートキー
   - GROUP BY 句のグループ化指定
   - HAVING 句の探索条件
   - 次のどれかを含む探索条件中

・OR 条件中に指定された列の表以外の表の列を指定している探索条件
・スカラ関数RANDOM またはRANDOM_NORMAL を指定している探索条件
・副問合せを指定している探索条件

- 次のどれかの探索条件を含むNOT の中に指定された探索条件中
  ・OR 条件中に指定された列の表以外の表の列を指定している探索条件
  ・スカラ関数RANDOM またはRANDOM_NORMAL を指定している探索条件
  ・副問合せを指定している探索条件

4. OR 条件に副問合せが指定されていないこと。
5. OR 条件中に行値構成子を指定した探索条件が指定されていないこと。
6. OR 条件にスカラ関数RANDOM またはRANDOM_NORMAL が指定されていないこと。
7. 論理演算子OR の数が 15 以下であること。
8. OR 条件に 2 表以上に対する探索条件が含まれていること。
9. 等価変換後の集合演算項の各探索条件中に，次のどれかの形式の探索条件が 1 つ以上含まれていること。また，これらの条件に対して論理演算子OR やNOT が指定されていないこと。

- 比較述語

```
{列指定 | 列指定[ANY〔(識別番号)〕]※}
   比較演算子  {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
```

```
{定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
   比較演算子  {列指定 | 列指定[ANY〔(識別番号)〕]※}
```

- BETWEEN 述語

```
{列指定 | 列指定[ANY〔(識別番号)〕]※}
   BETWEEN  {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
       AND  {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
```

- IN 述語

```
{列指定 | 列指定[ANY〔(識別番号)〕]※}
   IN ( {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
     〔, {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}〕…)
```

- LIKE 述語

```
{列指定 | 列指定[ANY〔(識別番号)〕]※}
   LIKE パターン文字列  〔ESCAPE エスケープ文字〕
   パターン文字列::= {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
   エスケープ文字::= {定数 | ?パラメタ}
```

- LIKE_REGEX 述語

```
{列指定 | 列指定[ANY〔(識別番号)〕]※}
   LIKE_REGEX 正規表現文字列  〔FLAG  {I | IGNORECASE}〕
   正規表現文字列::=定数
```

- NULL 述語

  {*列指定* | *列指定*[ANY〔(*識別番号*)〕]※} IS NULL

注※

WHERE 句中にこの配列要素参照と同じ識別番号を持つ配列要素参照が指定されている場合，そのSQL 文は等価変換されません。

> **メモ**
>
> スカラ演算の中に定数だけを指定している場合，そのスカラ演算は定数として扱われることがあります。定数と等価なスカラ演算については，マニュアル『HADB SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。

■等価変換が適用される問合せ指定の指定個所についての条件

問合せ指定が次の個所に指定された場合に，等価変換を適用するかどうかのチェックを HADB サーバが行います。

- 問合せ式中の問合せ式本体
- スカラ副問合せ
- 表副問合せ
- 集合演算の各集合演算項

  SQL 文が等価変換されてUNION ALL を指定した導出表が作成された結果，集合演算の個数の上限を超えた場合でも，SQL 文の等価変換が行われます。
- WITH リスト要素

  再帰的メンバの問合せ指定に対しては，等価変換を適用しません。
- CREATE VIEW 文中の問合せ式

5.11.5 スカラ演算に関する等価変換

探索条件の片方の項に，列指定※1 を含むスカラ演算が指定されている場合，スカラ演算の部分が移項されます（片方の項が列指定※1 だけになるように探索条件が等価変換されます)。次の条件をすべて満たす場合に，スカラ演算の移項が行われます。

- 演算項が次のどちらかの条件を満たしている
  - 演算項が「列指定※1 と定数」の四則演算（加減算だけ）である
  - 演算項が「列指定※1 とラベル付き間隔」の日時演算である
- 列指定の列のデータ型※2 が，SMALLINT，INTEGER，TIME，DATE，またはTIMESTAMP のどれかである
- スカラ演算が入れ子になっていない

注※1

列指定のほかに，配列値式が列指定でANY を指定した配列要素参照も該当します。配列要素参照の場合も列指定のときと同様に，スカラ演算の部分が移項されて，片方の項が配列要素参照だけになるように探索条件が等価変換されます。

注※2

配列要素参照の場合は，配列値式の結果のデータ型となります。

重要

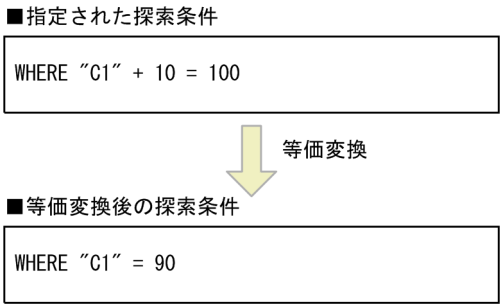
探索条件が等価変換された場合，検索時に使用されるインデクスは等価変換後の探索条件を基に決定されます。

等価変換の例を次に示します。例中のC1 は列名を意味しています。

なお，以降では，列指定のときの等価変換の例を記載しています。配列要素参照の場合は，列指定を配列要素参照に置き換えてください。

(1) 等価変換される例

(a) 例 1



[説明]

列指定を含む項に「+10」というスカラ演算があります。このスカラ演算が右辺に移項され，列指定だけの条件に等価変換されます。等価変換後，「列指定 比較演算子 定数」の形式になるため，検索時にインデクスが使用されます。

(b) 例 2

■指定された探索条件

```
WHERE "C1" + 1 DAY = DATE'2016-01-01'
```

■等価変換後の探索条件

```
WHERE "C1" = DATE'2015-12-31'
```

[説明]

列指定を含む項に「+1 DAY」というスカラ演算があります。このスカラ演算が右辺に移項され，列指定だけの条件に等価変換されます。等価変換後，「列指定 比較演算子 定数」の形式になるため，検索時にインデクスが使用されます。

(c) 例 3

■指定された探索条件

```
WHERE "C1" + 10 BETWEEN 50 AND 100
```

■等価変換後の探索条件

```
WHERE "C1" BETWEEN 40 AND 90
```

[説明]

列指定を含む項に「+10」というスカラ演算があります。このスカラ演算が右辺に移項され，列指定だけの条件に等価変換されます。等価変換後，「列指定 BETWEEN 定数」の形式になるため，検索時にインデクスが使用されます。

(d) 例 4

■指定された探索条件

```
WHERE "C1" + 10 IN(100,200,300)
```

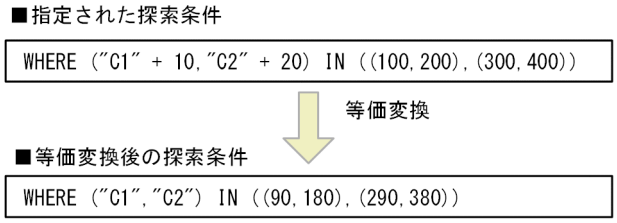
■等価変換後の探索条件

```
WHERE "C1" IN(90,190,290)
```

[説明]

列指定を含む項に「+10」というスカラ演算があります。このスカラ演算が右辺に移項され，列指定だけの条件に等価変換されます。等価変換後，「列指定 IN 定数」の形式になるため，検索時にインデクスが使用されます。

(e) 例 5

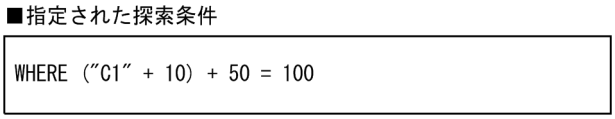


[説明]

列指定を含む行値構成子要素に「+10」および「+20」というスカラ演算があります。このスカラ演算が IN 述語の右側の対応する行値構成子要素に移項され，列指定だけの条件に等価変換されます。等価変換後，「列指定 IN 定数」の形式になるため，検索時にインデクスが使用されます。

(2) 等価変換されない例

(a) 例 1



[説明]

列指定を含むスカラ演算が入れ子になっているため，等価変換されません。

(3) 等価変換の規則

1. 比較述語で次に示す形式の場合は，スカラ演算の部分が移項され，列指定，または配列値式が列指定の ANY を指定した配列要素参照だけの条件に等価変換されます。

```
{列指定 | 列指定[ANY 〔(識別番号)〕]}   {+ | -}  定数
   比較演算子 定数
```

```
定数
    比較演算子  {列指定 | 列指定[ANY 〔(識別番号)〕]}   {+ | -}  定数
```

なお，次に示すすべての条件を満たす場合は，等価変換されません。

- 列指定，または配列値式が列指定のANY を指定した配列要素参照を含むスカラ演算に，ラベル付き間隔のYEAR またはMONTH が指定されている
- 比較述語の比較演算子に，<>，!=，^=のどれかが指定されている

2. BETWEEN 述語で次に示す形式の場合は，スカラ演算の部分が移項され，列指定，または配列値式が列指定のANY を指定した配列要素参照だけの条件に等価変換されます。

```
{列指定 | 列指定[ANY〔(識別番号)〕]}  {+ | -}  定数
    〔NOT〕 BETWEEN 定数 AND 定数
```

なお，次に示すすべての条件を満たす場合は，等価変換されません。

- 列指定，または配列値式が列指定のANY を指定した配列要素参照を含むスカラ演算に，ラベル付き間隔のYEAR またはMONTH が指定されている
- BETWEEN 述語でNOT BETWEEN が指定されている

3. 行値構成子を指定していないIN 述語で次に示す形式の場合は，スカラ演算の部分が移項され，列指定，または配列値式が列指定のANY を指定した配列要素参照だけの条件に等価変換されます。

```
{列指定 | 列指定[ANY〔(識別番号)〕]}  {+ | -}  定数
    〔NOT〕 IN (定数,…)
```

なお，列指定，または配列値式が列指定のANY を指定した配列要素参照を含むスカラ演算に，ラベル付き間隔のYEAR またはMONTH が指定されている場合は，等価変換されません。

4. 行値構成子を指定しているIN 述語で次に示す形式の場合は，スカラ演算の部分が移項され，列指定だけの条件に等価変換されます。

```
行値構成子1 〔NOT〕 IN (行値構成子2 〔,行値構成子2〕…)
  行値構成子1::=(列指定 {+ | -} 定数 〔,列指定 {+ | -} 定数〕…)
  行値構成子2::=(定数 〔,定数〕…)
```

なお，IN 述語の左側の行値構成子 1 に指定した列指定に，ラベル付き間隔のYEAR またはMONTH が指定されている場合は，等価変換されません。

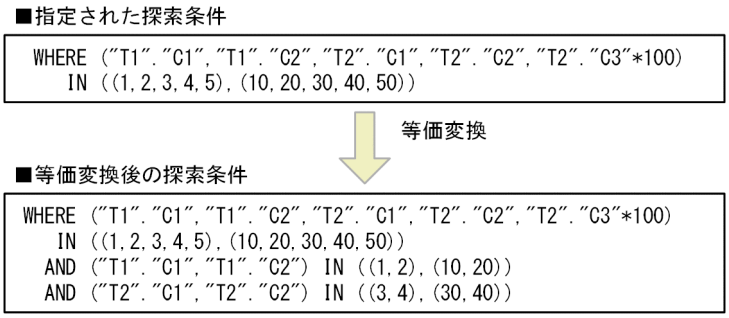
5.11.6 行値構成子を指定した IN 述語に関する等価変換（同じ表の列指定の抽出）

IN 述語の左側の行値構成子に，複数の表の列指定がある場合や，列指定とスカラ演算を含む指定が混在している場合，同じ表の列指定だけで構成されるIN 述語をAND 条件で追加する等価変換が行われます。ただし，論理演算子OR やNOT に指定されたIN 述語は，等価変換されません。

この等価変換によって，表を結合する前に追加したIN 述語を評価できるようになり，表の結合処理への入力行を削減できることがあります。また，表の検索時にインデクスが使用されることがあります。

等価変換の例を次に示します。例中のC1，C2，C3 は列名を意味しています。

(1) 例 1



[説明]

IN 述語の左側に，複数の表の列指定とスカラ演算が指定されている行値構成子があります。この行値構成子から同じ表の列指定が抽出され，IN 述語の条件が追加されます。等価変換後，追加されたIN 述語の条件で，探索時にインデクスが使用されることがあります。

(2) 等価変換される条件の形式

等価変換される条件の形式を次に示します。

- IN 述語

  行値構成子 IN（*行値構成子* 〔,*行値構成子*〕…）

  IN 述語の左側の行値構成子が，次の条件を満たす場合に等価変換されます。

  - 行値構成子要素に列指定が含まれている
  - 行値構成子要素がすべて列指定の場合，異なる表の列指定が含まれている

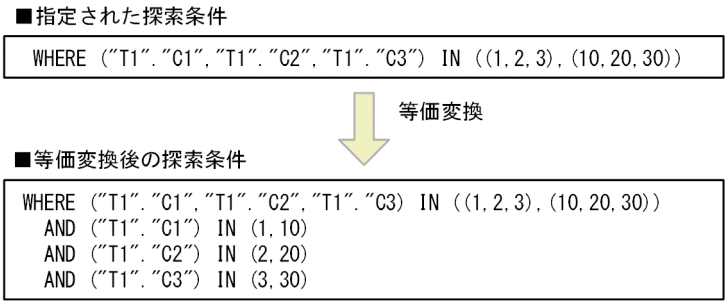
5.11.7 行値構成子を指定した IN 述語に関する等価変換（行値構成子要素ごとの条件の追加）

IN 述語の左側の行値構成子の行値構成子要素が同じ表の列指定だけの場合，各行値構成子要素に対するIN 述語をAND 条件で追加する等価変換が行われます。ただし，論理演算子OR やNOT に指定されたIN 述語は，等価変換されません。なお，この等価変換で追加されるIN 述語の上限は，SQL 文全体で 255 個です。255 個を超えた探索条件は追加されません。

この等価変換によって追加されたIN 述語で，表の検索時にインデクスが使用されることがあります。

等価変換の例を次に示します。例中のC1，C2，C3 は列名を意味しています。

(1) 例 1



[説明]

IN 述語の左側に，同じ表の複数の列指定が指定されている行値構成子があります。この行値構成子から同じ列指定が抽出され，行値構成子要素ごとのIN 述語の条件が追加されます。等価変換後，追加されたIN 述語の条件で，探索時にインデクスが使用されることがあります。

(2) 等価変換される条件の形式

等価変換される条件の形式を次に示します。

- IN 述語

  行値構成子 IN（*行値構成子* 〔,*行値構成子*〕…）

  IN 述語の左側の行値構成子が，次の条件を満たす場合に等価変換されます。

  - 行値構成子要素がすべて列指定である
  - 行値構成子要素の列指定がすべて同じ表の列指定である

5.11.8 IN 述語に関する等価変換

IN 条件の比較値の指定が 1 つだけの場合，指定した探索条件が等価変換されます。探索条件が等価変換された場合，検索時に使用されるインデクスは等価変換後の探索条件を基に決定されます。

等価変換の例を次に示します。例中のC1 は列名を意味しています。

(1) 例 1

■指定された探索条件

```
WHERE "C1" IN (100)
```

■等価変換後の探索条件

```
WHERE "C1" = 100
```

[説明]

IN 条件の比較値の指定が 1 つだけの場合，=条件に等価変換されます。

(2) 例 2

■指定された探索条件

```
WHERE "C1" NOT IN (100)
```

■等価変換後の探索条件

```
WHERE "C1" <> 100
```

[説明]

NOT IN 条件の比較値の指定が 1 つだけの場合，<>条件に等価変換されます。

(3) 例 3

■指定された探索条件

```
WHERE ("C1","C2") IN ((100,200))
```

■等価変換後の探索条件

```
WHERE ("C1","C2") = (100,200)
```

[説明]

IN 条件の比較値として行値構成子の指定が 1 つだけの場合，=条件に等価変換されます。

5.11.9　HAVING 句に関する等価変換（WHERE 句への変換）

HAVING 句の探索条件が，WHERE 句の探索条件に等価変換されることがあります。WHERE 句の探索条件に等価変換されると，グループ化の処理でむだな入力行を削減できたり，表の検索にインデクスが利用できたりすることがあります。

WHERE 句の探索条件に等価変換される条件の形式を，次に示します。探索条件が等価変換された場合，検索時に使用されるインデクスは等価変換後の探索条件を基に決定されます。

等価変換される条件の形式

- 比較述語

```
・列指定 比較演算子 {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
・{定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数} 比較演算子 列指定
```

- BETWEEN 述語

```
列指定 〔NOT〕 BETWEEN {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
                   AND {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
```

- IN 述語

```
列指定 〔NOT〕 IN ( {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
                  〔, {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数} 〕…)
```

- LIKE 述語

```
列指定 〔NOT〕 LIKE パターン文字列 〔ESCAPE エスケープ文字〕
  パターン文字列::= {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
  エスケープ文字::= {定数 | ?パラメタ}
```

- LIKE_REGEX 述語

```
列指定 〔NOT〕 LIKE_REGEX 正規表現文字列 〔FLAG {I | IGNORECASE} 〕
  正規表現文字列::=定数
```

- NULL 述語

```
列指定 IS 〔NOT〕 NULL
```

留意事項

- 論理演算のOR 条件の中に指定された条件の場合は，この等価変換の対象になりません。

  ただし，この論理演算のOR 条件に対して，次に示す等価変換が適用される場合は，次に示す等価変換が適用されたあとに，HAVING 句に関する等価変換が適用されます。

  ・OR 条件の中の条件をOR 条件の外側に抜き出す等価変換

  ・OR 条件の中の=条件から作成したIN 条件をOR の外側に追加する等価変換

  OR 条件に関する等価変換については，「5.11.2　OR 条件に関する等価変換（OR 条件の外側への抜き出し）」および「5.11.3　OR 条件に関する等価変換（IN 条件への変換）」を参照してください。

- 論理演算のNOT の中に指定された条件，および副問合せを含む条件の場合は，この等価変換の対象になりません。
- 列指定の列が外への参照列の場合，この等価変換の対象になりません。外への参照列については，マニュアル『HADB　SQL リファレンス』の『副問合せの指定形式および規則』を参照してください。
- スカラ演算の中に定数だけを指定している場合，そのスカラ演算は定数として扱われることがあります。定数と等価なスカラ演算については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。

5.11.10　導出問合せを指定した SQL 文の探索条件に関する等価変換（導出問合せの WHERE 句への移動）

導出問合せを指定した SQL 文のWHERE 句に指定された探索条件を，導出問合せのWHERE 句に移動する等価変換を行います。導出問合せについては，マニュアル『HADB　SQL リファレンス』の『導出問合せおよび導出問合せ名』を参照してください。

等価変換の例を次に示します。例中のC1，C2，C3 は列名を意味しています。

(1)　等価変換される例

(a)　例 1（導出問合せが問合せ指定の場合）

■指定されたSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT DISTINCT "C1","C2","C3"
          FROM "T1") "D1"("C1","C2","C3")
    WHERE "D1"."C1"<10
```

等価変換

■等価変換後のSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT DISTINCT "C1","C2","C3"
          FROM "T1" WHERE "T1"."C1"<10) "D1"("C1","C2","C3")
```

[説明]

導出問合せを指定した SQL 文のWHERE 句に指定された条件左側の列指定は，導出問合せに指定された問合せ指定の選択式（"T1"."C1"）を基に導出された導出列（"D1"."C1"）です。この条件は，等価変換の適用条件を満たしているため，導出問合せを指定した SQL 文の探索条件に関する等価変換が適用されます。

(b) 例 2（導出問合せが問合せ式の場合）

■指定されたSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT "C1","C2","C3" FROM "T1"
        UNION DISTINCT
        SELECT "C1","C2","C3" FROM "T2") "D1"("C1","C2","C3")
    WHERE "D1"."C1"<10
```

■等価変換後のSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT "C1","C2","C3" FROM "T1"
          WHERE "T1"."C1"<10
        UNION DISTINCT
        SELECT "C1","C2","C3" FROM "T2"
          WHERE "T2"."C1"<10) "D1"("C1","C2","C3")
```

[説明]

導出問合せを指定した SQL 文のWHERE 句に指定された条件左側の列指定は，導出問合せに指定された問合せ式の選択式（”T1”.”C1”，”T2”.”C1”）を基に導出された導出列（”D1”.”C1”）です。この条件は，等価変換の適用条件を満たしているため，導出問合せを指定した SQL 文の探索条件に関する等価変換が適用されます。

(c) 例 3（導出問合せが問合せ指定の場合）

■指定されたSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT DISTINCT "C1","C2","C3"
          FROM "T1") "D1"("C1","C2","C3")
    WHERE "D1"."C1"<10 OR ("D1"."C2">100 AND "D1"."C3"=200)
```

■等価変換後のSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT DISTINCT "C1","C2","C3" FROM "T1"
          WHERE "T1"."C1"<10 OR ("T1"."C2">100 AND "T1"."C3"=200)
        ) "D1"("C1","C2","C3")
```

[説明]

導出問合せを指定した SQL 文のWHERE 句に指定された論理演算のOR 条件に含まれる列指定は，導出問合せに指定された問合せ指定の選択式（”T1”.”C1”，”T1”.”C2”，”T1”.”C3”）を基に導出された導出列（”D1”.”C1”，”D1”.”C2”，”D1”.”C3”）です。この論理演算のOR 条件は，等価変換の適用条件を満たしているため，導出問合せを指定した SQL 文の探索条件に関する等価変換が適用されます。

(d) 例 4（導出問合せが問合せ式の場合）

■指定されたSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT "C1","C2","C3" FROM "T1"
        UNION DISTINCT
        SELECT "C1","C2","C3" FROM "T2"
        ) "D1"("C1","C2","C3")
    WHERE "D1"."C1"<10 OR ("D1"."C2">100 AND "D1"."C3"=200)
```

■等価変換後のSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT "C1","C2","C3" FROM "T1"
          WHERE "T1"."C1"<10 OR ("T1"."C2">100 AND "T1"."C3"=200)
        UNION DISTINCT
        SELECT "C1","C2","C3" FROM "T2"
          WHERE "T2"."C1"<10 OR ("T2"."C2">100 AND "T2"."C3"=200)
        ) "D1"("C1","C2","C3")
```

[説明]

導出問合せを指定した SQL 文のWHERE 句に指定された論理演算のOR 条件に含まれる列指定は，導出問合せに指定された問合せ式の選択式（"T1"."C1"，"T1"."C2"，"T1"."C3"，"T2"."C1"，"T2"."C2"，"T2"."C3"）を基に導出された導出列（"D1"."C1"，"D1"."C2"，"D1"."C3"）です。この論理演算のOR 条件は，等価変換の適用条件を満たしているため，導出問合せを指定した SQL 文の探索条件に関する等価変換が適用されます。

(e) 例 5（導出問合せが問合せ指定の場合）

■指定されたSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT DISTINCT "C1","C2","C3" FROM "T1"
        ) "D1"("C1","C2","C3")
    WHERE ("D1"."C1","D1"."C2") IN ((1,2),(3,4))
```

■等価変換後のSQL文

```
SELECT "C1","C2","C3"+10
  FROM (SELECT DISTINCT "C1","C2","C3" FROM "T1"
                WHERE ("T1"."C1","T1"."C2") IN ((1,2),(3,4))
        ) "D1"("C1","C2","C3")
```

[説明]

導出問合せを指定した SQL 文のWHERE 句に指定されたIN 述語の左側の行値構成子の列指定は，導出問合せに指定された問合せ指定の選択式（"T1"."C1"，"T1"."C2"）を基に導出された導出列（"D1"."C1"，"D1"."C2"）です。この条件は，等価変換の適用条件を満たしているため，導出問合せを指定した SQL 文の探索条件に関する等価変換が適用されます。

(2) 等価変換される条件の形式

等価変換される条件の形式を次に示します。

- 比較述語の場合

```
・列指定 比較演算子 {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
・ {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数} 比較演算子 列指定
・CONTAINS(列指定,検索条件式文字列) > 0
```

- BETWEEN 述語の場合

```
列指定 〔NOT〕 BETWEEN {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
                   AND {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
```

- 行値構成子を指定していないIN 述語の場合

```
列指定 〔NOT〕 IN (
       {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数} }
    〔, {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数} 〕…)
```

- 行値構成子を指定しているIN 述語の場合

```
行値構成子1 〔NOT〕 IN (行値構成子2 〔,行値構成子2〕…)
  行値構成子1::=(列指定 〔,列指定〕…)
  行値構成子2::=( {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
                〔, {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数} 〕…)
```

- LIKE 述語の場合

```
列指定 〔NOT〕 LIKE パターン文字列 〔ESCAPE エスケープ文字〕
  パターン文字列::= {定数 | ?パラメタ | 日時情報取得関数 | ユーザ情報取得関数}
  エスケープ文字::= {定数 | ?パラメタ}
```

- LIKE_REGEX 述語の場合

```
列指定 〔NOT〕 LIKE_REGEX 正規表現文字列 〔FLAG {I | IGNORECASE} 〕
  正規表現文字列::=定数
```

- NULL 述語の場合

```
列指定 IS 〔NOT〕 NULL
```

列指定に配列型の列が指定されている場合，等価変換は行われません。

(3) 留意事項

- 探索条件の列指定に指定された導出列の基となる導出問合せの選択式が，列指定の場合に等価変換が行われます。
- 導出問合せにウィンドウ関数が指定されている場合，等価変換は行われません。
- 導出問合せが導出問合せを指定した SQL 文に展開された場合，等価変換は行われません。導出問合せの展開については，マニュアル『HADB SQL リファレンス』の『内部導出表』を参照してください。
- 次のような導出問合せの場合，等価変換は行われません。
  - 導出問合せとしてビュー表名を指定し，同じビュー表名を SQL 文中に複数指定した場合

- 導出問合せとしてWITH リスト要素に指定した問合せ名を指定し，同じ問合せ名を SQL 文中に複数指定した場合
- WITH リスト要素が複数指定されていて，導出問合せとしてWITH リスト要素に指定した問合せ名を指定した場合
- INNER JOIN だけを指定した結合表以外の結合表中に，導出問合せを指定した場合
- 導出問合せとして表値構成子を指定した場合
- 導出問合せが再帰的問合せの場合

- 次に示すすべての条件を満たす場合に，導出問合せを指定した SQL 文の探索条件に関する等価変換が適用されます。
  - 論理演算のOR 条件中に指定された条件が，導出問合せを指定した SQL 文の探索条件に関する等価変換の対象となる探索条件の形式を満たしている
  - すべての列指定が同一の導出表の構成列である
- 論理演算のOR 条件に対して，次に示す等価変換が適用される場合は，次に示す等価変換が適用されたあとに，導出問合せを指定した SQL 文の探索条件に関する等価変換が適用されます。
  - OR 条件の中の条件をOR 条件の外側に抜き出す等価変換
  - OR 条件の中の=条件から作成したIN 条件をOR の外側に追加する等価変換

  上記の等価変換については，「5.11.2　OR 条件に関する等価変換（OR 条件の外側への抜き出し）」，および「5.11.3　OR 条件に関する等価変換（IN 条件への変換）」を参照してください。
- 論理演算のNOT 条件中に指定された条件，および副問合せを含む条件は，等価変換の対象になりません。
- 列指定に指定する列が外への参照列の場合，等価変換は行われません。外への参照列については，マニュアル『HADB　SQL リファレンス』の『副問合せの指定形式および規則』を参照してください。
- スカラ演算中に定数だけを指定している場合，そのスカラ演算は定数として扱われることがあります。定数と等価なスカラ演算については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』を参照してください。

5.12　アーカイブマルチチャンク表を検索する際の考慮点

アーカイブマルチチャンク表を検索する際の考慮点について説明します。

ここでの説明は，アーカイブレンジ列にDATE 型の日時データが格納されている場合の例を使用しています。

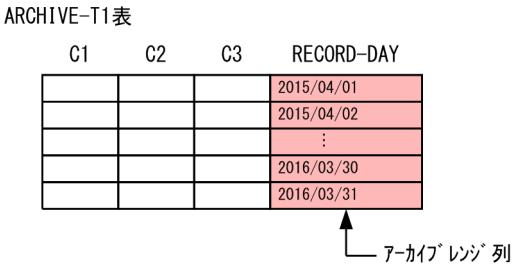
> **メモ**
>
> ここの説明は，マニュアル『HADB　システム構築・運用ガイド』の『チャンクアーカイブ機能（チャンク内のデータの圧縮)』をお読みいただいていることを前提としています。

5.12.1　アーカイブマルチチャンク表を検索する際のポイント

アーカイブマルチチャンク表を検索する際のポイントを例を使って説明します。

例中で使用するアーカイブマルチチャンク表の定義と，データのアーカイブ状態は次のとおりとします。

■アーカイブマルチチャンク表の定義



■データのアーカイブ状態

- データベースには，2015 年 4 月～2016 年 3 月までのデータが格納されています。
- 2015 年 4 月～12 月までのデータは，アーカイブされています。
- 2016 年 1 月～3 月までのデータは，アーカイブされていません。

(1)　検索時の基本的な考え方

アーカイブマルチチャンク表を検索する際は，アーカイブレンジ列の日時情報を探索条件に指定して，検索範囲を絞り込むようにしてください。

また，検索対象のデータがアーカイブされているかどうかを意識するようにしてください。アーカイブされているデータを検索する場合，検索処理時間が長くなることがあります。

(2) 探索条件の指定

アーカイブマルチチャンク表を検索する際の，探索条件の指定に関するポイントを次に示します。

- WHERE 句の探索条件にアーカイブレンジ列を指定した条件を必ず指定し，検索範囲を絞り込むようにしてください。

  （例）

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2016/02/01' AND DATE'2016/02/29'
```

  上記の下線部分の指定（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）を必ずしてください。

  なお，指定できる述語などに制限があります。詳細については，「5.12.2　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。

- AND 条件を追加して，検索範囲をさらに絞り込むようにしてください。

  （例）

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2016/02/01' AND DATE'2016/02/29'
    AND "C1"='P001'
    AND "C2"=100
```

- 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

```
WHERE "RECORD-DAY" BETWEEN DATE'2016/01/01' AND DATE'2016/03/31'
WHERE "RECORD-DAY" >= DATE'2016/02/01'
```

(3) アーカイブされていないデータを検索する場合

アーカイブされていないデータを検索する場合（この例では 2016 年 1 月以降のデータを検索する場合），探索条件には，アーカイブされていないデータだけを検索範囲とする条件を指定します。

（例）

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" >= DATE'2016/01/01'
```

```
    AND "C1"='P001'
    AND "C2"=100
```

この例の場合，2016/01/01 以降のデータだけを検索範囲とする探索条件を指定しています。

(4) アーカイブされているデータを検索する場合

アーカイブされているデータを検索する場合（この例では 2015 年 4 月～2015 年 12 月のデータを検索する場合），探索条件に指定するアーカイブレンジ列の日時情報の範囲をできる限り狭くしてください。

（例）

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2015/10/01' AND DATE'2015/10/05'
    AND "C1"='P001'
    AND "C2"=100
```

上記の下線部分の指定によって，検索範囲をできる限り絞り込み，読み込み対象のアーカイブファイルを減らします。読み込み対象のアーカイブファイルが増えると，それに比例して検索時間が長くなります。

5.12.2 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

アーカイブマルチチャンク表を検索する場合は，アーカイブレンジ列の日時情報を使用して検索範囲の絞り込みが行われるように探索条件を指定する必要があります。

重要

ここで説明している規則に従っていない場合，アーカイブされている全データが検索対象になるため，検索処理時間が長くなることがあります。アーカイブされている全データが検索対象になった場合，KFAA51121-W メッセージが出力されます。この場合，SQL 文を修正して，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してください。

(1) 探索条件の指定規則

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が行われる条件を次に示します。

- WHERE 句の探索条件にアーカイブレンジ列を指定した条件を指定している
- アーカイブレンジ列を指定した探索条件に，比較述語，IN 述語，またはBETWEEN 述語だけを指定している
- 比較述語，IN 述語，またはBETWEEN 述語の指定が，「(2) 比較述語の指定規則」以降で説明している条件を満たしている
- アーカイブレンジ列を指定した探索条件にNOT 条件を指定していない
- アーカイブレンジ列を指定した探索条件にOR 条件を指定していない（OR 条件の両辺にアーカイブレンジ列を指定した探索条件を指定していない）

・絞り込みが行われない例

```
SELECT * FROM "ARCHIVE-T1"
  WHERE "RECORD-DAY" BETWEEN DATE'2016/01/01' AND DATE'2016/01/10'
     OR "RECORD-DAY" BETWEEN DATE'2016/02/01' AND DATE'2016/02/10'
```

・絞り込みが行われる例

```
SELECT * FROM "ARCHIVE-T1"
  WHERE "RECORD-DAY" BETWEEN DATE'2016/01/01' AND DATE'2016/01/10'
  AND ("C1"='P001' OR "C2"='S002')
```

重要

DELETE 文またはUPDATE 文中に指定する探索条件についても，ここで説明している探索条件の指定規則が適用されます。ここで説明している指定規則に従っていない場合，DELETE 文または UPDATE 文がエラーになります。

アーカイブマルチチャンク表の行を削除する場合のDELETE 文の規則については，マニュアル『HADB　SQL リファレンス』の『DELETE 文の指定形式および規則』の『規則』を参照してください。

アーカイブマルチチャンク表の行を更新する場合のUPDATE 文の規則については，マニュアル『HADB　SQL リファレンス』の『UPDATE 文の指定形式および規則』の『規則』を参照してください。

(2)　比較述語の指定規則

ここで説明する指定規則に従って比較述語を指定した場合，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が行われます。

(a)　指定規則と推奨する指定例

比較述語の指定形式

比較述語 ：：＝*比較演算項1 比較演算子 比較演算項2*

指定規則

- 比較演算子には，=，<，<=，>=，>のどれかを指定していること
- 両方の比較演算項には，行値構成子を指定していないこと
- 片方の比較演算項には，アーカイブレンジ列（単独の列指定）を指定していること
- 反対の比較演算項には，値指定を指定していること

メモ

値指定には定数を指定することを推奨します。

推奨する指定例

```
SELECT * FROM "ARCHIVE-T1" WHERE "RECORD-DAY" >= DATE'2016/01/01'
```

推奨しない指定例

```
SELECT * FROM "ARCHIVE-T1" WHERE "RECORD-DAY" = ?
SELECT * FROM "ARCHIVE-T1" WHERE "RECORD-DAY" >= CURRENT_DATE
```

上記の例の場合，検索範囲の絞り込みは行われますが，定数以外の指定は推奨しません。

重要

値指定に定数を指定した場合，定数以外を指定したときに比べて，検索範囲の絞り込みに掛かる時間を短くできます。

(b) 検索範囲の絞り込みが行われない指定例

例 1

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" - 10 DAY > DATE'2016/02/10'
```

上記の例の場合，比較演算項にアーカイブレンジ列を使用した日時演算を指定しているため（単独の列指定ではないため），検索範囲の絞り込みは行われません。この場合，上記のSELECT 文を次のように修正すると，検索範囲の絞り込みが行われます。

修正例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" > DATE'2016/02/10' + 10 DAY
```

上記の例のように，片方の比較演算項にアーカイブレンジ列を単独の列指定で指定し，反対の比較演算項には定数と等価な値式を指定します。

例 2

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" > CURRENT_DATE - 1 YEAR
```

上記の例の場合，比較演算項に値指定を指定していないため，検索範囲の絞り込みは行われません。この場合，上記のSELECT 文を次のように修正すると，検索範囲の絞り込みが行われます。

修正例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" > DATE'2016/02/10' - 1 YEAR
```

上記の例のように，CURRENT_DATE を明示的に定数で指定してください。これによって，右側の比較演算項が定数と等価な値式になるため，検索範囲の絞り込みが行われます。

定数と等価な値式については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。

(3)　IN 述語の指定規則

ここで説明する指定規則に従ってIN 述語を指定した場合，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が行われます。

(a)　指定規則と推奨する指定例

IN 述語の指定形式

```
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}
```

指定規則

- IN 述語中に行値構成子を指定していないこと
- IN 述語の左側の値式には，アーカイブレンジ列（単独の列指定）を指定していること
- IN 述語の右側の値式には，値指定を指定していること
- IN 述語中に表副問合せを指定していないこと
- IN 述語中にNOT を指定していないこと

メモ

IN 述語の右側の値式に指定する値指定には，定数を指定することを推奨します。

推奨する指定例

```
SELECT * FROM ”ARCHIVE-T1”
    WHERE ”RECORD-DAY” IN (DATE’2016/01/01’,DATE’2016/02/01’)
```

推奨しない指定例

```
SELECT * FROM ”ARCHIVE-T1” WHERE ”RECORD-DAY” IN (?,?)
```

上記の例の場合，検索範囲の絞り込みは行われますが，定数以外の指定は推奨しません。

重要

値指定に定数を指定した場合，定数以外を指定したときに比べて，検索範囲の絞り込みに掛かる時間を短くできます。

(b) 検索範囲の絞り込みが行われない指定例

例 1

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" IN (CURRENT_DATE,
                           CURRENT_DATE - 7 DAY,
                           CURRENT_DATE - 14 DAY)
```

上記の例の場合，IN 述語の右側の値式に値指定以外を指定しているため，検索範囲の絞り込みは行われません。この場合，上記のSELECT 文を次のように修正すると，検索範囲の絞り込みが行われます。

修正例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" IN (DATE'2016/02/10',
                           DATE'2016/02/10' - 7 DAY,
                           DATE'2016/02/10' - 14 DAY)
```

上記の例のように，CURRENT_DATE を明示的に定数で指定してください。これによって，IN 述語の右側の値式が定数と等価な値式になるため，検索範囲の絞り込みが行われます。

定数と等価な値式については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。

例 2

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" IN (SELECT "SALES_DATE" FROM "SALESLIST"
                             WHERE "USERID" = 'U001')
```

上記の例の場合，IN 述語中に表副問合せを指定しているため，検索範囲の絞り込みは行われません。この場合，上記のSELECT 文を次のように修正すると，検索範囲の絞り込みが行われます。

修正例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" IN (SELECT "SALES_DATE" FROM "SALESLIST"
                             WHERE "USERID" = 'U001')
      AND "RECORD-DAY" > DATE'2015/10/01'
```

上記の例のように，検索範囲の絞り込みが行われる探索条件を追加してください。

(4) BETWEEN 述語の指定規則

ここで説明する指定規則に従ってBETWEEN 述語を指定した場合，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が行われます。

(a) 指定規則と推奨する指定例

BETWEEN 述語の指定形式

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

指定規則

- *値式 1* には，アーカイブレンジ列（単独の列指定）を指定していること
- *値式 2* および*値式 3* には，値指定を指定していること
- BETWEEN 述語中にNOT を指定していないこと

> **メモ**
>
> *値式 2* および*値式 3* に指定する値指定には，定数を指定することを推奨します。

推奨する指定例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2016/01/01' AND DATE'2016/01/10'
```

推奨しない指定例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN ? AND ?
```

上記の例の場合，検索範囲の絞り込みは行われますが，定数以外の指定は推奨しません。

> **重要**
>
> 値指定に定数を指定した場合，定数以外を指定したときに比べて，検索範囲の絞り込みに掛かる時間を短くできます。

(b) 検索範囲の絞り込みが行われない指定例

例 1

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN CURRENT_DATE - 2 YEAR
                           AND CURRENT_DATE - 1 YEAR
```

上記の例の場合，BETWEEN 述語中の*値式 2* または*値式 3* に，値指定以外を指定しているため，検索範囲の絞り込みは行われません。この場合，上記のSELECT 文を次のように修正すると，検索範囲の絞り込みが行われます。

修正例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2016/02/10' - 2 YEAR
                           AND DATE'2016/02/10' - 1 YEAR
```

上記の例のように，CURRENT_DATE を明示的に定数で指定してください。これによって，BETWEEN 述語中の*値式 2* または*値式 3* が定数と等価な値式になるため，検索範囲の絞り込みが行われます。

定数と等価な値式については，マニュアル『HADB　SQL リファレンス』の『値式の指定形式および規則』の『規則』にある表『定数と等価な値式となる条件』を参照してください。

例 2

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" NOT BETWEEN DATE'2016/01/01' AND DATE'2016/01/31'
```

上記の例の場合，BETWEEN 述語中にNOT を指定しているため，検索範囲の絞り込みは行われません。この場合，上記のSELECT 文を次のように修正すると，検索範囲の絞り込みが行われます。

修正例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2015/12/01' AND DATE'2015/12/31'
      AND "RECORD-DAY" BETWEEN DATE'2016/02/01' AND DATE'2016/02/29'
```

上記の例のように，BETWEEN 述語中にNOT を指定しないで，複数のBETWEEN 述語をAND 条件で指定してください。

5.12.3 JOIN（結合表）を指定した場合の留意事項

結合表を指定した場合，結合表の指定方法によっては，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が行われません。検索範囲の絞り込みが行われる結合表の指定方法について説明します。

例中のARCHIVE-T1 はアーカイブマルチチャンク表を意味し，RECORD-DAY はアーカイブレンジ列を意味しています。

(1) 例 1（LEFT OUTER JOIN の場合）

LEFT OUTER JOIN の右側の表参照にアーカイブマルチチャンク表を指定した場合，アーカイブレンジ列の日時情報を使用した検索範囲の絞り込みが行われません。

■検索範囲の絞り込みが行われない例（変更前）

```
SELECT "T1"."C1" FROM "T1"
                 LEFT OUTER JOIN "ADBUSER01"."ARCHIVE-T1" AS "DT"
                 ON "T1"."C1" = "DT"."C1"
  WHERE "RECORD-DAY" BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
  AND "C1"='P001'
```

[説明]

下線部のように，LEFT OUTER JOIN の右側の表参照にアーカイブマルチチャンク表を指定した場合，アーカイブレンジ列の日時情報を使用した検索範囲の絞り込みが行われません。この場合，アーカイブされている全データに対しても検索処理が行われるため，検索処理時間が長くなることがあります。

この場合，次のように SQL 文を変更してください。

■検索範囲の絞り込みが行われる例（変更後）

```
SELECT "T1"."C1" FROM "T1"
                 LEFT OUTER JOIN (SELECT * FROM "ADBUSER01"."ARCHIVE-T1"
                                  WHERE "RECORD-DAY"
```

```
                                        BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
                                    AND "C1"='P001') AS "DT"
                ON "T1"."C1" = "DT"."C1"
  WHERE "RECORD-DAY" BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
  AND "C1"='P001'
```

[説明]

下線部のように，アーカイブレンジ列を指定した探索条件を明示的に指定した導出表に書き換えます。上記のようにSQL文を変更すると，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が行われます。

メモ

LEFT OUTER JOINの左側の表参照にアーカイブマルチチャンク表を指定する場合は，特に考慮することはありません（SQL文を変更する必要はありません）。

(2) 例2（RIGHT OUTER JOINの場合）

RIGHT OUTER JOINの左側の表参照にアーカイブマルチチャンク表を指定した場合，アーカイブレンジ列の日時情報を使用した検索範囲の絞り込みが行われません。

■検索範囲の絞り込みが行われない例（変更前）

```
SELECT "T1"."C1" FROM "ADBUSER01"."ARCHIVE-T1" AS "DT"
                RIGHT OUTER JOIN "T1"
                ON "DT"."C1" = "T1"."C1"
  WHERE "RECORD-DAY" BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
  AND "C1"='P001'
```

[説明]

下線部のようにRIGHT OUTER JOINの左側の表参照にアーカイブマルチチャンク表を指定した場合，アーカイブレンジ列の日時情報を使用した検索範囲の絞り込みが行われません。この場合，アーカイブされている全データに対しても検索処理が行われるため，検索処理時間が長くなることがあります。

この場合，次のようにSQL文を変更してください。

■検索範囲の絞り込みが行われる例（変更後）

```
SELECT "T1"."C1" FROM (SELECT * FROM "ADBUSER01"."ARCHIVE-T1"
                        WHERE "RECORD-DAY" BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
                        AND "C1"='P001') AS "DT"
                RIGHT OUTER JOIN "T1"
                ON "DT"."C1" = "T1"."C1"
  WHERE "RECORD-DAY" BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
  AND "C1"='P001'
```

[説明]

下線部のように，アーカイブレンジ列を指定した探索条件を明示的に指定した導出表に書き換えます。上記のようにSQL文を変更すると，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が行われます。

メモ

RIGHT OUTER JOINの右側の表参照にアーカイブマルチチャンク表を指定する場合は，特に考慮することはありません（SQL文を変更する必要はありません）。

(3) 例3（FULL OUTER JOINの場合）

FULL OUTER JOINの左側または右側の表参照にアーカイブマルチチャンク表を指定した場合，アーカイブレンジ列の日時情報を使用した検索範囲の絞り込みが行われません。

■検索範囲の絞り込みが行われない例（変更前）

```
SELECT "T1"."C1" FROM "ADBUSER01"."ARCHIVE-T1" AS "DT"
                 FULL OUTER JOIN "T1"
                 ON "DT"."C1" = "T1"."C1"
  WHERE "RECORD-DAY" BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
  AND "C1"='P001'
```

[説明]

下線部のようにFULL OUTER JOINの左側または右側の表参照にアーカイブマルチチャンク表を指定した場合，アーカイブレンジ列の日時情報を使用した検索範囲の絞り込みが行われません。この場合，アーカイブされている全データに対しても検索処理が行われるため，検索処理時間が長くなることがあります。

この場合，次のようにSQL文を変更してください。

■検索範囲の絞り込みが行われる例（変更後）

```
SELECT "T1"."C1" FROM (SELECT * FROM "ADBUSER01"."ARCHIVE-T1"
                       WHERE "RECORD-DAY" BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
                       AND "C1"='P001') AS "DT"
                 FULL OUTER JOIN "T1"
                 ON "DT"."C1" = "T1"."C1"
  WHERE "RECORD-DAY" BETWEEN DATE'2016/01/15' AND DATE'2016/02/15'
  AND "C1"='P001'
```

[説明]

下線部のように，アーカイブレンジ列を指定した探索条件を明示的に指定した導出表に書き換えます。上記のようにSQL文を変更すると，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が行われます。

5.12.4　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

次に示す条件をすべて満たす場合，アーカイブマルチチャンク表を検索するSQL文はHADBサーバによって自動的に変換（等価変換）されます。

- アーカイブされているデータを検索する場合
- WHERE句の探索条件にアーカイブレンジ列を指定した条件を指定している場合
- WHERE句に指定した探索条件が，「5.12.2　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」で説明している規則を満たしている場合

メモ

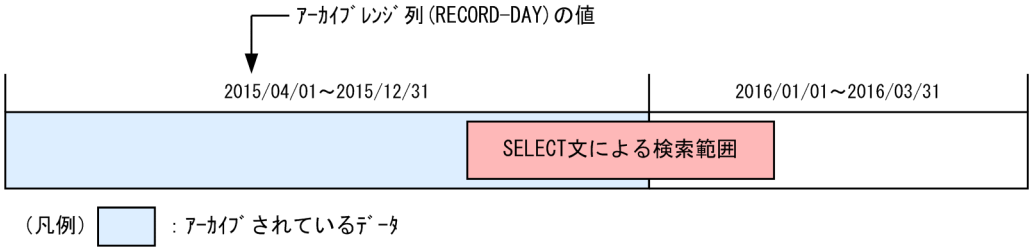
- 等価変換後のSQL文が，アクセスパスの情報に出力されます。
- 等価変換後のSQL文に対してSQL文の規則が適用されます。

等価変換の例を次に示します。

指定したSELECT文の例

```
SELECT * FROM "ARCHIVE-T1"
    WHERE "RECORD-DAY" BETWEEN DATE'2015/11/01' AND DATE'2016/01/31'
    AND "C1"='P001'
```

SELECT文による検索範囲



等価変換されたSELECT文の例

```
SELECT * FROM
  (SELECT * FROM "ARCHIVE-T1"                                    ...1
   UNION ALL                                                     ...2
   SELECT * FROM                                                 ...3
     TABLE (ADB_CSVREAD(MULTISET(                                ...4
       SELECT "ARCHIVE_FILE_NAME" FROM
                 "HADB"."LOCATION_TABLE_00020191" AS "LOC"
                ,"HADB"."STATUS_CHUNKS" AS "SCK"
         WHERE "RANGE_MAX" >= DATE'2015/11/01'                   ...5
           AND "RANGE_MIN" <= DATE'2016/01/31'                   ...5
           AND "SCK"."TABLE_SCHEMA" = 'ADBUSER01'
           AND "SCK"."TABLE_NAME" = 'ARCHIVE-T1'
           AND "SCK"."CHUNK_ID" = "LOC"."CHUNK_ID"
           AND "SCK"."CHUNK_STATUS" IS NULL
       )
       ,'中略')
```

```
    ) AS "TBLFUNC_00020191" ("C1" VARCHAR(10), "C2" INT, "RECORD_DAY" DATE)   ...6
)
  WHERE "RECORD_DAY" BETWEEN DATE'2015/11/01' AND DATE'2016/01/31'
  AND "C1"='P001'
```

[説明]

この例では，アーカイブされているデータと，アーカイブされていないデータを検索しています。この例では，アーカイブされているデータを検索する問合せと，アーカイブされていないデータを検索する問合せに書き換えられ，この 2 つの問合せの和集合（UNION ALL）を求めています。

1. アーカイブされていないデータを検索する問合せです。
2. 1.と 3.の問合せの和集合（UNION ALL）を求めています。
3. アーカイブされているデータを検索する問合せです。
4. ADB_CSVREAD 関数に変換されて，アーカイブされているデータが格納されているアーカイブファイルを読み込んでいます。
5. WHERE 句に指定した探索条件を基に，ロケーション表を検索する探索条件に書き換えています。※
6. TBLFUNC_00020191 は，表関数導出表の相関名です。相関名は次の規則に従って決定されます。

   TBLFUNC_nnnnnnnn

   nnnnnnnn：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（0～9，A～F）

注※

ロケーション表を検索する探索条件に書き換える例を次に示します。

（例 1）

■指定された探索条件

```
WHERE "RECORD_DAY" = DATE'2015/10/01'
```

書き換え

■書き換え後の探索条件

```
WHERE ("RANGE_MIN" <= DATE'2015/10/01') AND
      ("RANGE_MAX" >= DATE'2015/10/01')
```

（例 2）

■指定された探索条件

```
WHERE "RECORD_DAY" BETWEEN
      DATE'2015-10-01' AND DATE'2015/12/31'
```

■書き換え後の探索条件

```
WHERE ("RANGE_MAX" >= DATE'2015/10/01') AND
      ("RANGE_MIN" <= DATE'2015/12/31')
```

- RANGE_MAX

  ロケーション表の列です。アーカイブファイルごとのアーカイブレンジ列の値の最大値が格納されています。

- RANGE_MIN

  ロケーション表の列です。アーカイブファイルごとのアーカイブレンジ列の値の最小値が格納されています。

メモ

アーカイブマルチチャンク表を検索する SQL 文の等価変換によって生成された内部導出表は，展開の対象になりません。内部導出表の展開については，マニュアル『HADB　SQL リファレンス』の『内部導出表』を参照してください。

5.13　内部導出表の展開

HADB では，内部導出表を含む問合せ式を自動的に変更し，内部導出表を効率的に評価しています。内部導出表を含む問合せ式の変更を，内部導出表の展開といいます。内部導出表の展開規則については，マニュアル『HADB　SQL リファレンス』の『内部導出表』を参照してください。

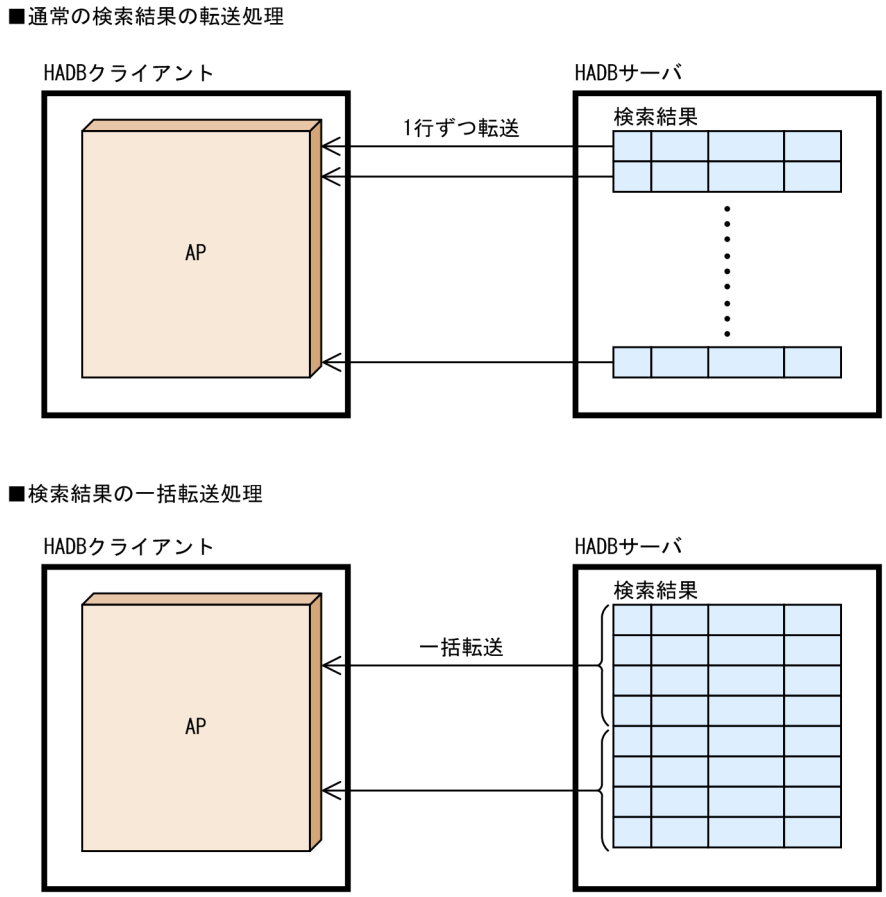
また，検索時に使用されるインデクスの優先順位と選択規則は，内部導出表の展開後の変更された問合せ式に対して適用されます。検索時に使用されるインデクスについては，「5.2　SQL 文の実行時に使用される B-tree インデクスおよびテキストインデクス」および「5.3　SQL 文の実行時に使用されるレンジインデクス」を参照してください。

5.14　検索結果の一括転送

HADB サーバから HADB クライアントに検索結果を転送する際，複数の行を一括して転送できます。大量のデータを検索する場合に効果があります。

検索結果の一括転送の概要を次の図に示します。

図 5-21　検索結果の一括転送の概要



検索結果を一括転送する場合，次に示す設定をしてください。

- JDBC ドライバを使用している場合
  次に示すどれかの方法で一括転送行数を指定してください。
  - システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティの`adb_clt_fetch_size`
  - `Statement` オブジェクトの`setFetchSize` メソッド
  - `ResultSet` オブジェクトの`setFetchSize` メソッド
- ODBC ドライバまたは CLI 関数を使用している場合
  クライアント定義の`adb_clt_fetch_size` オペランドに一括転送行数を指定してください。

上記の設定をすると，行の取り出しを実行した場合に検索結果が一括転送されます。

留意事項

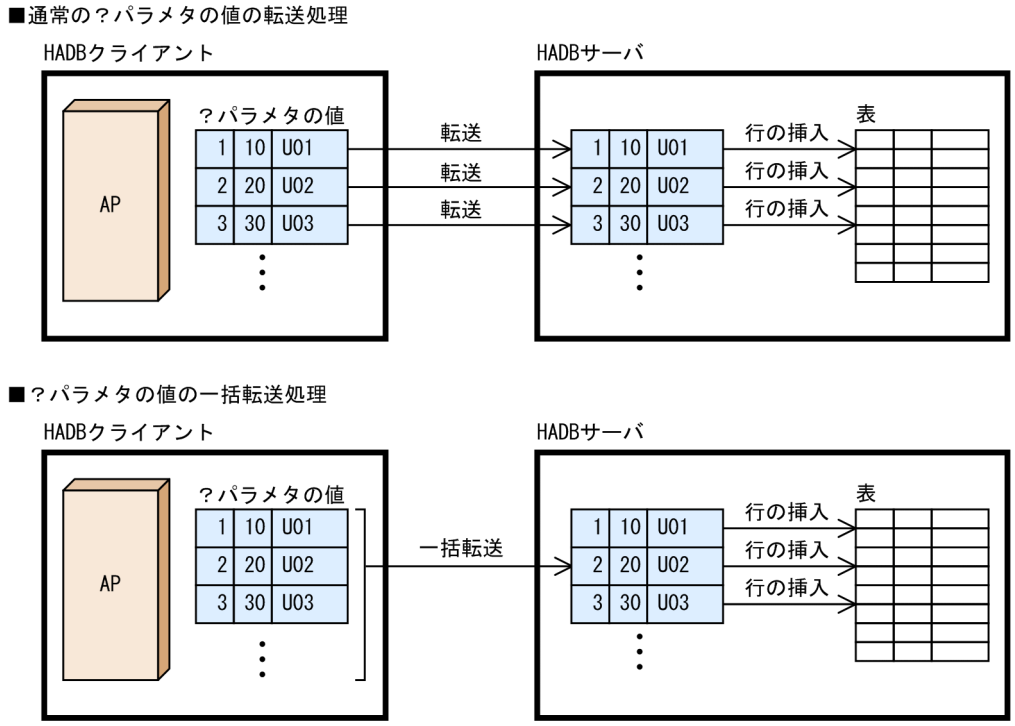
- 一括転送の行数に比例して，HADB サーバおよび HADB クライアントが使用するメモリ所要量が増加します。そのため，一括転送を行う場合は，メモリ所要量の再見積もりをしてください。HADB サーバが使用するメモリ所要量については，マニュアル『HADB　システム構築・運用ガイド』の『通常運用時のメモリ所要量の求め方』を参照してください。HADB クライアントが使用するメモリ所要量については，「付録 C　HADB クライアントのメモリ所要量の見積もり」を参照してください。
- 一括転送の実行中にエラーが発生した場合，HADB サーバでバッファリングされていた検索結果は破棄され，エラー情報だけが HADB クライアントに返されます。
- カーソルを使用した検索中に更新系 SQL を実行した場合，タイミングによっては更新操作の結果が検索結果に反映されることがありますが，一括転送を使用しているときは，更新操作の結果が検索結果に反映されることはありません。これは，HADB クライアントに検索結果が残っている間は，HADB サーバへのアクセスを行うことがないためです。

5.15　？パラメタの値の一括転送

HADB クライアントから HADB サーバに？パラメタの値を転送する際，複数の？パラメタの値を一括して転送できます。HADB クライアントから HADB サーバにアクセスし，？パラメタを使用してデータの追加，更新，または削除をする場合に効果があります。

？パラメタの値の一括転送の概要を次の図に示します。

図 5-22　？パラメタの値の一括転送の概要



[説明]

？パラメタの値を行の挿入値として，複数の行の挿入（INSERT 文）を実行する場合，通常の？パラメタの値の転送処理では，？パラメタの値を 1 組（1 つの行に挿入する？パラメタの値）ずつ HADB サーバに転送します。

一方，？パラメタの値の一括転送処理では，2 組以上の？パラメタの値をまとめて一括転送します。これによって，通信オーバヘッドが少なくなり，その分 AP の実行時間が短縮されます。

なお，この図では行の挿入（INSERT 文）を例にしていますが，UPDATE 文またはDELETE 文についても，？パラメタの値を一括転送できます。

JDBC ドライバを使用している場合は，PreparedStatement クラスのexecuteBatch メソッドまたは executeLargeBatch メソッドで SQL 文を実行すると，？パラメタの値が一括転送されます。

CLI 関数を使用している場合は，次に示す流れで SQL 文を実行すると，？パラメタの値が一括転送されます。

1. INSERT 文，UPDATE 文，またはDELETE 文を前処理する

2. a_rdb_SQLBindArrayParams()で，？パラメタの一括関連づけをする

3. 1.で前処理した文ハンドルを使用して，a_rdb_SQLExecute()で SQL 文を実行する

ODBC ドライバを使用している場合は，？パラメタの値の一括転送は使用できません。

留意事項

- SQL 文の実行中にエラーが発生してロールバックが実行された場合，処理結果行数は破棄され，エラー情報だけが HADB クライアントに返却されます。
- ？パラメタの値を一括転送すると，HADB サーバおよび HADB クライアントが使用するメモリ所要量が増加するため，メモリ所要量の再見積もりをしてください。HADB サーバが使用するメモリ所要量については，マニュアル『HADB　システム構築・運用ガイド』の『通常運用時のメモリ所要量の求め方』を参照してください。HADB クライアントが使用するメモリ所要量については，「付録 C　HADB クライアントのメモリ所要量の見積もり」を参照してください。
- スカラ関数RANDOMCURSOR を指定した SQL 文に対して，？パラメタの値の一括転送を実行した場合，スカラ関数RANDOMCURSOR は？パラメタの 1 組ごとに擬似乱数を生成します。

5.16　SQL パラレル実行機能が適用される条件

次の条件をすべて満たす場合に，検索系 SQL（SELECT 文）に SQL パラレル実行機能が適用されます。

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。

■SQL パラレル実行機能を使用するための条件

次のどちらかの条件を満たす必要があります。

- クライアント定義のadb_clt_sql_parallel_exec オペランドにY を指定していること。
- SQL 文に SQL パラレル実行指定を指定していること。

■実行環境に関する条件

次の 2 つの条件を満たす必要があります。

1. サーバ定義に関する条件

   次の条件をすべて満たす必要があります。

   - サーバ定義のadb_sys_uthd_num オペランドに0 を指定していないこと。
   - サーバ定義，またはクライアント定義のadb_sql_exe_max_rthd_num オペランドに0 を指定していないこと。
   - サーバ定義のadb_sys_max_parallel_exec_num オペランドに0 を指定していないこと。

2. 検索系 SQL を実行するトランザクションに関する条件

   次の条件をすべて満たす必要があります。

   - トランザクションアクセスモードが読み取り専用モードであること。
   - トランザクション隔離性水準がREAD COMMITTED であること。

■検索対象表に関する条件

検索系 SQL の検索対象表（検索系 SQL のFROM 句に指定する実表）が，次の条件をすべて満たす必要があります。

- 検索対象表がカラムストア表であること。
- 検索対象表がマルチチャンク表であること。
- バージョン 04-03 以降の HADB サーバで，検索対象表のコスト情報を収集していること。
- 検索対象表を格納しているデータ用 DB エリアが，2 つ以上の DB エリアファイルで構成されていること。※

注※

クラウドストレージ機能を使用している場合は該当しない条件です。

■SQL 文に関する条件

次の条件をすべて満たす必要があります。

- SQL 文がSELECT 文であること。

- SQL 文に集合演算を指定していないこと。
- SQL 文に副問合せを指定していないこと。

■SQL 文中の問合せ指定に関する条件

次の 6 つの条件を満たす必要があります。

1. SELECT DISTINCT が指定されていないこと。

   ただし，SELECT DISTINCT が指定されている場合でも，HADB サーバがSELECT DISTINCT の実行は不要と判断したときは，SQL パラレル実行機能が適用されます。

2. FROM 句には，実表が 1 つだけ指定されていること。または，実表 1 つと，実表に対する集まり導出表だけが指定されていること。

3. GROUP BY 句の指定がある場合，次の条件をすべて満たすこと。

   - グループ化方式としてグローバルハッシュグループ化が適用されていること。
   - 集合関数が指定されている場合，次のどれかだけが指定されていること。
     - 集合関数COUNT(*)
     - 集合関数MIN
     - 集合関数MAX
     - ALL 集合関数COUNT
     - ALL 集合関数SUM
     - ALL 集合関数AVG

4. GROUP BY 句の指定がない場合，次の条件をすべて満たすこと。

   - 次の集合関数のどれかが 1 つ以上指定されていること。
     - 集合関数COUNT(*)
     - 集合関数MIN
     - 集合関数MAX
     - ALL 集合関数COUNT
     - ALL 集合関数SUM
     - ALL 集合関数AVG
   - ORDER BY 句が指定されていて，ソートキーに集合関数を含む値式が指定されている場合，その値式が選択式に指定されていること。

5. HAVING 句の指定があり，HAVING 句の探索条件中に集合関数が指定されている場合，次のどれかだけが指定されていること。

   - 集合関数COUNT(*)
   - 集合関数MIN
   - 集合関数MAX
   - ALL 集合関数COUNT
   - ALL 集合関数SUM

- ALL 集合関数AVG

6. 値式は，次の条件をすべて満たすこと。
   - ？パラメタを指定していないこと。
   - 日時情報取得関数を指定していないこと。
   - ユーザ情報取得関数を指定していないこと。
   - 次のスカラ関数を指定していないこと。
     - RANDOM
     - RANDOM_NORMAL
     - RANDOMCURSOR
     - RANDOMROW
   - 同義語検索指定を指定したスカラ関数CONTAINS を指定していないこと。
   - ウィンドウ関数を指定していないこと。

重要

- 上記の条件をすべて満たしても，SQL パラレル実行機能を有効に利用できないと HADB サーバが判断した場合，検索系 SQL に SQL パラレル実行機能は適用されません。
- adbexport コマンドの SQL 記述ファイルに指定した SQL 文については，SQL パラレル実行機能は適用されません。

6 APのチューニング

この章では，アクセスパスの見方について説明します。

6.1　アクセスパスの見方（SQL 文の実行計画の見方）

ここでは，アクセスパス（SQL 文の実行計画）に出力される情報と，その見方について説明します。

> **メモ**
>
> HADB サーバのチューニングについては，マニュアル『HADB　システム構築・運用ガイド』の『チューニング』を参照してください。

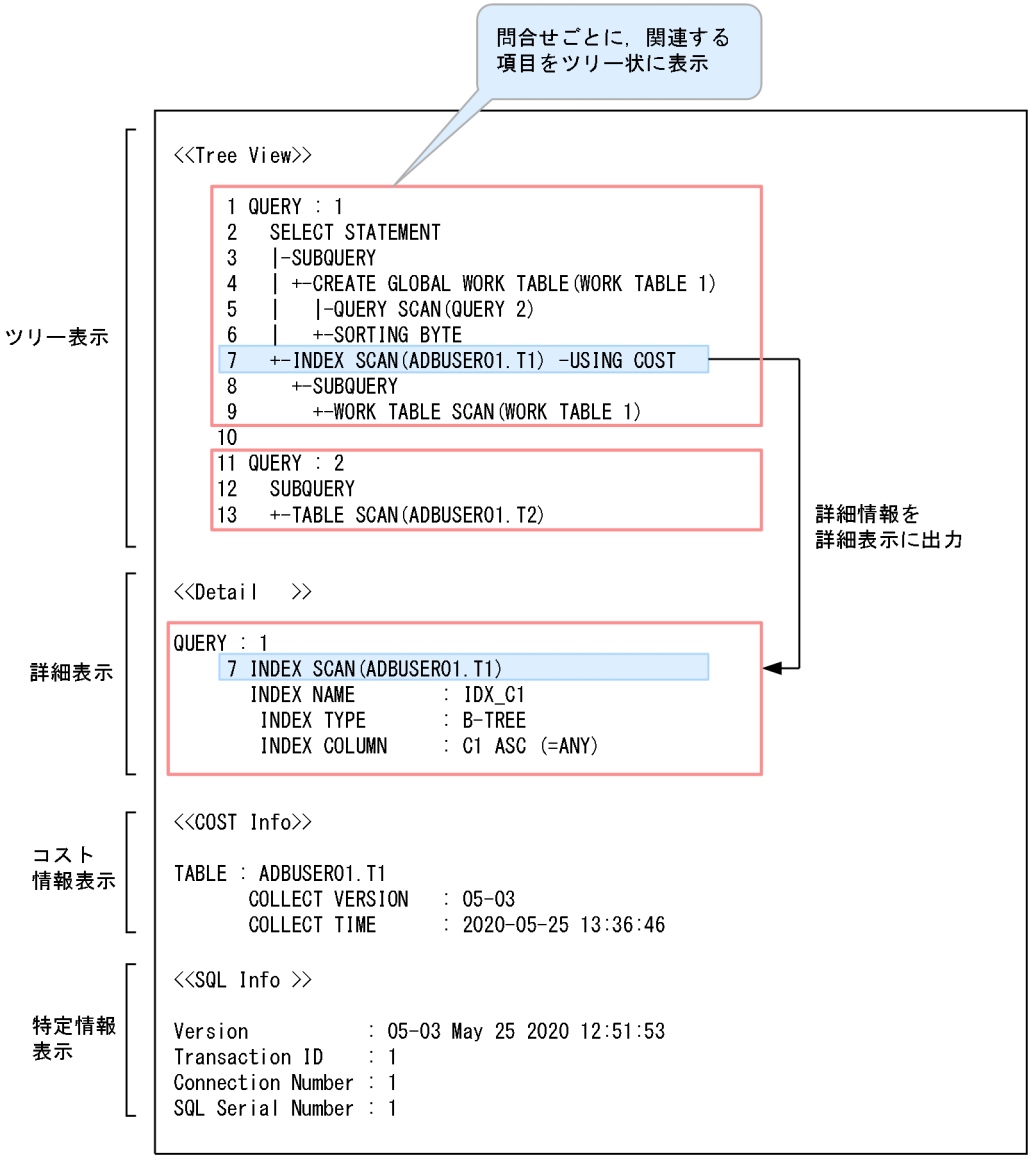
6.1.1　アクセスパスとは

アクセスパスとは，HADB が SQL 文を実行する際の実行計画のことです。アクセスパスを確認することで，実行する SQL 文がどのように処理されるかを確認できます。

(1)　アクセスパス情報の表示例

アクセスパス情報の表示例を次の図に示します。

図 6-1　アクセスパス情報の表示例



[説明]

アクセスパス情報には，ツリー表示，詳細表示，および特定情報表示が表示されます。

- ツリー表示

  表の検索方式，表の結合方式，作業表に関する情報などがツリー形式で表示されます。関連する項目がツリー状に表示されます。

  問合せごとにアクセスパス情報が表示されます。各問合せには，問合せツリー番号が割り振られます。

  ツリー表示のヘッダとして<<Tree View>>が表示されます。

- 詳細表示

  ツリー表示で表示された項目の詳細情報が表示されます。ツリー表示に表示されているツリー行番号と，詳細情報に表示されているツリー行番号が対応しています。上記の例では，ツリー表示に表示されているツリー行番号 7 の詳細情報が，詳細表示に表示されています。

  詳細表示のヘッダとして<<Detail >>が表示されます。

- コスト情報表示

  SQL 文中で参照される実表のうち，コスト情報が取得されている実表に関する情報（表コスト情報）が表示されます。

  コスト情報が取得されている実表の表名，コスト情報を取得したときの HADB サーバのバージョン，コスト情報を取得した日時が表示されます。

  コスト情報表示のヘッダとして<<COST Info>>が表示されます。

- 特定情報表示

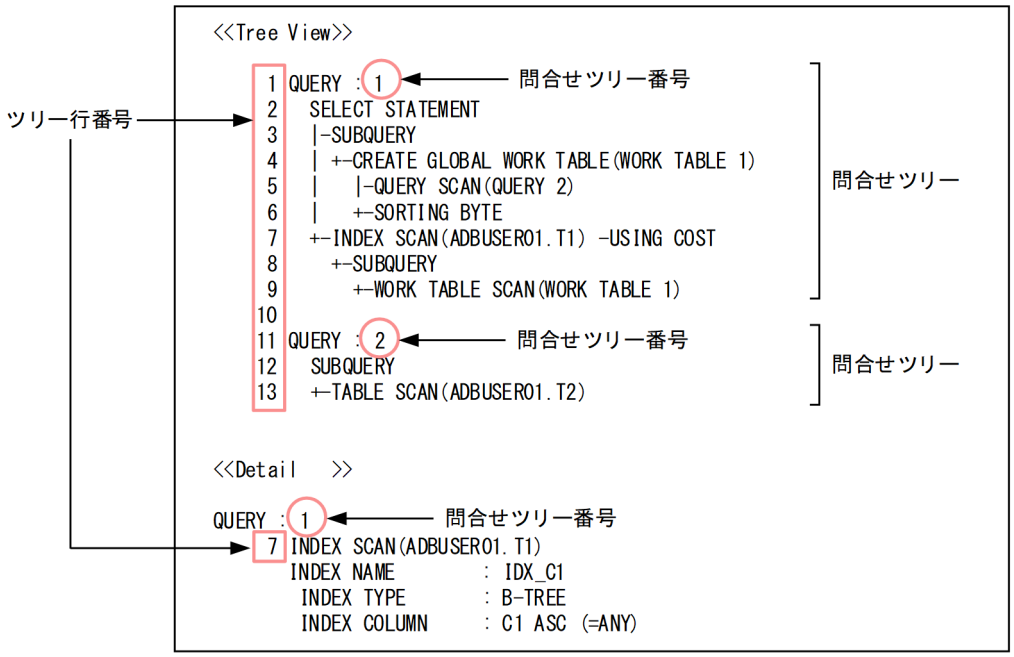
  特定情報表示には，SQL 文の特定情報が表示されます。SQL 文の特定情報には，アクセスパスの統計情報を取得した SQL 文を特定するための情報が表示されます。ここに表示された情報を基に，アクセスパス情報の出力結果とアクセスパスの統計情報の出力結果を対応付けることができます。

  特定情報表示のヘッダとして<<SQL Info >>が表示されます。

  アクセスパスの統計情報については，マニュアル『HADB　システム構築・運用ガイド』の『アクセスパスの統計情報の出力例と出力項目』を参照してください。

> **メモ**
>
> 詳細表示およびコスト情報表示は，該当する情報がない場合は出力されません。

■問合せツリー，問合せツリー番号，ツリー行番号



[説明]

- 問合せツリー

  アクセスパス情報は，問合せ（問合せ指定および表値構成子）ごとに問合せツリーの形式で出力されます。

- 問合せツリー番号

  各問合せツリーを識別するために，問合せツリーごとに番号（問合せツリー番号）が割り振られます。

副問合せに対応する問合せツリーの場合は，問合せツリー番号が 2 以降になります。ただし，次のどちらかの条件を満たす場合は，副問合せに対応する問合せツリーであっても，問合せツリー番号が 1 以降になります。

- INSERT 文にVALUES を指定し，VALUES 中に副問合せを指定した場合
- PURGE CHUNK 文中に副問合せを指定した場合

> **メモ**
>
> - 実行する SQL 文によっては，問合せツリー番号は 1，2，…の順に出力されないことがあります。
> - SQL 文中の問合せに含まれない部分の問合せツリーの場合は，問合せツリー番号が 0 になります。

- ツリー行番号
  ツリーで表示される行ごとに割り振られた番号です。

(2) アクセスパスの出力対象となる SQL 文

アクセスパスの出力対象となる SQL 文を次に示します。

- SELECT 文
- UPDATE 文
- INSERT 文
  次に示す場合にアクセスパスが出力されます。
  - INSERT 文中に問合せ式本体を指定している場合
  - INSERT 文中にVALUES を指定していて，かつ副問合せを指定している場合
- DELETE 文
- PURGE CHUNK 文
  PURGE CHUNK 文中に副問合せを指定した場合にアクセスパスが出力されます。

上記の SQL 文を実行するごとに，アクセスパスが出力されます。

6.1.2 アクセスパスを確認するには

アクセスパスを確認するには次に示す 2 つの方法があります。

- adbsql サブコマンドの#SET OPT REPORT を実行してアクセスパス情報を確認する
- SQL トレース情報中に出力されたアクセスパス情報を確認する

(1) #SET OPT REPORT を実行してアクセスパス情報を確認する方法

adbsql コマンドの#SET OPT REPORT で，アクセスパス情報を表示できます。アクセスパス情報の表示手順を次に示します。

手順

1. adbsql コマンドを実行する
2. adbsql サブコマンドの#SET OPT REPORT を実行する

```
#SET OPT REPORT ON TYPE=PATH EXEC=PREPARE;
```

> **メモ**
>
> この例では，SQL 文の実行はしないで，アクセスパス情報だけを表示するため，EXEC=PREPARE を指定しています。SQL 文を実行して，かつアクセスパス情報を表示したい場合は，EXEC=PREPARE を指定しないでください。

3. SQL 文を実行する

```
SELECT * FROM "T1" WHERE "C1" =1 AND "C2" > 0;
```

上記の SQL 文を実行すると，アクセスパス情報が表示されます。

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(ADBUSER01.T1) -USING COST


<<Detail   >>

QUERY : 1
     3 TABLE SCAN(ADBUSER01.T1)
       INDEX NAME         : RIDX
        INDEX TYPE        : RANGE
        SKIP COND         : CHUNK AND SEGMENT
        INDEX COLUMN      : C1

<<COST Info>>

TABLE : ADBUSER01.T1
       COLLECT VERSION    : 05-03
       COLLECT TIME       : 2020-05-25 13:36:46


<<SQL Info >>

Version           : 05-03 May 25 2020 12:51:53
Transaction ID    : 1
Connection Number : 1
SQL Serial Number : 1
```

なお，この例での表およびレンジインデクスの定義は，次のとおりとします。

```
CREATE TABLE ”T1”(”C1” INTEGER,”C2” INTEGER) IN ”DBAREA01”
CREATE INDEX ”RIDX” ON ”T1”(”C1”) IN ”DBAREA02” EMPTY INDEXTYPE RANGE
```

(2) SQL トレース情報中に出力されたアクセスパス情報を確認する方法

SQL トレースファイルに出力された SQL トレース情報を，テキストエディタを使用して参照してください。SQL トレースファイルのファイル名を次に示します。

- $ADBDIR/spool/adbsqltrc01.log～adbsqltrc08.log

SQL トレース情報に出力されるアクセスパス情報の例を次に示します。

（例）

```
[SQL]
SELECT * FROM ”T1” WHERE ”C1”=? AND ”C2”=? AND ”C3”=?

[access path]
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(T1)
```

SQL トレース情報については，マニュアル『HADB　システム構築・運用ガイド』の『SQL トレース機能の運用』を参照してください。

6.1.3 アクセスパスの見方の例

(1) 例 1

表および B-tree インデクスの定義

```
CREATE TABLE ”T1”(”C1” INT,”C2” DEC,”C3” CHAR(10)) IN ”DBAREA01”
CREATE INDEX ”IDX_C1” ON ”T1”(”C1”) IN ”DBAREA02” EMPTY
CREATE INDEX ”IDX_C3” ON ”T1”(”C3”) IN ”DBAREA02” EMPTY
CREATE TABLE ”T2”(”C1” INT,”C2” DEC,”C3” CHAR(10)) IN ”DBAREA01”
```

実行した SQL 文

```
SELECT * FROM ”T1”,”T2”
    WHERE ”T1”.”C1”=”T2”.”C2”
```

アクセスパスの表示例

```
<<Tree View>>
```

```
     1 QUERY : 1
     2   SELECT STATEMENT                                    ...1
     3   +-NESTED LOOP JOIN                                  ...2
     4     |-TABLE SCAN(ADBUSER01.T2)                        ...3
     5     +-INDEX SCAN(ADBUSER01.T1) -ORDER -USING COST   ...4


<<Detail   >>

QUERY : 1
     5 INDEX SCAN(ADBUSER01.T1)                 ...5
       INDEX NAME        : IDX_C1              ...6
        INDEX TYPE       : B-TREE              ...7
        INDEX COLUMN     : C1 ASC (=)          ...8


<<COST Info>>

TABLE : ADBUSER01.T1
       COLLECT VERSION   : 05-03
       COLLECT TIME      : 2020-05-25 13:36:46


<<SQL Info >>

Version           : 05-03 May 25 2020 12:51:53
Transaction ID    : 6197
Connection Number : 3
SQL Serial Number : 2
```

[説明]

1. SELECT 文が実行されることを示しています。
2. 表の結合処理で，ネストループジョインが実行されることを示しています。
3. 表T2 の検索処理で，テーブルスキャンが実行されることを示しています。
4. 表T1 の検索処理で，インデクススキャンが実行されることを示しています。
   - 「-ORDER」が表示されているため，非順序実行方式が適用されないで，順序実行方式が適用されることを示しています。
   - 「-USING COST」が表示されているため，表T1 に関するコスト情報が収集されていることを示しています。
5. 表T1 に対するインデクススキャンの詳細情報を示しています。
6. インデクススキャンの実行で，B-tree インデクスIDX_C1 が使用されることを示しています。
7. インデクスの種別が表示されます。B-TREE は，B-tree インデクスであることを示しています。
8. B-tree インデクスIDX_C1 の情報を示しています。
   - C1：B-tree インデクスIDX_C1 のインデクス構成列
   - ASC：キー値の並び順
   - (=)：サーチ条件として評価する条件の種別

(2) 例 2

表の定義

```
CREATE TABLE "T1"("C1" INT,"C2" DEC,"C3" CHAR(10)) IN "DBAREA01"
CREATE TABLE "T2"("C1" INT,"C2" DEC,"C3" CHAR(10)) IN "DBAREA01"
```

実行した SQL 文

```
SELECT * FROM "T1"
     WHERE "C1"=ANY(SELECT "C2" FROM "T2" WHERE "C1"="T1"."C2")
```

アクセスパスの表示例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT                                  ...1
     3   +-TABLE SCAN(ADBUSER01.T1) -USING COST            ...2
     4     +-SUBQUERY LOOP                                 ...3
     5       |-CREATE LOCAL WORK TABLE(WORK TABLE 1)       ...4
     6       | +-QUERY SCAN(QUERY 2)                       ...5
     7       +-WORK TABLE SCAN(WORK TABLE 1)               ...6
     8
     9 QUERY : 2                                           ...7
    10   SUBQUERY LOOP                                     ...8
    11   +-TABLE SCAN(ADBUSER01.T2)                        ...9


<<COST Info>>

TABLE : ADBUSER01.T1
       COLLECT VERSION   : 05-03
       COLLECT TIME      : 2020-05-25 13:36:46


<<SQL Info >>

Version           : 05-03 May 25 2020 12:51:53
Transaction ID    : 6197
Connection Number : 3
SQL Serial Number : 3
```

[説明]

1. SELECT 文が実行されることを示しています。
2. 表T1 の検索処理で，テーブルスキャンが実行されることを示しています。

   「-USING COST」が表示されているため，表T1 に関するコスト情報が収集されていることを示しています。
3. 限定述語（ANY）に指定した副問合せの処理方式で，ネストループ作業表実行またはネストループ行値実行が適用されることを示しています。
4. 副問合せの処理で，ローカル作業表が作成されることを示しています。

5. 問合せ検索が実行されることを示しています。ツリー行番号 9 に表示されている「QUERY : 2」の問合せ検索を行うことを示しています。
6. 副問合せの処理で，作業表の検索が実行されることを示しています。
7. 副問合せの詳細情報を示しています。
8. 副問合せの処理方式で，ネストループ作業表実行またはネストループ行値実行が適用されることを示しています。
9. 表T2 の検索処理で，テーブルスキャンが実行されることを示しています。

6.1.4 ツリー表示に出力される情報

ツリー表示には，表の検索方式，表の結合方式，副問合せの処理方式などが，問合せごとにツリー形式で表示されます。

(1) 実行した SQL 文の種類

次のどれかの情報が出力されます。

- SELECT STATEMENT

  SELECT 文が実行されることを示しています。
- UPDATE STATEMENT

  UPDATE 文が実行されることを示しています。
- INSERT STATEMENT

  INSERT 文が実行されることを示しています。
- DELETE STATEMENT

  DELETE 文が実行されることを示しています。
- PURGE CHUNK STATEMENT

  PURGE CHUNK 文が実行されることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(ADBUSER01.T2)
```

[説明]

SELECT 文が実行されることを示しています。

(2) 副問合せの処理方式

次のどれかの情報が出力されます。

- SUBQUERY

  ネストループ実行，ハッシュ実行以外の副問合せが適用されることを示しています。

- SUBQUERY LOOP

  副問合せの処理方式で，ネストループ作業表実行またはネストループ行値実行が適用されることを示しています。

- SUBQUERY HASH

  副問合せの処理方式で，ハッシュ実行が適用されることを示しています。

  SUBQUERY HASH の後ろにFILTER が表示されている場合，ハッシュ実行の際にハッシュフィルタが適用されることを示しています。

副問合せの処理方式については，「5.6 副問合せの処理方式」を参照してください。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-SUBQUERY HASH -FILTER
     4     |-QUERY SCAN(QUERY 2)
     5     +-TABLE SCAN(ADBUSER01.T1)
     6
     7 QUERY : 2
     8   SUBQUERY HASH
     9   +-TABLE SCAN(ADBUSER01.T2)
```

[説明]

副問合せの処理方式で，ハッシュ実行が適用されることを示しています。

ツリー行番号 3 のSUBQUERY HASH の後ろにFILTER が表示されているため，ハッシュ実行の際にハッシュフィルタが適用されることを示しています。

(3) 導出表の指定

次の情報が出力されます。

- DERIVED TABLE(*相関名*)

  次に示すどれかが指定されていることを示しています。

  - 導出表
  - ビュー表（相関名の指定がある場合）
  - 問合せ名（相関名の指定がある場合）

- DERIVED TABLE(*問合せ名*)

問合せ名（相関名の指定がない場合）が指定されていることを示しています。

- DERIVED TABLE(*表識別子*)

  ビュー表（相関名の指定がない場合）が指定されていることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-DERIVED TABLE(D1)
     4     +-QUERY SCAN(QUERY 2)
     5
     6 QUERY : 2
     7   DERIVED TABLE(D1)
     8   |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
     9   | |-TABLE SCAN(ADBUSER01.T2)
    10   | |-SORTING BYTE
    11   | +-LIMIT 10
    12   |-WORK TABLE SCAN(WORK TABLE 1)
    13   +-LIMIT 10
```

[説明]

SELECT 文中に導出表が指定されていることを示しています。（　）内には，相関名が表示されます。

(4)　集合演算の指定

次の情報が出力されます。

- SET OPERATION

  集合演算が指定されていることを示しています。

> **メモ**
>
> 次に示す場合に，上記の情報が表示されることがあります。
>
> - 結合表にFULL OUTER JOIN を指定した場合
> - 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を検索する際，SQL 文の等価変換によって集合演算を指定した SQL 文に自動的に書き換えられることがあります。詳細については，「5.12.4　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）が適用された場合
>
>   FROM 句にコンマ結合または結合表が指定されていて，かつWHERE 句にOR 条件が指定されている場合，集合演算UNION ALL を指定した導出表への等価変換によって SQL 文が自動的に書き換えられることがあります。詳細については，「5.11.4　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。

出力例

```
<<Tree View>>

     1 QUERY : 0
     2   SELECT STATEMENT
     3   +-SET OPERATION
     4     |-QUERY SCAN(QUERY 1)
     5     +-QUERY SCAN(QUERY 2)
     6
     7 QUERY : 1
     8   QUERY
     9   +-TABLE SCAN(ADBUSER01.T1)
    10
    11 QUERY : 2
    12   QUERY
    13   +-TABLE SCAN(ADBUSER01.T2)
```

[説明]

SELECT 文中に集合演算が指定されていることを示しています。

なお，SET OPERATION の後ろに，次の情報が表示されることがあります。

- RECURSIVE

  再帰的問合せが実行されることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 3
     2   SELECT STATEMENT
     3   +-DERIVED TABLE(REC)
     4     +-SET OPERATION -RECURSIVE
     5       |-QUERY SCAN(QUERY 1)
     6       +-QUERY SCAN(QUERY 2)
     7
     8 QUERY : 1
     9   QUERY
    10   |-CREATE GLOBAL WORK TABLE(WORK TABLE 1)
    11   | +-TABLE SCAN(ADBUSER01.T1)
    12   +-WORK TABLE SCAN(WORK TABLE 1)
    13
    14 QUERY : 2
    15   QUERY
    16   |-CREATE GLOBAL WORK TABLE(WORK TABLE 2)
    17   | +-WORK TABLE SCAN(WORK TABLE 2)
    18   +-WORK TABLE SCAN(WORK TABLE 2)
```

[説明]

再帰的問合せが実行されることを示しています。

SET OPERATION に続いて表示される最初の問合せ情報が，アンカーメンバの問合せ情報です。次の問合せ情報が，再帰的メンバの問合せ情報です。上記の出力例の場合，QUERY 1 がアンカーメンバの問合せ情報で，QUERY 2 が再帰的メンバの問合せ情報です。

(5) 集合演算方式指定

次の情報が出力されます。

- SPECIFIC

  集合演算方式指定が有効になることを示しています。

  集合演算方式指定については，マニュアル『HADB　SQL リファレンス』の『問合せ式の指定形式および規則』を参照してください。

出力例

```
<<Tree View>>

     1 QUERY : 0
     2   SELECT STATEMENT
     3   +-SET OPERATION -SPECIFIC
     4     |-QUERY SCAN(QUERY 1)
     5     +-QUERY SCAN(QUERY 2)
     6
     7 QUERY : 1
     8   QUERY
     9   |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
    10   | |-TABLE SCAN(ADBUSER01.T1)
    11   | +-SORTING BYTE -DISTINCT
    12   +-WORK TABLE SCAN(WORK TABLE 1)
    13
    14 QUERY : 2
    15   QUERY
    16   |-CREATE LOCAL WORK TABLE(WORK TABLE 2)
    17   | |-TABLE SCAN(ADBUSER01.T2)
    18   | +-SORTING BYTE -DISTINCT
    19   +-WORK TABLE SCAN(WORK TABLE 2)
```

[説明]

SELECT 文に指定した集合演算方式指定が有効になることを示しています。

(6) SQL パラレル実行機能の適用情報

次のどちらかの情報が出力されます。

- PARALLEL

  検索系 SQL に対して，SQL パラレル実行機能が適用されることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT -PARALLEL
     3   |-TABLE SCAN(ADBUSER01.T1) -COLUMN STORE -USING COST -UNNEST
     4   +-GROUPING
```

[説明]

このSELECT 文に対して SQL パラレル実行機能が適用されることを示しています。

- PARALLEL DISABLED

  検索系 SQL に対して，SQL パラレル実行機能が適用されないことを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT -PARALLEL DISABLED
     3   |-TABLE SCAN(ADBUSER01.T1) -COLUMN STORE -USING COST
     4   +-GROUPING -COLUMN
```

[説明]

このSELECT 文に対しては SQL パラレル実行機能が適用されないことを示しています。

なお，PARALLEL またはPARALLEL DISABLED が出力されるのは，次のどちらかの SQL パラレル実行機能を使用する指定をしている場合だけです。

- クライアント定義のadb_clt_sql_parallel_exec オペランドにY を指定している
- SQL 文中に SQL パラレル実行指定を指定している

(7) 問合せの種類

次の情報が出力されます。

- QUERY

  副問合せおよび導出表以外の問合せ式本体が指定されていることを示しています。

> **メモ**
>
> UPDATE 文の更新対象表，またはDELETE 文の削除対象表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）を検索する問合せが表示されることがあります。

出力例

```
<<Tree View>>

     1 QUERY : 0
     2   INSERT STATEMENT
     3   +-QUERY SCAN(QUERY 1)
     4
     5 QUERY : 1
     6   QUERY
     7   +-TABLE SCAN(ADBUSER01.T2) -ORDER
```

［説明］

INSERT 文中に問合せ式本体が指定されていることを示しています。

(8) 作業表の作成情報

次のどちらかの情報が出力されます。

- CREATE GLOBAL WORK TABLE (WORK TABLE *作業表番号*)

  グローバル作業表が作成されることを示しています。
- CREATE LOCAL WORK TABLE (WORK TABLE *作業表番号*)

  ローカル作業表が作成されることを示しています。

作業表番号とは，作業表を一意に識別するために割り当てられた番号のことです。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-SUBQUERY
     4   | +-CREATE GLOBAL WORK TABLE(WORK TABLE 1)
     5   |   |-QUERY SCAN(QUERY 2)
     6   |   +-SORTING BYTE
     7   +-INDEX SCAN(ADBUSER01.T1)
     8     +-SUBQUERY
     9       +-WORK TABLE SCAN(WORK TABLE 1)
    10
    11 QUERY : 2
    12   SUBQUERY
    13   +-TABLE SCAN(ADBUSER01.T2)
```

［説明］

副問合せの処理で，グローバル作業表が作成されることを示しています。

(9) 副問合せ処理方式指定

次の情報が出力されます。

- SPECIFIC

  副問合せ処理方式指定が有効になることを示しています。

  副問合せ処理方式指定については，マニュアル『HADB　SQL リファレンス』の『副問合せの指定形式および規則』を参照してください。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
```

```
 3   |-SUBQUERY -SPECIFIC
 4   | +-CREATE GLOBAL WORK TABLE(WORK TABLE 1)
 5   |   +-QUERY SCAN(QUERY 2)
 6   +-TABLE SCAN(ADBUSER01.T1(A))
 7     +-SUBQUERY
 8       +-WORK TABLE SCAN(WORK TABLE 1)
 9
10 QUERY : 2
11   SUBQUERY
12   +-TABLE SCAN(ADBUSER01.T2)
```

[説明]

SELECT 文に指定した副問合せ処理方式指定が有効になることを示しています。

(10) 副問合せ処理委譲指定

次の情報が出力されます。

- DELEGATION

  副問合せ処理委譲指定が有効になることを示しています。

  副問合せ処理委譲指定については，マニュアル『HADB SQL リファレンス』の『副問合せの指定形式および規則』を参照してください。

出力例

```
<<Tree View>>

 1 QUERY : 1
 2   SELECT STATEMENT
 3   +-TABLE SCAN(ADBUSER01.T1(A))
 4     +-SUBQUERY LOOP -DELEGATION -USING CACHE
 5       +-QUERY SCAN(QUERY 2)
 6
 7 QUERY : 2
 8   SUBQUERY LOOP
 9   |-SUBQUERY LOOP -SPECIFIC -DELEGATION
10   | +-QUERY SCAN(QUERY 3)
11   +-TABLE SCAN(ADBUSER01.T1(B))
12
13 QUERY : 3
14   SUBQUERY LOOP
15   +-TABLE SCAN(ADBUSER01.T1(C))
```

[説明]

SELECT 文に指定した副問合せ処理委譲指定が有効になることを示しています。

(11) 副問合せのキャッシュ使用情報

次の情報が出力されます。

- USING CACHE

副問合せの結果を格納するためのキャッシュが使用されることを示しています。副問合せの処理方式でネストループ行値実行が適用された場合に，キャッシュが使用されることがあります。

ネストループ行値実行については，「5.6.3　外への参照列を含む副問合せの処理方式とは」の「(2)　ネストループ行値実行」を参照してください。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(ADBUSER01.T1)
     4     +-SUBQUERY LOOP -USING CACHE
     5       +-QUERY SCAN(QUERY 2)
     6
     7 QUERY : 2
     8   SUBQUERY LOOP
     9   +-TABLE SCAN(ADBUSER01.T2)
```

[説明]

副問合せの処理方式でネストループ行値実行が適用されて，副問合せの結果を格納するためのキャッシュが使用されることを示しています。

(12)　表関数導出表の指定

次の情報が出力されます。

- TABLE FUNCTION DERIVED TABLE(*相関名*)

  表関数導出表が指定されていることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE FUNCTION DERIVED TABLE(T4)
     4     +-TABLE SCAN(ADBUSER01.T4)
```

[説明]

SELECT 文中に表関数導出表が指定されていることを示しています。（　）内には，相関名が表示されます。

(13)　重複排除の処理方式

次の情報が出力されます。

- GLOBAL HASH UNIQUE

  次のどれかの処理によって検索結果の重複を排除することを示しています。

- 集合演算の処理方式のハッシュ実行

  集合演算の処理方式のハッシュ実行については，「5.8.1　ハッシュ実行」を参照してください。

- SELECT DISTINCT の処理方式のハッシュ実行

  SELECT DISTINCT の処理方式のハッシュ実行については，「5.9.1　ハッシュ実行」を参照してください。

- グループ化の処理方式のグローバルハッシュグループ化

  グループ化の処理方式のグローバルハッシュグループ化については，「5.7.1　ハッシュグループ化とは」の「(2)　グローバルハッシュグループ化」を参照してください。

また，ハッシュテーブル領域で処理できない行があるため，作業表が作成される可能性がある場合にも GLOBAL HASH UNIQUE が表示されます。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-KEY SCAN(ADBUSER01.T1)
     4   |-GLOBAL HASH UNIQUE
     5   +-GROUPING
```

[説明]

グループ化の処理方式のグローバルハッシュグループ化を適用して検索結果の重複を排除することを示しています。

(14)　グループ化の処理方式

次のどれかの情報が出力されます。

- GROUPING

  作業表を使用しないグループ化が実行されることを示しています。

- SORT GROUPING

  ソートグループ化が実行されることを示しています。

- GLOBAL HASH GROUPING

  グローバルハッシュグループ化が実行されることを示しています。また，ハッシュテーブル領域で処理できない行がある場合，作業表を作成することがあることも示しています。

- LOCAL HASH GROUPING

  ローカルハッシュグループ化が実行されることを示しています。また，ハッシュグループ化領域で処理できない行がある場合，作業表を作成することがあることも示しています。

グループ化の処理方式については，「5.7　グループ化の処理方式」を参照してください。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-TABLE SCAN(ADBUSER01.T1)
     4   +-GLOBAL HASH GROUPING
```

[説明]

グループ化の処理方式で，グローバルハッシュグループ化が実行されることを示しています。

なお，グループ化の処理方式の各情報の後ろに，次の情報が表示されることがあります。

- SPECIFIC

  GROUP BY 句に指定したグループ化方式指定が有効になることを示しています。

  出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
     4   | |-TABLE SCAN(ADBUSER01.T1)
     5   | +-SORTING BYTE
     6   |-WORK TABLE SCAN(WORK TABLE 1)
     7   +-LOCAL HASH GROUPING -SPECIFIC
```

  [説明]

  GROUP BY 句に指定したグループ化方式指定が有効になり，グループ化の処理方式にLOCAL HASH GROUPING（ローカルハッシュグループ化）が適用されることを示しています。グループ化方式指定については，マニュアル『HADB　SQL リファレンス』の『GROUP BY 句の指定形式および規則』を参照してください。

- INDEX

  B-tree インデクスの特性を利用したグループ化の処理が実行されることを示しています。

  出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-KEY SCAN(ADBUSER01.T1)
     4   +-GROUPING -INDEX
```

  [説明]

  B-tree インデクスの特性を利用したグループ化の処理が，表T1 に対して実行されることを示しています。

- COLUMN

  カラムストア表の特性を利用したグループ化の処理が実行されることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-TABLE SCAN(ADBUSER01.T1) -COLUMN STORE
     4   +-GROUPING -COLUMN
```

[説明]

カラムストア表の特性を利用したグループ化の処理が，表T1 に対して実行されることを示しています。

- GROUPING SET

グループ化処理が複数回実行されることを示しています。

出力例

```
<<Tree View>>
     1 QUERY : 1
     2 SELECT STATEMENT
     3 |-DERIVED TABLE(##DRVTBL_0000000001)
     4 | |-TABLE SCAN(ADBUSER01.T1)
     5 | +-GLOBAL HASH GROUPING -GROUPING SET
     6 +-GLOBAL HASH GROUPING
```

[説明]

グローバルハッシュグループ化で，グループ化処理が複数回実行されることを示しています。

メモ

引数が異なるDISTINCT 集合関数を複数指定した場合に，GROUPING SET（グループ化集合情報）が出力されることがあります。

(15) HAVING 句の指定

次の情報が出力されます。

- HAVING

HAVING 句が指定されていることを示しています。また，HAVING 句を指定していない場合でも，導出表の展開によって表示されることがあります。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-TABLE SCAN(ADBUSER01.T1)
     4   |-GLOBAL HASH GROUPING
     5   +-HAVING
```

[説明]

SELECT 文中にHAVING 句が指定されていることを示しています。

(16) ソート処理の実行

次の情報が出力されます。

- SORTING {BYTE | ISO}
  - BYTE：バイトコード順に並び替えます。
  - ISO：ソートコード順（ISO/IEC14651:2011 準拠）に並び替えます。

  ORDER BY 句によるソート処理が実行されることを示しています。

  ただし，ORDER BY 句を指定した場合でも表示されないときがあります。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
     4   | |-TABLE SCAN(ADBUSER01.T1)
     5   | +-SORTING BYTE
     6   +-WORK TABLE SCAN(WORK TABLE 1)
```

[説明]

ORDER BY 句の指定によって，ソート処理が実行されることを示しています。

(17) 重複排除の情報

次の情報が出力されます。

- DISTINCT

  重複排除処理が行われることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2 SELECT STATEMENT
     3 |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
     4 | |-KEY SCAN(ADBUSER01.T1)
     5 | +-SORTING BYTE -DISTINCT
     6 +-WORK TABLE SCAN(WORK TABLE 1)
```

[説明]

ソート処理の中で，重複排除処理が実行されることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2 SELECT STATEMENT
     3 |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
     4 | |-KEY SCAN(ADBUSER01.T1) -DISTINCT
     5 | +-SORTING BYTE -DISTINCT
     6 +-WORK TABLE SCAN(WORK TABLE 1)
```

[説明]

キースキャン処理およびソート処理の中で，重複排除処理が実行されることを示しています。

(18) SELECT 重複排除方式指定

次の情報が出力されます。

- SPECIFIC

  SELECT 重複排除方式指定が有効になることを示しています。

  SELECT 重複排除方式指定については，マニュアル『HADB SQL リファレンス』の『問合せ指定の指定形式および規則』を参照してください。

  なお，「(17) 重複排除の情報」（DISTINCT）が出力されない場合は，SELECT 重複排除方式指定を指定してもSPECIFIC は出力されません。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
     4   | |-TABLE SCAN(ADBUSER01.T1)
     5   | +-SORTING BYTE -SPECIFIC -DISTINCT
     6   +-WORK TABLE SCAN(WORK TABLE 1)
```

[説明]

SELECT 文に指定したSELECT 重複排除方式指定が有効になることを示しています。

(19) LIMIT 句の指定

次の情報が出力されます。

- LIMIT 〔 {*オフセット行数* | ? PARAMETER} ,〕 {*リミット行数* | ? PARAMETER}
  - *オフセット行数*：

    オフセット行数を指定したLIMIT 句が指定されていることを示しています。

    オフセット行数に定数で0 を指定した場合，オフセット行数は表示されません。
  - *リミット行数*：

リミット行数を指定したLIMIT 句が指定されていることを示しています。

- ? PARAMETER：

  オフセット行数とリミット行数の両方，またはどちらか一方に？パラメタを指定したLIMIT 句が指定されていることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
     4   | |-TABLE SCAN(ADBUSER01.T1)
     5   | |-SORTING BYTE
     6   | +-LIMIT ? PARAMETER,5
     7   |-WORK TABLE SCAN(WORK TABLE 1)
     8   +-LIMIT ? PARAMETER,5
```

[説明]

オフセット行数に？パラメタを指定し，リミット行数に5 を指定したLIMIT 句が指定されていることを示しています。

(20) ウィンドウ関数の指定

次の情報が出力されます。

- WINDOW

  ウィンドウ関数が指定されていることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-CREATE LOCAL WORK TABLE(WORK TABLE 1)
     4   | |-TABLE SCAN(ADBUSER01.T1)
     5   | +-SORTING BYTE
     6   |-WORK TABLE SCAN(WORK TABLE 1)
     7   +-WINDOW
```

[説明]

ウィンドウ関数の指定によって，ウィンドウ関数が実行されることを示しています。

(21) 表の検索方式

次のどれかの情報が出力されます。

- TABLE SCAN

  表の検索処理で，テーブルスキャンが実行されることを示しています。

- INDEX SCAN(*スキーマ名.表識別子*(*問合せ名または相関名*))

  表の検索処理で，インデクススキャンが実行されることを示しています。問合せ名または相関名がある場合は，問合せ名または相関名が表示されます。

- KEY SCAN

  表の検索処理で，キースキャンが実行されることを示しています。

テーブルスキャン，インデクススキャン，およびキースキャンについては，「5.1　表の検索方式」を参照してください。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-INDEX SCAN(ADBUSER01.T1)
```

[説明]

表T1 の検索処理で，インデクススキャンが実行されることを示しています。

(22)　表データの格納形式

次の情報が出力されます。

- COLUMN STORE

  表データの格納形式がカラムストア形式であることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(ADBUSER01.T1) -COLUMN STORE
```

[説明]

表T1 のデータの格納形式がカラムストア形式であることを示しています。

(23)　順序実行方式

次の情報が出力されます。

- ORDER

  非順序実行方式が適用されないで，順序実行方式が適用されることを示しています。

出力例

```
<<Tree View>>
```

```
     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-NESTED LOOP JOIN
     4     |-TABLE SCAN(ADBUSER01.T2)
     5     +-INDEX SCAN(ADBUSER01.T1) -ORDER
```

[説明]

表T1 の検索処理で，インデクススキャンが順序実行方式で実行されることを示しています。

(24) インデクス指定

次のどちらかの情報が出力されます。

- SPECIFIC

  インデクス指定が有効になることを示しています。

- SPECIFIC DISABLED

  インデクス指定が無効になることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-INDEX SCAN(ADBUSER01.T1) -SPECIFIC
```

[説明]

SELECT 文に指定したインデクス指定が有効になり，表T1 の検索処理で，インデクススキャンが実行されることを示しています。

(25) コスト情報の収集

次の情報が出力されます。

- USING COST

  表またはインデクスに関するコスト情報が収集されていることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(ADBUSER01.T3) -USING COST
```

[説明]

表T3 に関するコスト情報が収集されていることを示しています。

(26) 集まり導出表の指定

次の情報が出力されます。

- UNNEST

  集まり導出表が指定されていることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(ADBUSER01.T1) -COLUMN STORE -UNNEST
```

[説明]

SQL 文中に集まり導出表が指定されていることを示しています。

(27) 作業表の検索実行

次の情報が出力されます。

- WORK TABLE SCAN(WORK TABLE *作業表番号*)

  作業表の検索が実行されることを示しています。作業表番号とは，作業表を一意に識別するために割り当てられた番号のことです。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-SUBQUERY
     4   | +-CREATE GLOBAL WORK TABLE(WORK TABLE 1)
     5   |   |-QUERY SCAN(QUERY 2)
     6   |   +-SORTING BYTE
     7   +-INDEX SCAN(ADBUSER01.T1)
     8     +-SUBQUERY
     9       +-WORK TABLE SCAN(WORK TABLE 1)
    10
    11 QUERY : 2
    12   SUBQUERY
    13   +-TABLE SCAN(ADBUSER01.T2)
```

[説明]

副問合せの処理で，作業表の検索が実行されることを示しています。

なお，ORDER BY 句を指定した SQL 文の実行時に作成された作業表に，FROM 句に指定した表の行 ID が格納される場合，作業表の検索直後に行 ID を使用してデータページからデータを取り出すための検索が実行されることがあります。作業表の用途と構成列については，「5.10　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。

(28)　問合せ検索の実行

次の情報が出力されます。

- QUERY SCAN(QUERY *問合せツリー番号*)
  問合せ検索が実行されることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-SUBQUERY HASH
     4     |-QUERY SCAN(QUERY 2)
     5     +-TABLE SCAN(ADBUSER01.T1)
     6
     7 QUERY : 2
     8   SUBQUERY HASH
     9   +-TABLE SCAN(ADBUSER01.T2)
```

[説明]

問合せ検索が実行されることを示しています。

ツリー行番号 7 に表示されている「QUERY : 2」の問合せ検索を行うことを示しています。

実行した SELECT 文の例

```
SELECT * FROM "T1" WHERE "C1"=(SELECT "C2" FROM "T2" WHERE "C1"="T1"."C1")
```

(29)　表の結合方式

次のどちらかの情報が出力されます。

- NESTED LOOP JOIN
  表の結合処理で，ネストループジョインが実行されることを示しています。
- HASH JOIN
  表の結合処理で，ハッシュジョインが実行されることを示しています。
  HASH JOIN の後ろにFILTER が表示されている場合，ハッシュジョインの際にハッシュフィルタが適用されることを示しています。
  HASH JOIN の後ろにDIVIDE が表示されている場合，ハッシュジョインが分割実行されることを示しています。
  2 つの表の列を比較する=条件がある場合，ハッシュジョインが実行されることがあります。また，ハッシュテーブル領域で処理できない行がある場合，作業表が作成されることがあります。

表の結合方式については，「5.5　表の結合方式」を参照してください。

出力例（NESTED LOOP JOIN の場合）

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-NESTED LOOP JOIN
     4     |-TABLE SCAN(ADBUSER01.T2)
     5     +-INDEX SCAN(ADBUSER01.T1) -ORDER
```

[説明]

- 表T1 と表T2 の結合処理で，ネストループジョインが実行されることを示しています。
- 表の結合方式がネストループジョインの場合，NESTED LOOP JOIN の下位に表示される情報は，外表から順に表示されます。上記の例の場合，ツリー行番号が 4 の情報が外表の情報で，ツリー行番号が 5 の情報が内表の情報です。

出力例（HASH JOIN の場合）

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-HASH JOIN -FILTER -DIVIDE
     4     |-TABLE SCAN(ADBUSER01.T1)
     5     +-TABLE SCAN(ADBUSER01.T2)
```

[説明]

- 表T1 と表T2 の結合処理で，ハッシュジョインが実行されることを示しています。
- 表の結合方式がハッシュジョインの場合，HASH JOIN の下位に表示される情報は，ハッシュジョインの外表，内表の順に表示されます。上記の例の場合，ツリー行番号が 4 の情報が外表の情報で，ツリー行番号が 5 の情報が内表の情報です。
- ツリー行番号が 3 のHASH JOIN の後ろにFILTER が表示されているため，ハッシュジョインの際にハッシュフィルタが適用されることを示しています。
- ツリー行番号が 3 のHASH JOIN の後ろにDIVIDE が表示されているため，ハッシュジョインが分割実行されることを示しています。

(30) 結合方式指定

次のどちらかの情報が出力されます。

- SPECIFIC

  結合方式指定が有効になることを示しています。
- SPECIFIC DISABLED

  結合方式指定が無効になることを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-HASH JOIN -SPECIFIC
     4     |-TABLE SCAN(ADBUSER01.T1(A))
     5     +-TABLE SCAN(ADBUSER01.T2(B))
```

[説明]

SELECT 文に指定した結合方式指定が有効になり，表の結合処理でハッシュジョインが実行されることを示しています。

(31) 表値構成子の検索情報

次の情報が出力されます。

- TABLE VALUE CONSTRUCTOR SCAN

  表値構成子を検索することを示しています。

出力例

```
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-NESTED LOOP JOIN
     4     |-TABLE SCAN(ADBUSER01.T1)
     5     +-DERIVED TABLE(DT)
     6     +-TABLE VALUE CONSTRUCTOR SCAN
```

[説明]

表値構成子を検索することを示しています。

6.1.5 詳細表示に出力される情報

詳細表示には，次の情報が出力されます。

- 表の検索方式，インデクス，および集まり導出表に関する情報
- 表の結合方式に関する情報
- 集合演算に関する情報
- 表関数導出表に関する情報
- 副問合せに関する情報
- グループ化に関する情報

なお，詳細表示は，問合せツリーに出力されている問合せに詳細情報がある場合に限り出力されます。

(1)　表の検索方式，インデクス，および集まり導出表に関する情報

表の検索方式，インデクス，および集まり導出表に関する情報の出力形式の例を次に示します。

出力形式の例 1

```
<<Detail   >>

QUERY : 1
     3 INDEX SCAN(ADBUSER01.T1)   <-- 表の検索方式に関する情報
       INDEX NAME          : IDX_C1C2
        INDEX TYPE         : B-TREE
        INDEX COLUMN       : C1 ASC (IN)
        INDEX COLUMN       : C2 ASC (>)
       INDEX NAME          : RIDX_C2
        INDEX TYPE         : RANGE
        SKIP COND          : CHUNK (HASH)
        INDEX COLUMN       : C2
```

出力形式の例 2

```
<<Detail   >>

QUERY : 1
     3 TABLE SCAN(ADBUSER01.T1)   <-- 表の検索方式に関する情報
       INDEX NAME          : RNG
        INDEX TYPE         : RANGE
        SKIP COND          : CHUNK AND SEGMENT
        INDEX COLUMN       : C1
       UNNEST DERIVED TABLE SCAN(DT)   <-- 集まり導出表に関する情報
```

(a)　表の検索方式に関する情報

表の検索方式に関する情報には，次のどれかが出力されます。

- TABLE SCAN

  表の検索処理で，テーブルスキャンが実行されることを示しています。

- INDEX SCAN(*スキーマ名*.*表識別子*(*問合せ名または相関名*))

  表の検索処理で，インデクススキャンが実行されることを示しています。問合せ名または相関名がある場合は，問合せ名または相関名が表示されます。

- KEY SCAN

  表の検索処理で，キースキャンが実行されることを示しています。

テーブルスキャン，インデクススキャン，およびキースキャンについては，「5.1　表の検索方式」を参照してください。

出力例

```
<<Detail   >>

QUERY : 1
     3 INDEX SCAN(ADBUSER01.T1)
       INDEX NAME         : IDX_C1C2
        INDEX TYPE        : B-TREE
        INDEX COLUMN      : C1 ASC (IN)
        INDEX COLUMN      : C2 ASC (>)
       INDEX NAME         : RIDX_C2
        INDEX TYPE        : RANGE
        SKIP COND         : CHUNK (HASH)
        INDEX COLUMN      : C2
```

[説明]

表T1 の検索処理で，インデクススキャンが実行されることを示しています。

(b) インデクスに関する情報

インデクスに関する情報の出力形式を次に示します。

出力形式（B-tree インデクスの場合）

```
INDEX NAME         : B-treeインデクス名   （一意性制約情報）
 INDEX TYPE        : インデクスの種別
 INDEX COLUMN      : インデクス構成列名   キー値の並び順   （サーチ条件として評価する条件の種別）
```

- *一意性制約情報*

  この B-tree インデクスがユニークインデクスの場合，一意性制約情報が出力されます。次のどちらかが出力されます。

  UNIQUE：このユニークインデクスが，一意性制約に違反していない状態であることを示しています。

  UNIQUE INVALID：このユニークインデクスが，一意性制約に違反している状態であることを示しています。

- *インデクスの種別*

  B-tree インデクスの場合，インデクスの種別にB-TREE が表示されます。

- *キー値の並び順*

  B-tree インデクスの定義時に指定した B-tree インデクスのキー値の並び順が出力されます。次のどちらかが出力されます。

  ASC：昇順にキー値が並んでいます。

  DESC：降順にキー値が並んでいます。

- *サーチ条件として評価する条件の種別*

  インデクス構成列に対してサーチ条件として評価する条件がある場合，その条件の種別を次のどれかの形式で出力します。

  =, <, <=, >, >=, =ANY, BETWEEN({<,< | <,<= | <=,< | <=,<=}), IN, LIKE, IS NULL

サーチ条件については，「5.4.1　B-tree インデクスによる評価方式」の「(1)　サーチ条件とは」を参照してください。

サーチ条件として評価する条件の種別の出力規則

- インデクス構成列に対してサーチ条件として評価する条件がない場合は，none が出力されます。
- 「IN *表副問合せ*」や「*限定述語*の=SOME」をサーチ条件として評価する場合は，=ANY が出力されます。
- 比較述語の左側に単独の列指定以外を指定した場合，探索条件が HADB サーバによって等価変換されます。このとき，サーチ条件として評価する条件の種別には，等価変換後の比較述語の比較演算子が出力されます。

  (例)

  SELECT 文のWHERE 句に指定された探索条件

  WHERE 10 ≦ C1

  HADB サーバによって等価変換された探索条件

  WHERE C1 ≧ 10

  サーチ条件として評価する条件の種別に出力される情報

  INDEX COLUMN: C1 ASC (≧)

出力形式（テキストインデクスの場合）

```
INDEX NAME        : テキストインデクス名
 INDEX TYPE       : インデクスの種別
 INDEX COLUMN     : インデクス構成列名
```

- *インデクスの種別*

  テキストインデクスの場合，インデクスの種別にTEXT が表示されます。

出力形式（レンジインデクスの場合）

```
INDEX NAME        : レンジインデクス名
 INDEX TYPE       : インデクスの種別
 SKIP COND        : レンジインデクス条件の種別
 INDEX COLUMN     : インデクス構成列名
```

- *インデクスの種別*

  レンジインデクスの場合，インデクスの種別にRANGE が表示されます。

- *レンジインデクス条件の種別*

  使用されるレンジインデクス条件として，次のどれかが出力されます。

  - CHUNK：チャンクのスキップ条件を使用します。
  - SEGMENT：セグメントのスキップ条件を使用します。
  - CHUNK AND SEGMENT：チャンクのスキップ条件とセグメントのスキップ条件の両方を使用します。

また，ハッシュジョインまたは副問合せの処理方式のハッシュ実行の際に，ハッシュテーブルとの突き合わせを行う条件に指定された列に対して，レンジインデクスが使用される場合，(HASH)が出力されます。

出力例

```
SKIP COND         : CHUNK (HASH)
```

ハッシュジョインの詳細については，「5.5.2　ハッシュジョインとは」を参照してください。

副問合せの処理方式のハッシュ実行については，「5.6.1　外への参照列を含まない副問合せの処理方式とは」の「(4)　ハッシュ実行」，または「5.6.3　外への参照列を含む副問合せの処理方式とは」の「(3)　ハッシュ実行」を参照してください。

出力例

```
<<Detail   >>

QUERY : 1
     3 INDEX SCAN(ADBUSER01.T1)
       INDEX NAME         : IDX_C1C2          ...1
        INDEX TYPE        : B-TREE            ...2
        INDEX COLUMN      : C1 ASC (IN)       ...3
        INDEX COLUMN      : C2 ASC (>)        ...3
       INDEX NAME         : RIDX_C2           ...4
        INDEX TYPE        : RANGE             ...5
        SKIP COND         : CHUNK (HASH)      ...6
        INDEX COLUMN      : C2                ...7
```

[説明]

1. 使用されるインデクスの名称です。
2. 1.のINDEX NAME に表示されたインデクスの種別です。B-TREE が表示されているため，IDX_C1C2 は B-tree インデクスです。

   1.および 2.の情報から，B-tree インデクスIDX_C1C2 を使用して，インデクススキャンが実行されることを示しています。
3. B-tree インデクスIDX_C1C2 についての情報です。

   C1，C2：インデクス構成列

   ASC：キー値の並び順

   (IN)，(>)：サーチ条件として評価する条件の種別
4. 使用されるインデクスの名称です。
5. 4.のINDEX NAME に表示されたインデクスの種別です。RANGE が表示されているため，RIDX_C2 はレンジインデクスです。
6. 使用されるレンジインデクス条件の種別です。

   CHUNK：レンジインデクスRIDX_C2 がチャンクのスキップ条件で使用されることを示しています。

   (HASH)：ハッシュジョインまたは副問合せの処理方式のハッシュ実行の際に，ハッシュテーブルとの突き合わせを行う条件に指定された列に対して，レンジインデクスRIDX_C2 が使用されることを示しています。

7. レンジインデクスRIDX_C2 のインデクス構成列です。

メモ

- B-tree インデクスの情報は，表の検索方式にインデクススキャンまたはキースキャンが実行される場合に表示されます。

  また，インデクス構成列が複数ある場合は，インデクス構成列ごとに情報が出力されます。この場合，CREATE INDEX 文で指定したインデクス構成列の定義順に情報が出力されます。
- テキストインデクスのインデクス情報は，表の検索方式にインデクススキャンが実行される場合に表示されます。
- レンジインデクスの情報は，レンジインデクス条件がある場合に表示されます。

(c) 集まり導出表に関する情報

SQL 文中に集まり導出表が指定されている場合，次の集まり導出表に関する情報が出力されます。

- UNNEST DERIVED TABLE SCAN(*相関名*)

  SQL 文中に集まり導出表が指定されていることを示しています。また，指定されている集まり導出表の相関名が表示されます。

  複数の集まり導出表を指定している場合，集まり導出表の指定順にこの情報が表示されます。

集まり導出表については，マニュアル『HADB　SQL リファレンス』の『表参照』を参照してください。

出力例

```
<<Detail   >>

QUERY : 1
     3 TABLE SCAN(ADBUSER01.T1)
       INDEX NAME         : RNG
        INDEX TYPE        : RANGE
        SKIP COND         : CHUNK AND SEGMENT
        INDEX COLUMN      : C1
       UNNEST DERIVED TABLE SCAN(DT)
```

[説明]

SQL 文中に集まり導出表が指定されていることを示しています。指定している集まり導出表の相関名はDT であることを示しています。

(2) 表の結合方式に関する情報

表の結合方式に関する情報の出力形式の例を次に示します。

出力形式の例

```
<<Detail   >>
```

```
QUERY : 1
     3 HASH JOIN                                  ←表の結合方式
       JOIN TYPE          : INNER JOIN           ←結合種別
       BUILD COLUMN       : ADBUSER01.T1.C3                ←ハッシュ検索情報
       PROBE COLUMN       : ADBUSER01.T2.C3                ←ハッシュ検索情報
```

(a)　表の結合方式

表の結合方式には，次のどちらかが出力されます。

- NESTED LOOP JOIN

  表の結合処理で，ネストループジョインが実行されることを示しています。

- HASH JOIN

  表の結合処理で，ハッシュジョインが実行されることを示しています。

  2 つの表の列を比較する=条件がある場合，ハッシュジョインが実行されることがあります。また，ハッシュテーブル領域で処理できない行がある場合，作業表が作成されることがあります。

表の結合方式については，「5.5　表の結合方式」を参照してください。

出力例

```
<<Detail   >>

QUERY : 1
     3 HASH JOIN
       JOIN TYPE          : INNER JOIN
       BUILD COLUMN       : ADBUSER01.T1.C3
       PROBE COLUMN       : ADBUSER01.T2.C3
```

[説明]

表の結合方式で，ハッシュジョインが実行されることを示しています。

(b)　結合種別（JOIN TYPE）

JOIN TYPE には，次のどれかが出力されます。

- CROSS JOIN

  交差結合が指定されていることを示しています。

- INNER JOIN

  内結合が指定されていることを示しています。

- LEFT OUTER JOIN

  LEFT OUTER JOIN による外結合が指定されていることを示しています。

- RIGHT OUTER JOIN

  RIGHT OUTER JOIN による外結合が指定されていることを示しています。

- FULL OUTER JOIN(LEFT)

FULL OUTER JOIN による外結合が指定されていることを示しています。

- FULL OUTER JOIN(RIGHT)

  FULL OUTER JOIN による外結合が指定されていることを示しています。

各結合の方式については，マニュアル『HADB　SQL リファレンス』の『結合表の指定形式および規則』を参照してください。

出力例

```
<<Detail   >>

QUERY : 1
     3 HASH JOIN
       JOIN TYPE          : INNER JOIN
       BUILD COLUMN       : ADBUSER01.T1.C3
       PROBE COLUMN       : ADBUSER01.T2.C3
```

[説明]

内結合が実行されることを示しています。

重要

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て SQL 文が実行されることがあります。この場合，結合種別（JOIN TYPE）は表示されません。コンマ結合については，マニュアル『HADB　SQL リファレンス』の『FROM 句の指定形式および規則』の『指定形式の説明』を参照してください。

出力規則を次に示します。

- 内部導出表を作成する SQL 文で内部導出表を展開する場合，ハッシュジョインの外表列および内表列の表名や相関名などに，内部導出表の展開結果が出力されます。内部導出表および内部導出表の展開規則については，マニュアル『HADB　SQL リファレンス』の『内部導出表』を参照してください。

(c)　ハッシュ検索情報（BUILD COLUMN，PROBE COLUMN）

ハッシュジョインが実行される場合，ハッシュ検索情報には，次の情報が出力されます。

- BUILD COLUMN

  外表の結合列の情報が出力されます。

- PROBE COLUMN

  内表の結合列の情報が出力されます。

出力例

```
<<Detail   >>

QUERY : 1
     3 HASH JOIN
```

```
      BUILD COLUMN      : ADBUSER01.T1.C1(CREATE FILTER 1)    ...1
      PROBE COLUMN      : ADBUSER01.T2.C1(USE FILTER 1)       ...2
```

[説明]

1. ハッシュジョインの際の，外表の結合列の列名（ADBUSER01.T1.C1）が出力されます。
2. ハッシュジョインの際の，内表の結合列の列名（ADBUSER01.T2.C1）が出力されます。

ハッシュジョインの際にハッシュフィルタが適用される場合，下線部分にハッシュフィルタの情報が出力されます。

出力規則を次に示します。

- BUILD COLUMN またはPROBE COLUMN には，次のどれかの形式で列名が出力されます。このうち，相関名の出力が優先されます。列名を表示できない場合は，***が出力されます。
  - *表名.列名*
  - *問合せ名.列名*
  - *相関名.列名*
- ハッシュジョインの際にハッシュフィルタが適用される場合，BUILD COLUMN には，「(CREATE FILTER XXXXX)」が出力されます。「XXXXX」は，出力された列名の列値を基に作成したハッシュフィルタの番号です。

  PROBE COLUMN には，「(USE FILTER XXXXX)」が出力されます。「XXXXX」は，使用するハッシュフィルタの番号です。
- 内部導出表を作成する SQL 文で内部導出表を展開する場合，ハッシュジョインの外表列および内表列の表名や相関名などに，内部導出表の展開結果が出力されます。内部導出表および内部導出表の展開規則については，マニュアル『HADB　SQL リファレンス』の『内部導出表』を参照してください。

(3)　集合演算に関する情報

集合演算に関する情報には，実行される集合演算の種類が，次の形式で出力されます。

SET OPERATION TYPE:集合演算の種別

集合演算の種別には，次のどれかが出力されます。

- UNION ALL

  集合演算UNION ALL が指定されていることを示しています。
- UNION DISTINCT

  集合演算UNION DISTINCT が指定されていることを示しています。
- EXCEPT ALL

  集合演算EXCEPT ALL が指定されていることを示しています。
- EXCEPT DISTINCT

  集合演算EXCEPT DISTINCT が指定されていることを示しています。

- INTERSECT ALL

  集合演算INTERSECT ALL が指定されていることを示しています。

- INTERSECT DISTINCT

  集合演算INTERSECT DISTINCT が指定されていることを示しています。

集合演算に関する情報の出力形式の例を次に示します。

出力形式の例

```
<<Detail   >>

QUERY : 0
     3 SET OPERATION
       SET OPERATION TYPE : UNION ALL
```

[説明]

集合演算UNION ALL が指定されていることを示しています。

留意事項

- SQL 文中に連続して指定した集合演算中に，ALL 指定とDISTINCT 指定の集合演算がある場合，ALL 指定の集合演算をDISTINCT 指定の集合演算として扱うアクセスパス情報が表示されることがあります。このとき，「6.1.4　ツリー表示に出力される情報」の「(4)　集合演算の指定」の表示では，連続して指定された集合演算を 1 つにまとめて出力します。また，対応する集合演算に関する情報も 1 つにまとめて出力します。
- 次に示す 2 つの条件を満たす場合，UNION DISTINCT 指定の集合演算を，UNION ALL 指定の集合演算として扱うアクセスパス情報が表示されることがあります。
  - SQL 文中に連続して指定した集合演算中に，UNION 指定，UNION ALL 指定，UNION DISTINCT 指定の集合演算がある
  - 集合演算の処理方式にハッシュ実行が適用される

  このとき，「6.1.4　ツリー表示に出力される情報」の「(4)　集合演算の指定」の表示では，連続して指定した集合演算を 1 つにまとめて出力します。また，対応する集合演算に関する情報も 1 つにまとめて出力します。

(4)　表関数導出表に関する情報

表関数導出表に関する情報には，実行されるシステム定義関数名が，次の形式で出力されます。

FUNCTION NAME：スキーマ名.システム定義関数名

システム定義関数名には，次のどちらかが出力されます。

- ADB_AUDITREAD

  ADB_AUDITREAD 関数が指定されていることを示しています。

- ADB_CSVREAD

ADB_CSVREAD 関数が指定されていることを示しています。

表関数導出表に関する情報の出力形式の例を次に示します。

出力形式の例

```
<<Detail   >>

QUERY : 1
     3 TABLE FUNCTION DERIVED TABLE(T5)
       FUNCTION NAME : MASTER.ADB_AUDITREAD
```

[説明]

ADB_AUDITREAD 関数が指定されていることを示しています。

(5) 副問合せに関する情報

副問合せの処理方式にハッシュ実行が適用される場合，副問合せに関する情報にハッシュ検索情報が出力されます。ハッシュ検索情報には，次の情報が出力されます。

- BUILD COLUMN

  外への参照列を含まない副問合せの処理方式にハッシュ実行が適用される場合は，副問合せの結果の列の情報が出力されます。

  外への参照列を含む副問合せの処理方式にハッシュ実行が適用される場合は，副問合せ中に指定された外への参照列と比較する列の情報が出力されます。

- PROBE COLUMN

  外への参照列を含まない副問合せの処理方式にハッシュ実行が適用される場合は，探索条件に指定された副問合せの結果と比較する列の情報が出力されます。

  外への参照列を含む副問合せの処理方式にハッシュ実行が適用される場合は，副問合せ中に指定された外への参照列の情報が出力されます。

出力例

```
<<Detail   >>

QUERY : 1
     3 SUBQUERY HASH
       BUILD COLUMN      : ADBUSER01.T2.C1(CREATE FILTER 1)    ...1
       PROBE COLUMN      : ADBUSER01.T1.C1(USE FILTER 1)       ...2
```

[説明]

上記は，外への参照列を含まない副問合せの処理方式にハッシュ実行が適用される場合の例です。

1. 副問合せの結果の列の列名（ADBUSER01.T2.C1）が出力されます。
2. 探索条件に指定された副問合せの結果と比較する列の列名（ADBUSER01.T1.C1）が出力されます。

ハッシュ実行の際にハッシュフィルタが適用される場合，下線部分にハッシュフィルタの情報が出力されます。

出力規則を次に示します。

- BUILD COLUMN またはPROBE COLUMN には，次のどれかの形式で列名が出力されます。このうち，相関名の出力が優先されます。列名を表示できない場合は，***が出力されます。
  - *表名.列名*
  - *問合せ名.列名*
  - *相関名.列名*
- ハッシュ実行の際にハッシュフィルタが適用される場合，BUILD COLUMN には，「(CREATE FILTER XXXXX)」が出力されます。「XXXXX」は，出力された列名の列値を基に作成したハッシュフィルタの番号です。

  PROBE COLUMN には，「(USE FILTER XXXXX)」が出力されます。「XXXXX」は，使用するハッシュフィルタの番号です。
- 内部導出表を作成する SQL 文で内部導出表を展開する場合，BUILD COLUMN とPROBE COLUMN に出力される表名や相関名などに，内部導出表の展開結果が出力されます。内部導出表および内部導出表の展開規則については，マニュアル『HADB　SQL リファレンス』の『内部導出表』を参照してください。

(6)　グループ化に関する情報

グループ化処理が複数回実行される場合に，グループ化に関する情報（グループ化集合情報）が次の形式で出力されます。

```
GROUPING SET :  {表名｜問合せ名｜相関名}.1番目のグループ化のグループ化列名1
                {表名｜問合せ名｜相関名}.1番目のグループ化のグループ化列名2
                              :
GROUPING SET :  {表名｜問合せ名｜相関名}.2番目のグループ化のグループ化列名1
                {表名｜問合せ名｜相関名}.2番目のグループ化のグループ化列名2
                              :
GROUPING SET :  {表名｜問合せ名｜相関名}.n番目のグループ化のグループ化列名1
                {表名｜問合せ名｜相関名}.n番目のグループ化のグループ化列名2
                              :
```

グループ化に関する情報の出力形式の例を次に示します。

出力形式の例

```
<<Detail   >>
QUERY : 1
     5 GLOBAL HASH GROUPING
       GROUPING SET : ADBUSER01.T1.C1
                      ADBUSER01.T1.C2
                      ADBUSER01.T1.C3
       GROUPING SET : ADBUSER01.T2.C1
                      ADBUSER01.T2.C2
       GROUPING SET : ADBUSER01.T3.C1
                      ADBUSER01.T3.C3
```

［説明］

グループ化処理ごとにグループ化列名が出力されます。

グループ化列名を出力できない場合は，***が出力されます。

出力規則を次に示します。

- 内部導出表を作成するSQL文で内部導出表を展開する場合，グループ化に関する情報に出力される表名や相関名などに，内部導出表の展開結果が出力されます。内部導出表および内部導出表の展開規則については，マニュアル『HADB　SQLリファレンス』の『内部導出表』を参照してください。

6.1.6　コスト情報表示に出力される情報

SQL文中で参照される実表のうち，コスト情報が取得されている実表に関する情報（表コスト情報）が出力されます。

出力例を次に示します。

出力例

```
<<COST Info>>

TABLE : ADBUSER01.T1
       COLLECT VERSION   : 05-03
       COLLECT TIME      : 2020-05-25 13:36:46
```

［説明］

- TABLE

  SQL文中で参照される実表のうち，コスト情報が取得されている実表の表名が出力されます。

- COLLECT VERSION

  TABLEに表示されている実表のコスト情報を取得したときのHADBサーバのバージョンが出力されます。

- COLLECT TIME

  TABLEに表示されている実表のコスト情報を取得したときの日時が出力されます。

メモ

- 表名の順に上記の情報が出力されます。
- SQL文中で同じ表が複数個所で参照される場合，重複排除して情報が出力されます。

6.1.7　特定情報表示（SQL 文の特定情報）に出力される情報

特定情報表示には，アクセスパスの統計情報を取得した SQL 文を特定するための情報（SQL 文の特定情報）が出力されます。ここに出力された情報を基に，アクセスパス情報の出力結果とアクセスパスの統計情報の出力結果を対応付けることができます。

アクセスパスの統計情報については，マニュアル『HADB　システム構築・運用ガイド』の『アクセスパスの統計情報の出力例と出力項目』を参照してください。

出力例を次に示します。

出力例

```
<<SQL Info >>

Version           : 05-03 May 25 2020 12:51:53
Transaction ID    : 1
Connection Number : 1
SQL Serial Number : 1
```

[説明]

- Version

  アクセスパスの統計情報を取得した SQL 文について，実行した HADB サーバのバージョンおよびバージョン付加情報が出力されます。

- Transaction ID

  アクセスパスの統計情報を取得した SQL 文のトランザクション ID が出力されます。

- Connection Number

  アクセスパスの統計情報を取得した SQL 文のコネクション通番が出力されます。

- SQL Serial Number

  アクセスパスの統計情報を取得した SQL 文の SQL 文通番が出力されます。

メモ

上記の情報は，アクセスパスの統計情報の前に出力される SQL 文の実行情報中の次の情報と対応しています。（　）内は，各情報のヘッダ名です。

- トランザクション ID（tran_id）
- コネクション通番（con_num）
- SQL 文通番（sql_serial_num）

SQL 文の実行情報については，マニュアル『HADB　システム構築・運用ガイド』の『SQL トレース情報に出力される情報』の『SQL 文の実行情報』を参照してください。

6.1.8　アクセスパスに表示される情報（アルファベット順）

アクセスパスに表示される情報を，アルファベット順に次の表に示します。

表 6-1　アクセスパスに表示される情報（アルファベット順）

先頭の文字	出力情報	説明	分類
B	BUILD COLUMN	表の結合方式にハッシュジョインが適用される場合，外表の結合列の情報が出力されます。また，ハッシュフィルタが適用される場合は，ハッシュフィルタの情報も出力されます。	ハッシュ検索情報（BUILD COLUMN，PROBE COLUMN）
		外への参照列を含まない副問合せの処理方式にハッシュ実行が適用される場合は，副問合せの結果の列の情報が出力されます。 外への参照列を含む副問合せの処理方式にハッシュ実行が適用される場合は，副問合せ中に指定された外への参照列と比較する列の情報が出力されます。 また，ハッシュフィルタが適用される場合は，ハッシュフィルタの情報も出力されます。	副問合せに関する情報
C	CHUNK	チャンクのスキップ条件が使用されることを示しています。	インデクスに関する情報
	CHUNK AND SEGMENT	チャンクのスキップ条件とセグメントのスキップ条件の両方が使用されることを示しています。	
	COLUMN	カラムストア表の特性を利用したグループ化の処理が実行されることを示しています。	グループ化の処理方式
	COLUMN STORE	表データの格納形式がカラムストア形式であることを示しています。	表データの格納形式
	CREATE FILTER	BUILD COLUMN に出力された列名の列値を基に作成したハッシュフィルタの番号が出力されます。	ハッシュ検索情報（BUILD COLUMN，PROBE COLUMN）
			副問合せに関する情報
	CREATE GLOBAL WORK TABLE（WORK TABLE *作業表番号*）	グローバル作業表が作成されることを示しています。	作業表の作成情報
	CREATE LOCAL WORK TABLE（WORK TABLE *作業表番号*）	ローカル作業表が作成されることを示しています。 作業表番号とは，作業表を一意に識別するために割り当てられた番号のことです。	
	CROSS JOIN	交差結合が指定されていることを示しています。	表の結合方式に関する情報
D	DELEGATION	副問合せ処理委譲指定が有効になることを示しています。	副問合せ処理委譲指定

先頭の文字	出力情報	説明	分類
	DELETE STATEMENT	DELETE 文が実行されることを示しています。	実行した SQL 文の種類
	DERIVED TABLE(*相関名，問合せ名，または表識別子*)	導出表，ビュー表，または問合せ名が指定されていることを示しています。	導出表の指定
	DISTINCT	重複排除処理が行われることを示しています。	重複排除の情報
	DIVIDE	ハッシュジョインが分割実行されることを示しています。	表の結合方式
F	FILTER	副問合せの処理方式のハッシュ実行の際に，ハッシュフィルタが適用されることを示しています。	副問合せの処理方式
		表の結合方式のハッシュジョインの際に，ハッシュフィルタが適用されることを示しています。	表の結合方式
	FULL OUTER JOIN(LEFT) FULL OUTER JOIN(RIGHT)	FULL OUTER JOIN による外結合が指定されていることを示しています。	表の結合方式に関する情報
	FUNCTION NAME	実行されるシステム定義関数の種類を示しています。	表関数導出表に関する情報
G	GLOBAL HASH GROUPING	グローバルハッシュグループ化が実行されることを示しています。また，ハッシュテーブル領域で処理できない行がある場合，作業表を作成することがあることも示しています。	グループ化の処理方式
	GLOBAL HASH UNIQUE	次のどれかの処理によって検索結果の重複を排除していることを示しています。 • 集合演算の処理方式のハッシュ実行 • SELECT DISTINCT の処理方式のハッシュ実行 • グループ化の処理方式のグローバルハッシュグループ化 また，ハッシュテーブル領域で処理できない行がある場合に，作業表が作成されていることも示しています。	重複排除の処理方式
	GROUPING	作業表を使用しないグループ化が実行されることを示しています。	グループ化の処理方式
	GROUPING SET	グループ化処理が複数回実行されることを示しています。	• グループ化の処理方式 • グループ化に関する情報
H	(HASH)	ハッシュジョインまたは副問合せの処理方式のハッシュ実行の際に，ハッシュテーブルとの突き合わせを行う条件に指定された列に対して，レンジインデクスが使用されることを示しています。	インデクスに関する情報

先頭の文字	出力情報	説明	分類
	HASH JOIN	表の結合処理で，ハッシュジョインが実行されることを示しています。 2 つの表の列を比較する=条件がある場合，ハッシュジョインが実行されることがあります。また，ハッシュテーブル領域で処理できない行がある場合，作業表が作成されることがあります。	• 表の結合方式（ツリー表示） • 表の結合方式（詳細表示）
	HAVING	HAVING 句が指定されていることを示しています。また，HAVING 句を指定していない場合でも，導出表の展開によって表示されることがあります。	HAVING 句の指定
I	INDEX	B-tree インデクスの特性を利用したグループ化の処理が実行されることを示しています。	グループ化の処理方式
	INDEX COLUMN	インデクス構成列に関する情報を示しています。 B-tree インデクスの場合は，インデクス構成列名，キー値の並び順，およびサーチ条件として評価する条件の種別が出力されます。 テキストインデクスまたはレンジインデクスの場合は，インデクス構成列名が出力されます。	インデクスに関する情報
	INDEX NAME	インデクス名です。	
	INDEX SCAN(*スキーマ名.表識別子*(*問合せ名または相関名*))	表の検索処理で，インデクススキャンが実行されることを示しています。問合せ名または相関名がある場合は，問合せ名または相関名が表示されます。	• 表の検索方式（ツリー表示） • 表の検索方式（詳細表示）
	INDEX TYPE	インデクス種別（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）を示しています。	インデクスに関する情報
	INNER JOIN	内結合が指定されていることを示しています。	表の結合方式に関する情報
	INSERT STATEMENT	INSERT 文が実行されることを示しています。	実行した SQL 文の種類
J	JOIN TYPE	結合種別を示しています。	表の結合方式に関する情報
K	KEY SCAN	表の検索処理で，キースキャンが実行されることを示しています。	• 表の検索方式（ツリー表示） • 表の検索方式（詳細表示）
L	LIMIT *オフセット行数*,*リミット行数*	LIMIT 句が指定されていることを示しています。オフセット行数の指定がない場合は，オフセット行数は表示されません。 オフセット行数とリミット行数の両方，またはどちらか一方に？パラメタが指定されている場合は，？PARAMETER と表示されます。	LIMIT 句の指定

先頭の文字	出力情報	説明	分類
	LEFT OUTER JOIN	LEFT OUTER JOIN による外結合が指定されていることを示しています。	表の結合方式に関する情報
	LOCAL HASH GROUPING	ローカルハッシュグループ化が実行されることを示しています。また，ハッシュグループ化領域で処理できない行がある場合，作業表を作成することがあることも示しています。	グループ化の処理方式
N	NESTED LOOP JOIN	表の結合処理で，ネストループジョインが実行されることを示しています。	• 表の結合方式（ツリー表示） • 表の結合方式（詳細表示）
O	ORDER	非順序実行方式が適用されないで，順序実行方式が適用されることを示しています。	順序実行方式
P	PARALLEL	検索系 SQL に対して，SQL パラレル実行機能が適用されることを示しています。	SQL パラレル実行機能の適用情報
	PARALLEL DISABLED	検索系 SQL に対して，SQL パラレル実行機能が適用されないことを示しています。	
	PROBE COLUMN	表の結合方式にハッシュジョインが適用される場合，内表の結合列の情報が出力されます。また，ハッシュフィルタが適用される場合は，ハッシュフィルタの情報も出力されます。	ハッシュ検索情報（BUILD COLUMN，PROBE COLUMN）
		外への参照列を含まない副問合せの処理方式にハッシュ実行が適用される場合は，探索条件に指定された副問合せの結果と比較する列の情報が出力されます。 外への参照列を含む副問合せの処理方式にハッシュ実行が適用される場合は，副問合せ中に指定された外への参照列の情報が出力されます。 また，ハッシュフィルタが適用される場合は，ハッシュフィルタの情報も出力されます。	副問合せに関する情報
	PURGE CHUNK STATEMENT	PURGE CHUNK 文が実行されることを示しています。	実行した SQL 文の種類
Q	QUERY	副問合せおよび導出表以外の問合せが指定されていることを示しています。	問合せの種類
	QUERY SCAN(QUERY *問合せツリー番号*)	問合せ検索が実行されることを示しています。	問合せ検索の実行
R	RECURSIVE	再帰的問合せが実行されることを示しています。	集合演算の指定
	RIGHT OUTER JOIN	RIGHT OUTER JOIN による外結合が指定されていることを示しています。	表の結合方式に関する情報
S	SEGMENT	セグメントのスキップ条件を使用します。	インデクスに関する情報

先頭の文字	出力情報	説明	分類
	SELECT STATEMENT	SELECT 文が実行されることを示しています。	実行した SQL 文の種類
	SET OPERATION	集合演算が指定されていることを示しています。	集合演算の指定
	SET OPERATION TYPE	実行される集合演算の種類を示しています。	集合演算に関する情報
	SKIP COND	レンジインデクス条件の種別です。	インデクスに関する情報
	SORT GROUPING	ソートグループ化が実行されることを示しています。	グループ化の処理方式
	SORTING {BYTE | ISO}	ORDER BY 句によるソート処理が実行されることを示しています。ただし，ORDER BY 句を指定した場合でも表示されないときがあります。	ソート処理の実行
	SPECIFIC	集合演算方式指定が有効になることを示しています。	集合演算方式指定
		副問合せ処理方式指定が有効になることを示しています。	副問合せ処理方式指定
		GROUP BY 句に指定したグループ化方式指定が有効になることを示しています。	グループ化の処理方式
		SELECT 重複排除方式指定が有効になることを示しています。	SELECT 重複排除方式指定
		インデクス指定が有効になることを示しています。	インデクス指定
		結合方式指定が有効になることを示しています。	結合方式指定
	SPECIFIC DISABLED	インデクス指定が無効になったことを示しています。	インデクス指定
		結合方式指定が無効になったことを示しています。	結合方式指定
	SUBQUERY	ネストループ実行，ハッシュ実行以外の副問合せが適用されることを示しています。	副問合せの処理方式
	SUBQUERY HASH	副問合せの処理方式で，ハッシュ実行が適用されることを示しています。	
	SUBQUERY LOOP	副問合せの処理方式で，ネストループ実行が適用されることを示しています。	
T	TABLE FUNCTION DERIVED TABLE(相関名)	表関数導出表が指定されていることを示しています。	表関数導出表の指定
	TABLE SCAN	表の検索処理で，テーブルスキャンが実行されることを示しています。	• 表の検索方式（ツリー表示） • 表の検索方式（詳細表示）

先頭の文字	出力情報	説明	分類
	TABLE VALUE CONSTRUCTOR SCAN	表値構成子を検索することを示しています。	表値構成子の検索情報
U	UNIQUE	このユニークインデクスが，一意性制約に違反していない状態であることを示しています。	インデクスに関する情報
	UNIQUE INVALID	このユニークインデクスが，一意性制約に違反している状態であることを示しています。	
	UNNEST	集まり導出表が指定されていることを示しています。	集まり導出表の指定
	UNNEST DERIVED TABLE SCAN(*相関名*)	集まり導出表が指定されていることを示しています。また，指定されている集まり導出表の相関名が表示されます。	集まり導出表に関する情報
	UPDATE STATEMENT	UPDATE 文が実行されることを示しています。	実行した SQL 文の種類
	USE FILTER	使用するハッシュフィルタの番号が出力されます。	ハッシュ検索情報（BUILD COLUMN，PROBE COLUMN）
			副問合せに関する情報
	USING CACHE	副問合せの結果を格納するためのキャッシュが使用されることを示しています。	副問合せのキャッシュ使用情報
	USING COST	表またはインデクスに関するコスト情報が収集されていることを示しています。	コスト情報の収集
W	WINDOW	ウィンドウ関数が指定されていることを示しています。	ウィンドウ関数の指定
	WORK TABLE SCAN(WORK TABLE *作業表番号*)	作業表の検索が実行されることを示しています。作業表番号とは，作業表を一意に識別するために割り当てられた番号のことです。	作業表の検索実行

7 APの作成

この章では，JDBC ドライバを使用した AP の作成方法について説明します。なお，JDBC ドライバの環境設定方法については，「3.　JDBC ドライバの環境設定」を参照してください。

7.1 HADB が提供している JDBC ドライバ

ここでは，HADB が提供している JDBC ドライバの JDBC 規格への準拠範囲，および JAR ファイルのパッケージ名称とディレクトリ構成について説明します。

7.1.1 JDBC 規格への準拠範囲

HADB は Type4 の JDBC ドライバを実装しています。HADB の JDBC ドライバの JDBC 規格への準拠範囲を次の表に示します。

表 7-1 HADB の JDBC ドライバの JDBC 規格への準拠範囲

項番	JDBC の規格	機能	準拠範囲
1	The JDBC™ API Version 1.20（JDBC 1.2 API）	Driver インタフェース	○
2		Connection インタフェース	○
3		Statement インタフェース	○
4		PreparedStatement インタフェース	○
5		CallableStatement インタフェース	×
6		ResultSet インタフェース	○
7		DatabaseMetaData インタフェース	○
8		ResultSetMetaData インタフェース	○
9		Blob インタフェース	×
10		Array インタフェース	○
11		SQLException インタフェース	○
12		SQLWarning インタフェース	○
13	JDBC™ 2.1 API Version 1.1（JDBC 2.1 コア API）	結果セットの拡張	△※1
14		バッチ更新	○
15		Java オブジェクトの持続化	×
16		JDBC SQL データ型追加	×
17		データ型カスタムマッピング	×
18	JDBC 2.0 Standard Extension API Version 1.0（JDBC 2.0 オプションパッケージ API）	JNDI	○
19		接続プール	○
20		分散トランザクション（JTA 対応）	×
21		Rowsets	×

項番	JDBC の規格	機能	準拠範囲
22	JDBC™ 3.0 Specification（JDBC 3.0 API）	セーブポイント	×
23		接続プール機能強化	×
24		パラメタメタデータ	○
25		自動生成キー	×
26		結果セットの複数同時オープン	×
27		CallableStatement でのパラメタ名の使用	×
28		ホールダブルカーソル	○
29		BOOLEAN 型	×
30		Blob クラス内でのデータ操作	×
31		参照型	×
32		DATALINK/URL 型	×
33		JCA 関連アーキテクチャ	×
34		データベースメタデータの追加 API	○
35	JDBC™ 4.0 Specification（JDBC 4.0 API）	java.sql.Driver 自動ローディング	○
36		ROWID データ型	×
37		各国文字データ型	×
38		BLOB/CLOB 機能拡張	×
39		XML サポート	×
40		ラッパーパターン	○
41		SQL 例外拡張	○
42		接続管理	△※2
43		スカラ関数追加	△※3
44		データベースメタデータの追加 API	○
45	JDBC™ 4.1 Specification（JDBC 4.1 API）	try-with-resources 文	○
46		getObject メソッドの変換先 Java 型指定	△※4
47		親ロガーの取得	×
48		スキーマ指定	×
49		物理接続の中止およびタイムアウト	×
50		依存オブジェクトクローズ時のStatement オブジェクトクローズ	○
51		データベースメタデータの追加 API	○

項番	JDBC の規格	機能	準拠範囲
52	JDBC™ 4.2 Specification（JDBC 4.2 API）	REF CURSOR	×
53		SQLType インタフェース	×
54		JDBCType 列挙（Enum）	×
55		大きい更新カウント	○
56		データベースメタデータの追加 API	○

（凡例）

○：サポート対象です。

△：一部の機能をサポート対象にしています。

×：サポート対象外です。

注※1

スクロール機能だけをサポートしています。

注※2

Connection#isValid()，およびStatement#isPoolable()だけをサポートしています。

注※3

CURRENT_DATE，CURRENT_TIME，CURRENT_TIMESTAMP，およびEXTRACT だけをサポートしています。

注※4

ResultSet インタフェースのgetObject メソッドです。また，一部の Java 型への変換だけをサポートしています。詳細については，「8.6.45　getObject(int columnIndex,Class<T> type)」を参照してください。

以降，Type4 JDBC ドライバを「JDBC ドライバ」と表記します。

7.1.2　JAR ファイルのパッケージ名称とディレクトリ構成

JAR ファイルのパッケージ名称とディレクトリ構成を次に示します。

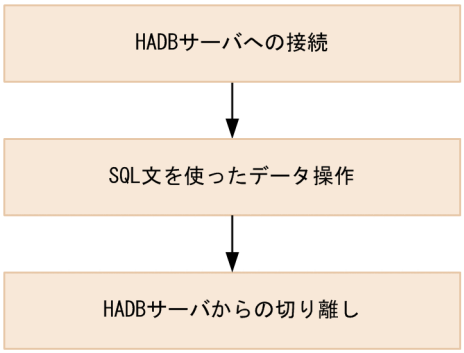
- パッケージ名称：com.hitachi.hadb.jdbc
- ディレクトリ構成：com/hitachi/hadb/jdbc/

> **メモ**
>
> HADB，hadb は，Hitachi Advanced Data Binder の略称です。

7.2　AP の処理の基本的な流れ

データベースを操作する際の AP の処理の基本的な流れを次の図に示します。

図 7-1　データベースを操作する際の AP の処理の基本的な流れ



［説明］

1. DriverManager クラスまたはDataSource クラスのgetConnection メソッドを使用して，HADB サーバに接続します。HADB サーバへの接続方法の詳細については，「7.3　HADB サーバへの接続方法」を参照してください。
2. データを操作する SQL 文を実行します。SELECT 文を実行する場合は，「7.4　データを検索する場合（SELECT 文を実行する場合）」を参照してください。

   INSERT 文，UPDATE 文，またはDELETE 文を実行する場合は，「7.5　データを追加，更新，または削除する場合（INSERT 文，UPDATE 文，または DELETE 文を実行する場合）」を参照してください。
3. Connection オブジェクトのclose メソッドを使用して，HADB サーバから AP を切り離します。

JDBC の API については，「8.　JDBC 1.2 API」以降で説明しています。

■マルチスレッドの AP を作成する際の注意事項

次に示す条件をすべて満たす場合，シリアライズによる待ち状態が発生することがあります。

- 同じConnection オブジェクトから生成された複数のStatement オブジェクトを，それぞれ異なるスレッドで使用するようにした場合
- それらのStatement オブジェクトを使用して SQL 文を同時に実行した場合

この待ち状態は，SQL 文の実行前に発生するため，SQL 文の実行処理のタイマ監視時間には含まれません。そのため，プロパティのadb_clt_rpc_sql_wait_time，またはsetQueryTimeout メソッドで指定した SQL 文の実行処理のタイマ監視時間を過ぎても，タイムアウトエラーが発生しないことがあります。

7.3 HADB サーバへの接続方法

HADB のデータベースにアクセスするには，HADB サーバに接続する必要があります。HADB サーバに接続するには，次に示す 2 つの方法があります。

- DriverManager クラスのgetConnection メソッドを使用する方法
- DataSource クラスのgetConnection メソッドを使用する方法

なお，getConnection メソッドを実行するにはCONNECT 権限が必要です。

各方法の詳細を以降で説明します。

7.3.1 DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法

DriverManager クラスのgetConnection メソッドを実行して HADB サーバに接続します。getConnection メソッドを実行する前に Java 仮想マシンにDriver クラスが自動で登録されます（なお，「(1) Java 仮想マシンへの Driver クラスの登録方法」を参照して登録作業をしても問題ありません)。そのあとに，getConnection メソッドを実行して HADB サーバに接続します。

各手順の詳細を以降で説明します。

(1) Java 仮想マシンへの Driver クラスの登録方法

Java 仮想マシンにDriver クラスを登録します。Java 仮想マシンにDriver クラスを登録する際に必要なドライバ名称は，「*パッケージ名称*.*クラス名称*」です。JDBC ドライバのパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：HADBDriver

> **メモ**
>
> HADB，hadb は，Hitachi Advanced Data Binder の略称です。

Driver クラスを登録するには次に示す 3 つの方法があります。

■登録方法 1（Class クラスのforName メソッドを使用して登録する方法）

アプリケーション内で，次のようにClass クラスのforName メソッドを実行します。

```
Class.forName("com.hitachi.hadb.jdbc.HADBDriver");
```

■登録方法 2（システムプロパティを使用して登録する方法）

Java 仮想マシンのシステムプロパティ（jdbc.drivers）に次の値を設定します。

```
System.setProperty(”jdbc.drivers”, ”com.hitachi.hadb.jdbc.HADBDriver”);
```

■登録方法３（Java 仮想マシンの動作設定ファイルを使用して登録する方法）

この方法は Java Applet の場合に限ります。

[JAVA_HOME]¥.hotjava¥properties ファイルに，次の内容を記述します（*[JAVA_HOME]*は，Java 実行環境によって異なります）。複数の JDBC ドライバを登録する場合には，コロン（:）で区切って記述してください。

```
jdbc.drivers=”com.hitachi.hadb.jdbc.HADBDriver”
```

(2) getConnection メソッドによる HADB サーバへの接続

DriverManager クラスのgetConnection メソッドを実行して HADB サーバに接続します。正常に HADB サーバに接続した場合，JDBC ドライバはこれらのメソッドを実行した結果として，Connection クラスのインスタンスの参照を返却します。HADB サーバへの接続に失敗した場合（次に示す場合）は，SQLException が投入されます。

- 必要な接続情報が引数に指定されていない場合
- 接続情報の指定内容が不正な場合
- 接続に失敗した場合（接続先の HADB サーバが開始していないときなど）

getConnection メソッドには次に示す３つの形式があり，引数（url，user，password，およびinfo）には HADB サーバへの接続情報を指定します。

- public static Connection getConnection(String url)
- public static Connection getConnection(String url, String user, String password)
- public static Connection getConnection(String url, Properties info)

各引数の指定内容を以降で説明します。

(a) 引数 url の指定内容（接続用の URL の指定）

引数url には接続用の URL を指定します。URL の指定形式を次に示します。

```
jdbc:hadb[://[host][:port]/[?property=value[&property=value]...]]
```

メモ

HADB，hadb は，Hitachi Advanced Data Binder の略称です。

URL の指定例

- 例 1：プロパティを指定しない場合

```
jdbc:hadb://localhost:23650/
```

- 例 2：プロパティを 1 つ指定する場合

```
jdbc:hadb://localhost:23650/?adb_clt_ap_name=AP001
```

- 例 3：複数のプロパティを指定する場合

```
jdbc:hadb://localhost:23650/?methodtrace=ON&tracenum=600
&sqlwarningkeep=FALSE&user=ADBUSER01&password=password01&adb_clt_ap_name=AP001
```

URL の指定規則

- URL 内の各項目および項目間にはスペースを入れないでください。
- 各項目名称の大文字と小文字は区別されます。
- ［ ］で囲まれている項目は省略できます。
- 最初のプロパティ（property）を指定する前に?を指定し，各プロパティの指定は&で区切ります。
- 同じプロパティを複数指定した場合，最初に指定された指定値が有効になります。
- プロパティの指定値として，&は使用できません。パスワードに'&'を含む場合は，URL でpassword プロパティを指定できないため，ほかの接続方法を利用してください。HADB のパスワードについては，マニュアル『HADB　システム構築・運用ガイド』の『パスワードの指定規則』を参照してください。
- URL 中に指定したプロパティに不正な値を指定しても，ユーザプロパティの同名のプロパティで正しい値を指定している場合は，SQLException は投入されません。

URL の各項目の説明

●jdbc:hadb

プロトコル名称，およびサブプロトコル名称です。必ず指定してください。大文字，小文字の指定は区別されます。

●host

接続先の HADB サーバのホスト名を指定します。ここで指定したホスト名は，HADB クライアントと HADB サーバ間の通信で使用します。

なお，HADB サーバのホスト名はほかの方法でも指定できます。ほかの指定方法および指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

ホスト名がどこにも指定されていない状態でgetConnection メソッドを実行すると，SQLException が投入されます。

コールドスタンバイ構成の場合は，HADB サーバと HADB クライアント間の通信で使用するエイリアス IP アドレスを指定してください。

[マルチノード機能]

- HA モニタありのマルチノード構成の場合

HADB サーバと HADB クライアント間の通信で使用するエイリアス IP アドレスを指定してください。

- HA モニタなしのマルチノード構成の場合

  HADB サーバと HADB クライアント間の通信で使用するプライマリノードのホスト名を指定してください。

●port

HADB クライアントと HADB サーバ間の通信で使用する，HADB サーバのポート番号を指定します。

なお，HADB サーバのポート番号はほかの方法でも指定できます。ほかの指定方法および指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

ポート番号がどこにも指定されていない状態でgetConnection メソッドを実行すると，SQLException が投入されます。

●property=value

各プロパティ（property）に値（value）を指定します。

引数url に指定できるプロパティを次の表に示します。

表 7-2　引数 url に指定できるプロパティ

項番	プロパティ名	説明
1	user	HADB サーバに接続する認可識別子を指定します。 認可識別子の名称規則については，マニュアル『HADB　SQL リファレンス』の『名前の指定』を参照してください。 なお，HADB サーバに接続する認可識別子はほかの方法でも指定できます。指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。 認可識別子がどこにも指定されていない状態でgetConnection メソッドを実行すると，SQLException が投入されます。
2	password	HADB サーバに接続する認可識別子のパスワードを指定します。
3	encodelang	String クラスで HADB サーバとのデータ受け渡しをする場合に，文字コード変換処理で使用する変換文字セットを指定します。指定できる変換文字セットは，『JavaTM Platform, Standard Edition JDK ドキュメント』の『国際化サポート』で示される『サポートされているエンコーディング』の一覧から選択してください。 この指定を省略した場合，「表 7-15　HADB サーバの文字コードに対応する文字セット名称」の組み合わせに従った変換文字セットで文字コードが変換されます。ただし，次に示す値については，Java 仮想マシンのデフォルトの変換文字セットで変換されます。 • AP 識別子（ユーザプロパティのadb_clt_ap_name などで設定）の指定値 • 認可識別子，またはパスワード（getConnection メソッドなどで指定） このプロパティは，「表 7-15　HADB サーバの文字コードに対応する文字セット名称」の組み合わせの変換文字セット以外の文字セットで文字コード変換したい場合だけ指定してください。「表 7-15　HADB サーバの文字コードに対応する文字セット名称」の組み合わせの変換文字セットで文字コード変換する場合は，このプロパティを指定する必要はありません。

項番	プロパティ名	説明
4	methodtrace	JDBC インタフェースメソッドトレースを取得するかどうかを指定します。 ON：取得します。 OFF：取得しません。 JDBC インタフェースメソッドトレースについては，「7.7.1　JDBC インタフェースメソッドトレース」を参照してください。 上記以外の値を指定した場合は，SQLException が投入されます。 この指定を省略した場合，OFF が仮定されます。 なお，setLogWriter メソッドで有効なログライターを設定していない場合，ON を指定しても JDBC インタフェースメソッドトレースは取得されません。
5	tracenum	JDBC インタフェースメソッドトレースのエントリ数を，10～1,000 の範囲で指定します。省略値は500 になります。 このプロパティの指定値は，次に示すすべての条件を満たしている場合に有効になります。 • setLogWriter メソッドで有効なログライターを設定している • methodtrace にON を指定している プロパティの指定が有効な状態のときに 10～1,000 以外の値を指定すると，SQLException が投入されます。
6	sqlwarningkeep	HADB サーバから返される警告情報を保持するかどうかを指定します。 TRUE：警告情報を保持します。 FALSE：警告情報を保持しません。 この指定を省略した場合，TRUE が仮定されます。上記以外の値を指定した場合，SQLException が投入されます。 Connection オブジェクトの警告情報保持レベルの詳細については，「8.10.1　SQLWarning オブジェクトの生成」を参照してください。
7	adb_clt_rpc_con_wait_time	HADB サーバへの接続処理の完了待ち時間を指定します。 このプロパティの機能は，クライアント定義のadb_clt_rpc_con_wait_time オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_con_wait_time オペランドを参照してください。
8	adb_clt_rpc_sql_wait_time	次に示す待ち時間を指定します。 • HADB クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間 • 同一コネクションで複数のSELECT 文を同時実行した際に，処理リアルスレッド数不足が発生したときの処理リアルスレッドの確保処理の待ち時間 このプロパティの機能は，クライアント定義のadb_clt_rpc_sql_wait_time オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_sql_wait_time オペランドを参照してください。
9	adb_clt_ap_name	HADB サーバに接続する AP の識別情報（AP 識別子）を指定します。 AP 識別子は Java 仮想マシンのデフォルトの変換文字セットで変換されるため，変換文字セットに依存しない半角英数字だけで構成される名称を指定することを推奨します。 なお，AP 識別子はほかの方法でも指定できます。指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

項番	プロパティ名	説明
		AP 識別子をどこにも指定しないで HADB サーバに接続した場合，AP 識別子には"********"が設定されます。 このプロパティの機能は，クライアント定義のadb_clt_ap_name オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_clt_ap_name オペランドを参照してください。
10	adb_clt_group_name	AP が属するクライアントグループ名を指定します。 このプロパティの機能は，クライアント定義のadb_clt_group_name オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_group_name オペランドを参照してください。
11	adb_clt_fetch_size	1 回のFETCH 処理で，HADB サーバから HADB クライアントに送信する検索結果の行数を指定します。 このプロパティの機能は，クライアント定義のadb_clt_fetch_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_fetch_size オペランドを参照してください。
12	adb_dbbuff_wrktbl_clt_blk_num	ローカル作業表用バッファのページ数を指定します。 このプロパティの機能は，クライアント定義の adb_dbbuff_wrktbl_clt_blk_num オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_dbbuff_wrktbl_clt_blk_num オペランドを参照してください。
13	adb_sql_exe_max_rthd_num	SQL 実行時に使用する処理リアルスレッド数の最大値を指定します。 このプロパティの機能は，クライアント定義のadb_sql_exe_max_rthd_num オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_sql_exe_max_rthd_num オペランドを参照してください。
14	adb_sql_exe_hashgrp_area_size	ハッシュグループ化領域の大きさをキロバイト単位で指定します。 このプロパティの機能は，クライアント定義の adb_sql_exe_hashgrp_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashgrp_area_size オペランドを参照してください。
15	adb_sql_exe_hashtbl_area_size	ハッシュテーブル領域サイズをメガバイト単位で指定します。 このプロパティの機能は，クライアント定義の adb_sql_exe_hashtbl_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashtbl_area_size オペランドを参照してください。
16	adb_sql_exe_hashflt_area_size	ハッシュフィルタ領域サイズをメガバイト単位で指定します。 このプロパティの機能は，クライアント定義の adb_sql_exe_hashflt_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashflt_area_size オペランドを参照してください。
17	adb_clt_sql_parallel_exec	SQL パラレル実行機能を適用して検索系 SQL を実行するかどうかを指定します。 このプロパティの機能は，クライアント定義のadb_clt_sql_parallel_exec オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_parallel_exec オペランドを参照してください。

項番	プロパティ名	説明	
18	adb_sql_prep_delrsvd_use_srvdef	サーバ定義のadb_sql_prep_delrsvd_words オペランドの指定に従って予約語を削除するかどうかを指定します。 このプロパティの機能は，クライアント定義のadb_sql_prep_delrsvd_use_srvdef オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_sql_prep_delrsvd_use_srvdef オペランドを参照してください。	
19	adb_clt_trn_iso_lv	トランザクション隔離性水準を指定します。 このプロパティの機能は，クライアント定義のadb_clt_trn_iso_lv オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_trn_iso_lv オペランドを参照してください。	
20	adb_clt_trn_access_mode	トランザクションアクセスモードを指定します。 このプロパティの機能は，クライアント定義のadb_clt_trn_access_mode オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_trn_access_mode オペランドを参照してください。	
21	adb_clt_sql_text_out	HADB クライアントが発行した SQL 文を，クライアントメッセージログファイルおよびサーバメッセージログファイルに出力するかどうかを指定します。 このプロパティの機能は，クライアント定義のadb_clt_sql_text_out オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_text_out オペランドを参照してください。	
22	adb_clt_sql_order_mode	ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を指定します。 このプロパティの機能は，クライアント定義のadb_clt_sql_order_mode オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_order_mode オペランドを参照してください。	
23	adb_sql_prep_dec_div_rs_prior	SQL 文中に指定した除算（四則演算）の結果のデータ型がDECIMAL 型の場合，除算結果の位取りの最小値を指定します。 このプロパティの機能は，クライアント定義のadb_sql_prep_dec_div_rs_prior オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_sql_prep_dec_div_rs_prior オペランドを参照してください。	
24	adb_clt_passwd_pubkey_path	ユーザ認証方式に PAM 認証を使用する場合，パスワードの暗号化に使用する公開鍵ファイルの絶対パスを指定します。 このプロパティの機能は，クライアント定義のadb_clt_passwd_pubkey_path オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_passwd_pubkey_path オペランドを参照してください。	
25	adb_jdbc_exc_trc_out_path	Exception トレースログを出力するディレクトリを，絶対パスで指定します。	これらのプロパティの機能詳細および指定できる値については，「7.7.2　Exception トレースログ」の「(1)　取得するメソッド，および取得するための設定」の「(b)　Exception トレースログを取得するための設定（プロパティの設定）」を参照してください。
26	adb_jdbc_info_max	1 ファイルへ出力する情報数の上限を指定します。	
27	adb_jdbc_cache_info_max	メモリ内で蓄積する情報数の上限を指定します。	

項番	プロパティ名	説明	
28	adb_jdbc_trc_out_lv	トレース取得レベルを指定します。	

注

各プロパティは，ほかの方法でも指定できます。ほかの指定方法，および指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

メモ

HADB 03-00 で，接続用の URL で指定する各プロパティ名を次のように変更しました。変更前のプロパティ名も使用できますが，HADB 03-00 以降にバージョンアップした場合は，プロパティ名を変更することを推奨します。

項番	変更前のプロパティ名（HADB 03-00 より前のプロパティ名）	変更後のプロパティ名（HADB 03-00 以降のプロパティ名）
1	apname	adb_clt_ap_name
2	extrcoutpath	adb_jdbc_exc_trc_out_path
3	extrcinfomax	adb_jdbc_info_max
4	extrccacheinfomax	adb_jdbc_cache_info_max
5	extrcoutlv	adb_jdbc_trc_out_lv

(b)　引数 user の指定内容（認可識別子の指定）

HADB サーバに接続する認可識別子を指定します。

認可識別子の名称規則については，マニュアル『HADB　SQL リファレンス』の『名前の指定』を参照してください。

なお，HADB サーバに接続する認可識別子はほかの方法でも指定できます。指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

認可識別子がどこにも指定されていない状態でgetConnection メソッドを実行すると，SQLException が投入されます。

null を指定した場合，認可識別子の指定を省略したと見なされます。

また，長さ 0 の文字列を指定した場合は，SQLException が投入されます。

(c)　引数 password の指定内容（パスワードの指定）

HADB サーバに接続する認可識別子のパスワードを指定します。

null を指定した場合，または長さ 0 の文字列を指定した場合，パスワードの指定を省略したと見なされます。

(d)　引数 info の指定内容（ユーザプロパティの指定）

引数info に指定できる情報（ユーザプロパティに指定できる情報）を次の表に示します。

表 7-3　引数 info に指定できる情報（ユーザプロパティに指定できる情報）

項番	プロパティ名	説明
1	user	HADB サーバに接続する認可識別子を指定します。 認可識別子の名称規則については，マニュアル『HADB　SQL リファレンス』の『名前の指定』を参照してください。 • HADB サーバに接続する認可識別子はほかの方法でも指定できます。指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。 • 認可識別子がどこにも指定されていない状態でgetConnection メソッドを実行すると，SQLException が投入されます。 • null を指定した場合，認可識別子の指定がないと見なされます。 • 長さ 0 の文字列を指定した場合，SQLException が投入されます。
2	password	HADB サーバに接続する認可識別子のパスワードを指定します。 null を指定した場合，または長さ 0 の文字列を指定した場合，パスワードの指定がないと見なされます。
3	encodelang	String クラスで HADB サーバとのデータ受け渡しをする場合に，文字コード変換処理で使用する変換文字セットを指定します。このプロパティの機能詳細および指定できる値については，「表 7-2　引数 url に指定できるプロパティ」を参照してください。 Java 仮想マシンがサポートしていない変換文字セット名称を指定した場合は，HADB サーバとの接続時にSQLException を投入します。 この指定を省略した場合，接続用の URL のencodelang の指定が適用されます。
4	methodtrace	JDBC インタフェースメソッドトレースを取得するかどうかを指定します。このプロパティの機能詳細および指定できる値については，「表 7-2　引数 url に指定できるプロパティ」を参照してください。 この指定を省略した場合，接続用の URL のmethodtrace の指定が適用されます。
5	tracenum	JDBC インタフェースメソッドトレースのエントリ数を指定します。このプロパティの機能詳細および指定できる値については，「表 7-2　引数 url に指定できるプロパティ」を参照してください。 この指定を省略した場合，接続用の URL のtracenum の指定が適用されます。
6	sqlwarningkeep	HADB サーバから返される警告情報を保持するかどうかを指定します。このプロパティの機能詳細および指定できる値については，「表 7-2　引数 url に指定できるプロパティ」を参照してください。 この指定を省略した場合，接続用の URL のsqlwarningkeep の指定が適用されます。
7	adb_clt_rpc_srv_host	接続先の HADB サーバのホスト名を指定します。

項番	プロパティ名	説明
		このプロパティの機能は，クライアント定義のadb_clt_rpc_srv_host オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_srv_host オペランドを参照してください。
8	adb_clt_rpc_srv_port	HADB クライアントと HADB サーバ間の通信で使用する，HADB サーバのポート番号を指定します。 このプロパティの機能は，クライアント定義のadb_clt_rpc_srv_port オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_srv_port オペランドを参照してください。
9	adb_clt_rpc_con_wait_time	HADB サーバへの接続処理の完了待ち時間を指定します。 このプロパティの機能は，クライアント定義のadb_clt_rpc_con_wait_time オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_con_wait_time オペランドを参照してください。
10	adb_clt_rpc_sql_wait_time	次に示す待ち時間を指定します。 • HADB クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間 • 同一コネクションで複数のSELECT 文を同時実行した際に，処理リアルスレッド数不足が発生したときの処理リアルスレッドの確保処理の待ち時間 このプロパティの機能は，クライアント定義のadb_clt_rpc_sql_wait_time オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_rpc_sql_wait_time オペランドを参照してください。
11	adb_clt_ap_name	HADB サーバに接続する AP の識別情報（AP 識別子）を指定します。 AP 識別子は Java 仮想マシンのデフォルトの変換文字セットで変換されるため，変換文字セットに依存しない半角英数字だけで構成される名称を指定することを推奨します。 なお，AP 識別子はほかの方法でも指定できます。指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。 AP 識別子をどこにも指定しないで HADB サーバに接続した場合，AP 識別子には"********"が設定されます。 このプロパティの機能は，クライアント定義のadb_clt_ap_name オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_ap_name オペランドを参照してください。
12	adb_clt_group_name	AP が属するクライアントグループ名を指定します。 このプロパティの機能は，クライアント定義のadb_clt_group_name オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_group_name オペランドを参照してください。
13	adb_clt_fetch_size	1 回のFETCH 処理で，HADB サーバから HADB クライアントに送信する検索結果の行数を指定します。 このプロパティの機能は，クライアント定義のadb_clt_fetch_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_fetch_size オペランドを参照してください。
14	adb_dbbuff_wrktbl_clt_blk_num	ローカル作業表用バッファのページ数を指定します。 このプロパティの機能は，クライアント定義の adb_dbbuff_wrktbl_clt_blk_num オペランドと同じです。機能詳細および指

項番	プロパティ名	説明
		定できる値については，クライアント定義の adb_dbbuff_wrktbl_clt_blk_num オペランドを参照してください。
15	adb_sql_exe_max_rthd_num	SQL 実行時に使用する処理リアルスレッド数の最大値を指定します。 このプロパティの機能は，クライアント定義のadb_sql_exe_max_rthd_num オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_sql_exe_max_rthd_num オペランドを参照してください。
16	adb_sql_exe_hashgrp_area_size	ハッシュグループ化領域の大きさをキロバイト単位で指定します。 このプロパティの機能は，クライアント定義の adb_sql_exe_hashgrp_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashgrp_area_size オペランドを参照してください。
17	adb_sql_exe_hashtbl_area_size	ハッシュテーブル領域サイズをメガバイト単位で指定します。 このプロパティの機能は，クライアント定義の adb_sql_exe_hashtbl_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashtbl_area_size オペランドを参照してください。
18	adb_sql_exe_hashflt_area_size	ハッシュフィルタ領域サイズをメガバイト単位で指定します。 このプロパティの機能は，クライアント定義の adb_sql_exe_hashflt_area_size オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_exe_hashflt_area_size オペランドを参照してください。
19	adb_clt_sql_parallel_exec	SQL パラレル実行機能を適用して検索系 SQL を実行するかどうかを指定します。 このプロパティの機能は，クライアント定義のadb_clt_sql_parallel_exec オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_parallel_exec オペランドを参照してください。
20	adb_sql_prep_delrsvd_use_srvdef	サーバ定義のadb_sql_prep_delrsvd_words オペランドの指定に従って予約語を削除するかどうかを指定します。 このプロパティの機能は，クライアント定義の adb_sql_prep_delrsvd_use_srvdef オペランドと同じです。機能詳細および指定できる値については，クライアント定義の adb_sql_prep_delrsvd_use_srvdef オペランドを参照してください。
21	adb_clt_trn_iso_lv	トランザクション隔離性水準を指定します。 このプロパティの機能は，クライアント定義のadb_clt_trn_iso_lv オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_trn_iso_lv オペランドを参照してください。
22	adb_clt_trn_access_mode	トランザクションアクセスモードを指定します。 このプロパティの機能は，クライアント定義のadb_clt_trn_access_mode オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_trn_access_mode オペランドを参照してください。
23	adb_clt_sql_text_out	HADB クライアントが発行した SQL 文を，クライアントメッセージログファイルおよびサーバメッセージログファイルに出力するかどうかを指定します。

項番	プロパティ名	説明	
		このプロパティの機能は，クライアント定義のadb_clt_sql_text_out オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_text_out オペランドを参照してください。	
24	adb_clt_sql_order_mode	ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を指定します。 このプロパティの機能は，クライアント定義のadb_clt_sql_order_mode オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_sql_order_mode オペランドを参照してください。	
25	adb_sql_prep_dec_div_rs_prior	SQL 文中に指定した除算（四則演算）の結果のデータ型がDECIMAL 型の場合，除算結果の位取りの最小値を指定します。 このプロパティの機能は，クライアント定義のadb_sql_prep_dec_div_rs_prior オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_sql_prep_dec_div_rs_prior オペランドを参照してください。	
26	adb_clt_passwd_pubkey_path	ユーザ認証方式に PAM 認証を使用する場合，パスワードの暗号化に使用する公開鍵ファイルの絶対パスを指定します。 このプロパティの機能は，クライアント定義のadb_clt_passwd_pubkey_path オペランドと同じです。機能詳細および指定できる値については，クライアント定義のadb_clt_passwd_pubkey_path オペランドを参照してください。	
27	adb_jdbc_exc_trc_out_path	Exception トレースログを出力するディレクトリを，絶対パスで指定します。	これらのプロパティの機能詳細および指定できる値については，「7.7.2　Exception トレースログ」の「(1)　取得するメソッド，および取得するための設定」の「(b)　Exception トレースログを取得するための設定（プロパティの設定）」を参照してください。
28	adb_jdbc_info_max	1 ファイルへ出力する情報数の上限を指定します。	
29	adb_jdbc_cache_info_max	メモリ内で蓄積する情報数の上限を指定します。	
30	adb_jdbc_trc_out_lv	トレース取得レベルを指定します。	

注

- 各プロパティは，ほかの方法でも指定できます。ほかの指定方法，および指定の優先順位については，「7.3.3　接続情報の優先順位」を参照してください。
- 各プロパティにnull を指定した場合，指定を省略したと見なされます。

メモ

HADB 03-00 で，ユーザプロパティの各プロパティ名を次のように変更しました。変更前のプロパティ名も使用できますが，HADB 03-00 以降にバージョンアップした場合は，プロパティ名を変更することを推奨します。

項番	変更前のプロパティ名（HADB 03-00 より前のプロパティ名）	変更後のプロパティ名（HADB 03-00 以降のプロパティ名）
1	apname	adb_clt_ap_name

項番	変更前のプロパティ名（HADB 03-00 より前のプロパティ名）	変更後のプロパティ名（HADB 03-00 以降のプロパティ名）
2	host	adb_clt_rpc_srv_host
3	port	adb_clt_rpc_srv_port

7.3.2 DataSource クラスの getConnection メソッドで HADB サーバに接続する方法

DataSource と JNDI を使用した DB 接続（HADB サーバへの接続）は，JDBC 2.0 Optional Package で使用できるようになりました。

JNDI を必ず使用する必要はありませんが，JNDI を使用することで接続情報の設定が 1 回で済むというメリットがあります。DataSource クラスのインタフェース定義，および JNDI は，JDK に標準で含まれていないため，AP 開発をする場合には，JavaSoft の Web サイトから入手する必要があります。

DataSource と JNDI を使用した HADB サーバへの接続手順を次に示します。

手順

1. DataSource オブジェクトの生成
2. 接続情報の設定
3. JNDI へのDataSource の登録
4. JNDI からのDataSource の取得
5. HADB サーバへの接続

JNDI を使用しない場合は，3 および 4 の操作をする必要はありません。

JNDI を使用する場合，1～3 の操作は 1 回だけ実行します。そのあと，4 および 5 の操作をするだけで，HADB サーバに接続できます。また，4 の操作のあと，必要に応じて接続情報を変更できます。

(1) DataSource オブジェクトの生成

JDBC ドライバが提供する，DataSource クラスのオブジェクトを生成します。

DataSource クラスのオブジェクト生成で必要となる，JDBC ドライバのDataSource クラス名は AdbDataSource となります。

DataSource クラスのオブジェクトの生成例を次に示します。

```
com.hitachi.hadb.jdbc.AdbDataSource  ds = null ;
ds = new com.hitachi.hadb.jdbc.AdbDataSource() ;
```

(2) 接続情報の設定

DataSource オブジェクトに対して，接続情報設定用メソッドを呼び出し，接続情報の設定をします。接続情報取得用のメソッドも使用できるため，現在の接続情報の確認もできます。接続情報設定および取得メソッドについては，「10.5 接続情報設定および取得インタフェース」を参照してください。

(3) JNDI への DataSource の登録

DataSource オブジェクトを JNDI に登録します。

JNDI は，その実行環境によって幾つかのサービスプロバイダを選択できます。

DataSource オブジェクトの JNDI への登録例を次に示します（Windows の場合の例です）。なお，登録例では，サービスプロバイダの一つである，File System サービスプロバイダを使用しています。ほかのサービスプロバイダについては，JNDI のドキュメントを参照してください。

```
// JDBCドライバが提供するDataSourceクラスのオブジェクトを生成する
com.hitachi.hadb.jdbc.AdbDataSource  ds;
ds = new com.hitachi.hadb.jdbc.AdbDataSource();

// 接続情報を設定する
        :

// システムプロパティを取得する
Properties sys_prop = System.getProperties() ;

// File Systemサービスプロバイダのプロパティを設定する
sys_prop.put(Context.INITIAL_CONTEXT_FACTORY,
             "com.sun.jndi.fscontext.RefFSContextFactory");

// File Systemサービスプロバイダで使用するディレクトリを設定する
//（この場合，c:¥JNDI_DIRの下に登録される）
sys_prop.put(Context.PROVIDER_URL, "file:c:¥¥" + "JNDI_DIR");

// システムプロパティを更新する
System.setProperties(sys_prop) ;

// JNDIを初期化する
Context ctx = new InitialContext();

// JDBCドライバが提供するDataSourceクラスのオブジェクトを，
// jdbc/TestDataSourceという論理名称でJNDIに登録する
ctx.bind("jdbc" + "¥¥" + "TestDataSource", ds);
    :
```

なお，JDBC 2.0 規格では，JNDI に登録する論理名称は，"jdbc"というサブコンテキスト下（登録例では jdbc/TestDataSource）に登録するように推奨されています。

(4) JNDI からの DataSource の取得

JNDI からDataSource オブジェクトを取得します。

JNDI からのDataSource オブジェクトの取得例を次に示します（Windows の場合の例です）。なお，取得例では，サービスプロバイダの一つである，File System サービスプロバイダを使用しています。ほかのサービスプロバイダについては，JNDI のドキュメントを参照してください。

```
// システムプロパティを取得する
Properties sys_prop = System.getProperties() ;

// File Systemサービスプロバイダのプロパティを設定する
sys_prop.put(Context.INITIAL_CONTEXT_FACTORY,
             "com.sun.jndi.fscontext.RefFSContextFactory");

// File Systemサービスプロバイダで使用するディレクトリを設定する
//（この場合，c:¥JNDI_DIRの下に登録されている）
sys_prop.put(Context.PROVIDER_URL, "file:c:¥¥" + "JNDI_DIR");

// システムプロパティを更新する
System.setProperties(sys_prop) ;

// JNDIを初期化する
Context ctx = new InitialContext();

// jdbc/TestDataSourceという論理名称のオブジェクトをJNDIから取得する
Object obj = ctx.lookup("jdbc" + "¥¥" + "TestDataSource") ;

// 取り出したオブジェクトを，DataSourceクラスの型にキャストする
DataSource ds = (DataSource)obj;
      :
```

(5)　HADB サーバへの接続

DataSource オブジェクトに対して，getConnection メソッドを呼び出します。getConnection メソッドの呼び出し例を次に示します。

```
DataSource ds

// JNDIからDataSourceオブジェクトを取得する
                      :

// getConnectionメソッドを発行する
Connection con = ds.getConnection();
  または
Connection con = ds.getConnection("USERID", "PASSWORD");※
```

注※

メソッドの引数（認可識別子，パスワード）は，DataSource オブジェクトに設定した接続情報よりも優先されます。次に示す場合はSQLException が投入されます。

- 必要な接続情報がDataSource オブジェクトに設定されていない場合
- 接続情報の内容が不正な場合
- HADB サーバとの接続に失敗した場合

JNDI からDataSource オブジェクトを取得後，必要に応じて接続情報を再度設定できます。この場合，DataSource オブジェクトを，JDBC ドライバが提供するDataSource クラスの型にキャストしてから設定する必要があります。例を次に示します。

```
DataSource ds
com.hitachi.hadb.jdbc.AdbDataSource  adb_ds;

// JNDIからDataSourceオブジェクトを取得する
                    :

// DataSourceオブジェクトを，JDBCドライバが提供する
// DataSourceクラスの型にキャストする
adb_ds = (com.hitachi.hadb.jdbc.AdbDataSource)ds;

// 接続情報を再設定する
          :
```

7.3.3　接続情報の優先順位

(1)　HADB サーバへの接続時に必要となる接続情報

HADB サーバへの接続時に必要となる接続情報を次に示します。

- HADB サーバのホスト名
- HADB サーバのポート番号
- HADB サーバに接続する認可識別子とパスワード
- AP 識別子
- 各プロパティに指定できる上記以外の項目

これらの接続情報は，幾つかの方法で設定できます。例えば，HADB サーバのホスト名は，システムプロパティのadb_clt_rpc_srv_host に指定する方法と，接続用の URL のhost に指定する方法があります。

このように接続情報が複数の方法で設定された場合の指定の優先順位を次の表に示します。

表 7-4　接続情報の指定の優先順位

接続情報	設定方法	優先順位	
		DM	DS
HADB サーバのホスト名	システムプロパティadb_clt_rpc_srv_host の値	1	1
	DriverManager クラスのgetConnection メソッドの引数info に指定したadb_clt_rpc_srv_host プロパティの値	2	－
	DriverManager クラスのgetConnection メソッドの引数url に指定したhost の値	3	－

接続情報	設定方法	優先順位	
		DM	DS
	接続情報設定および取得インタフェースのsetHostName メソッドで設定したホスト名	−	2
HADB サーバのポート番号	システムプロパティadb_clt_rpc_srv_port の値	1	1
	DriverManager クラスのgetConnection メソッドの引数info に指定したadb_clt_rpc_srv_port プロパティの値	2	−
	DriverManager クラスのgetConnection メソッドの引数url に指定したportの値	3	−
	接続情報設定および取得インタフェースのsetPort メソッドで設定したポート番号	−	2
接続時の認可識別子，パスワード	次に示すどちらかの値 • DriverManager クラスのgetConnection メソッドの引数user およびpassword の値 • DriverManager クラスのgetConnection メソッドの引数info に指定したuser およびpassword プロパティの値	1	−
	DriverManager クラスのgetConnection メソッドの引数url に指定したuserおよびpassword の値	2	−
	次に示すどちらかの値 • DataSource インタフェースのgetConnection メソッドの引数username およびpassword に指定した値 • ConnectionPoolDataSource インタフェースのgetPooledConnection メソッドの引数user およびpassword に指定した値	−	1
	• 接続情報設定および取得インタフェースのsetUser メソッドで設定した認可識別子 • 接続情報設定および取得インタフェースのsetPassword メソッドで設定したパスワード	−	2
AP 識別子	システムプロパティadb_clt_ap_name の値	1	1
	DriverManager クラスのgetConnection メソッドの引数info に指定したadb_clt_ap_name プロパティの値	2	−
	DriverManager クラスのgetConnection メソッドの引数url に指定したadb_clt_ap_name の値	3	−
	接続情報設定および取得インタフェースのsetApName メソッドで設定したAP 識別子	−	2
HADB サーバへの接続処理のタイムアウト時間	システムプロパティadb_clt_rpc_con_wait_time の値	1	1
	DriverManager クラスのgetConnection メソッドの引数info に指定したadb_clt_rpc_con_wait_time の値	2	−
	DriverManager クラスのgetConnection メソッドの引数url に指定したadb_clt_rpc_con_wait_time の値	3	−

接続情報	設定方法	優先順位	
		DM	DS
	DriverManager クラスのsetLoginTimeout メソッドで指定した値	4	−
	次に示すどちらかの値 • DataSource インタフェースのsetLoginTimeout メソッドで指定した値 • ConnectionPoolDataSource インタフェースのsetLoginTimeout メソッドで指定した値	−	2
各プロパティに指定できる上記以外の項目（次の項目が該当） • adb_clt_rpc_sql_wait_time • adb_clt_group_name • adb_clt_fetch_size • adb_clt_sql_text_out • adb_clt_trn_iso_lv • adb_clt_sql_order_mode • adb_sql_prep_dec_div_rs_prior • adb_clt_trn_access_mode • adb_dbbuff_wrktbl_clt_blk_num • adb_sql_prep_delrsvd_use_srvdef • adb_sql_exe_max_rthd_num • adb_sql_exe_hashgrp_area_size • adb_sql_exe_hashtbl_area_size • adb_sql_exe_hashflt_area_size • adb_clt_sql_parallel_exec • adb_clt_passwd_pubkey_path • adb_jdbc_exc_trc_out_path • adb_jdbc_info_max • adb_jdbc_cache_info_max • adb_jdbc_trc_out_lv	システムプロパティに指定したプロパティの値	1	1
	DriverManager クラスのgetConnection メソッドの引数info に指定したプロパティの値	2	−
	DriverManager クラスのgetConnection メソッドの引数url に指定したプロパティの値	3	−

（凡例）

DM：DriverManager クラスを使用して接続した場合

DS：DataSource クラスを使用して接続した場合

－：接続情報を指定できません。

注

優先順位の番号が小さいほど優先順位が高くなります。1 と 2 では 1 の指定が優先されます。

(2) 各プロパティで指定できる接続情報の一覧

HADB サーバへの接続時に必要となる接続情報は，システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティで指定できます。各プロパティで指定できる接続情報の一覧を次の表に示します。

表 7-5 各プロパティで指定できる接続情報の一覧

項番	分類	プロパティ名	各プロパティでの指定可否		
			システムプロパティ	ユーザプロパティ	接続用の URL のプロパティ
1	クライアント定義と同じ名称，同じ機能を持つプロパティ※1	adb_clt_rpc_srv_host	○	○	△※2
2		adb_clt_rpc_srv_port	○	○	△※3
3		adb_clt_rpc_con_wait_time	○	○	○
4		adb_clt_rpc_sql_wait_time	○	○	○
5		adb_clt_ap_name	○	○	○
6		adb_clt_group_name	○	○	○
7		adb_clt_fetch_size	○	○	○
8		adb_dbbuff_wrktbl_clt_blk_num	○	○	○
9		adb_sql_exe_max_rthd_num	○	○	○
10		adb_sql_exe_hashgrp_area_size	○	○	○
11		adb_sql_exe_hashtbl_area_size	○	○	○
12		adb_sql_exe_hashflt_area_size	○	○	○
13		adb_clt_sql_parallel_exec	○	○	○
14		adb_sql_prep_delrsvd_use_srvdef	○	○	○
15		adb_clt_trn_iso_lv	○	○	○
16		adb_clt_trn_access_mode	○	○	○
17		adb_clt_sql_text_out	○	○	○
18		adb_clt_sql_order_mode	○	○	○
19		adb_sql_prep_dec_div_rs_prior	○	○	○

項番	分類	プロパティ名	各プロパティでの指定可否		
			システムプロパティ	ユーザプロパティ	接続用の URL のプロパティ
20		adb_clt_passwd_pubkey_path	○	○	○
21	Exception トレースログに関するプロパティ	adb_jdbc_exc_trc_out_path	○	○	○
22		adb_jdbc_info_max	○	○	○
23		adb_jdbc_cache_info_max	○	○	○
24		adb_jdbc_trc_out_lv	○	○	○
25	上記以外のプロパティ	user	×	○	○
26		password	×	○	○
27		encodelang	×	○	○
28		methodtrace	×	○	○
29		tracenum	×	○	○
30		sqlwarningkeep	×	○	○

（凡例）

○：指定できるプロパティ

△：指定できないプロパティではあるが，代替となる指定あり

×：指定できないプロパティ

注※1

これらのプロパティは，クライアント定義のオペランド名と同じプロパティ名であり，クライアント定義のオペランドと同じ機能を持っています。

注※2

接続用の URL のプロパティでは，接続先の HADB サーバのホスト名をadb_clt_rpc_srv_host で指定できません。接続先の HADB サーバのホスト名は，接続用の URL のhost で指定します。

注※3

接続用の URL のプロパティでは，HADB サーバのポート番号をadb_clt_rpc_srv_port で指定できません。HADB サーバのポート番号は，接続用の URL のport で指定します。

重要

JDBC ドライバを使用している場合，クライアント定義のオペランドの指定は適用されません。システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティの指定が適用されます。

システムプロパティについては，「3.1.6　システムプロパティの設定」を参照してください。

ユーザプロパティについては，「7.3.1　DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法」の「(2)　getConnection メソッドによる HADB サーバへの接続」の「(d)　引数 info の指定内容（ユーザプロパティの指定)」を参照してください。

接続用の URL のプロパティについては，「7.3.1　DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法」の「(2)　getConnection メソッドによる HADB サーバへの接続」の「(a)　引数 url の指定内容（接続用の URL の指定)」を参照してください。

7.4　データを検索する場合（SELECT 文を実行する場合）

ここでは，JDBC ドライバを使用したデータの検索方法について説明します。

7.4.1　データの検索方法

SELECT 文を実行してデータを検索する場合，次に示す順序で処理を実行します。

- Statement オブジェクトを生成する
- SELECT 文を実行する
- 検索結果を取得する

(1)　Statement オブジェクトの生成

Statement オブジェクトを生成して，SELECT 文を HADB サーバに送ります。

なお，HADB サーバへの接続がすでに確立されている場合は，Connection オブジェクトのcreateStatement メソッドを使用してStatement オブジェクトを生成できます。

Statement オブジェクトの生成例を次に示します。

```
// HADBサーバへ接続します
Connection con = DriverManager.getConnection(url,info);

// Statementオブジェクトを生成します
Statement stmt = con.createStatement();
```

(2)　SELECT 文の実行

executeQuery メソッドの引数にSELECT 文を指定して SELECT 文を実行します。SELECT 文の実行例を次に示します。

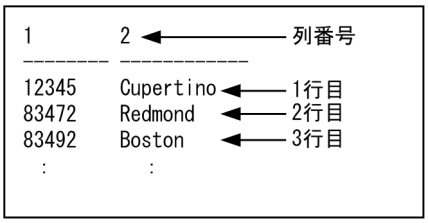
```
Statement stmt = con.createStatement();

// SELECT文を実行して，ResultSetオブジェクトを取得します
ResultSet rs = stmt.executeQuery("SELECT ¥"CODE¥", ¥"STATE¥" FROM ¥"SAMPLE¥"");
```

SELECT 文を実行すると，ResultSet オブジェクトに検索結果が格納されます。

(3)　検索結果の取得

ResultSet オブジェクトには，列番号と検索結果に対応する値から構成される表の形式で検索結果が格納されます。ResultSet オブジェクトの形式の例を次の図に示します。

図 7-2　ResultSet オブジェクトの形式の例



ResultSet オブジェクトは，現在行を指し示すカーソルを保持しています。ResultSet オブジェクトから検索結果を取得するには，next メソッドでカーソルを移動し，get*XXX* メソッドで現在行のデータを取得します。

ResultSet オブジェクトの作成時，先頭行の前にカーソルが位置づけられています。最初のnext メソッドの呼び出しによってカーソルは先頭行に移動します。next メソッドが呼び出されるたびにカーソルが 1 行ずつ下に移動します。

検索結果データの取得例を次に示します。

```
ResultSet rs = stmt.executeQuery("SELECT ¥"CODE¥", ¥"STATE¥" FROM ¥"SAMPLE¥"");

// 結果行がなくなるまで繰り返します
while(rs.next())
{
  // 1列目のデータを取得します
  int i = rs.getInt(1);
  // 2列目のデータを取得します
  String s = rs.getString(2);
  // 結果データを出力します
  System.out.println("検索結果: " + i + ", " + s);
}
```

(4)　同一コネクションで複数の SELECT 文を同時実行する際の注意事項

同一コネクションで複数のSELECT 文を同時に実行すると，SELECT 文の実行に必要な処理リアルスレッド数が不足することがあります。この場合，必要な処理リアルスレッド数が確保されるまで，処理リアルスレッドの確保処理が繰り返されます。そのため，次に示すどれかの対策を実施して，処理リアルスレッドの確保処理が無限に繰り返されることを防いでください。まずは，対策の 1.が実施できるかどうかを検討してください。

対策

1. AP を修正する

   AP の修正ができる場合は，不要となったResultSet オブジェクトをクローズするように AP を修正してください。AP の修正ができない場合は，2.の対策の実施を検討してください。

2. サーバ定義を変更する

   次の計算式を満たすようにサーバ定義を変更できる場合は，サーバ定義を変更してください。

$$A \geqq B \times C \times D$$

A：処理スレッド数（adb_sys_rthd_num オペランドの値）

B：最大 SQL 処理リアルスレッド数（adb_sql_exe_max_rthd_num オペランドの値）

C：1 コネクション内で同時に実行する（カーソルオープン状態にする）SELECT 文の数

D：同時に SQL 文を実行するコネクション数

3. 待ち時間を設定する

上記の 1.または 2.の対策を実施できない場合は，必要な処理リアルスレッド数を確保できるまでの待ち時間を，次のメソッドまたはプロパティで指定してください。

- Statement インタフェースのsetQueryTimeout メソッド
- システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティの adb_clt_rpc_sql_wait_time

7.4.2 ？パラメタの使用方法

？パラメタを使用してデータを検索する場合，次に示す順序で処理を実行します。

- PreparedStatement オブジェクトを生成する
- SELECT 文を実行する
- 検索結果を取得する

(1) PreparedStatement オブジェクトの生成

PreparedStatement オブジェクトを生成して，？パラメタを指定したSELECT 文を HADB サーバに送ります。

なお，HADB サーバへの接続がすでに確立されている場合は，Connection オブジェクトのprepareStatement メソッドを使用してPreparedStatement オブジェクトを生成できます。

PreparedStatement オブジェクトの生成例を次に示します。

```
// HADBサーバへ接続します
Connection con = DriverManager.getConnection(url, info);

// PreparedStatementオブジェクトを生成します
PreparedStatement pstmt =
con.prepareStatement("SELECT * FROM ¥"SAMPLE¥" WHERE ¥"CODE¥" = ? AND ¥"STATE¥" = ?");
```

prepareStatement メソッドの引数にSELECT 文を指定すると，このSELECT 文が前処理されて PreparedStatement オブジェクトが生成されます。

SQL 文に？パラメタを指定している場合，SQL 文の実行前に各？パラメタの値を設定する必要があります。？パラメタに値を設定するには，set*XXX* メソッドを使用します。

(2)　SELECT 文の実行

引数指定なしのexecuteQuery メソッドを使用してSELECT 文を実行します。？パラメタを使用したSELECT 文の実行例を次に示します。

```
PreparedStatement pstmt =
con.prepareStatement("SELECT * FROM ¥"SAMPLE¥" WHERE ¥"CODE¥" = ? AND ¥"STATE¥" = ?");

// 1番目の？パラメタに値を設定します
pstmt.setInt(1, 12345);
// 2番目の？パラメタに値を設定します
pstmt.setString(2, "Boston");

// 前処理済みのSQL文を実行して，ResultSetオブジェクトを取得します
ResultSet rs = pstmt.executeQuery();
```

SELECT 文を実行すると，ResultSet オブジェクトに検索結果が格納されます。

(3)　検索結果の取得

検索結果の取得方法については，「7.4.1　データの検索方法」の「(3)　検索結果の取得」を参照してください。

7.5 データを追加，更新，または削除する場合（INSERT 文，UPDATE 文，またはDELETE 文を実行する場合）

INSERT 文，UPDATE 文，またはDELETE 文などの操作系 SQL によるデータの追加，更新，または削除をするには，Statement オブジェクト（？パラメタを使用する場合はPreparedStatement オブジェクト）のexecuteUpdate メソッドまたはexecuteLargeUpdate メソッドを使用します。

データの更新および削除の実行例を次に示します。

```
Connection con = DriverManager.getConnection(url, info);
Statement stmt = con.createStatement();

// 条件に合致するデータを更新します
stmt.executeUpdate("UPDATE ¥"SAMPLE¥" SET ¥"CODE¥"=98765 WHERE ¥"STATE¥" = 'Redmond'");

// すべての行を削除します
stmt.executeUpdate("DELETE FROM ¥"SAMPLE¥" ");
```

■更新操作によるカーソルを使用した検索への影響

カーソルを使用した検索中に更新操作を行うと，タイミングによっては更新操作の結果が，検索の結果に反映されることがあります。更新操作の結果を検索の結果に反映させないようにするには，次のように運用してください。

- カーソルを閉じたあとに，行の追加または更新を行う
- 追加または更新する行が，検索結果と一致しないようにデータや探索条件などを工夫する

7.6　データ処理

この章では，JDBC ドライバと HADB サーバ間のデータ処理方式について説明します。

7.6.1　データ型のマッピング

ここでは，HADB のデータ型と JDBC の SQL データ型のマッピングについて説明します。

(1)　HADB のデータ型と JDBC の SQL データ型の対応

HADB のデータ型と JDBC の SQL データ型は，完全には一致していません。そのため，JDBC ドライバが，HADB のデータ型と JDBC の SQL データ型をマッピング（変換）します。マッピングできないデータ型を使用してアクセスしようとすると，SQLException が投入されます。

データ型のマッピングは，ResultSet またはPreparedStatement クラスのget*XXX* メソッドおよびset*XXX* メソッドで実行します。get*XXX* メソッドおよびset*XXX* メソッドのマッピング規則については，JDBC1.0 規格または JDBC 2.0 基本規格のドキュメントを参照してください。

HADB のデータ型と JDBC の SQL データ型の対応を次の表に示します。

表 7-6　HADB のデータ型と JDBC の SQL データ型の対応

HADB のデータ型	JDBC の SQL データ型
INTEGER	BIGINT
SMALLINT	INTEGER，SMALLINT※1
DECIMAL，NUMERIC	DECIMAL（，NUMERIC）※2
DOUBLE PRECISION，FLOAT	DOUBLE（，FLOAT）※2
CHAR	CHAR
VARCHAR	VARCHAR（，LONGVARCHAR）※2
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
BINARY	BINARY（，VARBINARY，LONGVARBINARY）※2
VARBINARY	VARBINARY（，BINARY，LONGVARBINARY）※2
ROW	BINARY（，VARBINARY，LONGVARBINARY）※2
BOOLEAN※3	BOOLEAN
ARRAY	ARRAY

注※1

DatabaseMetaData から生成したResultSet にだけ存在する列です。データ型がshort で規定されている列の場合，ResultSetMetaData で取得できるメタデータにはSMALLINT に対応した値を返却します。ただし，メタデータ以外はINTEGER に対応した値を返却します。

注※2

（ ）内のデータ型は，setNull メソッドまたはsetObject メソッドの引数に JDBC の SQL データ型を指定する場合にだけ対応します。HADB のデータ型から JDBC の SQL データ型へマッピングする際には対応しません。

注※3

DatabaseMetaData のgetTypeInfo メソッドなどで生成される，ResultSet オブジェクトが持つBOOLEAN 型の列を指します。

(2) 検索データ取得時のマッピング

ResultSet クラスのget*XXX* メソッドと，JDBC の SQL データ型とのマッピングを次の表に示します。マッピングできない JDBC の SQL データ型に対してget*XXX* メソッドが呼び出された場合は，SQLException が投入されます。

表 7-7 getXXX メソッドと JDBC の SQL データ型とのマッピング

メソッド名	JDBC の SQL データ型										
	BIGINT	INTEGER	DECIMAL	DOUBLE	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY, VARBINARY	ARRAY
getByte	○	○	○	○	○※	○※	×	×	×	×	×
getShort	○	○	○	○	○※	○※	×	×	×	×	×
getInt	○	◎	○	○	○※	○※	×	×	×	×	×
getLong	◎	○	○	○	○※	○※	×	×	×	×	×
getFloat	○	○	○	○	○※	○※	×	×	×	×	×
getDouble	○	○	○	◎	○※	○※	×	×	×	×	×
getBigDecimal	○	○	◎	○	○※	○※	×	×	×	×	×
getBoolean	○	○	○	○	○	○	×	×	×	×	×
getString	○	○	○	○	◎	◎	○	○	○	○	×
getBytes	×	×	×	×	×	×	×	×	×	◎	×
getDate	×	×	×	×	○※	○※	◎	○	○	×	×
getTime	×	×	×	×	○※	○※	×	◎	○	×	×

メソッド名	JDBC の SQL データ型										
	BIGINT	INTEGER	DECIMAL	DOUBLE	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY, VARBINARY	ARRAY
getTimestamp	×	×	×	×	○※	○※	○	○	◎	×	×
getAsciiStream	×	×	×	×	○	○	×	×	×	○	×
getObject	○	○	○	○	○	○	○	○	○	○	○
getCharacterStream	×	×	×	×	○	○	×	×	×	○	×
getBinaryStream	×	×	×	×	×	×	×	×	×	○	×
getArray	×	×	×	×	×	×	×	×	×	×	◎

（凡例）

◎：マッピングすることを推奨します。

○：マッピングできます。ただし，変換元のデータ形式によっては，データの欠落や変換エラーになることがあるため，注意してください。

×：マッピングできません。

注※

このメソッドでの変換の際に，データベースから取得した文字列データの前後に半角空白がある場合は，半角空白を取り除きます。また，半角空白を取り除いたあと，get*XXX* メソッドが返却する Java のデータ型に変換します。

Java のデータ型に変換する場合の注意事項を次に示します。

- 文字列データに小数点以下の表現がある場合，getByte メソッド，getInt メソッド，getShort メソッド，またはgetLong メソッドを実行すると，小数点以下を切り捨てて整数だけを変換し，返却します。
- 文字列データに全角文字が含まれている場合，変換はしないでSQLException が投入されます。
- 文字列データを Java のデータ型に変換した結果，オーバフローが発生した場合，SQLException が投入されます。

(3) ？パラメタ設定時のマッピング

PreparedStatement クラスのset*XXX* メソッドと，マッピングされる JDBC の SQL データ型を次の表に示します。使用できない JDBC の SQL データ型の場合，SQLException が投入されます。

PreparedStatement クラスのset*XXX* メソッドと各 JDBC の SQL データ型とのマッピングを次の表に示します。

表 7-8　setXXX メソッドと JDBC の SQL データ型とのマッピング

メソッド名	JDBC の SQL データ型										
	BIGINT	INTEGER	DECIMAL※1	DOUBLE	CHAR	VARCHAR	DATE	TIME	TIMESTAMP	BINARY，VARBINARY	ARRAY
setByte	○	○	○	○	○	○	×	×	×	×	×
setShort	○	○	○	○	○	○	×	×	×	×	×
setInt	○	◎	○	○	○	○	×	×	×	×	×
setLong	◎	○	○	○	○	○	×	×	×	×	×
setFloat	○	○	○	○	○	○	×	×	×	×	×
setDouble	○	○	○	◎	○	○	×	×	×	×	×
setBigDecimal	○	○	◎	○	○	○	×	×	×	×	×
setBoolean	○	○	○	○	○	○	×	×	×	×	×
setString	○	○	○	○	◎	◎	○	○	○	○	×
setBytes	×	×	×	×	×	×	×	×	×	◎	×
setDate	×	×	×	×	○	○	◎	○	○	×	×
setTime	×	×	×	×	○	○	×	◎	○	×	×
setTimestamp※2	×	×	×	×	○	○	○	○	◎	×	×
setAsciiStream	×	×	×	×	○	○	×	×	×	○	×
setObject※3	○	○	○	○	○	○	○	○	○	○	×
setCharacterStream	×	×	×	×	○※4	○※4	×	×	×	○	×
setBinaryStream	×	×	×	×	×	×	×	×	×	○	×

（凡例）

◎：マッピングすることを推奨します。

○：マッピングできます。ただし，変換元データの形式によっては，データの欠落や変換エラーとなることがあるため，注意してください。

×：マッピングできません。

注※1

HADB のデータ型であるDECIMAL 型またはNUMERIC 型の？パラメタに対してset*XXX* メソッドで値を指定する場合に，？パラメタの精度および位取りと，値の精度および位取りが一致していないときの動作を次に示します。

- ？パラメタの精度よりも大きいとき：SQLException が投入されます。
- ？パラメタの精度よりも小さいとき：拡張します。
- ？パラメタの位取りよりも大きいとき：実際の位取りで切り捨てます。
- ？パラメタの位取りよりも小さいとき：0 で補完して，拡張します。

注※2

HADB のデータ型であるTIME 型またはTIMESTAMP 型の？パラメタに対してset*XXX* メソッドで値を指定する場合に，？パラメタの小数秒精度と値の小数秒精度が一致していないときの動作を次に示します。

- ？パラメタの小数秒精度よりも大きいとき：切り捨てます。
- ？パラメタの小数秒精度よりも小さいとき：拡張します。

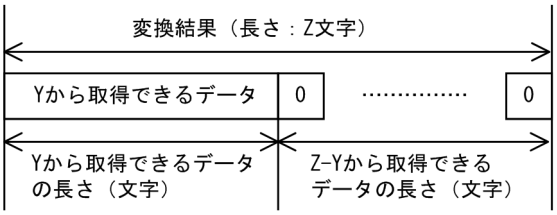
注※3

setObject メソッドにInputStream クラスおよびReader クラス（サブクラスを含む）のオブジェクトは指定できません。

注※4

java.io.Reader オブジェクトから取得できるデータの長さが，引数で指定した長さより短い場合，次に示すように引数で指定した長さまで 0 を補完します。

```
int X,Z;
java.io.Reader Y;
setCharacterStream(X,Y,Z)
```



7.6.2 データの変換処理

ここでは，set*XXX* およびget*XXX* メソッド実行時のデータ変換処理について説明します。

(1) setXXX メソッド実行時のデータ変換処理（DATE 型，TIME 型，または TIMESTAMP 型の場合）

setTime，setDate，setTimestamp，またはsetString メソッドに，HADB のデータ型のDATE 型，TIME 型，またはTIMESTAMP 型のデータを設定した場合の変換処理について説明します。

HADB のデータ型のDATE 型，TIME 型，またはTIMESTAMP 型の列に対して，setTime，setDate，setTimestamp，またはsetString メソッドを使用してデータを設定した場合，次の表に示す規則に従ってデータ変換が行われます。

表 7-9　setXXX メソッド実行時の変換処理

実行するメソッド	DATE 型の変換処理	TIME 型の変換処理	TIMESTAMP 型の変換処理
setDate(Date Obj)※1	AP の設定値どおりにデータベースに格納されます。	00:00:00 がデータベースに格納されます。	AP の設定値 *YYYY-MM-DD* の後ろに「00:00:00」を付与したデータがデータベースに格納されます。
setTime(Time Obj)※2	SQLException が投入されます。	AP の設定値 *hh*:*mm*:*ss*.*fff* のデータがデータベースに格納されます。	AP の設定値 *hh*:*mm*:*ss*.*fff* の前に「1970-01-01」を付与したデータがデータベースに格納されます。
setTimestamp(Timestamp Obj)※3	AP の設定値から「*YYYY-MM-DD*」を抜き出したデータがデータベースに格納されます。	AP の設定値から「*hh*:*mm*:*ss*.*ffffffff*」を抜き出したデータがデータベースに格納されます。	AP の設定値 *YYYY-MM-DD*△*hh*:*mm*:*ss*.*ffffffff* のデータがデータベースに格納されます。※4
setString(*YYYY-MM-DD* 形式の文字列)	指定された日付が java.sql.Date.valueOf()で変換されてデータベースに格納されます。※5	00:00:00 がデータベースに格納されます。	SQLException が投入されます。
setString(*hh*:*mm*:*ss*[.*f*...]形式の文字列)	SQLException が投入されます。	AP の設定値 *hh*:*mm*:*ss*[.*f*...]のデータがデータベースに格納されます。	SQLException が投入されます。
setString(*YYYY-MM-DD*△*hh*:*mm*:*ss*[.*f*...]形式の文字列)※4	SQLException が投入されます。	AP の設定値 *hh*:*mm*:*ss*[.*f*...]のデータがデータベースに格納されます。	AP の設定値 *YYYY-MM-DD*△*hh*:*mm*:*ss*[.*f*...]のデータがデータベースに格納されます。※5

注

実際に存在しない日時を指定した場合は，Java 仮想マシンが返す値となります。

注※1

「Date Obj」は，java.sql.Date オブジェクト「*年-月-日*」の値を持つオブジェクトです。

注※2

「Time Obj」は，java.sql.Time オブジェクト「*時*：*分*：*秒*.*ミリ秒*」の値を持つオブジェクトです。

注※3

「Timestamp Obj」は，java.sql.Timestamp オブジェクト「*年-月-日　時*：*分*：*秒*．*ナノ秒*」の値を持つオブジェクトです。

注※4

△は半角空白を示します。

注※5

存在しない日時を指定した場合の結果はjava.sql.Date.valueOf()，java.sql.Time.valueOf()，およびjava.sql.Timestamp.valueOf()に依存します。

例 1：25:00:00 は，01:00:00 となります。

例 2：2000-01-32 は，2000-02-01 となります。

例 3：1582-10-05 は，1582-10-15 となります（ユリウス暦とグレゴリオ暦が切り替わります）。

(2) getXXX メソッド実行時のデータ変換処理（DATE 型，TIME 型，TIMESTAMP 型，または文字列型の場合）

getTime，getDate，またはgetTimestamp メソッドに，HADB のデータ型のDATE 型，TIME 型，TIMESTAMP 型，または文字列型（CHAR 型，VARCHAR 型）のデータを設定した場合の変換処理について説明します。

HADB のデータ型のDATE 型，TIME 型，TIMESTAMP 型，または文字列型の列に対して，getTime，getDate，またはgetTimestamp メソッドを使用してデータを設定した場合，次の表に示す規則に従ってデータ変換が行われます。

表 7-10　getXXX メソッド実行時の変換処理

実行するメソッド	DATE 型の変換処理	TIME 型の変換処理	TIMESTAMP 型の変換処理	文字列型の変換処理
getDate()※1	データベースに格納されている値どおりに java.sql.Date のオブジェクトとして取得されます。※2	1970-01-01 のデータを持つjava.sql.Date のオブジェクトとして取得されます。※2	データベースから取得したTIMESTAMP 型のデータから，「*年-月-日*」のデータを抜き出したデータが java.sql.Date のオブジェクトとして取得されます。※2	文字列表現「*YYYY-MM-DD*」だけが java.sql.Date のオブジェクトとして取得されます。それ以外は SQLException が投入されます。
getTime()※1	SQLException が投入されます。	データベースから取得したTIME 型のデータから，「*時*：*分*：*秒*.*ミリ秒*」のデータを抜き出したデータが java.sql.Time のオブジェクトとして取得されます。※2	データベースから取得したTIMESTAMP 型のデータから，「*時*：*分*：*秒*.*ミリ秒*」のデータを抜き出したデータが java.sql.Time のオブジェクトとして取得されます。※2	文字列表現「*hh*:*mm*:*ss*[.*f*...]」だけがjava.sql.Time のオブジェクトとして取得されます。それ以外はSQLException が投入されます。
getTimestamp()※1	データベースから取得したDATE 型のデータの後ろに「00:00:00.000000000」を付加したデータが java.sql.Timestamp の	データベースから取得したTIME 型のデータの前に「1970-01-01」を付加したデータが java.sql.Timestamp の	データベースから取得したTIMESTAMP 型のデータから「*年-月-日 時*：*分*：*秒*.*ナノ秒*」のデータを抜き出したデータが	TIMESTAMP 型の文字列表現「*YYYY-MM-DD*△*hh*:*mm*:*ss*[.*f*...]」（△は半角空白）だけが java.sql.Timestamp の

実行するメソッド	DATE 型の変換処理	TIME 型の変換処理	TIMESTAMP 型の変換処理	文字列型の変換処理
	オブジェクトとして取得されます。	オブジェクトとして取得されます。	java.sql.Timestamp のオブジェクトとして取得されます。	オブジェクトとして取得されます。それ以外はSQLException が投入されます。

注※1

データベースに格納されている日時とjava.sql.Time，java.sql.Date，およびjava.sql.Timestamp から得られる日時は異なることがあります。

例 1：25:00:00 は，01:00:00 となります。

例 2：2000-01-32 は，2000-02-01 となります。

例 3：1582-10-05 と1582-10-15 は，どちらも1582-10-15 となります（ユリウス暦とグレゴリオ暦が切り替わります）。

注※2

値の設定について記載していない日付項目（*年-月-日*）の設定値は「1970-01-01」，時間項目（*時*：*分*：*秒*）の設定値は「00：00：00」とします。

7.6.3 オーバフローが発生したときの処理

ここでは，set*XXX* およびget*XXX* メソッド実行時のオーバフローの可能性について説明します。

(1) setXXX メソッド（setObject メソッドを除く）実行時のオーバフローの可能性

set*XXX* メソッド実行時のオーバフローの可能性（setObject メソッドを除く）を次の表に示します。

表 7-11 setXXX メソッド実行時のオーバフローの可能性（setObject メソッドを除く）

実行するメソッド	HADB のデータ型									
	SMALLINT	INTEGER	DOUBLE PRECISION, FLOAT	DECIMAL, NUMERIC	CHAR, VARCHAR	DATE※	TIME※	TIMESTAMP※	ROW	BINARY, VARBINARY
setByte	○	○	○	×	○	−	−	−	−	−
setShort	○	○	○	×	○	−	−	−	−	−
setInt	○	○	○	×	○	−	−	−	−	−
setLong	×	○	○	×	○	−	−	−	−	−
setFloat	×	×	○	×	○	−	−	−	−	−

実行するメソッド	HADB のデータ型									
	SMALLINT	INTEGER	DOUBLE PRECISION, FLOAT	DECIMAL, NUMERIC	CHAR, VARCHAR	DATE※	TIME※	TIMESTAMP※	ROW	BINARY, VARBINARY
setDouble	×	×	○	×	○	−	−	−	−	−
setBigDecimal	×	×	○	×	○	−	−	−	−	−
setBoolean	○	○	○	○	○	−	−	−	−	−
setString	×	×	○	×	○	×	○	×	−	○
setBytes	−	−	−	−	−	−	−	−	○	○
setDate	−	−	−	−	○	×	○	×	−	−
setTime	−	−	−	−	○	−	○	×	−	−
setTimestamp	−	−	−	−	○	×	○	×	−	−
setAsciiStream	−	−	−	−	○	−	−	−	−	○
setBinaryStream	−	−	−	−	○	−	−	−	−	○
setCharacterStream	−	−	−	−	○	−	−	−	−	○

（凡例）

○：値に関係なく，オーバフローしません。

×：値によっては，オーバフローすることがあります。

−：この組み合わせでは使用できません。

注※

java.sql.Date，java.sql.Time，またはjava.sql.Timestamp クラスのgetTime メソッドで取得した値が 253,402,268,399,999 より大きいか，または-62,135,802,000,000 より小さいオブジェクトである場合，オーバフローします。なお，getTime メソッドは，1970 年 1 月 1 日 0 時 0 分 0 秒（グリニッジ標準時）からのミリ秒数を返します。

253,402,268,399,999 は HADB のTIMESTAMP 型に格納できる最大値から，-62,135,802,000,000 は java.sql.Timestamp クラスで表現できる最小値から，次に示すメソッドで取得できます。

253,402,268,399,999：

```
Timestamp.valueOf("9999-12-31 23:59:59.999999").getTime()
```

-62,135,802,000,000：

```
Timestamp.valueOf("0001-01-01 00:00:00.0").getTime()
```

(2) setObject メソッド実行時のオーバフローの可能性

setObject メソッド実行時のオーバフローの可能性を次の表に示します。

表 7-12 setObject メソッド実行時のオーバフローの可能性

setObject メソッドで指定するオブジェクトのデータ型	HADB のデータ型									
	SMALLINT	INTEGER	DOUBLE PRECISION, FLOAT	DECIMAL, NUMERIC	CHAR, VARCHAR	DATE※	TIME※	TIMESTAMP※	ROW	BINARY, VARBINARY
Byte	○	○	○	×	○	−	−	−	−	−
Short	○	○	○	×	○	−	−	−	−	−
Integer	○	○	○	×	○	−	−	−	−	−
Long	×	○	○	×	○	−	−	−	−	−
Decimal	×	×	○	×	○	−	−	−	−	−
Float	×	×	○	×	○	−	−	−	−	−
Double	×	×	○	×	○	−	−	−	−	−
Boolean	○	○	○	○	○	−	−	−	−	−
String	×	×	○	×	○	×	○	×	−	−
Date	−	−	−	−	○	×	○	×	−	−
Time	−	−	−	−	○	−	○	−	−	−
Timestamp	−	−	−	−	○	×	○	×	−	−
byte[]	−	−	−	−	○	−	−	−	○	○

（凡例）

○：値に関係なく，オーバフローしません。

×：値によっては，オーバフローすることがあります。

−：この組み合わせでは使用できません。

注※

java.sql.Date，java.sql.Time，またはjava.sql.Timestamp クラスのgetTime メソッドで取得した値が253,402,268,399,999 より大きいか，または-62,135,802,000,000 より小さいオブジェクトである場合，オーバフローします。なお，getTime メソッドは，1970 年 1 月 1 日 0 時 0 分 0 秒（グリニッジ標準時）からのミリ秒数を返します。

253,402,268,399,999 は HADB のTIMESTAMP 型に格納できる最大値から，-62,135,802,000,000 はjava.sql.Timestamp クラスで表現できる最小値から，次に示すメソッドで取得できます。

253,402,268,399,999：

```
Timestamp.valueOf("9999-12-31 23:59:59.999999").getTime()
```

-62,135,802,000,000：

```
Timestamp.valueOf("0001-01-01 00:00:00.0").getTime()
```

(3) getXXX メソッド（getObject メソッドを除く）実行時のオーバフローの可能性

get*XXX* メソッド実行時のオーバフローの可能性（getObject メソッドを除く）を次の表に示します。

表 7-13 getXXX メソッド実行時のオーバフローの可能性（getObject メソッドを除く）

実行するメソッド	HADB のデータ型									
	SMALLINT	INTEGER	DOUBLE PRECISION, FLOAT	DECIMAL, NUMERIC	CHAR, VARCHAR	DATE	TIME	TIMESTAMP	ROW	BINARY, VARBINARY
getByte	×	×	×	×	×	−	−	−	−	−
getShort	×	×	×	×	×	−	−	−	−	−
getInt	○	×	×	×	×	−	−	−	−	−
getLong	○	○	×	×	×	−	−	−	−	−
getFloat	○	○	○	○	○	−	−	−	−	−
getDouble	○	○	○	○	○	−	−	−	−	−
getBigDecimal	○	○	○	○	○	−	−	−	−	−
getBoolean	○	○	○	○	○	−	−	−	−	−
getString	○	○	○	○	○	○	○	○	−	○
getBytes	−	−	−	−	−	−	−	−	○	○

実行するメソッド	HADB のデータ型									
	SMALLINT	INTEGER	DOUBLE PRECISION, FLOAT	DECIMAL, NUMERIC	CHAR, VARCHAR	DATE	TIME	TIMESTAMP	ROW	BINARY, VARBINARY
getDate	−	−	−	−	○	○	○	○	−	−
getTime	−	−	−	−	○	−	○	○	−	−
getTimestamp	−	−	−	−	○	○	○	○	−	−
getAsciiStream	−	−	−	−	○	−	−	−	−	○
getBinaryStream	−	−	−	−	○	−	−	−	−	○
getCharacterStream	−	−	−	−	○	−	−	−	−	○
getArray	−	−	−	−	○	−	−	−	−	○

（凡例）

○：値に関係なく，オーバフローしません。

×：値によっては，オーバフローすることがあります。

−：この組み合わせでは使用できません。

(4) getObject メソッド実行時のオーバフローの可能性

getObject メソッド実行時のオーバフローの可能性を次の表に示します。

表 7-14 getObject メソッド実行時のオーバフローの可能性

getObject メソッドで取得するオブジェクトのデータ型	HADB のデータ型									
	SMALLINT	INTEGER	DOUBLE PRECISION, FLOAT	DECIMAL, NUMERIC	CHAR, VARCHAR	DATE	TIME	TIMESTAMP	ROW	BINARY, VARBINARY
Byte	×	×	×	×	×	−	−	−	−	−
Short	×	×	×	×	×	−	−	−	−	−
Int	○	×	×	×	×	−	−	−	−	−
Long	○	○	×	×	×	−	−	−	−	−

getObject メソッドで取得するオブジェクトのデータ型	HADB のデータ型									
	SMALLINT	INTEGER	DOUBLE PRECISION, FLOAT	DECIMAL, NUMERIC	CHAR, VARCHAR	DATE	TIME	TIMESTAMP	ROW	BINARY, VARBINARY
Float	○	○	×	×	×	−	−	−	−	−
Double	○	○	○	×	×	−	−	−	−	−
BigDecimal	○	○	○	×	×	−	−	−	−	−
Boolean	○	○	○	○	○	−	−	−	−	−
String	○	○	○	○	○	○	○	○	−	○
Bytes	−	−	−	−	−	−	−	−	○	○
Date	−	−	−	−	○	○	○	○	−	−
Time	−	−	−	−	○	−	○	○	−	−
Timestamp	−	−	−	−	○	○	○	○	−	−
AsciiStream	−	−	−	−	○	−	−	−	−	○
BinaryStream	−	−	−	−	○	−	−	−	−	○
Object	○	○	○	○	○	○	○	○	○	○
CharacterStream	−	−	−	−	○	−	−	−	−	○

（凡例）

○：値に関係なく，オーバフローしません。

×：値によっては，オーバフローすることがあります。

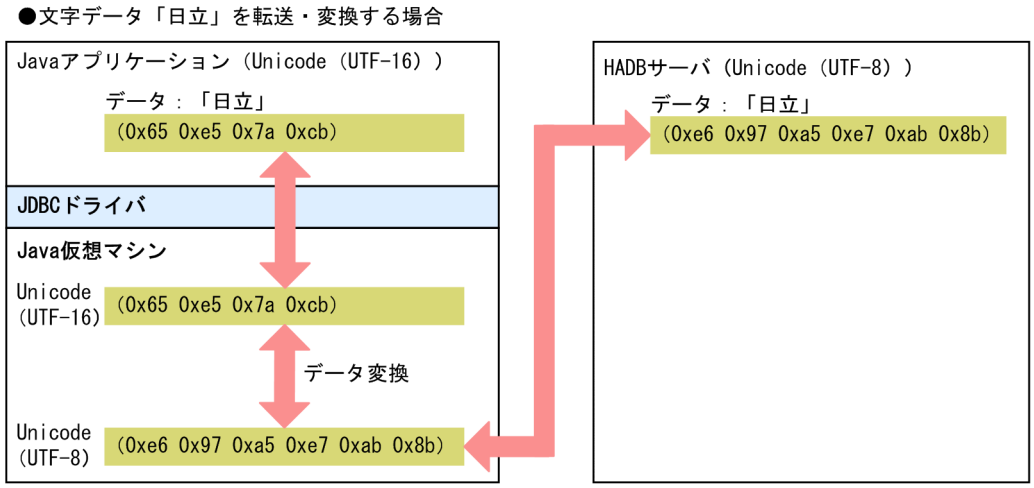
−：この組み合わせでは使用できません。

7.6.4 文字コードの変換

Java プログラムでは，文字コードは Unicode（UTF-16）で扱うため，JDBC ドライバが HADB の文字データと Unicode（UTF-16）との相互文字コード変換をします。この文字コード変換処理で，JDBC ドライバは Java 仮想マシンが提供するエンコーダを利用します。

HADB の文字データと Unicode（UTF-16）との相互文字コード変換の流れを次の図に示します。

図 7-3　HADB の文字データと Unicode（UTF-16）との相互文字コード変換の流れ



HADB との文字データのやり取りの際，Java 仮想マシンのエンコーダに対して，JDBC ドライバが文字セット名称を指定します。このとき，HADB サーバの文字コード Unicode（UTF-8）を取得し，それに相当する文字セット名称にします。

HADB サーバの文字コードに対応する，Java 仮想マシンのエンコーダに対して指定する文字セット名称を次の表に示します。

表 7-15　HADB サーバの文字コードに対応する文字セット名称

項番	HADB サーバの文字コード（環境変数 ADBLANG に指定した文字コード）	Java 仮想マシンのエンコーダに指定する文字セット名称※
1	Unicode（UTF-8）（UTF8）	UTF-8
2	Shift-JIS（SJIS）	Windows-31j（MS932）

注※

Java 仮想マシンのエンコーダに表中の文字セット名称が指定されるのは，HADB サーバとのコネクション確立後になります。

コネクション確立前は，Java 仮想マシンのデフォルトの文字セットで文字コード変換します。

7.7　トラブルシュート

この章では，トラブルシュート機能であるJDBCインタフェースメソッドトレースとExceptionトレースログについて説明します。

7.7.1　JDBCインタフェースメソッドトレース

トラブルシュート情報として，JDBCインタフェースのメソッド呼び出し時に，JDBCインタフェースメソッドトレースを取得できます。

(1)　環境設定

(a)　DriverManagerクラスによる接続の場合

環境設定手順を次に示します。

手順

1. DriverManagerクラスのsetLogWriterメソッドを実行して，有効なログライターを指定します。
2. DriverManagerクラスのgetConnectionメソッドを実行してHADBサーバに接続します。その際，getConnectionメソッドの引数urlまたはinfoに，JDBCインタフェースメソッドトレースを取得する指定をします（methodtraceおよびtracenumを指定します）。

getConnectionメソッドの引数urlの指定については，「7.3.1　DriverManagerクラスのgetConnectionメソッドでHADBサーバに接続する方法」の「(2)　getConnectionメソッドによるHADBサーバへの接続」の「(a)　引数urlの指定内容（接続用のURLの指定）」を参照してください。

getConnectionメソッドの引数infoの指定については，「7.3.1　DriverManagerクラスのgetConnectionメソッドでHADBサーバに接続する方法」の「(2)　getConnectionメソッドによるHADBサーバへの接続」の「(d)　引数infoの指定内容（ユーザプロパティの指定）」を参照してください。

(b)　DataSourceクラスによる接続の場合

環境設定手順を次に示します。

手順

1. DataSourceまたはConnectionPoolDataSourceインタフェースのsetLogWriterメソッドを実行して，有効なログライターを指定します。
2. 接続情報設定および取得インタフェースのsetInterfaceMethodTraceおよびsetTraceNumberメソッドを実行して，JDBCインタフェースメソッドトレースを取得する指定をします。

DataSource インタフェースのsetLogWriter メソッドについては「10.2.7　setLogWriter(PrintWriter out)」を，ConnectionPoolDataSource インタフェースのsetLogWriter メソッドについては「10.3.7 setLogWriter(PrintWriter out)」を参照してください。

setInterfaceMethodTrace メソッドについては「10.5.14　setInterfaceMethodTrace(boolean flag)」を，setTraceNumber メソッドについては「10.5.18　setTraceNumber(int num)」を参照してください。

(2)　JDBC インタフェースメソッドトレースの取得規則

JDBC インタフェースメソッドトレースの取得規則を次に示します。

- JDBC インタフェースのメソッドの呼び出し時，およびメソッドからの戻り時に，トレース情報が取得されます。ただし，データベース接続前に実行できるメソッドついては，トレース情報が取得されません。トレース情報が取得されないメソッドを次に示します。

  Driver インタフェース

  - acceptsURL(String url)
  - getMajorVersion()
  - getMinorVersion()
  - getPropertyInfo(String url, Properties info)
  - jdbcCompliant()

  DataSource インタフェースおよび ConnectionPoolDataSource インタフェース

  - getLoginTimeout()
  - getLogWriter()
  - setLoginTimeout(int seconds)
  - setLogWriter(PrintWriter out)

  Wrapper インタフェース

  - isWrapperFor(Class<?> iface)
  - unwrap(Class<T> iface)
- トレース情報はエントリ数分保持され，次に示すタイミングで指定したログライターに出力されます。
  - Connection.close メソッドの呼び出し時（正常終了時）
  - SQLException の投入時（エラー発生時）
  - BatchUpdateException の投入時（エラー発生時）
  - SQLClientInfoException の投入時（エラー発生時）
  - UnsupportedOperationException の投入時（エラー発生時）
- トレース情報の数がエントリ数を超えた場合，保持されていたトレース情報は古い順に破棄され，新しいトレース情報が保持されます。

- JDBC インタフェースメソッドトレースは，Entry およびReturn でそれぞれ 1 エントリのトレース領域を使用します。

(3) 出力例

JDBC インタフェースメソッドトレースの出力例を次に示します。図中の番号は，説明の項番と対応しています。

出力例

```
[1]                                      [2]                     [3]
[Hitachi Advanced Data Binder JDBC Driver][JDBC Interface Entry ][AdbStatement.executeQuery]
                                                                 [4]
[Hitachi Advanced Data Binder JDBC Driver]                       sql=select * from pp
[Hitachi Advanced Data Binder JDBC Driver][JDBC Interface Return][AdbStatement.executeQuery]
                                                                 [5]
[Hitachi Advanced Data Binder JDBC Driver]                       Return=com.hitachi.hadb.jdbc.Adb...
[Hitachi Advanced Data Binder JDBC Driver][JDBC Interface Entry ][AdbResultSet.getMetaData]
[Hitachi Advanced Data Binder JDBC Driver][JDBC Interface Return][AdbResultSet.getMetaData]
[Hitachi Advanced Data Binder JDBC Driver]                       Return=com.hitachi.hadb.jdbc.Adb...
```

説明

1. [Hitachi Advanced Data Binder JDBC Driver]
   JDBC ドライバの名称
2. [JDBC Interface Entry], [JDBC Interface Return]
   [JDBC Interface Entry]：JDBC メソッドの呼び出し
   [JDBC Interface Return]：JDBC メソッドからの戻り
3. [*XXXXX*.*YYYYY*]
   XXXXX クラスの *YYYYY* メソッド
4. select * from pp
   JDBC メソッドの引数（パスワードを示す引数については，"password=*"のように，"*"1 個を出力）
5. com.hitachi.hadb.jdbc.Adb...
   JDBC メソッドの戻り値

7.7.2 Exception トレースログ

トラブルシュート情報として，Exception トレースログを取得できます。Exception トレースログには，JDBC ドライバ内で例外による障害が発生した際の障害要因が出力されます。

出力内容を次に示します。

- 例外発生時の情報（エラーメッセージなど）
- 例外が発生するまでの，JDBC の API メソッドの実行記録

この機能を使用すると，AP から呼び出される JDBC の API メソッドの情報が JDBC ドライバのメモリ上に蓄積され，SQLException，BatchUpdateException，SQLClientInfoException，または UnSupportedOperationException の発生を契機として，例外を投入する前に，メモリ上に蓄積された情報がファイルに出力されます。

(1) 取得するメソッド，および取得するための設定

(a) Exception トレースログの取得対象メソッド

Exception トレースログの取得対象は，Java Platform Standard Edition 6 の API 仕様にあるパッケージjava.sql，javax.sql に記述されているメソッドの呼び出しと戻りです。

次の条件を満たすメソッドが取得されます。

- 「表 7-16　Exception トレースログの取得対象であるメソッドとトレース取得レベル」に記載されていて，かつ取得に必要なトレース取得レベルを指定している

なお，ResultSet オブジェクトのget*XXX* メソッド，PreparedStatement オブジェクトのset*XXX* メソッド，Connection オブジェクトのisClosed メソッドなど，オブジェクト内の情報を参照して返すだけのメソッドや，オブジェクト内に情報を格納するだけのメソッドは取得対象になりません。

Exception トレースログの取得対象であるメソッドと，そのメソッドのトレース取得レベルを次の表に示します。

表 7-16　Exception トレースログの取得対象であるメソッドとトレース取得レベル

クラス	メソッド	トレース取得レベル				
		1	2	3	4	5※1
Connection	void close()	○	○	○	○	○
	void commit()	×	○	○	○	○
	Statement createStatement()※2	○	○	○	○	○
	Statement createStatement(int resultSetType, int resultSetConcurrency)※3	○	○	○	○	○
	DatabaseMetaData getMetaData()	×	○	○	○	○
	boolean isValid(int timeout)	×	○	○	○	○
	PreparedStatement prepareStatement(String sql)※2	○	○	○	○	○
	PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)※3	○	○	○	○	○
	void rollback()※2	×	○	○	○	○
	void setAutoCommit(boolean autoCommit)	×	○	○	○	○

クラス	メソッド	トレース取得レベル				
		1	2	3	4	5※1
DatabaseMetaData	boolean autoCommitFailureClosesAllResultSets()	×	○	○	○	○
	ResultSet getAttributes(String catalog, String schemaPattern, String typeNamePattern, String attributeNamePattern)	×	○	○	○	○
	ResultSet getBestRowIdentifier(String catalog, String schema, String table, int scope, boolean nullable)	×	○	○	○	○
	ResultSet getCatalogs()	×	○	○	○	○
	ResultSet getClientInfoProperties()	×	○	○	○	○
	ResultSet getColumnPrivileges(String catalog, String schema, String table, String columnNamePattern)	×	○	○	○	○
	ResultSet getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)	×	○	○	○	○
	Connection getConnection()	×	○	○	○	○
	ResultSet getCrossReference(String parentCatalog, String parentSchema, String parentTable, String foreignCatalog, String foreignSchema, String foreignTable)	×	○	○	○	○
	ResultSet getExportedKeys(String catalog, String schema, String table)	×	○	○	○	○
	ResultSet getFunctions(String catalog, String schemaPattern, String functionNamePattern)	×	○	○	○	○
	ResultSet getFunctionColumns(String catalog, String schemaPattern, String functionNamePattern, String columnNamePattern)	×	○	○	○	○
	ResultSet getImportedKeys(String catalog, String schema, String table)	×	○	○	○	○
	ResultSet getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate)	×	○	○	○	○
	ResultSet getPrimaryKeys(String catalog, String schema, String table)	×	○	○	○	○
	ResultSet getProcedureColumns(String catalog, String schemaPattern, String	×	○	○	○	○

クラス	メソッド	トレース取得レベル				
		1	2	3	4	5※1
	procedureNamePattern, String columnNamePattern)					
	ResultSet getProcedures(String catalog, String schemaPattern, String procedureNamePattern)	×	○	○	○	○
	ResultSet getPseudoColumns(String catalog,String schemaPattern,String tableNamePattern,String columnNamePattern)	×	○	○	○	○
	RowIdLifetime getRowIdLifetime()	×	○	○	○	○
	ResultSet getSchemas()	×	○	○	○	○
	ResultSet getSuperTables(String catalog, String schemaPattern, String tableNamePattern)	×	○	○	○	○
	ResultSet getSuperTypes(String catalog, String schemaPattern, String typeNamePattern)	×	○	○	○	○
	ResultSet getTablePrivileges(String catalog, String schemaPattern, String tableNamePattern)	×	○	○	○	○
	ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)	×	○	○	○	○
	ResultSet getTableTypes()	×	○	○	○	○
	ResultSet getTypeInfo()	×	○	○	○	○
	ResultSet getUDTs(String catalog, String schemaPattern, String typeNamePattern, int[] types)	×	○	○	○	○
	ResultSet getVersionColumns(String catalog, String schema, String table)	×	○	○	○	○
Driver	Connection connect(String url, Properties info)	○	○	○	○	○
PreparedStatement	boolean execute()※2	×	○	○	○	○
	ResultSet executeQuery()※2	×	○	○	○	○
	int executeUpdate()※2	×	○	○	○	○
	long executeLargeUpdate()※2	×	○	○	○	○
	ResultSetMetaData getMetaData()	×	○	○	○	○
	boolean execute(String sql)※3, ※4	×	○	○	○	○

クラス	メソッド	トレース取得レベル				
		1	2	3	4	5※1
	int[] executeBatch()※4	×	○	○	○	○
	long[] executeLargeBatch()※4	×	○	○	○	○
	ResultSet executeQuery(String sql)※3, ※4	○	○	○	○	○
	int executeUpdate(String sql)※3, ※4	○	○	○	○	○
	long executeLargeUpdate(String sql)※3, ※4	○	○	○	○	○
	ParameterMetaData getParameterMetaData()	×	○	○	○	○
ResultSet	boolean absolute(int row)	×	○	○	○	○
	void afterLast()	×	○	○	○	○
	void beforeFirst()	×	○	○	○	○
	void close()	×	○	○	○	○
	boolean first()	×	○	○	○	○
	ResultSetMetaData getMetaData()	×	○	○	○	○
	int getHoldability()	×	○	○	○	○
	Statement getStatement()	×	○	○	○	○
	boolean isClosed()	×	○	○	○	○
	boolean last()	×	○	○	○	○
	boolean next()	×	○	○	○	○
	boolean relative(int rows)	×	○	○	○	○
	boolean isAfterLast()	×	○	○	○	○
	boolean isBeforeFirst()	×	○	○	○	○
	boolean isLast()	×	○	○	○	○
Statement	void cancel()	×	○	○	○	○
	void close()	○	○	○	○	○
	boolean execute(String sql)	○	○	○	○	○
	int[] executeBatch()	×	○	○	○	○
	long[] executeLargeBatch()	×	○	○	○	○
	ResultSet executeQuery(String sql)	○	○	○	○	○
	int executeUpdate(String sql)	○	○	○	○	○
	long executeLargeUpdate(String sql)	○	○	○	○	○
	ResultSet getResultSet()	×	○	○	○	○

クラス	メソッド	トレース取得レベル				
		1	2	3	4	5※1
DataSource	getConnection()※2	○	○	○	○	○
	getConnection(String username, String password)※3	○	○	○	○	○
ConnectionPoolDataSource	getPooledConnection()※2	○	○	○	○	○
	getPooledConnection(String username, String password)※3	○	○	○	○	○
PooledConnection	close()	○	○	○	○	○
	getConnection()	○	○	○	○	○

（凡例）

○：Exception トレースログを取得します。

×：Exception トレースログを取得しません。

注※1

トレース取得レベルが5 の場合，内部呼び出しも含めて Exception トレースログを取得します。

注※2

メソッド名として，「*メソッド名*(1)」と出力されます。

注※3

メソッド名として，「*メソッド名*(2)」と出力されます。

注※4

Statement クラスのメソッドをオーバライドしたメソッドです。

(b) Exception トレースログを取得するための設定（プロパティの設定）

Exception トレースログのファイル出力先，ファイルへの出力数，メモリ内取得情報数，およびトレース取得レベルをシステムプロパティ，ユーザプロパティ，または接続用の URL のプロパティで設定します。

プロパティに設定する項目を次の表に示します。

表 7-17 プロパティに設定する項目

項目	プロパティ	内容	デフォルト値※
ファイル出力先	adb_jdbc_exc_trc_out_path	Exception トレースログを出力するディレクトリを，絶対パスで指定します。Exception トレースログは，指定したディレクトリの直下に出力されます。	カレントディレクトリ
ファイルへの出力数	adb_jdbc_info_max	1 ファイルへ出力する情報数の上限を指定します。指定できる範囲は 1～50 です。	5

項目	プロパティ	内容	デフォルト値※
		実際に 1 ファイルへ出力する情報数の上限は，「**ファイルへの出力数**」×「**メモリ内取得情報数**」個となります。 ファイルへの出力数は，「(2)　Exception トレースログの出力形式」に示す形式 2～形式 4 をそれぞれ 1 個と数えます。 なお，ファイルには，メモリに蓄積した順番に出力されます。 また，上限値を超えてファイルに出力する場合は，2 ファイルでラップアラウンドします。ファイル名は次のとおりです。 • adbjdbcexception01.trc • adbjdbcexception02.trc ただし，「(2)　Exception トレースログの出力形式」に示す形式 1 と形式 2 の間で出力先のファイルが切り替わることはありません。	
メモリ内取得情報数	adb_jdbc_cache_info_max	メモリ内で蓄積する情報数の上限を指定します。指定できる範囲は 500～10,000 です。 なお，メモリ内取得情報は，「表 7-16　Exception トレースログの取得対象であるメソッドとトレース取得レベル」に示すメソッド 1 つを 1 個と数えます。 また，上限値を超えて情報を蓄積しようとすると，古い情報から順に，新しい情報に上書きされます。	1,000
トレース取得レベル	adb_jdbc_trc_out_lv	トレース取得レベルを指定します。指定できる範囲は 0～5 です。 5 を指定すると，内部呼び出しを含めたすべてのトレース取得対象メソッドを取得します。 0 を指定すると，Exception トレースログを取得しません。	1

注※

次のような場合に取得する Exception トレースログでは，プロパティに値が指定されなかったものと見なされます。その場合はデフォルト値が仮定されます。

- プロパティに不正な値が指定され，データベース接続時にSQLException が投入された場合
- Java 仮想マシンが，セキュリティマネジャによって JDBC ドライバに対するプロパティの受け渡しを拒否した場合
- Java 仮想マシンの，最初の接続が確立する前

(2)　Exception トレースログの出力形式

Exception トレースログには，次の 4 種類の形式があります。

形式 1：ヘッダ部分

```
[AA....AA] Hitachi Advanced Data Binder JDBC Driver BB-CC
```

形式 2：メソッドの実行履歴（メソッドの実行開始）

```
AAAAAAAAAAAAAAAAAAAAAAA BB....BB:[C][DD....DD]
                        ConnectionID(EE....EE) : SID(FF....FF)
                        GG....GG
```

形式 3：メソッドの実行履歴（メソッドの正常終了）

```
AAAAAAAAAAAAAAAAAAAAAAA BB....BB:[C][DD....DD]
                        ConnectionID(EE....EE) : SID(FF....FF)
                        HH....HH
```

形式 4：発生した出力契機

```
AAAAAAAAAAAAAAAAAAAAAAA BB....BB:Exception:
II....II
```

形式 2 および形式 3 は，時系列順に，メソッドを実行した分だけ繰り返して出力されます。

(a) 形式 1 の変数の説明

AA....AA：

出力情報の通番です。

通番は 1 回の出力（出力エラーによる失敗も含む）ごとに 1 増加します。2,147,483,647 を超えると，0 に戻ります。

BB：

JDBC ドライバのバージョンです。

CC：

JDBC ドライバのリビジョンです。

(b) 形式 2～形式 4 の変数の説明

AAAAAAAAAAAAAAAAAAAAAAA：

Exception トレースログの取得日時を，次の形式で出力します。それぞれの変数には，すべて 0～9 の値が入ります。

```
YYYY/MM/DD hh:mm:ss.sss
```

YYYY：年（西暦）

MM：月

DD：日

hh：時（24 時間形式）

mm：分

ss.sss：秒（小数点以下 3 桁を含みます）

BB....BB：

該当するスレッドのスレッド識別情報を，次の形式で出力します。

```
Thread[aa....aa]@bb....bb
```

aa....aa：

スレッド名，優先順位，スレッドグループ名を含む，スレッド情報です。形式は Java 仮想マシンによって決まります。

bb....bb：

オブジェクトのハッシュコードです。形式は Java 仮想マシンによって決まります。

C：

メソッドの呼び出し識別情報です。

E：

メソッドが開始したときの履歴であることを示します。

R：

メソッドが正常終了したときの履歴であることを示します。

DD....DD：

オブジェクト識別子およびメソッド名を，次の形式で出力します。

```
aa....aa.bb....bb
```

aa....aa：

オブジェクト識別子です（最大 32 文字）。形式は Java 仮想マシンによって決まります。

bb....bb：

メソッド名です。

EE....EE：

接続 ID を出力します（最大 4 文字）。

FF....FF：

セクション ID を出力します（最大 4 文字）。

GG....GG：

メソッドの引数を次の形式で出力します。引数がないメソッドでは出力しません。

```
aa....aa=bb....bb
aa....aa=bb....bb
          :
aa....aa=bb....bb
```

aa....aa：

引数の名称です。

bb....bb：

引数の内容です（最大 256 文字）。参照型値の場合，形式はオブジェクトによって決まります。

なお，次のメソッドの引数password については，*bb....bb* に"*"1 個を出力します。

- DataSource クラスのgetConnection(String username, String password)
- ConnectionPoolDataSource クラスのgetPooledConnection(String username, String password)

また，Driver クラスのconnect(String url,Properties info)の，引数info 中の次のプロパティについては，値を"*"1 個に置き換えて出力します。

- password

HH....HH：

メソッドの戻り値を次の形式で出力します。戻り値がないメソッドでは出力しません。戻り値が参照型値の場合，形式は Java 仮想マシンによって決まります。

```
Return=aa....aa
```

aa....aa：

メソッドの戻り値です。

II....II：

トラブルシュート情報を，次の形式で出力します。

```
ExceptionClass: aa....aa
ConnectionInformation: bb....bb
Message: cc....cc
ErrorCode: dd....dd
SQLState: eeeee
UpdateCounts: ff....ff, ..<省略>.. ,ff....ff
gg....gg
```

aa....aa：

投入した例外オブジェクトの実行クラス名です。

bb....bb：

例外オブジェクトの接続情報を次の形式で出力します。出力しない場合は，"*"1 個に置き換えて出力します。

```
yy....yy (zz....zz), ..<省略>.., yy....yy (zz....zz)
```

yy....yy：各接続情報の名称です。次に示す情報が出力されます。

- host（HADB サーバのホスト名）
- port（HADB サーバのポート番号）
- user（認可識別子）
- sqlwaittime（HADB クライアントの応答最大待ち時間（単位：秒））

zz....zz：上記の接続情報の内容です。なお，user のパスワード部分は出力しません。

cc....cc：

例外オブジェクトが持つメッセージです。

複数のメッセージを持つ場合は，SQLCODE に対応するメッセージのあとに，改行で区切って出力します。この場合，例外オブジェクトのgetMessage メソッドで返却するメッセージなども，同様に改行で区切られた文字列になります。

dd....dd：

SQLCODE のエラーコードです（最大 11 文字)。

投入した例外オブジェクトの実行クラスが次のクラスまたはサブクラスの場合に出力します。

- SQLException

eeeee：

例外オブジェクトのSQLSTATE です。

ff....ff：

この例外が発生するまでに正常に実行されたバッチ更新の，各更新文の更新行数を出力します（最大 11 文字)。

例外オブジェクトの実行クラスがBatchUpdateException の場合に出力します。

更新行数が取得できない場合，"*"を出力します。

gg....gg：

例外投入メソッドを基点としたスタックトレースを出力します。形式は Java 仮想マシンによって決まります。

(3) 出力例と解析方法

(a) 出力例

```
[1] Hitachi Advanced Data Binder JDBC Driver VV-RR※
2011/07/06 23:07:09.129 Thread[main,5,main]@1259414:[E][AdbConnection@82c01f.createStatemen
t(1)]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:09.160 Thread[main,5,main]@1259414:[R][AdbConnection@82c01f.createStatemen
t(1)]
                        ConnectionID(1) : SID(0)
                        Return=com.hitachi.hadb.jdbc.AdbStatement@1e4cbc4
2011/07/06 23:07:09.160 Thread[main,5,main]@1259414:[E][AdbStatement@1e4cbc4.execute]
                        ConnectionID(1) : SID(0)
                        sql=DELETE FROM SEINO_TABLE
2011/07/06 23:07:14.285 Thread[main,5,main]@1259414:[E][AdbConnection@82c01f.commit]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:14.301 Thread[main,5,main]@1259414:[R][AdbConnection@82c01f.commit]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:14.301 Thread[main,5,main]@1259414:[R][AdbStatement@1e4cbc4.execute]
                        ConnectionID(1) : SID(1)
                        Return=false
2011/07/06 23:07:14.301 Thread[main,5,main]@1259414:[E][AdbConnection@82c01f.prepareStatemen
t(1)]
                        ConnectionID(1) : SID(0)
                        sql=INSERT INTO SEINO_TABLE VALUES(?, ?)
2011/07/06 23:07:14.348 Thread[main,5,main]@1259414:[R][AdbConnection@82c01f.prepareStatemen
t(1)]
```

```
                        ConnectionID(1) : SID(0)
                        Return=com.hitachi.hadb.jdbc.AdbPreparedStatement@15d56d5
2011/07/06 23:07:26.567 Thread[main,5,main]@1259414:[E][AdbConnection@82c01f.commit]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:26.567 Thread[main,5,main]@1259414:[R][AdbConnection@82c01f.commit]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:26.567 Thread[main,5,main]@1259414:[E][AdbStatement@1e4cbc4.executeQuery]
                        ConnectionID(1) : SID(0)
                        sql=SELECT * FROM SEINO_TABLE
2011/07/06 23:07:26.676 Thread[main,5,main]@1259414:[R][AdbStatement@1e4cbc4.executeQuery]
                        ConnectionID(1) : SID(1)
                        Return=com.hitachi.hadb.jdbc.AdbResultSet@3eca90
2011/07/06 23:07:28.332 Thread[main,5,main]@1259414:[E][AdbResultSet@3eca90.close]
                        ConnectionID(1) : SID(1)
2011/07/06 23:07:28.332 Thread[main,5,main]@1259414:[E][AdbConnection@82c01f.commit]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:28.332 Thread[main,5,main]@1259414:[R][AdbConnection@82c01f.commit]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:28.332 Thread[main,5,main]@1259414:[R][AdbResultSet@3eca90.close]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:28.332 Thread[Thread-0,5,main]@30090737:[E][AdbConnection@82c01f.prepareSta
tement(1)]
                        ConnectionID(1) : SID(0)
                        sql=SELECT * FROM SEINO_TABLE
2011/07/06 23:07:28.332 Thread[Thread-0,5,main]@30090737:[R][AdbConnection@82c01f.prepareSta
tement(1)]
                        ConnectionID(1) : SID(0)
                        Return=com.hitachi.hadb.jdbc.AdbPreparedStatement@2808b3
2011/07/06 23:07:28.348 Thread[Thread-1,5,main]@5462872:[E][AdbConnection@82c01f.prepareStat
ement(1)]
                        ConnectionID(1) : SID(0)
                        sql=DELETE FROM SEINO_TABLE WHERE I1=?
2011/07/06 23:07:28.358 Thread[Thread-1,5,main]@5462872:[E][AdbConnection@82c01f.commit]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:29.672 Thread[Thread-1,5,main]@5462872:[R][AdbConnection@82c01f.commit]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:30.098 Thread[Thread-1,5,main]@5462872:[R][AdbConnection@82c01f.prepareStat
ement(1)]
                        ConnectionID(1) : SID(0)
                        Return=com.hitachi.hadb.jdbc.AdbPreparedStatement@922804
2011/07/06 23:07:30.332 Thread[Thread-2,5,main]@25253977:[E][AdbConnection@82c01f.rollback(1
)]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:42.098 Thread[Thread-2,5,main]@25253977:[R][AdbConnection@82c01f.rollback(1
)]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:42.098 Thread[Thread-2,5,main]@25253977:[E][AdbConnection@82c01f.close]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:42.098 Thread[Thread-2,5,main]@25253977:[R][AdbConnection@82c01f.close]
                        ConnectionID(1) : SID(0)
2011/07/06 23:07:42.535 Thread[Thread-1,5,main]@5462872:Exception:
ExceptionClass: SQLNonTransientException
ConnectionInformation: *
Message: KFAA71206-E Processing cannot continue because the connection is already closed. [A
dbPreparedStatement.setInt]
ErrorCode: -1071206
SQLState : R2416
java.sql.SQLNonTransientException: KFAA71206-E Processing cannot continue because the connec
```

```
tion is already closed. [AdbPreparedStatement.setInt]
at com.hitachi.hadb.jdbc.JdbMakeException.generateSQLSubException(JdbMakeException.java:271)
at com.hitachi.hadb.jdbc.JdbMakeException.generateSQLException(JdbMakeException.java:37)
at com.hitachi.hadb.jdbc.AdbStatement.generateClosedSQLException(AdbStatement.java:3005)
at com.hitachi.hadb.jdbc.AdbPreparedStatement.setInt(AdbPreparedStatement.java:1170)
at Exception1.run(ExceptionTraceSample.java:57)
[2] Hitachi Advanced Data Binder JDBC Driver VV-RR※
2011/07/06 23:07:25.723 Thread[Thread-3,5,main]@13249998:[E][AdbConnection@119cca4.prepareSt
atement(1)]
                         ConnectionID(1) : SID(0)
                         sql=SELECT * FROM SEINO_TABLE
2011/07/06 23:07:25.770 Thread[Thread-4,5,main]@25839584:[E][AdbConnection@119cca4.rollback(
1)]
                         ConnectionID(1) : SID(0)
2011/07/06 23:07:25.770 Thread[Thread-4,5,main]@25839584:[R][AdbConnection@119cca4.rollback(
1)]
                         ConnectionID(1) : SID(0)
2011/07/06 23:07:25.770 Thread[Thread-5,5,main]@24431647:[E][AdbConnection@119cca4.prepareSt
atement(1)]
                         ConnectionID(1) : SID(0)
                         sql=SELECT ** FROM SEINO_TABLE
2011/07/06 23:07:25.863 Thread[Thread-5,5,main]@24431647:Exception:
ExceptionClass: SQLSyntaxErrorException
ConnectionInformation: user(ADBUSER01), sqlwaittime(0), host(dragon2), port(20249)
Message: KFAA30105-E Token "*"(non-reserved word), which is after token "*", is invalid.[Adb
Statement.prepare]
ErrorCode: -105
SQLState : 42602
java.sql.SQLSyntaxErrorException: KFAA30105-E Token "*"(non-reserved word), which is after t
oken "*", is invalid.[AdbStatement.prepare]
at com.hitachi.hadb.cli.CtlStatement.prepare(CtlStatement.java:451)
at com.hitachi.hadb.jdbc.JdbSection.prepare(JdbSection.java:1497)
at com.hitachi.hadb.jdbc.AdbStatement.prepare(AdbStatement.java:2834)
at com.hitachi.hadb.jdbc.AdbPreparedStatement.<init>(AdbPreparedStatement.java:109)
at com.hitachi.hadb.jdbc.AdbConnection.prepareStatement(AdbConnection.java:1041)
at Exception1.run(ExceptionTraceSample.java:64)
```

注※

VV-RR には，JDBC ドライバのバージョンが出力されます。

(b)　解析方法

Exception トレースログの解析方法について説明します。Exception トレースログはテキストエディタなどで参照できます。

ここでは，「(a)　出力例」の Exception トレースログを解析する例を示します。

解析例

1. 調査対象の例外を含む通番の情報を抜き出します。
2. 情報をThread 識別情報で分類し，スレッドごとに分割します。
3. 取得時刻によって，情報を時系列に並べます。

次の表に示すようになります。

表 7-18　Exception トレースログを時系列に並べた例

日時	スレッド 1	スレッド 2	スレッド 3	スレッド 4
	Thread[main,5,main]@1259414	Thread[Thread-0,5,main]@30090737	Thread[Thread-1,5,main]@5462872	Thread[Thread-2,5,main]@25253977
2011/07/06 23:07:09.129	AdbConnection@82c01f.createStatement(1)			
2011/07/06 23:07:09.160	AdbStatement@1e4cbc4.execute			
2011/07/06 23:07:14.285	AdbConnection@82c01f.commit			
2011/07/06 23:07:14.301	AdbConnection@82c01f.prepareStatement(1)			
2011/07/06 23:07:26.567	AdbConnection@82c01f.commit			
2011/07/06 23:07:26.567	AdbConnection@82c01f.commit			
2011/07/06 23:07:26.567	AdbStatement@1e4cbc4.executeQuery			
2011/07/06 23:07:28.332	AdbResultSet@3eca90.close	AdbConnection@82c01f.prepareStatement(1)		
2011/07/06 23:07:28.332	AdbConnection@82c01f.commit			
2011/07/06 23:07:28.348			AdbConnection@82c01f.prepareStatement(1)	
2011/07/06 23:07:28.358			AdbConnection@82c01f.commit	
2011/07/06 23:07:30.332				AdbConnection@82c01f.rollback(1)
2011/07/06 23:07:42.098				AdbConnection@82c01f.close
2011/07/06 23:07:42.535			SQLException発生 ”KFAA71206-E Processing cannot continue because the connection is already closed.”	

4. Exception のエラーの内容を確認します。

2011/07/06 23:07:42.535 のスレッド 3 でSQLException が発生し，Statement オブジェクトまたは Connection オブジェクトがすでにクローズされていることがわかります。

5. 時系列でオブジェクトの操作を確認します。

次のスレッドのConnection オブジェクトのオブジェクト ID が同じであることから，4 つのスレッドが同一のコネクションで処理されていることがわかります。

- 2011/07/06 23:07:09.129 のスレッド 1
- 2011/07/06 23:07:28.332 のスレッド 2
- 2011/07/06 23:07:28.348 のスレッド 3
- 2011/07/06 23:07:30.332 のスレッド 4

6. エラーの原因である個所を探します。

4 つのスレッドが同一のコネクションであることはわかったので，Statement.close メソッドまたは Connection.close メソッドを実行している個所を探すと，スレッド 4 が 2011/07/06 23:07:42.098 でConnection.close メソッドを実行していることがわかります。このことから，2011/07/06 23:07:42.535 のスレッド 3 で発生したSQLException の原因は，スレッド 4 が 2011/07/06 23:07:42.098 でConnection.close メソッドを実行したためであるとわかります。

(4) 必要となるメモリ所要量およびファイルサイズ

(a) メモリ所要量

Exception トレースログを取得するためのメモリ所要量は，次に示す計算式で求められます。

計算式

↑360×*n*÷1024↑　　（単位：キロバイト）

変数の説明

n：メモリ内取得情報数（システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティのadb_jdbc_cache_info_max の指定値）

(b) 必要となるファイルサイズ

Exception トレースログを取得するためのファイルサイズの概算は，次に示す計算式で求められます。

計算式

↑180×*n*×*m*÷1024↑＋2　　（単位：キロバイト）

変数の説明

n：メモリ内取得情報数（システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティのadb_jdbc_cache_info_max の指定値）

m：ファイル出力情報数（システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティのadb_jdbc_info_max の指定値）

(5) 注意事項

(a) Java 仮想マシン起動後の最初の出力

Java 仮想マシン起動後，最初にファイルに Exception トレースログを出力する場合は，更新日時が古い方のファイルに出力されます。同じ場合は，adbjdbcexception01.trc に出力されます。

(b) ファイル出力先の指定

複数のプロセスから Exception トレースログを取得する場合，同じファイル出力先を指定していると，同一ファイルに異なるプロセスのトレースが出力されます。プロセスごとにトレースを取得する場合は，プロセスごとに異なるファイル出力先を指定してください。

JDBC ドライバは，Java 仮想マシンの機能を通じて OS が提供するファイルシステム上にログファイルを作成します。そのため，次の点については，使用する Java 仮想マシンおよびファイルシステムに依存します。

- 絶対パス名の接頭辞
- パスの区切り文字
- 出力先ファイル（絶対パス）の最大文字数
- 1 ファイル当たりのサイズ

(c) エラー発生時の処理

ファイルの作成や出力に失敗しても，Exception トレースログには情報が出力されません。また，エラーメッセージが AP に返されたり，ファイル出力をリトライしたりすることもありません。

(d) 文字コード

Exception トレースログは，使用する Java 仮想マシンのデフォルトの変換文字セットで出力されます。

7.8　エスケープ句で指定できるスカラ関数

エスケープ句で指定できるスカラ関数を次の表に示します。

表 7-19　エスケープ句で指定できるスカラ関数

スカラ関数	スカラ関数の標準形式
数学関数	ABS(number)
	ACOS(float)
	ASIN(float)
	ATAN(float)
	ATAN2(float1, float2)
	CEILING(number)
	COS(float)
	DEGREES(number)
	EXP(float)
	FLOOR(number)
	LOG(float)
	LOG10(float)
	MOD(integer1, integer2)
	PI()
	POWER(number, power)
	RADIANS(number)
	RAND([number, number])
	ROUND(number, places)
	SIGN(number)
	SIN(float)
	SQRT(float)
	TAN(float)
	TRUNCATE(number[, places])
文字列関数	ASCII(string)
	CHAR(code)
	CONCAT(string1,string2)
	LCASE(string)

スカラ関数	スカラ関数の標準形式
	LEFT(string,count)
	LENGTH(string)
	LTRIM(string)
	OCTET_LENGTH(string)
	REPLACE(string1,string2[,string3])
	RIGHT(string,count)
	RTRIM(string)
	SUBSTRING(string, start[, length])
	UCASE(string)
時刻と日付の関数	CURDATE()
	CURRENT_DATE()
	CURRENT_TIME()
	CURRENT_TIMESTAMP()
	CURTIME()
	DAYOFWEEK(date)
	DAYOFYEAR(date)
	EXTRACT(extract-field FROM extract-source)
	NOW()
	TIMESTAMPADD(interval,integer_exp,timestamp_exp)
	TIMESTAMPDIFF(interval,timestamp_exp1,timestamp_exp2)
システム関数	USER()
データ型変換関数	CONVERT(value, SQLtype)

8 JDBC 1.2 API

この章では，JDBC 1.2 API の各インタフェースとメソッドについて説明します。

8.1　Array インタフェース

ここでは，Array インタフェースで提供されているメソッドについて説明します。

8.1.1　Array インタフェースのメソッド一覧

(1)　Array インタフェースの主な機能

Array インタフェースでは，配列型の列へのアクセス手段として主に次の機能が提供されています。

- SQL Array 値（配列型の列の配列要素の値）の取得
- SQL Array 値（配列型の列の配列要素の値）を格納した結果セットの取得

(2)　HADB でサポートしている Array インタフェースのメソッド

HADB でサポートしているArray インタフェースのメソッドの一覧を次の表に示します。

表 8-1　Array インタフェースのメソッドの一覧

項番	Array インタフェースのメソッド	機能
1	getArray()	配列型の列の全配列要素を Object 配列として取得します。
2	getArray(long index, int count)	配列型の列の一部の配列要素を Object 配列として取得します。
3	getBaseType()	Array オブジェクトの配列要素の要素データ型を JDBC の SQL データ型で取得します。
4	getBaseTypeName()	Array オブジェクトの配列要素の要素データ型を HADB のデータ型名で取得します。
5	getResultSet()	配列型の列の全配列要素を格納した結果セットを取得します。
6	getResultSet(long index, int count)	配列型の列の一部の配列要素を格納した結果セットを取得します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されます。

(3)　必要なパッケージ名称とクラス名称

Array インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc

- クラス名称：AdbArray

8.1.2 getArray()

(1) 機能

配列型の列の全配列要素を Object 配列として取得します。

(2) 形式

```
public synchronized Object getArray() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Array オブジェクト内に保持している配列型のすべての配列要素が，Object 配列として返却されます。

返却される配列要素の HADB でのデータ型を次の表に示します。

表 8-2 返却される配列要素の HADB でのデータ型

配列要素の HADB のデータ型（要素データ型）	返却される配列
INTEGER	java.lang.Long[]
SMALLINT	java.lang.Integer[]
DECIMAL またはNUMERIC	java.math.BigDecimal[]
DOUBLE PRECISION またはFLOAT	java.lang.Double[]
CHAR	java.lang.String[]
VARCHAR	java.lang.String[]
DATE	java.sql.Date[]
TIME	java.sql.Time[]
TIMESTAMP	java.sql.Timestamp[]
BINARY	java.io.InputStream[]
VARBINARY	java.io.InputStream[]

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このArray オブジェクトを生成したResultSet オブジェクトがクローズされている場合
  このArray オブジェクトを生成したResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- 上記のStatement オブジェクトを作成したConnection オブジェクトがクローズされている場合
- トランザクションの決着によって，ResultSet オブジェクトが無効になった場合
- 列の値がArray 値として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.1.3 getArray(long index, int count)

(1) 機能

配列型の列の一部の配列要素を Object 配列として取得します。

(2) 形式

```
public synchronized Object getArray(long index,int count) throws SQLException
```

(3) 引数

long index：

最初に取り出す配列要素の順序番号を指定します。配列要素の先頭から取り出す場合は1 を指定します。

int count：

連続して取得する配列要素の数を指定します。

(4) 戻り値

Array オブジェクト内に保持している配列要素のうち，引数index に指定した順序番号から，最大で引数count に指定した数分の連続する配列要素が，Object 配列として返却されます。

返却される配列要素の HADB でのデータ型については，「表 8-2　返却される配列要素の HADB でのデータ型」を参照してください。

引数index と引数count に指定した値と返却される配列の関係を次の表に示します。

表 8-3　引数に指定した値と返却される配列

index の指定値	count の指定値	index の指定値＋ count の指定値	返却される配列
1≦*index の指定値*≦*配列要素数*	0≦*count の指定値*	(*index の指定値*＋*count の指定値*）－1≦*配列要素数*	長さが *count の指定値*分の配列
		(*index の指定値*＋*count の指定値*）－1 ＞*配列要素数*	*配列要素数*－（*index の指定値*－1）分の長さを持つ配列
	count の指定値＜0	－	SQLExceptionを投入
index の指定値＞*配列要素数*	－	－	SQLExceptionを投入
index の指定値＜1	－	－	SQLExceptionを投入

（凡例）

－：該当しません。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このArray オブジェクトを生成したResultSet オブジェクトがクローズされている場合
  このArray オブジェクトを生成したResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- 上記のStatement オブジェクトを作成したConnection オブジェクトがクローズされている場合
- トランザクションの決着によって，ResultSet オブジェクトが無効になった場合
- 列の値がArray 値として取得できない場合
- 引数index に指定した値が1 より小さい場合，または配列要素数より大きい場合
- 引数count に指定した値が0 より小さい場合
- JDBC ドライバ内でエラーが発生した場合

8.1.4　getBaseType()

(1)　機能

Array オブジェクトの配列要素の要素データ型を JDBC の SQL データ型で取得します。

(2)　形式

```
public synchronized int getBaseType() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Array オブジェクトの配列要素の要素データ型が，JDBC の SQL データ型（java.sql.Types クラス）の定数で返却されます。

配列要素の要素データ型と返却値の対応については，「7.6.1 データ型のマッピング」の「(1) HADB のデータ型と JDBC の SQL データ型の対応」を参照してください。

(5) 発生する例外

なし。

8.1.5 getBaseTypeName()

(1) 機能

Array オブジェクトの配列要素の要素データ型を HADB のデータ型名で取得します。

(2) 形式

```
public synchronized String getBaseTypeName() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Array オブジェクトの配列要素の要素データ型が，HADB のデータ型名（String オブジェクト）で返却されます。

getBaseTypeName メソッドの戻り値と要素データ型（HADB のデータ型）の対応を次の表に示します。

表 8-4 getBaseTypeName メソッドの戻り値と要素データ型（HADB のデータ型）の対応

要素データ型（HADB のデータ型）	戻り値（返却される文字列）
INTEGER	"INTEGER"
SMALLINT	"SMALLINT"※1

要素データ型（HADB のデータ型）	戻り値（返却される文字列）
DECIMAL またはNUMERIC	”DECIMAL”
DOUBLE PRECISION またはFLOAT	”DOUBLE PRECISION”
CHAR	”CHAR”
VARCHAR	”VARCHAR”
DATE	”DATE”
TIME	”TIME”
TIMESTAMP	”TIMESTAMP”
BINARY	”BINARY”
VARBINARY	”VARBINARY”
ROW	”ROW”
BOOLEAN	”BOOLEAN”※2

注※1

DatabaseMetaData から生成したResultSet にだけ存在する列で，データ型がshort で規定されている列の場合はSMALLINT が返却されます。

注※2

DatabaseMetaData から生成したResultSet にだけ存在する列で，データ型がBOOLEAN で規定されている列の場合はBOOLEAN が返却されます。

(5) 発生する例外

なし。

8.1.6 getResultSet()

(1) 機能

配列型の列の全配列要素を格納した結果セットを取得します。

(2) 形式

```
public synchronized ResultSet getResultSet() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

配列データを格納したResultSet オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このArray オブジェクトを生成したResultSet オブジェクトがクローズされている場合
  このArray オブジェクトを生成したResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- 上記のStatement オブジェクトを作成したConnection オブジェクトがクローズされている場合
- トランザクションの決着によって，ResultSet オブジェクトが無効になった場合
- JDBC ドライバ内でエラーが発生した場合

8.1.7 getResultSet(long index, int count)

(1) 機能

配列型の列の一部の配列要素を格納した結果セットを取得します。

(2) 形式

```
public synchronized ResultSet getResultSet(long index,int count) throws SQLException
```

(3) 引数

long index：

最初に取り出す配列要素の順序番号を指定します。配列要素の先頭から取り出す場合は1 を指定します。

int count：

連続して取得する配列要素の数を指定します。

(4) 戻り値

引数index に指定した順序番号から，最大で引数count に指定した数分の連続する配列要素が，ResultSet オブジェクトとして返却されます。

引数index と引数count に指定した値と返却されるResultSet オブジェクトの関係を次の表に示します。

表 8-5　引数に指定した値と返却される ResultSet オブジェクト

index の指定値	count の指定値	index の指定値＋ count の指定値	返却される ResultSet オブジェクト
1≦*index の指定値*≦*配列要素数*	0≦*count の指定値*	（*index の指定値*＋*count の指定値*）－1≦*配列要素数*	長さが*count* 分の結果セット
		（*index の指定値*＋*count の指定値*）－1 ＞*配列要素数*	*配列要素数*－（*index の指定値*－1）分の長さを持つ結果セット
	count の指定値＜0	－	SQLExceptionを投入
index の指定値＞*配列要素数*	－	－	SQLExceptionを投入
index の指定値＜1	－	－	SQLExceptionを投入

（凡例）

－：該当しません。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このArray オブジェクトを生成したResultSet オブジェクトがクローズされている場合
  このArray オブジェクトを生成したResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- 上記のStatement オブジェクトを作成したConnection オブジェクトがクローズされている場合
- トランザクションの決着によって，ResultSet オブジェクトが無効になった場合
- 列の値がArray 値として取得できない場合
- 引数index に指定した値が1 より小さい場合，または配列要素数より大きい場合
- 引数count に指定した値が0 より小さい場合
- JDBC ドライバ内でエラーが発生した場合

8.2 Driver インタフェース

ここでは，Driver インタフェースで提供されているメソッドについて説明します。

8.2.1 Driver インタフェースのメソッド一覧

(1) Driver インタフェースの主な機能

Driver インタフェースでは，主に次の機能が提供されています。

- データベースへの接続
- 接続用の URL の妥当性チェック
- DriverManager.getConnection メソッドで指定する接続プロパティの情報取得
- JDBC ドライバのバージョンの返却

(2) HADB でサポートしている Driver インタフェースのメソッド

HADB でサポートしているDriver インタフェースのメソッドの一覧を次の表に示します。

表 8-6 Driver インタフェースのメソッドの一覧

項番	Driver インタフェースのメソッド	機能
1	acceptsURL(String url)	接続用の URL の接続情報で HADB サーバに接続できるかどうかを確認します。
2	connect(String url, Properties info)	接続情報に従って HADB サーバに接続します。
3	getMajorVersion()	JDBC ドライバのメジャーバージョンを取得します。
4	getMinorVersion()	JDBC ドライバのマイナーバージョンを取得します。
5	getPropertyInfo(String url, Properties info)	JDBC ドライバの有効なプロパティについての情報を取得します。
6	jdbcCompliant()	JDBC ドライバが JDBC Compliant™ であるかどうかを返します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されます。

(3) 必要なパッケージ名称とクラス名称

Driver インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：HADBDriver

8.2.2　acceptsURL(String url)

(1)　機能

URL に指定されているデータベースに接続できるかどうかを確認します。

(2)　形式

```
public boolean acceptsURL(String url) throws SQLException
```

(3)　引数

String url：

接続用の URL を指定します。

接続用の URL の指定形式については，「7.3.1　DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法」の「(2)　getConnection メソッドによる HADB サーバへの接続」の「(a)　引数 url の指定内容（接続用の URL の指定）」を参照してください。

(4)　戻り値

URL に指定したデータベースに接続できる場合はtrue が，接続できない場合はfalse が返却されます。

(5)　発生する例外

なし。

8.2.3　connect(String url, Properties info)

(1)　機能

接続情報に従って HADB サーバに接続します。

なお，connect メソッドを実行するにはCONNECT 権限が必要です。

(2)　形式

```
public Connection connect(String url, Properties info) throws SQLException
```

(3) 引数

`String url`：

接続用の URL を指定します。

接続用の URL の指定形式については，「7.3.1 DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法」の「(2) getConnection メソッドによる HADB サーバへの接続」の「(a) 引数 url の指定内容（接続用の URL の指定）」を参照してください。

`Properties info`：

接続引数としてのプロパティ名称と値のペアのリストを指定します。指定形式については，「7.3.1 DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法」の「(2) getConnection メソッドによる HADB サーバへの接続」の「(d) 引数 info の指定内容（ユーザプロパティの指定）」を参照してください。

(4) 戻り値

`Connection` オブジェクトが返却されます。

指定された URL が不正な場合（URL に指定したデータベースに接続できない場合），`null` が返却されます。

(5) 発生する例外

次に示す場合に`SQLException` が投入されます。

- データベースのアクセスエラーが発生した場合
- 指定した接続情報に誤りがある場合

(6) 留意事項

接続情報は，各プロパティやメソッドなど複数の個所で設定できます。適用される接続情報の優先順位については，「7.3.3 接続情報の優先順位」の「(1) HADB サーバへの接続時に必要となる接続情報」を参照してください。

8.2.4 getMajorVersion()

(1) 機能

JDBC ドライバのメジャーバージョンを取得します。

(2) 形式

```
public synchronized int getMajorVersion()
```

(3) 引数

なし。

(4) 戻り値

JDBC ドライバのメジャーバージョン番号が返却されます。

(5) 発生する例外

なし。

8.2.5 getMinorVersion()

(1) 機能

JDBC ドライバのマイナーバージョンを取得します。

(2) 形式

```
public synchronized int getMinorVersion()
```

(3) 引数

なし。

(4) 戻り値

JDBC ドライバのマイナーバージョン番号が返却されます。

(5) 発生する例外

なし。

8.2.6 getPropertyInfo(String url, Properties info)

(1) 機能

JDBC ドライバの有効なプロパティについての情報を取得します。

(2) 形式

```
public synchronized DriverPropertyInfo[] getPropertyInfo(String url, Properties info) throws
 SQLException
```

(3) 引数

String url：

接続用の URL を指定します。

接続用の URL の指定形式については，「7.3.1 DriverManager クラスの getConnection メソッドで HADB サーバに接続する方法」の「(2) getConnection メソッドによる HADB サーバへの接続」の「(a) 引数 url の指定内容（接続用の URL の指定)」を参照してください。

Properties info：

接続引数としてのプロパティ名称，および値のペアのリストを指定します。

(4) 戻り値

有効なプロパティを記述するDriverPropertyInfo オブジェクトの配列が返却されます。プロパティが必要ない場合は，この配列は空になることもあります。

DriverPropertyInfo の各フィールドの設定値を次の表に示します。

表 8-7 DriverPropertyInfo の各フィールドの設定値

プロパティ名	DriverPropertyInfo フィールド				
	name	value	description	required	choices
adb_clt_rpc_srv_host	プロパティ名と同じ	null	"Host Name"	true	null
adb_clt_rpc_srv_port		null	"Port Number"	true	null
adb_clt_rpc_con_wait_time		"300"	"Connect Wait Time"	false	null
adb_clt_rpc_sql_wait_time		"0"	"Sql Wait Time"	false	null
adb_clt_ap_name		"********"	"Application Name"	false	null
adb_clt_group_name		null	"Client Group Name"	false	null
adb_clt_fetch_size		"1024"	"Fetch Size"	false	null
adb_clt_sql_text_out		"N"	"Text Out"	false	{"Y","N"}
adb_clt_trn_iso_lv		"READ_COMMITTED"	"Isolation Level"	false	{"READ_COMMITTED","REPEATABLE_READ"}

プロパティ名	DriverPropertyInfo フィールド				
	name	value	description	required	choices
adb_clt_sql_order_mode		"BYTE"	"Order Mode"	false	{"BYTE","ISO"}
adb_clt_trn_access_mode		"READ_WRITE"	"Access Mode"	false	{"READ_WRITE","READ_ONLY"}
adb_clt_sql_parallel_exec		"N"	"Sql Parallel Execution"	false	{"Y","N"}
adb_clt_passwd_pubkey_path		null	"Password Public Key File Path"	false	null
adb_dbbuff_wrktbl_clt_blk_num		"256"	"Work Table Block Number"	false	null
adb_sql_prep_delrsvd_use_srvdef		"Y"	"Delete Reserved Word Using Server Definition"	false	{"Y","N"}
adb_sql_prep_dec_div_rs_prior		"INTEGRAL_PART"	"Decimal Division Result Prior"	false	{"INTEGRAL_PART","FRACTIONAL_PART"}
adb_sql_exe_max_rthd_num		"4"	"Sql Execute Max Real Thread Number"	false	null
adb_sql_exe_hashgrp_area_size		"4800"	"Hash Group Area Size"	false	null
adb_sql_exe_hashtbl_area_size		"2000"	"Hash Table Area Size"	false	null
adb_sql_exe_hashflt_area_size		"200"	"Hash Filter Area Size"	false	null
adb_jdbc_exc_trc_out_path		null	"Exception Trace Out Path"	false	null
adb_jdbc_info_max		"5"	"Exception Trace Information Max Number"	false	null
adb_jdbc_cache_info_max		"1000"	"Exception Trace Cache Information Max Number"	false	null
adb_jdbc_trc_out_lv		"1"	"Exception Trace Out Level"	false	null

プロパティ名	DriverPropertyInfo フィールド				
	name	value	description	required	choices
encodelang		null	"Encode Lang"	false	null
methodtrace		"OFF"	"JDBC Interface Trace"	false	{"ON","OFF"}
tracenum		"500"	"Trace Entry Number"	false	null
sqlwarningkeep		"TRUE"	"Keeping up the Warning Objects"	false	{"TRUE","FALSE"}
user		null	"UserID"	true	null
password		null	"Password"	true	null

url，info に指定された情報を解析し，HADB サーバに接続するための情報が返却されます。

なお，acceptsURL メソッドの戻り値がfalse の場合，戻り値にはnull が返却されます。

(5) 発生する例外

なし。

8.2.7 jdbcCompliant()

(1) 機能

JDBC ドライバが JDBC Compliant^{TM} であるかどうかを返します。

(2) 形式

```
public synchronized boolean jdbcCompliant()
```

(3) 引数

なし。

(4) 戻り値

JDBC ドライバが JDBC Compliant の場合はtrue が，そうでない場合はfalse が返却されます。

(5)　発生する例外

なし。

8.2.8　エスケープ句

SQL 文中で｛｝で囲まれた部分をエスケープ句と呼びます。エスケープ句は 1 つのキーワードと複数のパラメタで構成されます。キーワードの大文字と小文字は区別しません。エスケープ句の一覧を次の表に示します。

表 8-8　エスケープ句の一覧

エスケープ句の種別	キーワード
日付	d
時刻	t
時刻印	ts
LIKE エスケープ文字	escape
外結合	oj
スカラ関数	fn

エスケープ句で指定できるスカラ関数については，「7.8　エスケープ句で指定できるスカラ関数」を参照してください。

エスケープ構文の解析

エスケープ構文の解析を有効にするかどうかは，Statement クラスのsetEscapeProcessing メソッドで指定します。指定がない場合は，有効となります。エスケープ構文の解析が有効な場合，SQL 文内にエスケープ句がないか解析します。SQL 文内にエスケープ句があった場合は，HADB が実行できる SQL 文に変換します。

エスケープ構文を使用しない場合は，Statement オブジェクトのsetEscapeProcessing メソッドでエスケープ構文の解析を無効にすると，構文解析のオーバヘッドを削減できます。

8.3 Connection インタフェース

ここでは，Connection インタフェースで提供されているメソッドについて説明します。

8.3.1 Connection インタフェースのメソッド一覧

(1) Connection インタフェースの主な機能

Connection インタフェースでは，主に次の機能が提供されています。

- Statement クラスおよびPreparedStatement クラスのオブジェクト生成
- トランザクションの決着（コミットまたはロールバック）
- 自動コミットモードの設定

(2) HADB でサポートしている Connection インタフェースのメソッド

HADB でサポートしているConnection インタフェースのメソッドの一覧を次の表に示します。

表 8-9 Connection インタフェースのメソッドの一覧

項番	Connection インタフェースのメソッド	機能
1	clearWarnings()	Connection オブジェクトに通知されたすべての警告をクリアします。
2	close()	HADB サーバとの接続を切断します。
3	commit()	直前のコミットまたはロールバック以降に行われた変更をすべて有効とします。
4	createStatement()	SQL 文を HADB サーバに送るためのStatement オブジェクトを生成します。
5	createStatement(int resultSetType, int resultSetConcurrency)	
6	createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)	
7	getAutoCommit()	このConnection オブジェクトの現在の自動コミットモードを取得します。
8	getCatalog()	このConnection オブジェクトの現在のカタログ名を取得します。
9	getHADBConnectionID()	このConnection オブジェクトに割り当てられているコネクション ID を取得します。
10	getHADBConnectionSerialNum()	このConnection オブジェクトに割り当てられているコネクション通番を取得します。

項番	Connection インタフェースのメソッド	機能
11	getHADBOrderMode()	このConnection オブジェクトの，ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を取得します。
12	getHADBSQLHashFltSize()	このConnection オブジェクトに設定されているハッシュフィルタ領域サイズを取得します。
13	getHADBSQLHashTblSize()	このConnection オブジェクトに設定されているハッシュテーブル領域サイズを取得します。
14	getHADBSQLMaxRthdNum()	このConnection オブジェクトに設定されている最大 SQL 処理リアルスレッド数を取得します。
15	getHADBTransactionID()	実行中のトランザクションのトランザクション ID を取得します。
16	getHoldability()	このConnection オブジェクトを使用して生成されるResultSet オブジェクトの現在の保持機能を取得します。
17	getMetaData()	DatabaseMetaData オブジェクトを生成します。
18	getSchema()	このConnection オブジェクトの現在のスキーマ名を取得します。
19	getTransactionIsolation()	このConnection オブジェクトの現在のトランザクション隔離性水準を取得します。
20	getTypeMap()	このConnection オブジェクトに関連したMap オブジェクトを取得します。
21	getWarnings()	このConnection オブジェクトに関する呼び出しによって報告される警告を，SQLWarning オブジェクトに取得します。
22	isClosed()	Connection オブジェクトがクローズされているかどうかを返します。
23	isReadOnly()	このConnection オブジェクトが読み込み専用モードかどうかを取得します。
24	isValid(int timeout)	現在の接続状態を取得します。
25	nativeSQL(String sql)	指定した SQL 文内のエスケープ句を，HADB が実行できる形式に変換します。
26	prepareStatement(String sql)	パラメタ付きの SQL 文を HADB サーバに送るための PreparedStatement オブジェクトを生成します。
27	prepareStatement(String sql, int resultSetType, int resultSetConcurrency)	
28	prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	
29	rollback()	トランザクションによって実行された変更をすべて元に戻し，Connection オブジェクトが保持するロックをすべて解除します。
30	setAutoCommit(boolean autoCommit)	この接続の自動コミットモードを設定します。

項番	Connection インタフェースのメソッド	機能
31	setCatalog(String catalog)	渡されたカタログ名を設定し，Connection オブジェクトのデータベースの作業用サブスペースを選択します。
32	setHADBAuditInfo(int pos,String userinfo)	ユーザ任意の接続情報（ユーザ付加情報）を設定します。
33	setHADBOrderMode(int mode)	このConnection オブジェクトの，ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を設定します。
34	setHADBSQLHashFltSize(int areaSize)	このConnection オブジェクトに対してハッシュフィルタ領域サイズを設定します。
35	setHADBSQLHashTblSize(int areaSize)	このConnection オブジェクトに対してハッシュテーブル領域サイズを設定します。
36	setHADBSQLMaxRthdNum(int rthdNum)	このConnection オブジェクトに対して最大 SQL 処理リアルスレッド数を設定します。
37	setHoldability(int holdability)	このConnection オブジェクトを使用して生成されたResultSet オブジェクトの保持機能を設定します。
38	setReadOnly(boolean readOnly)	このConnection オブジェクトを読み取り専用モードに設定します。トランザクションアクセスモードを設定します。
39	setSchema(String schema)	アクセスするスキーマ名を設定します。
40	setTransactionIsolation(int level)	このConnection オブジェクトのトランザクション隔離性水準を設定します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されることがあります。

(3) 必要なパッケージ名称とクラス名称

Connection インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbConnection

8.3.2 clearWarnings()

(1) 機能

Connection オブジェクトに通知されたすべての警告をクリアします。

このメソッドを実行した場合，このConnection オブジェクトに対する新しい警告が報告されるまでは，getWarnings メソッドの戻り値はnull になります。

(2) 形式

```
public synchronized void clearWarnings() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.3 close()

(1) 機能

HADB サーバとの接続を切断します。

通常接続時は，HADB サーバとの接続を切断するとともに，該当するオブジェクトを無効にし，不要なリソースを解放します。

(2) 形式

```
public synchronized void close() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

なし。

(6) 留意事項

- プーリング環境下でこのメソッドを実行した場合，HADB サーバとの物理的な接続は切断されません。この場合，PooledConnection オブジェクトのclose メソッドで物理的な接続の切断をします。
- プーリング環境下でこのメソッドを実行した結果，致命的なエラーが発生してコネクションプーリングが使用できなくなった場合でも，ConnectionEventListener.connectionErrorOccurred は発生しません。
- すでにクローズされたConnection オブジェクトでこのメソッドを実行しても，このメソッドは何もしません。
- 行の取り出し処理でエラーが発生していた場合，決着していなかったトランザクションはコミットされないでロールバックされます。なお，HADB サーバからは正常に切断されて，リソースも解放されます。

8.3.4 commit()

(1) 機能

直前のコミットまたはロールバック以降に行われた変更をすべて有効とします。

自動コミットモードを有効にしている状態でこのメソッドを呼び出しても，例外を投入しないでコミット処理を行います。

(2) 形式

```
public synchronized void commit() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- データベースのアクセスエラーが発生した場合
- 行の取り出し処理で発生したエラーを，コミット処理の延長で検知した場合

(6) 留意事項

- 行の取り出し処理のエラーを検知した例外が発生した場合，トランザクションはコミットされないでロールバックされます。
- コミット処理が失敗した場合，HADB サーバが異常終了します。

8.3.5 createStatement()

(1) 機能

SQL 文を HADB サーバに送るためのStatement オブジェクトを生成します。

(2) 形式

```
public synchronized Statement createStatement() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Statement オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- Statement オブジェクトの生成でエラーが発生した場合
- データベースのアクセスエラーが発生した場合

(6) 留意事項

このメソッドで生成したStatement オブジェクトから生成されるResultSet オブジェクトの保持機能は，常にResultSet.HOLD_CURSORS_OVER_COMMIT になります。

8.3.6 createStatement(int resultSetType, int resultSetConcurrency)

(1) 機能

SQL 文を HADB サーバに送るためのStatement オブジェクトを生成します。

(2) 形式

```
public synchronized Statement createStatement(int resultSetType, int resultSetConcurrency) t
hrows SQLException
```

(3) 引数

int resultSetType：

結果セットタイプを指定します。

int resultSetConcurrency：

並行処理モードを指定します。

(4) 戻り値

Statement オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- Statement オブジェクトの生成でエラーが発生した場合
- 結果セットタイプにResultSet 定数以外を指定した場合
- 並行処理モードにResultSet 定数以外を指定した場合
- データベースのアクセスエラーが発生した場合

(6) 留意事項

- このメソッドで生成したStatement オブジェクトから生成されるResultSet オブジェクトの保持機能は，常にResultSet.HOLD_CURSORS_OVER_COMMIT になります。
- 結果セットタイプにResultSet.TYPE_SCROLL_SENSITIVE を指定した場合，ResultSet.TYPE_SCROLL_INSENSITIVE に切り替え，SQLWarning を設定します。
- 並行処理モードはResultSet.CONCUR_READ_ONLY だけサポートしています。ResultSet.CONCUR_UPDATABLE を指定した場合，ResultSet.CONCUR_READ_ONLY に切り替えてSQLWarning を設定します。

8.3.7 createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)

(1) 機能

SQL 文を HADB サーバに送るためのStatement オブジェクトを生成します。

(2) 形式

```
public synchronized Statement createStatement(int resultSetType, int resultSetConcurrency, i
nt resultSetHoldability) throws SQLException
```

(3) 引数

int resultSetType：

結果セットタイプを指定します。

int resultSetConcurrency：

並行処理モードを指定します。

int resultSetHoldability：

ResultSet オブジェクトの保持機能を指定します。

(4) 戻り値

Statement オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- Statement オブジェクトの生成でエラーが発生した場合
- 結果セットタイプにResultSet 定数以外を指定した場合
- 並行処理モードにResultSet 定数以外を指定した場合
- ResultSet オブジェクトの保持機能にResultSet 定数以外を指定した場合
- データベースのアクセスエラーが発生した場合

(6) 留意事項

- 結果セットタイプにResultSet.TYPE_SCROLL_SENSITIVE を指定した場合，ResultSet.TYPE_SCROLL_INSENSITIVE に切り替えて，SQLWarning を設定します。

- 並行処理モードはResultSet.CONCUR_READ_ONLY だけをサポートします。ResultSet.CONCUR_UPDATABLE を指定した場合，ResultSet.CONCUR_READ_ONLY に切り替えて，SQLWarning を設定します。
- ResultSet オブジェクトの保持機能は，ResultSet.HOLD_CURSORS_OVER_COMMIT だけをサポートしています。ResultSet.CLOSE_CURSORS_AT_COMMIT を指定した場合は，ResultSet.HOLD_CURSORS_OVER_COMMIT に切り替えて，SQLWarning を設定します。

8.3.8 getAutoCommit()

(1) 機能

このConnection オブジェクトの現在の自動コミットモードを取得します。

(2) 形式

```
public synchronized boolean getAutoCommit() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Connection オブジェクトでの現在の自動コミットモードの状態が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.9 getCatalog()

(1) 機能

このConnection オブジェクトの現在のカタログ名を取得します。

(2) 形式

```
public synchronized String getCatalog() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にnull が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.10 getHADBConnectionID()

(1) 機能

このConnection オブジェクトに割り当てられているコネクション ID を取得します。

(2) 形式

```
public int getHADBConnectionID() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このConnection オブジェクトに割り当てられているコネクション ID が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6) 留意事項

このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。

8.3.11　getHADBConnectionSerialNum()

(1)　機能

このConnection オブジェクトに割り当てられているコネクション通番を取得します。

(2)　形式

```
public int getHADBConnectionSerialNum() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

このConnection オブジェクトに割り当てられているコネクション通番が返却されます。

(5)　発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6)　留意事項

このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2　Wrapper インタフェース」を参照してください。

8.3.12　getHADBOrderMode()

(1)　機能

このConnection オブジェクトの，現在のORDER BY 句を指定したSELECT 文の文字データの並び替え順序を取得します。

(2)　形式

```
public int getHADBOrderMode() throws SQLException
```

(3)　引数

なし。

(4) 戻り値

現在のORDER BY 句を指定したSELECT 文の文字データの並び替え順序が返却されます。次に示すどちらかの値が返却されます。

- AdbConnection.HADB_SQL_ORDER_MODE_BYTE
- AdbConnection.HADB_SQL_ORDER_MODE_ISO

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6) 留意事項

このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。

8.3.13 getHADBSQLHashFltSize()

(1) 機能

このConnection オブジェクトに設定されているハッシュフィルタ領域サイズを取得します。

このメソッドの使い方（実行する SQL 文ごとにハッシュフィルタ領域サイズを変更する方法）については，「8.3.37 setHADBSQLMaxRthdNum(int rthdNum)」の「(7) 使用例」を参照してください。

(2) 形式

```
public int getHADBSQLHashFltSize() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このConnection オブジェクトに設定されているハッシュフィルタ領域サイズ（単位：メガバイト）が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6) 留意事項

このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。

8.3.14 getHADBSQLHashTblSize()

(1) 機能

このConnection オブジェクトに設定されているハッシュテーブル領域サイズを取得します。

このメソッドの使い方（実行する SQL 文ごとにハッシュテーブル領域サイズを変更する方法）については，「8.3.37 setHADBSQLMaxRthdNum(int rthdNum)」の「(7) 使用例」を参照してください。

(2) 形式

```
public int getHADBSQLHashTblSize() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このConnection オブジェクトに設定されているハッシュテーブル領域サイズ（単位：メガバイト）が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6) 留意事項

このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。

8.3.15 getHADBSQLMaxRthdNum()

(1) 機能

このConnection オブジェクトに設定されている最大 SQL 処理リアルスレッド数を取得します。

このメソッドの使い方（実行する SQL 文ごとに最大 SQL 処理リアルスレッド数を変更する方法）については，「8.3.37　setHADBSQLMaxRthdNum(int rthdNum)」の「(7)　使用例」を参照してください。

(2)　形式

```
public int getHADBSQLMaxRthdNum() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

このConnection オブジェクトに設定されている最大 SQL 処理リアルスレッド数が返却されます。

(5)　発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6)　留意事項

このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2　Wrapper インタフェース」を参照してください。

8.3.16　getHADBTransactionID()

(1)　機能

実行中のトランザクションのトランザクション ID を取得します。

(2)　形式

```
public long getHADBTransactionID() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

getHADBTransactionID メソッドを発行したときに実行中のトランザクションのトランザクション ID が返却されます。

なお，SQL 文の実行前にgetHADBTransactionID メソッドを発行した場合は，0 が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6) 留意事項

このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。

8.3.17 getHoldability()

(1) 機能

このConnection オブジェクトを使用して生成されるResultSet オブジェクトの現在の保持機能を取得します。

(2) 形式

```
public synchronized int getHoldability() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にResultSet.HOLD_CURSORS_OVER_COMMIT が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.18 getMetaData()

(1) 機能

DatabaseMetaData オブジェクトを生成します。

(2) 形式

```
public synchronized DatabaseMetaData getMetaData() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

DatabaseMetaData オブジェクトが返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.19 getSchema()

(1) 機能

このConnection オブジェクトの現在のスキーマ名を取得します。

(2) 形式

```
public synchronized String getSchema() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にnull が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.20 getTransactionIsolation()

(1) 機能

このConnection オブジェクトの現在のトランザクション隔離性水準を取得します。

(2) 形式

```
public synchronized int getTransactionIsolation() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

現在のトランザクション隔離性水準が返却されます。次に示すどちらかの値が返却されます。

- Connection.TRANSACTION_READ_COMMITTED
  トランザクション隔離性水準にREAD COMMITTED が適用されている場合に返却されます。
- Connection.TRANSACTION_REPEATABLE_READ
  トランザクション隔離性水準にREPEATABLE READ が適用されている場合に返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.21 getTypeMap()

(1) 機能

このConnection オブジェクトに関連したMap オブジェクトを取得します。

(2) 形式

```
public synchronized java.util.Map getTypeMap() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に情報を格納していない空のjava.util.Map オブジェクトが返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.22 getWarnings()

(1) 機能

このConnection オブジェクトに関する呼び出しによって報告される警告を，SQLWarning オブジェクトに取得します。

該当するConnection オブジェクトが保持するSQLWarning オブジェクトを取得します。取得したSQLWarning オブジェクトのgetNextWarning メソッドを実行すれば，２つ目以降の警告を取得できます。

(2) 形式

```
public synchronized SQLWarning getWarnings() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

最初のSQLWarning オブジェクトが返却されます。SQLWarning オブジェクトがない場合はnull が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.23 isClosed()

(1) 機能

Connection オブジェクトがクローズされているかどうかを返します。

HADB サーバへの接続は，close メソッドが呼び出されるか，または特定の致命的なエラーが発生した場合に切断されます。close メソッドを実行したあとに，このメソッドが実行された場合にだけ，true を返すことを保証します。

なお，HADB サーバへの接続が，有効または無効かをこのメソッドで判定することはできません。

(2) 形式

```
public synchronized boolean isClosed() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Connection オブジェクトがクローズされている場合はtrue が，クローズされていない場合はfalse が返却されます。

(5) 発生する例外

なし。

8.3.24 isReadOnly()

(1) 機能

このConnection オブジェクトが読み込み専用モードかどうかを取得します。

(2) 形式

```
public synchronized boolean isReadOnly() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このConnection オブジェクトが読み込み専用モードの場合はtrue が，読み込み専用モードではない状態の場合はfalse が返却されます。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.25 isValid(int timeout)

(1) 機能

現在の接続状態を取得します。

(2) 形式

```
public synchronized boolean isValid(int timeout) throws SQLException
```

(3) 引数

int timeout：

待ち時間（秒）を 0～65,535 の範囲で指定します。

0 を指定した場合は無制限となります。

65,536 以上を指定した場合には，65,535 が仮定されます。

(4) 戻り値

接続中であることを確認できた場合は，true が返却されます。Connection オブジェクトがクローズされている場合，または引数で指定された待ち時間を過ぎた場合はタイムアウトとなり，false が返却されます。

(5) 発生する例外

引数に-1 以下が指定された場合は，SQLException が投入されます。

8.3.26 nativeSQL(String sql)

(1) 機能

指定した SQL 文内のエスケープ句を，HADB が実行できる形式に変換します。

(2) 形式

```
public synchronized String nativeSQL(String sql) throws SQLException
```

(3) 引数

String sql：

SQL 文を指定します。

(4) 戻り値

HADB が実行できる SQL 文が返却されます。

sql にnull を指定した場合，null が返却されます。sql に空文字を指定した場合，空文字が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- 指定された SQL のエスケープ句の形式が次のように不正である場合
  - "{"およびキーワードはあるが，"}"がない
- 指定された SQL 文の長さが，16,000,000 文字を超えている場合

(6) エスケープ句の構文規則

指定された SQL 文内のエスケープ句を，HADB が実行できる形式に変換して返します。エスケープ句の構文規則を次に示します。

```
エスケープ句 ::= 日付・時刻・時刻印のエスケープシーケンス
               | LIKE述語のエスケープ文字のエスケープシーケンス
               | 外結合のエスケープシーケンス
               | スカラ関数のエスケープシーケンス
日付・時刻・時刻印のエスケープシーケンス ::= 日付のエスケープシーケンス
                                            | 時刻のエスケープシーケンス
                                            | 時刻印のエスケープシーケンス
日付のエスケープシーケンス ::=
        エスケープ開始子 d 日付データの既定の文字列表現※1 エスケープ終了子
時刻のエスケープシーケンス ::=
        エスケープ開始子 t 時刻データの既定の文字列表現※2 エスケープ終了子
時刻印のエスケープシーケンス ::=
        エスケープ開始子 ts 時刻印データの既定の文字列表現※3 エスケープ終了子
LIKE述語のエスケープ文字のエスケープシーケンス ::=
        エスケープ開始子 escape エスケープ文字 エスケープ終了子
外結合のエスケープシーケンス ::= エスケープ開始子 oj 結合表 エスケープ終了子
スカラ関数のエスケープシーケンス ::= エスケープ開始子 fn スカラ関数 エスケープ終了子
スカラ関数 ::= 標準形式のスカラ関数※4
エスケープ開始子 ::= '{'
エスケープ終了子 ::= '}'
```

注※1

'*YYYY-MM-DD*'で表される文字列表現のことです。

注※2

'*hh*:*mm*:*ss*[.*f*...]'で表される文字列表現のことです。

注※3

'*YYYY-MM-DD hh*:*mm*:*ss*[.*f*...]'で表される文字列表現のことです。

注※4

標準形式のスカラ関数については，「7.8　エスケープ句で指定できるスカラ関数」を参照してください。

なお，下線部には，エスケープ句を指定できません。また，JDBC ドライバでは構文解析をしないで，変換後もそのままとし，HADB サーバの構文解析に任せます。

エスケープシーケンスのキーワードを次に示します。キーワードについては，大文字と小文字を区別しません。

1. 日付のエスケープシーケンス内の"d"
2. 時刻のエスケープシーケンス内の"t"
3. 時刻印のエスケープシーケンス内の"ts"
4. LIKE 述語のエスケープ文字のエスケープシーケンス内の"escape"
5. 外結合のエスケープシーケンス内の"oj"
6. スカラ関数のエスケープシーケンス内の"fn"

エスケープ句の入力規則を次に示します。

- エスケープ句内の区切り文字には，半角空白が指定できます。
- 区切り文字は，エスケープ開始子の後ろ，キーワードの後ろ，およびエスケープ終了子の前に挿入できます。
- 1 つの SQL 文中に，複数のエスケープ句を指定できます。
- JDBC ドライバは指定された SQL 文内のエスケープ句を，HADB が実行できる形式に変換します。変換するのは，{ }で囲まれたエスケープ句内だけです。エスケープ句外は，何も変換しません。

エスケープ句の変換規則を次の表に示します。

表 8-10　エスケープ句の変換規則

対象エスケープ句	変換前	変換後
日付	*エスケープ開始子* d *日付データの既定の文字列表現 エスケープ終了子*	*DATE 日付データの既定の文字列表現*
時刻	*エスケープ開始子* t *時刻データの既定の文字列表現 エスケープ終了子*	*TIME 時刻データの既定の文字列表現*
時刻印	*エスケープ開始子* ts *時刻印データの既定の文字列表現 エスケープ終了子*	*TIMESTAMP 時刻印データの既定の文字列表現*

対象エスケープ句	変換前	変換後
LIKE	*エスケープ開始子* escape *エスケープ文字 エスケープ終了子*	escape *エスケープ文字*
外結合	*エスケープ開始子* oj *結合表 エスケープ終了子*	*結合表*
スカラ関数	*エスケープ開始子* fn *スカラ関数 エスケープ終了子*	*HADB 形式のスカラ関数*※

注※

標準形式のスカラ関数を，HADB 形式に変換します。

標準形式と HADB 形式が異なるスカラ関数の変換内容を次の表に示します。

基本的に，スカラ関数の引数の個数チェックはしません。

表 8-11　標準形式と HADB 形式が異なるスカラ関数の変換内容一覧

スカラ関数	変換前の形式	変換後の形式（HADB 形式）
数学関数	CEILING(number)	CEIL(number)
	LOG(float)	LN(float)
	LOG10(float)	LOG(10,float)
	RAND([number, number])	RANDOM([number, number])
	TRUNCATE(number[, places])	TRUNC(number[, places])
文字列関数	CHAR(code)	CHR(code)
	LCASE(string)	LOWER(string)
	OCTET_LENGTH(string)	LENGTHB(string)
	SUBSTRING(string, start[, length])	SUBSTR(string, start[, length])
	UCASE(string)	UPPER(string)
時刻と日付の関数	CURDATE()	CURRENT_DATE
	CURRENT_DATE()	CURRENT_DATE
	CURRENT_TIME()	CURRENT_TIME
	CURRENT_TIMESTAMP()	CURRENT_TIMESTAMP
	CURTIME()	CURRENT_TIME
	NOW()	CURRENT_TIMESTAMP
システム関数	USER()	CURRENT_USER

8.3.27 prepareStatement(String sql)

(1) 機能

パラメタ付きの SQL 文を HADB サーバに送るためのPreparedStatement オブジェクトを生成します。

(2) 形式

```
public synchronized PreparedStatement prepareStatement(String sql) throws SQLException
```

(3) 引数

String sql：

実行する SQL 文を指定します。

(4) 戻り値

PreparedStatement オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- PreparedStatement オブジェクトの生成でエラーが発生した場合
- データベースのアクセスエラーが発生した場合
- 指定された SQL 文の長さが，16,000,000 文字を超えている場合

(6) 留意事項

このメソッドで生成したPreparedStatement オブジェクトから生成されるResultSet オブジェクトの保持機能は，常にResultSet.HOLD_CURSORS_OVER_COMMIT となります。

8.3.28 prepareStatement(String sql, int resultSetType, int resultSetConcurrency)

(1) 機能

パラメタ付きの SQL 文を HADB サーバに送るためのPreparedStatement オブジェクトを生成します。

(2) 形式

```
public synchronized PreparedStatement prepareStatement(String sql, int resultSetType, int re
sultSetConcurrency) throws SQLException
```

(3) 引数

String sql：

実行する SQL 文を指定します。

int resultSetType：

結果セットタイプを指定します。

int resultSetConcurrency：

並行処理モードを指定します。

(4) 戻り値

PreparedStatement オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- PreparedStatement オブジェクトの生成でエラーが発生した場合
- 結果セットタイプにResultSet 定数以外を指定した場合
- 並行処理モードにResultSet 定数以外を指定した場合
- データベースのアクセスエラーが発生した場合
- 指定された SQL 文の長さが，16,000,000 文字を超えている場合

(6) 留意事項

- 結果セットタイプにResultSet.TYPE_SCROLL_SENSITIVE を指定した場合，ResultSet.TYPE_SCROLL_INSENSITIVE に切り替えて，SQLWarning を設定します。
- 並行処理モードはResultSet.CONCUR_READ_ONLY だけをサポートしています。ResultSet.CONCUR_UPDATABLE を指定した場合，ResultSet.CONCUR_READ_ONLY に切り替えて，SQLWarning を設定します。
- このメソッドで生成したPreparedStatement オブジェクトから生成されるResultSet オブジェクトの保持機能は，常にResultSet.HOLD_CURSORS_OVER_COMMIT となります。

8.3.29 prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)

(1) 機能

パラメタ付きの SQL 文を HADB サーバに送るためのPreparedStatement オブジェクトを生成します。

(2) 形式

```
public synchronized PreparedStatement prepareStatement(String sql, int resultSetType, int re
sultSetConcurrency, int resultSetHoldability) throws SQLException
```

(3) 引数

String sql：

実行する SQL 文を指定します。

int resultSetType：

結果セットタイプを指定します。

int resultSetConcurrency：

並行処理モードを指定します。

int resultSetHoldability：

ResultSet オブジェクトの保持機能を指定します。

(4) 戻り値

PreparedStatement オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- PreparedStatement オブジェクトの生成でエラーが発生した場合
- 結果セットタイプにResultSet 定数以外を指定した場合
- 並行処理モードにResultSet 定数以外を指定した場合
- ResultSet オブジェクトの保持機能にResultSet 定数以外を指定した場合
- データベースのアクセスエラーが発生した場合
- 指定された SQL 文の長さが，16,000,000 文字を超えている場合

(6) 留意事項

- 結果セットタイプにResultSet.TYPE_SCROLL_SENSITIVE を指定した場合，ResultSet.TYPE_SCROLL_INSENSITIVE に切り替えて，SQLWarning を設定します。
- 並行処理モードはResultSet.CONCUR_READ_ONLY だけをサポートします。ResultSet.CONCUR_UPDATABLE を指定した場合，ResultSet.CONCUR_READ_ONLY に切り替えて，SQLWarning を設定します。
- ResultSet オブジェクトの保持機能はResultSet.HOLD_CURSORS_OVER_COMMIT だけをサポートしています。ResultSet.CLOSE_CURSORS_AT_COMMIT を指定した場合は，ResultSet.HOLD_CURSORS_OVER_COMMIT に切り替えて，SQLWarning を設定します。

8.3.30 rollback()

(1) 機能

トランザクションによって実行された変更をすべて元に戻し，Connection オブジェクトが保持するロックをすべて解除します。

自動コミットモードを有効にしている状態で，このメソッドを呼び出しても例外を投入しないで，ロールバック処理を行います。

(2) 形式

```
public synchronized void rollback() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- データベースのアクセスエラーが発生した場合

(6) 留意事項

- ロールバック処理が成功すると，ResultSet オブジェクトが無効になります。

- ロールバック処理が失敗した場合，HADB サーバが異常終了します。

8.3.31　setAutoCommit(boolean autoCommit)

(1)　機能

この接続の自動コミットモードを設定します。

(2)　形式

```
public synchronized void setAutoCommit(boolean autoCommit) throws SQLException
```

(3)　引数

boolean autoCommit：

自動コミットモードを有効にする場合はtrue を，無効にする場合はfalse を指定します。

(4)　戻り値

なし。

(5)　発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6)　留意事項

- 自動コミットモードが有効な場合，SQL 文の処理が終わると自動的にコミットされます。そのため，1 つの SQL 文は 1 つのトランザクションとして扱われます。自動コミットモードが無効な場合は，commit メソッドまたはrollback メソッドの実行によって SQL 文が完了します。デフォルトでは，自動コミットモードは有効です。
- SQL 文の完了時に自動コミットが実行されます。SQL 文がResultSet オブジェクトを返す場合，ResultSet オブジェクトがクローズされたときに，SQL 文が完了します。
- トランザクションの途中でこのメソッドが呼び出されても，そのトランザクションはコミットされません。

8.3.32 setCatalog(String catalog)

(1) 機能

渡されたカタログ名を設定し，Connection オブジェクトのデータベースの作業用サブスペースを選択します。

(2) 形式

```
public synchronized void setCatalog(String catalog) throws SQLException
```

(3) 引数

String catalog：

この引数を指定しても無視されます。

(4) 戻り値

なし。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.33 setHADBAuditInfo(int pos,String userinfo)

(1) 機能

HADB サーバにアクセスするアプリケーションのアカウント情報などのユーザ付加情報を設定します。設定したユーザ付加情報は，解除するまで有効になります。

なお，ここで設定したユーザ付加情報は，次に示すタイミングで監査証跡として出力されます。

- このConnection オブジェクトを使用して生成したStatement オブジェクトを使用した SQL 文の実行時
- このConnection オブジェクトを使用して生成したPreparedStatement オブジェクトを使用した SQL 文の実行時
- このConnection オブジェクトのクローズ時

(2) 形式

```
public synchronized void setHADBAuditInfo(int pos,String userinfo) throws SQLException
```

(3) 引数

int pos：

userinfo で指定したユーザ付加情報を，監査証跡のどの列に出力するかを指定します。次のどれかの値を指定してください。

1：userinfo で指定したユーザ付加情報を，監査証跡のユーザ付加情報 1 に出力する場合に指定します。

2：userinfo で指定したユーザ付加情報を，監査証跡のユーザ付加情報 2 に出力する場合に指定します。

3：userinfo で指定したユーザ付加情報を，監査証跡のユーザ付加情報 3 に出力する場合に指定します。

String userinfo：

ユーザ付加情報を指定します。

ここで指定したユーザ付加情報は，HADB サーバが使用している文字コードに変換されます。文字コード変換後のバイト数が 100 バイト以内になるように，ユーザ付加情報を指定してください。

なお，NULL 文字（0x00）は使用できません。

また，ユーザ付加情報の設定を解除する場合は，null を指定してください。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- pos に 1～3 以外の値を指定した場合
- ユーザ付加情報の文字コード変換に失敗した場合
- ユーザ付加情報の文字コード変換後の長さが 100 バイトを超えた場合
- ユーザ付加情報中に NULL 文字（0x00）を含んでいる場合
- トランザクションがすでに開始している場合

(6) 留意事項

- このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。
- setHADBAuditInfo メソッドで指定したユーザ付加情報は，Connection オブジェクトをいったんプールして再度使用した場合，前回設定した付加情報を再利用しません。setHADBAuditInfo メソッドを実行していない状態と同じになります。

8.3.34　setHADBOrderMode(int mode)

(1)　機能

このConnection オブジェクトの，ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を設定します。

このメソッドで設定する内容は，クライアント定義のadb_clt_sql_order_mode オペランドに対応しています。

(2)　形式

```
public void setHADBOrderMode(int mode) throws SQLException
```

(3)　引数

int mode：

ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を指定します。次のどちらかの値を指定してください。

- AdbConnection.HADB_SQL_ORDER_MODE_BYTE
  文字データをバイトコード順に並び替えます。
- AdbConnection.HADB_SQL_ORDER_MODE_ISO
  文字データをソートコード順（ISO/IEC14651:2011 準拠）に並び替えます。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- トランザクションがすでに開始している場合
- 並び替え順序に次の値以外を指定した場合
  ・AdbConnection.HADB_SQL_ORDER_MODE_BYTE
  ・AdbConnection.HADB_SQL_ORDER_MODE_ISO

(6) 留意事項

- ORDER BY 句を指定したSELECT 文の文字データの並び替え順序は，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 と 2 では 1 の指定が優先されます。
  1. setHADBOrderMode メソッドで指定した並び替え順序
  2. システムプロパティのadb_clt_sql_order_mode で指定した並び替え順序
  3. DriverManager クラスのgetConnection メソッドの引数info に指定したadb_clt_sql_order_mode プロパティの値
  4. DriverManager クラスのgetConnection メソッドの引数url に指定したadb_clt_sql_order_mode の値
  5. サーバ定義のadb_sql_order_mode オペランドで指定した並び替え順序
- setHADBOrderMode メソッドで指定した，ORDER BY 句を指定したSELECT 文の文字データの並び替え順序は，Connection オブジェクトをいったんプールしたあとに再度使用した場合，前回の並び替え順序を引き継ぎません。setHADBOrderMode メソッドを実行していない状態と同じになります。
- このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。

8.3.35 setHADBSQLHashFltSize(int areaSize)

(1) 機能

このConnection オブジェクトに対してハッシュフィルタ領域サイズを設定します。

このメソッドで設定する内容は，クライアント定義のadb_sql_exe_hashflt_area_size オペランドに対応しています。

> **重要**
>
> このメソッドを指定したConnection オブジェクトから生成されたStatement オブジェクトまたはPreparedStatement オブジェクトは，Connection オブジェクトがクローズされるまでは，このメソッドで指定したハッシュフィルタ領域サイズが適用されます。

このメソッドの使い方（実行する SQL 文ごとにハッシュフィルタ領域サイズを変更する方法）については，「8.3.37 setHADBSQLMaxRthdNum(int rthdNum)」の「(7) 使用例」を参照してください。

(2) 形式

```
public void setHADBSQLHashFltSize(int areaSize) throws SQLException
```

(3) 引数

int areaSize：

設定するハッシュフィルタ領域サイズをメガバイト単位で指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- areaSize に不正な値を指定した場合

(6) 留意事項

- このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。
- setHADBSQLHashFltSize メソッドで指定したハッシュフィルタ領域サイズは，Connection オブジェクトをいったんプールしたあとに再度使用した場合，前回のハッシュフィルタ領域サイズを引き継ぎません。setHADBSQLHashFltSize メソッドを実行していない状態と同じになります。
- setHADBSQLHashFltSize メソッドでハッシュフィルタ領域サイズを再設定しても，生成済みのStatement オブジェクトまたはPreparedStatement オブジェクトには，再設定後のハッシュフィルタ領域サイズは適用されません。
- ハッシュフィルタ領域サイズの決定方法については，「2.2.3 性能に関するオペランド」の adb_sql_exe_hashflt_area_size オペランドの説明を参照してください。

8.3.36 setHADBSQLHashTblSize(int areaSize)

(1) 機能

このConnection オブジェクトに対してハッシュテーブル領域サイズを設定します。

このメソッドで設定する内容は，クライアント定義のadb_sql_exe_hashtbl_area_size オペランドに対応しています。

重要

このメソッドを指定したConnection オブジェクトから生成されたStatement オブジェクトまたはPreparedStatement オブジェクトは，Connection オブジェクトがクローズされるまでは，このメソッドで指定したハッシュテーブル領域サイズが適用されます。

このメソッドの使い方（実行する SQL 文ごとにハッシュテーブル領域サイズを変更する方法）については，「8.3.37　setHADBSQLMaxRthdNum(int rthdNum)」の「(7)　使用例」を参照してください。

(2)　形式

```
public void setHADBSQLHashTblSize(int areaSize) throws SQLException
```

(3)　引数

int areaSize：

設定するハッシュテーブル領域サイズをメガバイト単位で指定します。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- areaSize に不正な値を指定した場合

(6)　留意事項

- このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2　Wrapper インタフェース」を参照してください。
- setHADBSQLHashTblSize メソッドで指定したハッシュテーブル領域サイズは，Connection オブジェクトをいったんプールしたあとに再度使用した場合，前回のハッシュテーブル領域サイズを引き継ぎません。setHADBSQLHashTblSize メソッドを実行していない状態と同じになります。
- setHADBSQLHashTblSize メソッドでハッシュテーブル領域サイズを再設定しても，生成済みのStatement オブジェクトまたはPreparedStatement オブジェクトには，再設定後のハッシュテーブル領域サイズは適用されません。
- ハッシュテーブル領域サイズの決定方法については，「2.2.3　性能に関するオペランド」の adb_sql_exe_hashtbl_area_size オペランドの説明を参照してください。

8.3.37　setHADBSQLMaxRthdNum(int rthdNum)

(1)　機能

このConnection オブジェクトに対して最大 SQL 処理リアルスレッド数を設定します。

このメソッドで設定する内容は，クライアント定義のadb_sql_exe_max_rthd_num オペランドに対応しています。

> **重要**
>
> このメソッドを指定したConnection オブジェクトから生成されたStatement オブジェクトまたはPreparedStatement オブジェクトは，Connection オブジェクトがクローズされるまでは，このメソッドで指定した最大 SQL 処理リアルスレッド数が適用されます。

(2)　形式

```
public void setHADBSQLMaxRthdNum(int rthdNum) throws SQLException
```

(3)　引数

int rthdNum：

設定する最大 SQL 処理リアルスレッド数を指定します。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合
- rthdNum に不正な値を指定した場合

(6)　留意事項

- このメソッドは，AdbConnection インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2　Wrapper インタフェース」を参照してください。
- setHADBSQLMaxRthdNum メソッドで指定した最大 SQL 処理リアルスレッド数は，Connection オブジェクトをいったんプールしたあとに再度使用した場合，前回の最大 SQL 処理リアルスレッド数を引き継ぎません。setHADBSQLMaxRthdNum メソッドを実行していない状態と同じになります。

- setHADBSQLMaxRthdNum メソッドで最大 SQL 処理リアルスレッド数を再設定しても，生成済みの Statement オブジェクトまたはPreparedStatement オブジェクトには，再設定後の最大 SQL 処理リアルスレッド数は適用されません。
- 最大 SQL 処理リアルスレッド数の決定方法については，「2.2.3　性能に関するオペランド」の adb_sql_exe_max_rthd_num オペランドの説明を参照してください。

(7)　使用例

setHADBSQLMaxRthdNum およびgetHADBSQLMaxRthdNum メソッドを使用して，実行する SQL 文ごとに最大 SQL 処理リアルスレッド数を変更する例を次に示します。

(例 1)

```
   :
   :
Connection cnct = ds.getConnection();
AdbConnection con = cnct.unwrap(com.hitachi.hadb.jdbc.AdbConnection.class);

// 最大SQL処理リアルスレッド数のデフォルト値を退避しておく
int default_sql_exe_rthd_num = con.getHADBSQLMaxRthdNum();

// 最大SQL処理リアルスレッド数を0に変更する
con.setHADBSQLMaxRthdNum(0);

// PreparedStatementオブジェクトを生成する
// このPreparedStatementオブジェクトで実行するすべてのSQL文は,
// 最大SQL処理リアルスレッド数が0で動作する
PreparedStatement pstmt = con.prepareStatement("SELECT * FROM MASTER.SQL_USERS WHERE USER
_NAME=?");

// 最大SQL処理リアルスレッド数をデフォルト値に戻す
con.setHADBSQLMaxRthdNum(default_sql_exe_rthd_num);
// この操作は上記のpstmtに影響を与えない

pstmt.setString(1, "HOGE");
ResultSet rs = pstmt.executeQuery(); // 最大SQL処理リアルスレッド数が0でSQL文を実行する
while (rs.next()) {
   :
   :
}
rs.close();
pstmt.close();
   :
   :
```

(例 2)

ステートメントごとに異なる最大 SQL 処理リアルスレッド数で，SQL 文を実行する例を次に示します。

```
   :
   :
Connection cnct = ds.getConnection();
AdbConnection con = cnct.unwrap(com.hitachi.hadb.jdbc.AdbConnection.class);
```

```
// 最大SQL処理リアルスレッド数のデフォルト値を退避しておく
int default_sql_exe_rthd_num = con.getHADBSQLMaxRthdNum();

// 最大SQL処理リアルスレッド数を0に変更する
con.setHADBSQLMaxRthdNum(0);

// PreparedStatementオブジェクトを生成する
// pstmt1で実行するすべてのSQL文は，最大SQL処理リアルスレッド数が0で動作する
PreparedStatement pstmt1 = con.prepareStatement("SELECT * FROM MASTER.SQL_USERS WHERE USE
R_NAME=?");

// 最大SQL処理リアルスレッド数を4に変更する
con.setHADBSQLMaxRthdNum(4); // この操作は上記の pstmt1 に影響を与えない

// PreparedStatementオブジェクトを生成する
// pstmt2で実行するすべてのSQL文は，最大SQL処理リアルスレッド数が4で動作する
PreparedStatement pstmt2 = con.prepareStatement("SELECT * FROM MASTER.SQL_TABLE_PRIVILEGE
S WHERE GRANTOR=?");

// 最大SQL処理リアルスレッド数をデフォルト値に戻す
con.setHADBSQLMaxRthdNum(default_sql_exe_rthd_num);
// この操作は上記のpstmt1およびpstmt2に影響を与えない

pstmt1.setString(1, "HOGE");
ResultSet rs1 = pstmt1.executeQuery(); // 最大SQL処理リアルスレッド数が0でSQL文を実行する
while (rs1.next()) {
   :
   :
}
rs1.close();
pstmt1.close();

pstmt2.setString(1, "FOO");
ResultSet rs2 = pstmt2.executeQuery(); // 最大SQL処理リアルスレッド数が4でSQL文を実行する
while (rs2.next()) {
   :
   :
}
rs2.close();
pstmt2.close();
   :
   :
```

メモ

- 上記の例では，例外処理などのエラー処理は省いています。
- setHADBSQLHashTblSize およびgetHADBSQLHashTblSize メソッドを使用して，実行する SQL 文ごとにハッシュテーブル領域サイズを変更する場合も，上記の例を参考にしてください。
- setHADBSQLHashFltSize およびgetHADBSQLHashFltSize メソッドを使用して，実行する SQL 文ごとにハッシュフィルタ領域サイズを変更する場合も，上記の例を参考にしてください。

8.3.38　setHoldability(int holdability)

(1)　機能

このConnection オブジェクトを使用して生成されたResultSet オブジェクトの保持機能を設定します。

(2)　形式

```
public synchronized void setHoldability(int holdability) throws SQLException
```

(3)　引数

int holdability：

この引数を指定しても無視されます。常にResultSet.HOLD_CURSORS_OVER_COMMIT が指定されたと見なします。

(4)　戻り値

なし。

(5)　発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.39　setReadOnly(boolean readOnly)

(1)　機能

このConnection オブジェクトを読み取り専用モードに設定します。トランザクションアクセスモードを設定します。

(2)　形式

```
public synchronized void setReadOnly(boolean readOnly) throws SQLException
```

(3)　引数

boolean readOnly：

読み取り専用モードにする場合はtrue を，そうでない場合はfalse を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- Connectionオブジェクトがクローズされている場合
- トランザクションがすでに開始している場合
- 当該Connectionオブジェクトで，保持機能にHOLD_CURSORS_OVER_COMMITを指定して作成したResultSetオブジェクトがクローズされていない場合

(6) 留意事項

- トランザクションアクセスモードは，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と2では1の指定が優先されます。
  1. setReadOnlyメソッドで指定したトランザクションアクセスモード
  2. システムプロパティのadb_clt_trn_access_modeで指定したトランザクションアクセスモード
  3. DriverManagerクラスのgetConnectionメソッドの引数infoに指定したadb_clt_trn_access_modeプロパティの値
  4. DriverManagerクラスのgetConnectionメソッドの引数urlに指定したadb_clt_trn_access_modeの値
- setReadOnlyメソッドで指定したトランザクションアクセスモードは，Connectionオブジェクトをいったんプールしたあとに再使用した場合，前回のトランザクションアクセスモードを引き継ぎません。setReadOnlyメソッドを実行しないときと同じ状態になります。

8.3.40 setSchema(String schema)

(1) 機能

アクセスするスキーマ名を設定します。ただし，このメソッドでは引数の指定値は無視され，スキーマ名の設定は行われません。

(2) 形式

```
public synchronized void setSchema(String schema) throws SQLException
```

(3) 引数

String schema：

スキーマ名を指定します。ただし，指定しても無視されます。

(4) 戻り値

なし。

(5) 発生する例外

Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.3.41 setTransactionIsolation(int level)

(1) 機能

このConnection オブジェクトのトランザクション隔離性水準を設定します。

(2) 形式

```
public synchronized void setTransactionIsolation(int level) throws SQLException
```

(3) 引数

int level：

適用するトランザクション隔離性水準を指定します。次に示すどちらかの値を指定します。

- Connection.TRANSACTION_READ_COMMITTED
  トランザクション隔離性水準にREAD COMMITTED を適用する場合に指定します。
- Connection.TRANSACTION_REPEATABLE_READ
  トランザクション隔離性水準にREPEATABLE READ を適用する場合に指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Connection オブジェクトがクローズされている場合

- トランザクションがすでに開始している場合
- このConnection オブジェクトで，保持機能にHOLD_CURSORS_OVER_COMMIT を指定して作成したResultSet オブジェクトがクローズされていない場合
- トランザクション隔離性水準に次の値以外を指定した場合
  - Connection.TRANSACTION_READ_COMMITTED
  - Connection.TRANSACTION_REPEATABLE_READ

(6) 留意事項

- トランザクション隔離性水準は，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 と 2 では 1 の指定が優先されます。
  1. setTransactionIsolation メソッドで指定したトランザクション隔離性水準
  2. システムプロパティのadb_clt_trn_iso_lv で指定したトランザクション隔離性水準
  3. DriverManager クラスのgetConnection メソッドの引数info に指定したadb_clt_trn_iso_lv プロパティの値
  4. DriverManager クラスのgetConnection メソッドの引数url に指定したadb_clt_trn_iso_lv の値
  5. サーバ定義のadb_sys_trn_iso_lv オペランドで指定したトランザクション隔離性水準
- setTransactionIsolation メソッドで指定したトランザクション隔離性水準は，Connection オブジェクトをいったんプールしたあとに再度使用した場合，前回のトランザクション隔離性水準を引き継ぎません。setTransactionIsolation メソッドを実行していない状態と同じになります。

8.4 Statement インタフェース

ここでは，Statement インタフェースで提供されているメソッドについて説明します。

8.4.1 Statement インタフェースのメソッド一覧

(1) Statement インタフェースの主な機能

Statement インタフェースでは，主に次の機能が提供されています。

- SQL 文の実行
- 検索結果としての結果セット（ResultSet オブジェクト）の生成
- 更新結果としての更新行数の返却
- 最大検索行数の設定
- 検索制限時間の設定

(2) HADB でサポートしている Statement インタフェースのメソッド

HADB でサポートしているStatement インタフェースのメソッドの一覧を次の表に示します。

表 8-12 Statement インタフェースのメソッド一覧

項番	Statement インタフェースのメソッド	機能
1	addBatch(String sql)	Statement オブジェクトのバッチに SQL 文を登録します。
2	cancel()	該当するオブジェクト，および該当するオブジェクトと同一接続のオブジェクトが実行している SQL の処理を取り消します。
3	clearBatch()	Statement オブジェクトのバッチに登録されている SQL 文をすべてクリアします。
4	clearWarnings()	このStatement オブジェクトに関して報告されたすべての警告をクリアします。
5	close()	Statement オブジェクト，およびこのStatement オブジェクトから生成されたResultSet オブジェクトのクローズを行います。
6	closeOnCompletion()	このStatement オブジェクトに依存するすべての結果セットがクローズされたときに，このStatement オブジェクトをクローズします。
7	execute(String sql)	SQL 文を実行します。
8	executeBatch()	バッチに登録された SQL 文を実行し，更新行数の int 型の配列を返却します。
9	executeLargeBatch()	バッチに登録された SQL 文を実行し，更新行数の long 型の配列を返却します。

項番	Statement インタフェースのメソッド	機能
10	executeLargeUpdate(String sql)	SQL 文（検索系 SQL 以外の SQL 文）を実行し，更新行数をlong 型で返却します。
11	executeQuery(String sql)	検索系 SQL を実行し，検索結果を格納したResultSet オブジェクトを返却します。
12	executeUpdate(String sql)	SQL 文（検索系 SQL 以外の SQL 文）を実行し，更新行数をint 型で返却します。
13	getConnection()	Statement オブジェクトを生成したConnection オブジェクトを返却します。
14	getFetchDirection()	このStatement オブジェクトから生成される結果集合のデフォルトのフェッチ方向を取得します。
15	getFetchSize()	Statement オブジェクトから生成されるResultSet オブジェクトの，デフォルトのフェッチサイズ（HADB サーバから HADB クライアントに検索結果を一括転送する行数）を取得します。
16	getHADBSQLSerialNum()	このStatement オブジェクトに割り当てられている SQL 文通番を取得します。
17	getHADBStatementHandle()	このStatement オブジェクトに割り当てられている文ハンドルを取得します。
18	getLargeMaxRows()	このStatement オブジェクトによって生成されるResultSet オブジェクトの最大格納行数をlong 型で取得します。
19	getLargeUpdateCount()	更新行数をlong 型で返します。
20	getMaxFieldSize()	このStatement オブジェクトによって生成されるResultSet オブジェクトの，CHAR およびVARCHAR の列の最大バイト数を取得します。
21	getMaxRows()	このStatement オブジェクトによって生成されるResultSet オブジェクトの最大格納行数をint 型で取得します。
22	getMoreResults()	次の結果集合に移動します。
23	getQueryTimeout()	setQueryTimeout メソッドで設定した SQL 処理のタイムアウト時間を取得します。
24	getResultSet()	ResultSet オブジェクトとして，検索結果を取得します。
25	getResultSetConcurrency()	このStatement オブジェクトから生成されるResultSet オブジェクトの並行処理モードを取得します。
26	getResultSetHoldability()	このStatement オブジェクトから生成されるResultSet オブジェクトの保持機能を取得します。
27	getResultSetType()	このStatement オブジェクトから生成されるResultSet オブジェクトの，結果セットタイプを取得します。
28	getUpdateCount()	更新行数をint 型で返します。
29	getWarnings()	このStatement オブジェクトに関する呼び出しによって報告される，最初の警告を取得します。

項番	Statement インタフェースのメソッド	機能
30	isClosed()	このStatement オブジェクトがクローズされているかどうかを示す値を取得します。
31	isCloseOnCompletion()	このStatement オブジェクトに依存するすべての結果セットがクローズされたときに，このStatement オブジェクトがクローズされるかどうかを示す値を取得します。
32	isPoolable()	Statement オブジェクトがプールできるかどうかを示す値を取得します。
33	setCursorName(String name)	後続のStatement オブジェクトのexecute メソッドによって使用されるSQL カーソル名を設定します。
34	setEscapeProcessing(boolean enable)	このStatement オブジェクトによるエスケープ構文の解析を有効にするかどうかを設定します。
35	setFetchDirection(int direction)	このStatement オブジェクトから生成される結果集合のフェッチ方向を設定します。
36	setFetchSize(int rows)	このStatement オブジェクトから生成されるResultSet オブジェクトの，デフォルトのフェッチサイズ（HADB サーバから HADB クライアントに検索結果を一括転送する行数）を設定します。
37	setLargeMaxRows(long max)	このStatement オブジェクトによって生成されるResultSet オブジェクトの最大格納行数をlong 型で設定します。
38	setMaxFieldSize(int max)	このStatement オブジェクトによって生成されるResultSet オブジェクトの，CHAR およびVARCHAR の列の最大バイト数を設定します。
39	setMaxRows(int max)	このStatement オブジェクトによって生成されるResultSet オブジェクトの最大格納行数をint 型で設定します。
40	setQueryTimeout(int seconds)	SQL 処理のタイムアウト時間を設定します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されることがあります。

(3) 必要なパッケージ名称とクラス名称

Statement インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbStatement

8.4.2 addBatch(String sql)

(1) 機能

Statement オブジェクトのバッチに SQL 文を登録します。最大 2,147,483,647 個の SQL 文を登録できます。

(2) 形式

```
public synchronized void addBatch(String sql) throws SQLException
```

(3) 引数

String sql：

登録する SQL 文を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 上限値の 2,147,483,647 個を超える SQL 文を登録しようとした場合
- SQL 文にnull または長さ 0 の文字列を指定している場合
- 指定された SQL 文の長さが，16,000,000 文字を超えている場合

8.4.3 cancel()

(1) 機能

該当するオブジェクト，および該当するオブジェクトと同一接続のオブジェクトが実行している SQL の処理を取り消します。

このメソッドを使用して，実行中の SQL に非同期キャンセルを実行できます。

(2) 形式

```
public void cancel() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

(6) 留意事項

- このメソッドは，該当するStatement オブジェクトが SQL の実行中でなくても，同一接続オブジェクトに対して，ほかのオブジェクトが SQL を実行している場合は，非同期キャンセルを行います。
- 該当するStatement オブジェクトが SQL の実行中ではなく，かつ同一接続オブジェクトに対してほかのオブジェクトが SQL を実行していない場合，このメソッドはキャンセルを実行しません。
- SQL の非同期キャンセルに成功した場合，トランザクションがロールバックされます。

8.4.4 clearBatch()

(1) 機能

Statement オブジェクトのバッチに登録されている SQL 文をすべてクリアします。

(2) 形式

```
public synchronized void clearBatch() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- Statementオブジェクトがクローズされている場合
- Statementオブジェクトを生成したConnectionオブジェクトがクローズされている場合

8.4.5 clearWarnings()

(1) 機能

このStatementオブジェクトに関して報告されたすべての警告をクリアします。

(2) 形式

```
public synchronized void clearWarnings() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

なし。

8.4.6 close()

(1) 機能

Statementオブジェクト，およびこのStatementオブジェクトから生成されたResultSetオブジェクトのクローズを行います。

(2) 形式

```
public synchronized void close() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

データベースのアクセスエラーが発生した場合，SQLException が投入されます。

8.4.7 closeOnCompletion()

(1) 機能

このStatement オブジェクトに依存するすべての結果セットがクローズされたときに，このStatement オブジェクトをクローズします。Statement オブジェクトの実行によって結果セットが生成されない場合は，このメソッドは無効になります。

このメソッドを複数回呼び出しても，このStatement オブジェクトへの効果は切り替わりません。

このメソッドの呼び出しは，Statement オブジェクトのその後の実行と，現在開いていて依存されている結果セットがあるStatement オブジェクトの両方に影響します。

(2) 形式

```
public synchronized void closeOnCompletion() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.8 execute(String sql)

(1) 機能

SQL 文を実行します。ResultSet オブジェクトや更新行数をgetResultSet メソッドおよびgetUpdateCount メソッド（またはgetLargeUpdateCount メソッド）で取得できます。

実行した SQL 文の種類とgetResultSet メソッドおよびgetUpdateCount メソッド（または getLargeUpdateCount メソッド）の戻り値の関係を次の表に示します。

表 8-13 実行した SQL 文の種類と getResultSet メソッドおよび getUpdateCount メソッド（または getLargeUpdateCount メソッド）の戻り値の関係

実行した SQL 文の種類	getResultSet メソッドの戻り値	getUpdateCount メソッドまたは getLargeUpdateCount メソッドの戻り値
検索系 SQL の場合	実行結果のResultSet オブジェクト	-1
検索系 SQL 以外の SQL の場合	null	0 以上の値※
SQL の実行がエラーになった場合	null	-1

注※

更新行数がInteger.MAX_VALUE を超える可能性がある場合は，getUpdateCount メソッドではなく，getLargeUpdateCount メソッドを使用してください。getUpdateCount メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

メモ

検索系 SQL とはSELECT 文のことです。また，検索系 SQL 以外の SQL とは，UPDATE 文などの更新系 SQL のほかに，CREATE TABLE などの定義系 SQL も含んでいます。

更新系 SQL とは，INSERT 文，UPDATE 文，DELETE 文，PURGE CHUNK 文，およびTRUNCATE TABLE 文のことです。

(2) 形式

```
public synchronized boolean execute(String sql) throws SQLException
```

(3) 引数

String sql：

実行する SQL 文を指定します。

(4) 戻り値

実行した SQL 文が検索系 SQL の場合はtrue が，そうでない場合はfalse が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- sql にnull または長さ 0 の文字列を指定した場合
- データベースのアクセスエラーが発生した場合
- 指定された SQL 文の長さが，16,000,000 文字を超えている場合

8.4.9 executeBatch()

(1) 機能

バッチに登録された SQL 文を実行し，更新行数のint 型の配列を返却します。

バッチに登録されているすべての SQL 文を実行したあと，バッチに登録されている SQL 文をすべてクリアします。途中でエラーが発生した場合でも，バッチに登録されている SQL 文はすべてクリアされます。

更新行数がInteger.MAX_VALUE を超える可能性がある場合は，executeBatch メソッドではなく，executeLargeBatch メソッドを使用してください。executeBatch メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(2) 形式

```
public synchronized int[] executeBatch() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

実行したSQL文ごとの更新行数をint型の配列にして返却します。配列は，バッチに登録されたSQL文の順序になります。バッチにSQL文が1つも登録されていない場合，またはバッチの1つ目のSQL文がエラーだった場合，要素数0の配列が返却されます。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- Statementオブジェクトがクローズされている場合
- Statementオブジェクトを生成したConnectionオブジェクトがクローズされている場合

次に示す場合は，例外BatchUpdateException（SQLExceptionのサブクラス）が投入されます。

- 検索系SQLがバッチで実行された場合
- データベースのアクセスエラーが発生した場合
- キャンセルを受け付けた場合

8.4.10 executeLargeBatch()

(1) 機能

バッチに登録されたSQL文を実行し，更新行数のlong型の配列を返却します。

バッチに登録されているすべてのSQL文を実行したあと，バッチに登録されているSQL文をすべてクリアします。途中でエラーが発生した場合でも，バッチに登録されているSQL文はすべてクリアされます。

更新行数がInteger.MAX_VALUEを超える可能性がある場合は，executeBatchメソッドではなく，executeLargeBatchメソッドを使用してください。executeBatchメソッドを使用した場合に，Integer.MAX_VALUEを超えると0が返却されます。

(2) 形式

```
public synchronized long[] executeLargeBatch() throws SQLException
```

(3) 引数

なし。

(4)　戻り値

実行したSQL文ごとの更新行数をlong型の配列にして返却します。配列は，バッチに登録されたSQL文の順序になります。バッチにSQL文が1つも登録されていない場合，またはバッチの1つ目のSQL文がエラーだった場合，要素数0の配列が返却されます。

(5)　発生する例外

発生する例外については，「8.4.9　executeBatch()」の「(5)　発生する例外」を参照してください。

8.4.11　executeLargeUpdate(String sql)

(1)　機能

SQL文（検索系SQL以外のSQL文）を実行し，更新行数をlong型で返却します。

更新行数がInteger.MAX_VALUEを超える可能性がある場合は，executeUpdate(String sql)メソッドではなく，executeLargeUpdate(String sql)メソッドを使用してください。executeUpdate(String sql)メソッドを使用した場合に，Integer.MAX_VALUEを超えると0が返却されます。

(2)　形式

```
public synchronized long executeLargeUpdate(String sql) throws SQLException
```

(3)　引数

String sql：

実行するSQL文（検索系SQL以外のSQL文）を指定します。

(4)　戻り値

INSERT文，UPDATE文，およびDELETE文を実行した場合は，更新行数がlong型で返却されます。これら以外のSQL文を実行した場合は，0が返却されます。

(5)　発生する例外

発生する例外については，「8.4.13　executeUpdate(String sql)」の「(5)　発生する例外」を参照してください。

8.4.12　executeQuery(String sql)

(1)　機能

検索系 SQL を実行し，検索結果を格納したResultSet オブジェクトを返却します。

(2)　形式

```
public synchronized ResultSet executeQuery(String sql) throws SQLException
```

(3)　引数

String sql：

　実行する SQL 文（検索系 SQL）を指定します。

(4)　戻り値

検索結果を格納したResultSet オブジェクトが返却されます。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 検索系 SQL 以外の SQL 文（INSERT 文など）を実行した場合
- sql にnull または長さ 0 の文字列を指定した場合
- データベースのアクセスエラーが発生した場合
- 指定された SQL 文の長さが，16,000,000 文字を超えている場合

8.4.13　executeUpdate(String sql)

(1)　機能

SQL 文（検索系 SQL 以外の SQL 文）を実行し，更新行数をint 型で返却します。更新行数が Integer.MAX_VALUE を超える可能性がある場合は，executeUpdate(String sql)メソッドではなく，executeLargeUpdate(String sql)メソッドを使用してください。executeUpdate(String sql)メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(2) 形式

```
public synchronized int executeUpdate(String sql) throws SQLException
```

(3) 引数

String sql：

実行する SQL 文（検索系 SQL 以外の SQL 文）を指定します。

(4) 戻り値

INSERT 文，UPDATE 文，およびDELETE 文を実行した場合は，更新行数がint 型で返却されます。これら以外の SQL 文を実行した場合は，0 が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 検索系 SQL（SELECT 文）を実行した場合
- sql にnull または長さ 0 の文字列を指定した場合
- データベースのアクセスエラーが発生した場合
- 指定された SQL 文の長さが，16,000,000 文字を超えている場合

8.4.14 getConnection()

(1) 機能

Statement オブジェクトを生成したConnection オブジェクトを返却します。

(2) 形式

```
public synchronized Connection getConnection() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Connection オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.15 getFetchDirection()

(1) 機能

このStatement オブジェクトから生成される結果集合のデフォルトのフェッチ方向を取得します。

(2) 形式

```
public synchronized int getFetchDirection() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にResultSet.FETCH_FORWARD が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.16　getFetchSize()

(1)　機能

Statement オブジェクトから生成されるResultSet オブジェクトの，デフォルトのフェッチサイズ（HADB サーバから HADB クライアントに検索結果を一括転送する行数）を取得します。

(2)　形式

```
public synchronized int getFetchSize() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

このStatement オブジェクトから生成されるResultSet オブジェクトの，デフォルトのフェッチサイズ（HADB サーバから HADB クライアントに検索結果を一括転送する行数）が返却されます。

setFetchSize メソッドで0 を設定した場合，フェッチサイズにはシステムプロパティ，ユーザプロパティ，または接続用の URL のプロパティのadb_clt_fetch_size の値が適用されますが，戻り値には0 が返却されます。フェッチサイズと戻り値の関係を次の表に示します。

表 8-14　フェッチサイズと戻り値の関係

setFetchSize メソッドの設定値（m）	戻り値
0	0
1≦*m*≦65,535	*m*

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.17　getHADBSQLSerialNum()

(1)　機能

このStatement オブジェクトに割り当てられている SQL 文通番を取得します。

(2) 形式

```
public long getHADBSQLSerialNum() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このStatement オブジェクトに割り当てられている SQL 文通番が返却されます。

SQL 文の実行前にこのメソッドを実行した場合，0 が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

(6) 留意事項

このメソッドは，AdbStatement インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。

8.4.18 getHADBStatementHandle()

(1) 機能

このStatement オブジェクトに割り当てられている文ハンドルを取得します。

(2) 形式

```
public int getHADBStatementHandle() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このStatement オブジェクトに割り当てられている文ハンドルが返却されます。

SQL 文の実行前にこのメソッドを実行した場合，0 が返却されます。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

(6)　留意事項

このメソッドは，AdbStatement インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2　Wrapper インタフェース」を参照してください。

8.4.19　getLargeMaxRows()

(1)　機能

このStatement オブジェクトによって生成されるResultSet オブジェクトの最大格納行数をlong 型で取得します。最大格納行数を超えた場合，その行はResultSet オブジェクトに格納されません。また，格納されなかったことは通知されません。

setLargeMaxRows メソッドでInteger.MAX_VALUE を超える値を指定した場合は，getMaxRows メソッドではなく，getLargeMaxRows メソッドを使用してください。getMaxRows メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(2)　形式

```
public synchronized long getLargeMaxRows() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

このStatement オブジェクトによって生成される，ResultSet オブジェクトの最大格納行数がlong 型で返却されます。setMaxRows メソッドまたはsetLargeMaxRows メソッドで設定した値が返却されます。0 が返却された場合，最大格納行数の設定がないことを意味します。

(5)　発生する例外

発生する例外については，「8.4.22　getMaxRows()」の「(5)　発生する例外」を参照してください。

8.4.20　getLargeUpdateCount()

(1)　機能

更新行数をlong 型で返します。

更新行数がInteger.MAX_VALUE を超える可能性がある場合は，getUpdateCount メソッドではなく，getLargeUpdateCount メソッドを使用してください。getUpdateCount メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(2)　形式

```
public synchronized long getLargeUpdateCount() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

戻り値の詳細については，「8.4.29　getUpdateCount()」の「(4)　戻り値」を参照してください。

(5)　発生する例外

発生する例外については，「8.4.29　getUpdateCount()」の「(5)　発生する例外」を参照してください。

8.4.21　getMaxFieldSize()

(1)　機能

このStatement オブジェクトによって生成されるResultSet オブジェクトの，CHAR およびVARCHAR の列の最大バイト数を取得します。最大バイト数を超えた分のデータは切り捨てられます。

(2)　形式

```
public synchronized int getMaxFieldSize() throws SQLException
```

(3)　引数

なし。

(4) 戻り値

CHAR およびVARCHAR の列の最大バイト数が返却されます。setMaxFieldSize メソッドで設定した値が返却されます。0 が返却された場合，最大バイト数の設定がないことを意味します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.22 getMaxRows()

(1) 機能

このStatement オブジェクトによって生成されるResultSet オブジェクトの最大格納行数をint 型で取得します。最大格納行数を超えた場合，その行はResultSet オブジェクトに格納されません。また，格納されなかったことは通知されません。

setLargeMaxRows メソッドでInteger.MAX_VALUE を超える値を指定した場合は，getMaxRows メソッドではなく，getLargeMaxRows メソッドを使用してください。getMaxRows メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(2) 形式

```
public synchronized int getMaxRows() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このStatement オブジェクトによって生成される，ResultSet オブジェクトの最大格納行数がint 型で返却されます。setMaxRows メソッドまたはsetLargeMaxRows メソッドで設定した値が返却されます。0 が返却された場合，最大格納行数の設定がないことを意味します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合

- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.23 getMoreResults()

(1) 機能

次の結果集合に移動します。

(2) 形式

```
public synchronized boolean getMoreResults() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

次の結果集合が存在する場合はtrue が，そうでない場合はfalse が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.24 getQueryTimeout()

(1) 機能

setQueryTimeout メソッドで設定した SQL 処理のタイムアウト時間を取得します。

(2) 形式

```
public synchronized int getQueryTimeout() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

setQueryTimeout メソッドで設定した SQL 処理のタイムアウト時間（単位：秒）が返却されます。
setQueryTimeout メソッドを実行していない場合は，0 が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.25 getResultSet()

(1) 機能

ResultSet オブジェクトとして，検索結果を取得します。

(2) 形式

```
public synchronized ResultSet getResultSet() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Statement オブジェクトが保持しているResultSet オブジェクトが返却されます。ResultSet オブジェクト内に検索結果がない場合は，null が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.26　getResultSetConcurrency()

(1)　機能

このStatement オブジェクトから生成されるResultSet オブジェクトの並行処理モードを取得します。

(2)　形式

```
public synchronized int getResultSetConcurrency() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にResultSet.CONCUR_READ_ONLY が返却されます。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.27　getResultSetHoldability()

(1)　機能

このStatement オブジェクトから生成されるResultSet オブジェクトの保持機能を取得します。

(2)　形式

```
public synchronized int getResultSetHoldability() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にResultSet.HOLD_CURSORS_OVER_COMMIT が返却されます。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- Statementオブジェクトがクローズされている場合
- Statementオブジェクトを生成したConnectionオブジェクトがクローズされている場合

8.4.28 getResultSetType()

(1) 機能

このStatementオブジェクトから生成されるResultSetオブジェクトの，結果セットタイプを取得します。

(2) 形式

```
public synchronized int getResultSetType() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

ResultSet.TYPE_FORWARD_ONLYまたはResultSet.TYPE_SCROLL_INSENSITIVEが返却されます。

ResultSet.TYPE_FORWARD_ONLY：
　カーソルが順方向だけ移動できます。

ResultSet.TYPE_SCROLL_INSENSITIVE：
　カーソルがスクロールできますが，値の変更は反映されません。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- Statementオブジェクトがクローズされている場合
- Statementオブジェクトを生成したConnectionオブジェクトがクローズされている場合

8.4.29 getUpdateCount()

(1) 機能

更新行数をint 型で返します。

更新行数がInteger.MAX_VALUE を超える可能性がある場合は，getUpdateCount メソッドではなく，getLargeUpdateCount メソッドを使用してください。getUpdateCount メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(2) 形式

```
public synchronized int getUpdateCount() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

戻り値の詳細を次の表に示します。

表 8-15 getUpdateCount メソッドまたは getLargeUpdateCount メソッドの戻り値の詳細

Statement オブジェクトのメソッドの実行条件				getUpdateCount または getLargeUpdateCount メソッドの戻り値
execute*XXX* メソッドを実行していない場合				-1
execute*XXX* メソッドを実行している場合	最後に実行したexecute*XXX* メソッドのあとに，getMoreResults メソッドを実行した場合			-1
	最後に実行したexecute*XXX* メソッドでエラーが発生した場合			-1
	最後にexecuteBatch メソッドまたはexecuteLargeBatch メソッドを実行した場合			-1
	最後にexecuteBatch メソッドまたはexecuteLargeBatch メソッド以外のexecute*XXX* メソッドを実行した場合	最後に実行した SQL 文が検索系 SQL の場合		-1
		最後に実行した SQL が検索系 SQL 以外の SQL 文の場合	INSERT，UPDATE，DELETE	更新行数※
			そのほか	0

注※

更新行数がInteger.MAX_VALUE を超える可能性がある場合は，getUpdateCount メソッドではなく，getLargeUpdateCount メソッドを使用してください。getUpdateCount メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.30 getWarnings()

(1) 機能

このStatement オブジェクトに関する呼び出しによって報告される，最初の警告を取得します。2 つ以上の警告がある場合，後続の警告は，最初の警告にチェーンされ，直前に取得された警告のSQLWarning オブジェクトのgetNextWarning メソッドを呼び出すことによって取得できます。

(2) 形式

```
public synchronized SQLWarning getWarnings() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

最初のSQLWarning オブジェクトが返却されます。SQLWarning オブジェクトがない場合は，null が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.31　isClosed()

(1)　機能

このStatement オブジェクトがクローズされているかどうかを示す値を取得します。

(2)　形式

```
public synchronized boolean isClosed() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

Statement オブジェクトがクローズされている場合はtrue が返却されます。Statement オブジェクトがクローズされていない場合はfalse が返却されます。

(5)　発生する例外

なし。

8.4.32　isCloseOnCompletion()

(1)　機能

このStatement オブジェクトに依存するすべての結果セットがクローズされたときに，このStatement オブジェクトがクローズされるかどうかを示す値を取得します。

(2)　形式

```
public synchronized boolean isCloseOnCompletion() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

このStatement オブジェクトに依存するすべての結果セットがクローズされたときに，このStatement オブジェクトがクローズされる場合はtrue が返却されます。そうでない場合はfalse が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.33 isPoolable()

(1) 機能

Statement オブジェクトがプールできるかどうかを示す値を取得します。

(2) 形式

```
public synchronized boolean isPoolable() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.34 setCursorName(String name)

(1) 機能

後続のStatement オブジェクトのexecute メソッドによって使用される SQL カーソル名を設定します。

(2) 形式

```
public synchronized void setCursorName(String name) throws SQLException
```

(3) 引数

String name：

SQL カーソル名を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.4.35 setEscapeProcessing(boolean enable)

(1) 機能

このStatement オブジェクトによるエスケープ構文の解析を有効にするかどうかを設定します。

(2) 形式

```
public synchronized void setEscapeProcessing(boolean enable) throws SQLException
```

(3) 引数

boolean enable：

エスケープ構文の解析を有効にする場合はtrue を，無効にする場合はfalse を指定します。

このメソッドを実行しない場合，true が仮定されます。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- Statementオブジェクトがクローズされている場合
- Statementオブジェクトを生成したConnectionオブジェクトがクローズされている場合

8.4.36 setFetchDirection(int direction)

(1) 機能

このStatementオブジェクトから生成される結果集合のフェッチ方向を指定します。

(2) 形式

```
public synchronized void setFetchDirection(int direction) throws SQLException
```

(3) 引数

int direction：

フェッチ方向を指定します。ResultSet.FETCH_FORWARDだけ指定できます。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- Statementオブジェクトがクローズされている場合
- Statementオブジェクトを生成したConnectionオブジェクトがクローズされている場合
- directionにResultSet.FETCH_FORWARD以外が指定された場合

8.4.37 setFetchSize(int rows)

(1) 機能

このStatementオブジェクトから生成されるResultSetオブジェクトの，デフォルトのフェッチサイズ（HADBサーバからHADBクライアントに検索結果を一括転送する行数）を設定します。

(2) 形式

```
public synchronized void setFetchSize(int rows) throws SQLException
```

(3) 引数

int rows：

一括転送する行数を 0～65,535 の範囲で指定します。

1 以上の値を指定した場合，指定値に従って複数の行データを HADB サーバから HADB クライアントに一括転送します。

0 を指定した場合，またはこのメソッドを実行しない場合は，システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティのadb_clt_fetch_size の値が適用されます。

setFetchSize メソッドの設定値とプロパティのadb_clt_fetch_size の設定値の関係を次の表に示します。

表 8-16　setFetchSize メソッドの設定値とプロパティの adb_clt_fetch_size の設定値の関係

setFetchSize メソッドの設定値 (m)	プロパティの adb_clt_fetch_size の設定値 (n)	一括転送される行数
0	1≦*n*≦65,535	*n*
	指定なし	1
1≦*m*≦65,535	1≦*n*≦65,535	*m*
	指定なし	*m*

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- rows に 0～65,535 以外の値を指定した場合
- rows に指定した値が，最大格納行数（setMaxRows メソッドの設定値）よりも大きい場合
- rows に指定した値が，最大格納行数（setLargeMaxRows メソッドの設定値）よりも大きい場合

(6) 留意事項

HADB サーバに対して一度の通信で要求する行数は，次の表に示す優先順位に従って決定されます。

表 8-17　HADB サーバに対して一度の通信で要求する行数の優先順位

優先順位	HADB サーバに対して一度の通信で要求する行数
1	ResultSet クラスのsetFetchSize メソッドの引数で指定した値
2	Statement クラスのsetFetchSize メソッドの引数で指定した値
3	システムプロパティadb_clt_fetch_size で設定した値
4	DriverManager クラスのgetConnection メソッドの引数info に指定したadb_clt_fetch_size プロパティの値
5	DriverManager クラスのgetConnection メソッドの引数url に指定したadb_clt_fetch_size の値

なお，検索結果の行数が上記の表に示す転送行数より多い場合，検索が終了するまで（または AP からの検索要求がなくなるまで），HADB サーバに対して転送を要求します。

8.4.38　setLargeMaxRows(long max)

(1)　機能

このStatement オブジェクトによって生成されるResultSet オブジェクトの最大格納行数をlong 型で設定します。最大格納行数を超えた場合，その行はResultSet オブジェクトに格納されません。また，格納されなかったことは通知されません。

生成済みのResultSet オブジェクトに対しては，このメソッドの設定値は適用されません。

(2)　形式

```
public synchronized void setLargeMaxRows(long max) throws SQLException
```

(3)　引数

long max：

最大格納行数を指定します。

0 を指定した場合，最大格納行数を設定しません。ただし，結果セットタイプがResultSet.TYPE_SCROLL_INSENSITIVE の場合は，0 を指定してもInteger.MAX_VALUE が最大格納行数となります。

このメソッドを実行しない場合，0（最大格納行数を設定しない）が仮定されます。

(4)　戻り値

なし。

(5)　発生する例外

発生する例外については，「8.4.40　setMaxRows(int max)」の「(5)　発生する例外」を参照してください。

8.4.39　setMaxFieldSize(int max)

(1)　機能

このStatement オブジェクトによって生成されるResultSet オブジェクトの，CHAR およびVARCHAR の列の最大バイト数を設定します。最大バイト数を超えた分のデータは切り捨てられます。

生成済みのResultSet オブジェクトに対しては，このメソッドの設定値は適用されません。

(2)　形式

```
public synchronized void setMaxFieldSize(int max) throws SQLException
```

(3)　引数

int max：

CHAR およびVARCHAR の列に適用する最大バイト数を指定します。

0 を指定した場合，最大バイト数の設定をしません。

このメソッドを実行しない場合，0（最大バイト数を設定しない）が仮定されます。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- max に0 未満の値を指定した場合

8.4.40　setMaxRows(int max)

(1)　機能

このStatement オブジェクトによって生成されるResultSet オブジェクトの最大格納行数をint 型で設定します。最大格納行数を超えた場合，その行はResultSet オブジェクトに格納されません。また，格納されなかったことは通知されません。

生成済みのResultSet オブジェクトに対しては，このメソッドの設定値は適用されません。

(2)　形式

```
public synchronized void setMaxRows(int max) throws SQLException
```

(3)　引数

int max：

最大格納行数を指定します。

0 を指定した場合，最大格納行数を設定しません。ただし，結果セットタイプがResultSet.TYPE_SCROLL_INSENSITIVE の場合は，0 を指定してもInteger.MAX_VALUE が最大格納行数となります。

このメソッドを実行しない場合，0（最大格納行数を設定しない）が仮定されます。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- max に0 未満の値を指定した場合

8.4.41　setQueryTimeout(int seconds)

(1)　機能

SQL 処理のタイムアウト時間を設定します。

(2) 形式

```
public synchronized void setQueryTimeout(int seconds) throws SQLException
```

(3) 引数

int seconds：

SQL 処理のタイムアウト時間（単位：秒）を 0～65,535 の範囲で指定します。

0 を指定した場合，またはこのメソッドを実行しない場合は，システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティのadb_clt_rpc_sql_wait_time の設定値が有効になります。

このメソッドを実行すると，次に示す 2 つの待ち時間が監視対象になります。

- HADB クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間

  上記の待ち時間を超えた場合，SQLCODE が-732（KFAA30732-E）のタイムアウトエラーが AP に返されます。このとき，SQL 文の処理はキャンセルされ，トランザクションはロールバックされます。そのあと，HADB サーバから AP が切り離されます。

- 同一コネクションで複数のSELECT 文を同時実行した際に，処理リアルスレッド数不足が発生したときの，処理リアルスレッドを確保できるまでの待ち時間

  上記の待ち時間を超えた場合，SQLCODE が-1071570（KFAA71570-E）のタイムアウトエラーが AP に返されます。このとき，SQL 文の処理はキャンセルされますが，トランザクションはロールバックされません。また，AP は HADB サーバから切り離されません。

このメソッドによる待ち時間の監視の目的については，「7.4.1　データの検索方法」の「(4)　同一コネクションで複数の SELECT 文を同時実行する際の注意事項」を参照してください。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- Statement オブジェクトがクローズされている場合
- Statement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- seconds に0 未満の値を指定した場合

(6) 留意事項

seconds に 65,536（プロパティのadb_clt_rpc_sql_wait_time の最大値）以上の値を指定した場合，このメソッドの指定値は無視されます。

8.4.42　Statement インタフェースに関する注意事項

(1)　executeXXX メソッド実行時の注意事項

該当するStatement オブジェクトが生成したResultSet オブジェクトがクローズされていない状態でexecute*XXX* メソッドを実行すると，以前生成したResultSet オブジェクトをクローズします。このため，execute*XXX* メソッド実行後，以前に生成したResultSet オブジェクトを使用して検索結果を取得しようとすると，SQLException が投入されます。SQLException が発生する例を次に示します。

■SQLException が発生する例

```
Statement st  = con.createStatement();
ResultSet rs1 = st.executeQuery("select * from tb1");
ResultSet rs2 = st.executeQuery("select * from tb2");
rs1.next();  // SQLExceptionを投入する。
rs2.next();
```

(2)　Statement オブジェクトのクローズ

Statement オブジェクトを使用したあとは，必ず明示的にclose メソッドでStatement オブジェクトをクローズしてください。Statement オブジェクトを明示的にクローズすると，HADB 内の対応する文ハンドルが解放されます。Statement オブジェクトをクローズしないと，文ハンドルが不足することがあります。

ただし，文ハンドルはCOMMIT またはROLLBACK でトランザクションを決着した場合も解放されます。したがって，トランザクションをある程度の間隔で決着させると，文ハンドルの不足を回避できます。

8.5 PreparedStatement インタフェース

ここでは，PreparedStatement インタフェースで提供されているメソッドについて説明します。

8.5.1 PreparedStatement インタフェースのメソッド一覧

(1) PreparedStatement インタフェースの主な機能

PreparedStatement インタフェースでは，主に次の機能が提供されています。

- ？パラメタ指定の SQL の実行
- ？パラメタの設定
- 検索結果としてのResultSet オブジェクトの生成，返却
- 更新結果としての更新行数の返却

PreparedStatement インタフェースはStatement インタフェースのサブインタフェースであるため，Statement インタフェースの機能をすべて継承します。

(2) HADB でサポートしている PreparedStatement インタフェースのメソッド

HADB でサポートしているPreparedStatement インタフェースのメソッドの一覧を次の表に示します。

表 8-18 PreparedStatement インタフェースのメソッド一覧

項番	PreparedStatement インタフェースのメソッド	機能
1	addBatch()	PreparedStatement オブジェクトのバッチに，現在のパラメタセットを追加します。
2	clearParameters()	現在設定されているパラメタセットの値をすべてクリアします。
3	execute()	前処理済みの SQL 文を実行します。
4	executeLargeUpdate()	前処理済みの SQL 文（検索系 SQL 以外の SQL 文）を実行し，更新行数をlong 型で返却します。
5	executeQuery()	前処理済みの検索系 SQL を実行し，実行結果を格納したResultSet オブジェクトを返却します。
6	executeUpdate()	前処理済みの SQL 文（検索系 SQL 以外の SQL 文）を実行し，更新行数をint 型で返却します。
7	getHADBSQLSerialNum()	このPreparedStatement オブジェクトに割り当てられているSQL 文通番を取得します。

項番	PreparedStatement インタフェースのメソッド	機能
8	getHADBStatementHandle()	このPreparedStatement オブジェクトに割り当てられている文ハンドルを取得します。
9	getMetaData()	このPreparedStatement が実行されるときに返されるResultSet オブジェクトの列に関する情報を格納するResultSetMetaData オブジェクトを返却します。
10	getParameterMetaData()	このPreparedStatement オブジェクト内のパラメタのメタ情報を表すParameterMetaData オブジェクトを返却します。
11	setAsciiStream(int parameterIndex, InputStream x, int length)	指定したInputStream オブジェクトの持つ値を，？パラメタ値に設定します。
12	setBigDecimal(int parameterIndex, BigDecimal x)	指定したBigDecimal オブジェクトを？パラメタ値に設定します。
13	setBinaryStream(int parameterIndex, InputStream x, int length)	指定したInputStream オブジェクトが持つ値を？パラメタ値に設定します。
14	setBoolean(int parameterIndex, boolean x)	指定したboolean 値を？パラメタ値に設定します。
15	setByte(int parameterIndex, byte x)	指定したbyte 値を？パラメタ値に設定します。
16	setBytes(int parameterIndex, byte[] x)	指定したbyte 配列を？パラメタ値に設定します。
17	setCharacterStream(int parameterIndex, Reader reader, int length)	指定したReader オブジェクトを？パラメタ値に設定します。
18	setDate(int parameterIndex, Date x)	指定したjava.sql.Date オブジェクトを？パラメタ値に設定します。
19	setDate(int parameterIndex, Date x, Calendar cal)	ローカルタイムで指定したjava.sql.Date オブジェクトを，指定したカレンダーのタイムゾーンの時間に変換し，？パラメタ値に設定します。
20	setDouble(int parameterIndex, double x)	指定したdouble 値を？パラメタ値に設定します。
21	setFloat(int parameterIndex, float x)	指定したfloat 値を？パラメタ値に設定します。
22	setInt(int parameterIndex, int x)	指定したint 値を？パラメタ値に設定します。
23	setLong(int parameterIndex, long x)	指定したlong 値を？パラメタ値に設定します。
24	setNull(int parameterIndex,int sqlType)	指定した？パラメタにナル値を設定します。
25	setObject(int parameterIndex, Object x)	指定したオブジェクトが持つ値を？パラメタ値に設定します。
26	setObject(int parameterIndex, Object x, int targetSqlType)	
27	setObject(int parameterIndex, Object x, int targetSqlType, int scale)	
28	setShort(int parameterIndex, short x)	指定したshort 値を？パラメタ値に設定します。
29	setString(int parameterIndex, String x)	指定したString オブジェクトを？パラメタ値に設定します。

項番	PreparedStatement インタフェースのメソッド	機能
30	setTime(int parameterIndex, Time x)	指定したjava.sql.Time オブジェクトを？パラメタ値に設定します。
31	setTime(int parameterIndex, Time x, Calendar cal)	ローカルタイムで指定したjava.sql.Time オブジェクトを，指定したカレンダーのタイムゾーンの時間に変換し，？パラメタ値に設定します。
32	setTimestamp(int parameterIndex, Timestamp x)	指定したjava.sql.Timestamp オブジェクトを？パラメタ値に設定します。
33	setTimestamp(int parameterIndex, Timestamp x, Calendar cal)	ローカルタイムで指定したjava.sql.Timestamp オブジェクトを，指定したカレンダーのタイムゾーンの時間に変換し，？パラメタ値に設定します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されます。

(3) 必要なパッケージ名称とクラス名称

PreparedStatement インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbPreparedStatement

8.5.2 addBatch()

(1) 機能

PreparedStatement オブジェクトのバッチに，現在のパラメタセットを追加します。最大 2,147,483,647 個パラメタセットを登録できます。

(2) 形式

```
public synchronized void addBatch() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- PreparedStatementオブジェクトがクローズされている場合
- このPreparedStatementオブジェクトを生成したConnectionオブジェクトがクローズされている場合
- すべての？パラメタに値がセットされていない場合
- バッチの登録数が2,147,483,647個を超えた場合

8.5.3 clearParameters()

(1) 機能

現在設定されているパラメタセットの値をすべてクリアします。

(2) 形式

```
public synchronized void clearParameters() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- PreparedStatementオブジェクトがクローズされている場合
- このPreparedStatementオブジェクトを生成したConnectionオブジェクトがクローズされている場合

8.5.4　execute()

(1)　機能

前処理済みの SQL 文を実行します。

実行結果のResultSet オブジェクトや更新行数を，PreparedStatement オブジェクトのgetResultSet メソッド，およびPreparedStatement オブジェクトのgetUpdateCount メソッド（またはgetLargeUpdateCount メソッド）で取得できます。

execute メソッド実行後のgetResultSet メソッドおよびgetUpdateCount メソッド（または getLargeUpdateCount メソッド）の戻り値については，「表 8-13　実行した SQL 文の種類と getResultSet メソッドおよび getUpdateCount メソッド（または getLargeUpdateCount メソッド）の戻り値の関係」を参照してください。

(2)　形式

```
public synchronized boolean execute() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

実行した SQL が検索系 SQL の場合はtrue が，そうでない場合はfalse が返却されます。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 値を設定していない？パラメタがある場合
- データベースのアクセスエラーが発生した場合

8.5.5　executeLargeUpdate()

(1)　機能

前処理済みの SQL 文（検索系 SQL 以外の SQL 文）を実行し，更新行数をlong 型で返却します。

更新行数がInteger.MAX_VALUE を超える可能性がある場合は，executeUpdate()メソッドではなく，executeLargeUpdate()メソッドを使用してください。executeUpdate()メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(2)　形式

```
public synchronized long executeLargeUpdate() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

INSERT 文，UPDATE 文，およびDELETE 文を実行した場合は，更新行数がlong 型で返却されます。これら以外の SQL 文を実行した場合は，0 が返却されます。

(5)　発生する例外

発生する例外については，「8.5.7　executeUpdate()」の「(5)　発生する例外」を参照してください。

8.5.6　executeQuery()

(1)　機能

前処理済みの検索系 SQL を実行し，実行結果を格納したResultSet オブジェクトを返却します。

(2)　形式

```
public synchronized ResultSet executeQuery() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

実行結果を格納したResultSet オブジェクトが返却されます。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 検索系 SQL 以外の SQL 文（INSERT 文など）を実行した場合
- 値を設定していない？パラメタがある場合
- データベースのアクセスエラーが発生した場合

8.5.7 executeUpdate()

(1) 機能

前処理済みの SQL 文（検索系 SQL 以外の SQL 文）を実行し，更新行数をint 型で返却します。

更新行数がInteger.MAX_VALUE を超える可能性がある場合は，executeUpdate()メソッドではなく，executeLargeUpdate()メソッドを使用してください。executeUpdate()メソッドを使用した場合に，Integer.MAX_VALUE を超えると0 が返却されます。

(2) 形式

```
public synchronized int executeUpdate() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

INSERT 文，UPDATE 文，およびDELETE 文を実行した場合は，更新行数がint 型で返却されます。これら以外の SQL 文を実行した場合は，0 が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このオブジェクトを生成したConnection オブジェクトがクローズされている場合
- 検索系 SQL を実行した場合
- 値を設定していない？パラメタがある場合
- データベースのアクセスエラーが発生した場合

8.5.8　getHADBSQLSerialNum()

(1)　機能

このPreparedStatement オブジェクトに割り当てられている SQL 文通番を取得します。

(2)　形式

```
public long getHADBSQLSerialNum() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

このPreparedStatement オブジェクトに割り当てられている SQL 文通番が返却されます。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合

(6)　留意事項

このメソッドは，AdbPreparedStatement インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2　Wrapper インタフェース」を参照してください。

8.5.9　getHADBStatementHandle()

(1)　機能

このPreparedStatement オブジェクトに割り当てられている文ハンドルを取得します。

(2)　形式

```
public int getHADBStatementHandle() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このPreparedStatement オブジェクトに割り当てられている文ハンドルが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合

(6) 留意事項

このメソッドは，AdbPreparedStatement インタフェースで提供される HADB 独自のメソッドです。実行方法については，「12.2 Wrapper インタフェース」を参照してください。

8.5.10 getMetaData()

(1) 機能

このPreparedStatement が実行されるときに返されるResultSet オブジェクトの列に関する情報を格納するResultSetMetaData オブジェクトを返却します。

(2) 形式

```
public synchronized ResultSetMetaData getMetaData() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このPreparedStatement オブジェクトのメタ情報が，ResultSetMetaData オブジェクトに格納されて返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.5.11 getParameterMetaData()

(1) 機能

このPreparedStatement オブジェクト内のパラメタのメタ情報を表すParameterMetaData オブジェクトを返却します。返却するParameterMetaData は，Connection.prepareStatement()実行時点でサーバから取得したパラメタのメタ情報となります。

(2) 形式

```
public synchronized ParameterMetaData getParameterMetaData() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このPreparedStatement オブジェクトのメタ情報が，ParametertMetaData オブジェクトに格納されて返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.5.12 setAsciiStream(int parameterIndex, InputStream x, int length)

(1) 機能

指定したInputStream オブジェクトの持つ値を，？パラメタ値に設定します。

(2) 形式

```
public synchronized void setAsciiStream(int parameterIndex, InputStream x, int length) throw
s SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

InputStream x：

？パラメタに設定する値を持つjava.io.InputStream オブジェクトを指定します。

int length：

設定するバイト数を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- length に0 未満の値を指定した場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

(6) 留意事項

setAsciiStream メソッドの実行時，x からの入力が終わったあとでも，x に対してclose メソッドは実行されません。

8.5.13 setBigDecimal(int parameterIndex, BigDecimal x)

(1) 機能

指定したBigDecimal オブジェクトを？パラメタ値に設定します。

(2) 形式

```
public synchronized void setBigDecimal(int parameterIndex, BigDecimal x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

BigDecimal x：

？パラメタに設定するjava.math.BigDecimal オブジェクトを指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.14 setBinaryStream(int parameterIndex, InputStream x, int length)

(1) 機能

指定したInputStream オブジェクトが持つ値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setBinaryStream(int parameterIndex, InputStream x, int length) thro
ws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

InputStream x：

？パラメタに設定する値を持つjava.io.InputStream オブジェクトを指定します。

int length：

設定するバイト数を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- length に0 未満の値を指定した場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

(6) 留意事項

setBinaryStream メソッドの実行時，x からの入力が終わったあとでも，x に対してclose メソッドは実行されません。

8.5.15 setBoolean(int parameterIndex, boolean x)

(1) 機能

指定したboolean 値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setBoolean(int parameterIndex, boolean x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

boolean x：

？パラメタに設定する値を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合

8.5.16 setByte(int parameterIndex, byte x)

(1) 機能

指定したbyte 値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setByte(int parameterIndex, byte x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

byte x：

？パラメタに設定する値を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- PreparedStatementオブジェクトがクローズされている場合
- このPreparedStatementオブジェクトを生成したConnectionオブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタのHADBのデータ型がこのメソッドでは設定できないデータ型の場合

8.5.17 setBytes(int parameterIndex, byte[] x)

(1) 機能

指定したbyte配列を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setBytes(int parameterIndex, byte[] x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

byte[] x：

？パラメタに設定する値を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- PreparedStatementオブジェクトがクローズされている場合
- このPreparedStatementオブジェクトを生成したConnectionオブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタのHADBのデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.18 setCharacterStream(int parameterIndex, Reader reader, int length)

(1) 機能

指定したReader オブジェクトを？パラメタ値に設定します。

(2) 形式

```
public synchronized void setCharacterStream(int parameterIndex, Reader reader, int length) t
hrows SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

Reader reader：

？パラメタに設定する値を持つjava.io.Reader オブジェクトを指定します。

int length：

文字数を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- length に0 未満の値を指定した場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合
- エンコードに失敗した場合

8.5.19　setDate(int parameterIndex, Date x)

(1)　機能

指定したjava.sql.Date オブジェクトを？パラメタ値に設定します。

(2)　形式

```
public synchronized void setDate(int parameterIndex, Date x) throws SQLException
```

(3)　引数

int parameterIndex：

　？パラメタの番号を指定します。

Date x：

　？パラメタに設定する値を持つjava.sql.Date オブジェクトを指定します。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.20　setDate(int parameterIndex, Date x, Calendar cal)

(1)　機能

ローカルタイムで指定したjava.sql.Date オブジェクトを，指定したカレンダーのタイムゾーンの時間に変換し，？パラメタ値に設定します。

(2) 形式

```
public synchronized void setDate(int parameterIndex, Date x, Calendar cal) throws SQLExcepti
on
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

Date x：

？パラメタに設定する値を持つjava.sql.Date オブジェクトを指定します。

Calendar cal：

データベースに格納する値のタイムゾーンが設定されたカレンダーを指定します。null を指定した場合，Java 仮想マシンのデフォルトのタイムゾーンのカレンダーが適用されます。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.21 setDouble(int parameterIndex, double x)

(1) 機能

指定したdouble 値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setDouble(int parameterIndex, double x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

double x：

？パラメタに設定する値を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.22 setFloat(int parameterIndex, float x)

(1) 機能

指定したfloat 値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setFloat(int parameterIndex, float x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

float x：

？パラメタに設定する値を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- PreparedStatementオブジェクトがクローズされている場合
- このPreparedStatementオブジェクトを生成したConnectionオブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタのHADBのデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.23 setInt(int parameterIndex, int x)

(1) 機能

指定したint値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setInt(int parameterIndex, int x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

int x：

？パラメタに設定する値を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- PreparedStatementオブジェクトがクローズされている場合

- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.24 setLong(int parameterIndex, long x)

(1) 機能

指定したlong 値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setLong(int parameterIndex, long x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

long x：

？パラメタに設定する値を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.25　setNull(int parameterIndex,int sqlType)

(1)　機能

指定した？パラメタにナル値を設定します。

(2)　形式

```
public synchronized void setNull(int parameterIndex,int sqlType) throws SQLException
```

(3)　引数

int parameterIndex：

　？パラメタの番号を指定します。

int sqlType：

　JDBC の SQL データ型を指定します。

　この引数を指定しても無視されます。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合

8.5.26　setObject(int parameterIndex, Object x)

(1)　機能

指定したオブジェクトが持つ値を？パラメタ値に設定します。

(2)　形式

```
public synchronized void setObject(int parameterIndex, Object x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

Object x：

？パラメタに設定する値を持つオブジェクトを指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.27 setObject(int parameterIndex, Object x, int targetSqlType)

(1) 機能

指定したオブジェクトが持つ値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setObject(int parameterIndex, Object x, int targetSqlType) throws S
QLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

Object x：

？パラメタに設定する値を持つオブジェクトを指定します。

int targetSqlType：

JDBC の SQL データ型を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合
- targetSqlType が次のどれかの場合

  Types.ARRAY，Types.BLOB，Types.CLOB，Types.JAVA_OBJECT，Types.REF，Types.STRUCT

8.5.28 setObject(int parameterIndex, Object x, int targetSqlType, int scale)

(1) 機能

指定したオブジェクトが持つ値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setObject(int parameterIndex, Object x, int targetSqlType, int scal
e) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

Object x：

？パラメタに設定する値を持つオブジェクトを指定します。

int targetSqlType：

JDBC の SQL データ型を指定します。

int scale：

位取りを指定します。ただし，指定値は無視されます。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合
- targetSqlType が次のどれかの場合
  Types.ARRAY，Types.BLOB，Types.CLOB，Types.JAVA_OBJECT，Types.REF，Types.STRUCT

8.5.29 setShort(int parameterIndex, short x)

(1) 機能

指定したshort 値を？パラメタ値に設定します。

(2) 形式

```
public synchronized void setShort(int parameterIndex, short x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

short x：

？パラメタに設定する値を指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- PreparedStatementオブジェクトがクローズされている場合
- このPreparedStatementオブジェクトを生成したConnectionオブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタのHADBのデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.30 setString(int parameterIndex, String x)

(1) 機能

指定したStringオブジェクトを？パラメタ値に設定します。

(2) 形式

```
public synchronized void setString(int parameterIndex, String x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

String x：

？パラメタに設定する値を持つStringオブジェクトを指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- PreparedStatementオブジェクトがクローズされている場合
- このPreparedStatementオブジェクトを生成したConnectionオブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタのHADBのデータ型がこのメソッドでは設定できないデータ型の場合

- 指定した値が列のデータ型の範囲外，または変換できない形式の場合
- エンコードに失敗した場合

8.5.31　setTime(int parameterIndex, Time x)

(1)　機能

指定したjava.sql.Time オブジェクトを？パラメタ値に設定します。

(2)　形式

```
public synchronized void setTime(int parameterIndex, Time x) throws SQLException
```

(3)　引数

int parameterIndex：

　？パラメタの番号を指定します。

Time x：

　？パラメタに設定する値を持つjava.sql.Time オブジェクトを指定します。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.32　setTime(int parameterIndex, Time x, Calendar cal)

(1)　機能

ローカルタイムで指定したjava.sql.Time オブジェクトを，指定したカレンダーのタイムゾーンの時間に変換し，？パラメタ値に設定します。

(2)　形式

```
public synchronized void setTime(int parameterIndex, Time x, Calendar cal) throws SQLExcepti
on
```

(3)　引数

int parameterIndex：

？パラメタの番号を指定します。

Time x：

？パラメタに設定する値を持つjava.sql.Time オブジェクトを指定します。

Calendar cal：

データベースに格納する値のタイムゾーンが設定されたカレンダーを指定します。null を指定した場合，Java 仮想マシンのデフォルトのタイムゾーンのカレンダーが適用されます。

(4)　戻り値

なし。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.33 setTimestamp(int parameterIndex, Timestamp x)

(1) 機能

指定したjava.sql.Timestamp オブジェクトを？パラメタ値に設定します。

(2) 形式

```
public synchronized void setTimestamp(int parameterIndex, Timestamp x) throws SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

Timestamp x：

？パラメタに設定する値を持つjava.sql.Timestamp オブジェクトを指定します。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.34 setTimestamp(int parameterIndex, Timestamp x, Calendar cal)

(1) 機能

ローカルタイムで指定したjava.sql.Timestamp オブジェクトを，指定したカレンダーのタイムゾーンの時間に変換し，？パラメタ値に設定します。

(2) 形式

```
public synchronized void setTimestamp(int parameterIndex, Timestamp x, Calendar cal) throws
SQLException
```

(3) 引数

int parameterIndex：

？パラメタの番号を指定します。

Timestamp x：

？パラメタに設定する値を持つjava.sql.Timestamp オブジェクトを指定します。

Calendar cal：

データベースに格納する値のタイムゾーンが設定されたカレンダーを指定します。null を指定した場合，Java 仮想マシンのデフォルトのタイムゾーンのカレンダーが適用されます。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- PreparedStatement オブジェクトがクローズされている場合
- このPreparedStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合
- 存在しない？パラメタの番号を指定した場合
- ？パラメタの HADB のデータ型がこのメソッドでは設定できないデータ型の場合
- 指定した値が列のデータ型の範囲外，または変換できない形式の場合

8.5.35 PreparedStatement インタフェースに関する注意事項

PreparedStatement インタフェースはStatement インタフェースのサブインタフェースであるため，Statement インタフェースの注意事項はすべて該当します。

ここでは，それ以外のPreparedStatement インタフェース固有の注意事項について説明します。

(1) ？パラメタの設定

- set*XXX* メソッドによってマッピングできるかどうかについては，「7.6.1 データ型のマッピング」の「(3) ？パラメタ設定時のマッピング」を参照してください。

- set*XXX* メソッドに指定した列番号が存在しない場合は，SQLException が投入されます。
- set*XXX* メソッドに指定した値が，対応する？パラメタのデータ型が表現できる値の範囲を超えた場合，オーバフローが発生してSQLException が投入されます。オーバフローが発生するおそれのある set*XXX* メソッドと HADB のデータ型の組み合わせについては，「7.6.3　オーバフローが発生したときの処理」を参照してください。
- set*XXX* メソッドで設定した値は，次に示すどれかの操作をするまで有効になります。
  - 該当するPreparedStatement オブジェクトに対して，clearParameters メソッドを実行する
  - 該当するPreparedStatement オブジェクトに対して，set*XXX* メソッドを実行し，かつ設定対象の？パラメタは同じである
  - 該当するPreparedStatement オブジェクトに対して，close メソッドを実行する

(2)　HADB の DECIMAL 型または NUMERIC 型の？パラメタに対する値指定

HADB のDECIMAL 型またはNUMERIC 型の？パラメタに対してset*XXX* メソッドで値を設定する場合，？パラメタの精度および位取りと，値の精度および位取りが一致していないときの処理を次に示します。

- 実際の精度よりも大きいとき：SQLException が投入されます。
- 実際の精度よりも小さいとき：拡張します。
- 実際の位取りよりも大きいとき：実際の位取りで切り捨てます。
- 実際の位取りよりも小さいとき：0 で補完し拡張します。

(3)　HADB の TIME 型および TIMESTAMP 型の？パラメタに対する値指定

小数秒精度が低いデータ型に小数秒精度が高いデータ型を指定した場合，差分の小数秒精度が切り捨てられます。逆に，小数秒精度が高いデータ型に小数秒精度が低いデータ型を指定した場合，差分の小数秒精度に 0 を補い拡張されます。

(4)　HADB の CHAR または VARCHAR 型の？パラメタに対する値指定

HADB のCHAR 型またはVARCHAR 型の？パラメタに対してset*XXX* メソッドで値を指定する場合，？パラメタの定義長よりも値を文字列表現にしたときの長さが大きいと，SQLException が投入されます。

(5)　setObject で指定できるオブジェクト

setObject メソッドの引数x に指定できるオブジェクトは，次に示す型のオブジェクトです。

- byte[]
- java.lang.Byte
- java.lang.Double
- java.lang.Float

- java.lang.Integer
- java.lang.Long
- java.lang.Short
- java.lang.String
- java.math.BigDecimal
- java.sql.Boolean
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp

8.6　ResultSet インタフェース

ここでは，ResultSet インタフェースで提供されているメソッドについて説明します。

8.6.1　ResultSet インタフェースのメソッド一覧

(1)　ResultSet インタフェースの主な機能

ResultSet インタフェースでは，主に次の機能が提供されています。

- 行単位の結果セット内の移動
- 結果データの返却
- 検索結果データがナル値かどうかの通知

(2)　HADB でサポートしている ResultSet インタフェースのメソッド

HADB でサポートしているResultSet インタフェースのメソッドの一覧を次の表に示します。

表 8-19　ResultSet インタフェースのメソッドの一覧

項番	ResultSet インタフェースのメソッド	機能
1	absolute(int row)	ResultSet オブジェクト内の指定された行にカーソルを移動します。
2	afterLast()	ResultSet オブジェクト内の最終行の後ろにカーソルを移動します。
3	beforeFirst()	ResultSet オブジェクト内の先頭行の前にカーソルを移動します。
4	clearWarnings()	このResultSet オブジェクトに関して報告された，すべての警告をクリアします。
5	close()	ResultSet オブジェクトでオープンしていたカーソルをクローズし，JDBC リソースを解放します。
6	findColumn(String columnName)	指定した列名の列番号を返します。
7	first()	ResultSet オブジェクト内の先頭行にカーソルを移動します。
8	getArray(int columnIndex)	ResultSet オブジェクトの現在行の列（配列型）の値をArray オブジェクトに取得します。
9	getArray(String columnName)	
10	getAsciiStream(int columnIndex)	ResultSet オブジェクトの現在行の列の値を java.io.InputStream オブジェクトに取得します。
11	getAsciiStream(String columnName)	

項番	ResultSet インタフェースのメソッド	機能
12	getBigDecimal(int columnIndex)	ResultSet オブジェクトの現在行の列の値を java.math.BigDecimal オブジェクトに取得します。
13	getBigDecimal(String columnName)	
14	getBinaryStream(int columnIndex)	ResultSet オブジェクトの現在行の列の値を java.io.InputStream オブジェクトに取得します。
15	getBinaryStream(String columnName)	
16	getBoolean(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のboolean で取得します。
17	getBoolean(String columnName)	
18	getByte(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のbyte で取得します。
19	getByte(String columnName)	
20	getBytes(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のbyte 配列で取得します。
21	getBytes(String columnName)	
22	getCharacterStream(int columnIndex)	ResultSet オブジェクトの現在行の列の値をjava.io.Reader オブジェクトに取得します。
23	getCharacterStream(String columnName)	
24	getConcurrency()	このResultSet オブジェクトの並行処理モードを取得します。
25	getCursorName()	このResultSet オブジェクトが使用する SQL カーソルの名前を取得します。
26	getDate(int columnIndex)	ResultSet オブジェクトの現在行の列の値をjava.sql.Date オブジェクトに取得します。
27	getDate(int columnIndex, Calendar cal)	
28	getDate(String columnName)	
29	getDate(String columnName, Calendar cal)	
30	getDouble(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のdouble で取得します。
31	getDouble(String columnName)	
32	getFetchDirection()	このResultSet オブジェクトのフェッチ方向を取得します。
33	getFetchSize()	ResultSet オブジェクトのフェッチサイズを取得します。
34	getFloat(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のfloat で取得します。
35	getFloat(String columnName)	
36	getHoldability()	このResultSet オブジェクトの保持機能の状態を表す値を取得します。
37	getInt(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のint で取得します。
38	getInt(String columnName)	
39	getLong(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のlong で取得します。
40	getLong(String columnName)	

項番	ResultSet インタフェースのメソッド	機能
41	getMetaData()	このResultSet オブジェクトのメタ情報を取得します。
42	getObject(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のObject で取得します。
43	getObject(String columnName)	
44	getObject(int columnIndex,Class<T> type)	ResultSet オブジェクトの現在行の列の値を取得し，指定されたクラスの Java データ型に変換します。
45	getObject(String columnLabel,Class<T> type)	
46	getRow()	現在の行の番号を取得します。
47	getShort(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のshort で取得します。
48	getShort(String columnName)	
49	getStatement()	このResultSet オブジェクトを生成したStatement オブジェクトを取得します。
50	getString(int columnIndex)	ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のString で取得します。
51	getString(String columnName)	
52	getTime(int columnIndex)	ResultSet オブジェクトの現在行の列の値をjava.sql.Time オブジェクトに取得します。
53	getTime(int columnIndex, Calendar cal)	
54	getTime(String columnName)	
55	getTime(String columnName, Calendar cal)	
56	getTimestamp(int columnIndex)	ResultSet オブジェクトの現在行の列の値を java.sql.Timestamp オブジェクトに取得します。
57	getTimestamp(int columnIndex, Calendar cal)	
58	getTimestamp(String columnName)	
59	getTimestamp(String columnName, Calendar cal)	
60	getType()	ResultSet オブジェクトの型を返します。
61	getWarnings()	このResultSet オブジェクトに関する呼び出しによって報告される，最初の警告を取得します。
62	isAfterLast()	ResultSet オブジェクトの最終行の後ろに，カーソルがあるかどうかの情報を取得します。
63	isBeforeFirst()	ResultSet オブジェクトの先頭行の前に，カーソルがあるかどうかの情報を取得します。
64	isClosed()	このResultSet オブジェクトがクローズされているかどうかを示す値を取得します。
65	isFirst()	ResultSet オブジェクトの先頭行に，カーソルがあるかどうかの情報を取得します。
66	isLast()	ResultSet オブジェクトの最終行に，カーソルがあるかどうかの情報を取得します。

項番	ResultSet インタフェースのメソッド	機能
67	last()	ResultSet オブジェクトの最終行に，カーソルを移動します。
68	next()	カーソルを次の行に移動します。
69	previous()	カーソルを 1 つ前の行に移動します。
70	relative(int rows)	カーソルを移動します。
71	setFetchDirection(int direction)	ResultSet オブジェクトのフェッチ方向を設定します。
72	setFetchSize(int rows)	ResultSet オブジェクトを検索する際のフェッチサイズ（フェッチする行数）を設定します。
73	wasNull()	最後に取得した列の値がナル値かどうかの情報を返します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されることがあります。

(3) 必要なパッケージ名称とクラス名称

ResultSet インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbResultSet

8.6.2 absolute(int row)

(1) 機能

ResultSet オブジェクト内の指定された行にカーソルを移動します。

(2) 形式

```
public synchronized boolean absolute(int row) throws SQLException
```

(3) 引数

int row：

カーソルの移動先の行番号を指定します。正の番号を指定した場合，行番号は結果セットの先頭からカウントされます。負の番号を指定した場合，行番号は結果セットの終端からカウントされます。

(4) 戻り値

absolute メソッドを呼び出したあとのカーソル位置が，先頭行の前または最終行の後ろの場合はfalse が，そうでない場合はtrue が返却されます。

absolute メソッド実行時のカーソルの移動先と戻り値を次の表に示します。

表 8-20 absolute メソッド実行時のカーソルの移動先と戻り値

結果集合の行数※	row の指定値	カーソルの移動先	戻り値
0	0 以外	先頭行の前のまま	false
n	*n* ＜row	最終行の後ろ	false
	1≦row≦*n*	row	true
	-*n*≦row≦-1	(*n* ＋ 1)＋row	true
	row ＜-*n*	先頭行の前	false

注※

setMaxRows の値より実際の行数の方が多い場合は，setMaxRows の値になります。

setLargeMaxRows の値より実際の行数の方が多い場合は，setLargeMaxRows の値になります。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- このResultSet オブジェクトの型がResultSet.TYPE_FORWARD_ONLY の場合
- row に0 を指定した場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.3 afterLast()

(1) 機能

ResultSet オブジェクト内の最終行の後ろにカーソルを移動します。このメソッド実行時のカーソルの移動先を次の表に示します。

表 8-21 afterLast メソッド実行時のカーソルの移動先

結果集合の行数※	現在のカーソルの位置	カーソルの移動先
0	先頭行の前	先頭行の前のまま
n	先頭行の前	最終行の後ろ
	1≦現在の行≦*n*	最終行の後ろ
	最終行の後ろ	最終行の後ろのまま

注※

setMaxRows の値より実際の行数の方が多い場合は，setMaxRows の値になります。

setLargeMaxRows の値より実際の行数の方が多い場合は，setLargeMaxRows の値になります。

(2) 形式

```
public synchronized void afterLast() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- このResultSet オブジェクトの型がResultSet.TYPE_FORWARD_ONLY の場合

- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.4 beforeFirst()

(1) 機能

ResultSet オブジェクト内の先頭行の前にカーソルを移動します。このメソッド実行時のカーソルの移動先を次の表に示します。

表 8-22 beforeFirst メソッド実行時のカーソルの移動先

結果集合の行数※	現在のカーソルの位置	カーソルの移動先
0	先頭行の前	先頭行の前のまま
n	先頭行の前	先頭行の前のまま
	1≦現在の行≦*n*	先頭行の前
	最終行の後ろ	先頭行の前

注※

setMaxRows の値より実際の行数の方が多い場合は，setMaxRows の値になります。

setLargeMaxRows の値より実際の行数の方が多い場合は，setLargeMaxRows の値になります。

(2) 形式

```
public synchronized void beforeFirst() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。

- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- このResultSet オブジェクトの型がResultSet.TYPE_FORWARD_ONLY の場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.5 clearWarnings()

(1) 機能

このResultSet オブジェクトに関して報告された，すべての警告をクリアします。

(2) 形式

```
public synchronized void clearWarnings() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

トランザクションの決着によってResultSet オブジェクトが無効になった場合，SQLException が投入されます。

8.6.6 close()

(1) 機能

ResultSet オブジェクトでオープンしていたカーソルをクローズし，JDBC リソースを解放します。

(2) 形式

```
public synchronized void close() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

データベースのアクセスエラーが発生した場合，SQLException が投入されます。

8.6.7 findColumn(String columnName)

(1) 機能

指定した列名の列番号を返します。

(2) 形式

```
public synchronized int findColumn(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。この列名に対応する列番号が返却されます。

列名指定時の注意事項を次に示します。

- 大文字と小文字を区別しません。
- 指定した文字列すべてを列名として扱うため，文字列に二重引用符（"）が含まれている場合，二重引用符（"）も列名の一部として扱われます。
- 指定した列名が複数ある場合，列番号が小さい列が優先されます。

(4) 戻り値

指定した列名に対応する列番号が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。

- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 指定した列がない場合
- columnName にnull を指定するか，または長さ 0 の文字列を指定した場合

8.6.8 first()

(1) 機能

ResultSet オブジェクト内の先頭行にカーソルを移動します。

(2) 形式

```
public synchronized boolean first() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

結果集合の行数が 0 の場合はfalse が，そうでない場合はtrue が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- このResultSet オブジェクトの型がResultSet.TYPE_FORWARD_ONLY の場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.9 getArray(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値をArray オブジェクトに取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized Array getArray(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

検索結果を格納したArray オブジェクトが返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-23 検索結果と戻り値の関係（getArray メソッドの場合）

HADB のデータ型	検索結果	戻り値
ARRAY	ナル値	null
	上記以外	検索結果を格納したArray オブジェクト
そのほか	該当しません	SQLException が投入される

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.10　getArray(String columnName)

(1)　機能

ResultSet オブジェクトの現在行の列の値をArray オブジェクトに取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized Array getArray(String columnName) throws SQLException
```

(3)　引数

String columnName：

　列名を指定します。

(4)　戻り値

検索結果を格納したArray オブジェクトが返却されます。検索結果と戻り値の関係については，「表 8-23 検索結果と戻り値の関係（getArray メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.11　getAsciiStream(int columnIndex)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.io.InputStream オブジェクトに取得します。値を取得する列は，引数で指定します。

なお，ASCII 文字への変換は行いません。

(2)　形式

```
public synchronized InputStream getAsciiStream(int columnIndex) throws SQLException
```

(3)　引数

int columnIndex：

　列番号を指定します。

(4)　戻り値

検索結果を格納したjava.io.InputStream オブジェクトが返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-24　検索結果と戻り値の関係（getAsciiStream メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR BINARY VARBINARY	ナル値	null
	上記以外	検索結果を格納したjava.io.InputStream オブジェクト
そのほか	該当しません	SQLException が投入される

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.12 getAsciiStream(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値をjava.io.InputStream オブジェクトに取得します。値を取得する列は，引数で指定します。

なお，ASCII 文字への変換は行いません。

(2) 形式

```
public synchronized InputStream getAsciiStream(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

検索結果を格納したjava.io.InputStream オブジェクトが返却されます。検索結果と戻り値の関係については，「表 8-24 検索結果と戻り値の関係（getAsciiStream メソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.13　getBigDecimal(int columnIndex)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.math.BigDecimal オブジェクトに取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized BigDecimal getBigDecimal(int columnIndex) throws SQLException
```

(3)　引数

int columnIndex：

　列番号を指定します。

(4)　戻り値

指定された列番号に対応する列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-25　検索結果と戻り値の関係（getBigDecimal メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	null
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]	検索結果を持つjava.math.BigDecimal オブジェクト（文字列の前後の半角空白を取り除いた値をjava.math.BigDecimal オブジェクトにします）
	上記以外	SQLException が投入される
SMALLINT	ナル値	null
	上記以外	検索結果を持つjava.math.BigDecimal オブジェクト
INTEGER	ナル値	null
	上記以外	検索結果を持つjava.math.BigDecimal オブジェクト
DECIMAL NUMERIC	ナル値	null
	上記以外	検索結果を持つjava.math.BigDecimal オブジェクト
DOUBLE PRECISION FLOAT	ナル値	null
	上記以外	検索結果を持つjava.math.BigDecimal オブジェクト
BOOLEAN※	ナル値	null
	true	BigDecimal(1)でjava.math.BigDecimal オブジェクトにしたもの

HADB のデータ型	検索結果	戻り値
	false	BigDecimal(0)でjava.math.BigDecimal オブジェクトにしたもの
そのほか	該当しません	SQLException が投入される

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がBigDecimal として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.14 getBigDecimal(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値をjava.math.BigDecimal オブジェクトに取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized BigDecimal getBigDecimal(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

指定された列名に対応する列値が返却されます。

検索結果と戻り値の関係については，「表 8-25 検索結果と戻り値の関係（getBigDecimal メソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がBigDecimal として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.15 getBinaryStream(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値をjava.io.InputStream オブジェクトに取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized InputStream getBinaryStream(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

検索結果を格納したjava.io.InputStream オブジェクトが返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-26 検索結果と戻り値の関係（getBinaryStream メソッドの場合）

HADB のデータ型	検索結果	戻り値
BINARY VARBINARY	ナル値	null
	上記以外	検索結果を格納したjava.io.InputStream オブジェクト
そのほか	該当しません	SQLException が投入される

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.16 getBinaryStream(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値をjava.io.InputStream オブジェクトに取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized InputStream getBinaryStream(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

検索結果を格納した java.io.InputStream オブジェクトが返却されます。検索結果と戻り値の関係については，「表 8-26 検索結果と戻り値の関係（getBinaryStream メソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.17 getBoolean(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のboolean で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized boolean getBoolean(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

trueまたはfalseが返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-27 検索結果と戻り値の関係（getBoolean メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	false
	[*半角空白*]1[*半角空白*]	true
	上記以外	false
SMALLINT	ナル値	false
	0	false
	上記以外	true
INTEGER	ナル値	false
	0	false
	上記以外	true
DECIMAL NUMERIC	ナル値	false
	0[.00…0]	false
	上記以外	true
DOUBLE PRECISION FLOAT	ナル値	false
	0.0 または-0.0	false
	上記以外	true
BOOLEAN※	ナル値	false
	ナル値以外	*検索結果*
そのほか	該当しません	SQLException が投入される

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。

- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.18　getBoolean(String columnName)

(1)　機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のboolean で取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized boolean getBoolean(String columnName) throws SQLException
```

(3)　引数

String columnName：

　列名を指定します。

(4)　戻り値

true またはfalse が返却されます。検索結果と戻り値の関係については，「表 8-27　検索結果と戻り値の関係（getBoolean メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合

- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.19 getByte(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のbyte で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized byte getByte(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-28 検索結果と戻り値の関係（getByte メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	0
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]であり，かつByte.MIN_VALUE 以上で，Byte.MAX_VALUE 以下	検索結果をbyte 値にしたもの
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現，または浮動小数点数の文字列表現*[*半角空白*]であり，かつByte.MAX_VALUE より大きいか，Byte.MIN_VALUE より小さい	SQLException が投入される
	[*半角空白*][+]Infinity[*半角空白*]	
	[*半角空白*]-Infinity[*半角空白*]	
	[*半角空白*][+|-]NaN[*半角空白*]	
	上記以外	
SMALLINT	ナル値	0

HADB のデータ型	検索結果	戻り値
	Byte.MIN_VALUE 以上かつByte.MAX_VALUE 以下	検索結果をbyte 値にしたもの
	上記以外	SQLException が投入される
INTEGER	ナル値	0
	Byte.MIN_VALUE 以上かつByte.MAX_VALUE 以下	検索結果をbyte 値にしたもの
	上記以外	SQLException が投入される
DECIMAL NUMERIC	ナル値	0
	Byte.MIN_VALUE 以上かつByte.MAX_VALUE 以下	検索結果の整数部分の値をbyte 値にしたもの
	上記以外	SQLException が投入される
DOUBLE PRECISION FLOAT	ナル値	0
	Byte.MIN_VALUE 以上かつByte.MAX_VALUE 以下	検索結果の整数部分の値をbyte 値にしたもの
	上記以外	SQLException が投入される
BOOLEAN※	ナル値	0
	true	1
	false	0
そのほか	該当しません	SQLException が投入される

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がbyte として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.20　getByte(String columnName)

(1)　機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のbyte で取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized byte getByte(String columnName) throws SQLException
```

(3)　引数

String columnName：

　列名を指定します。

(4)　戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-28　検索結果と戻り値の関係（getByte メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がbyte として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.21　getBytes(int columnIndex)

(1)　機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のbyte 配列で取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized byte[] getBytes(int columnIndex) throws SQLException
```

(3)　引数

int columnIndex：

　列番号を指定します。

(4)　戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-29　検索結果と戻り値の関係（getBytes メソッドの場合）

HADB のデータ型	検索結果	戻り値
BINARY VARBINARY	ナル値	null
	上記以外	検索結果をByte 配列にしたもの
ROW	検索結果がナル値になることはありません	検索結果をByte 配列にしたもの
そのほか	該当しません	SQLException が投入される

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合

- JDBC ドライバ内でエラーが発生した場合

8.6.22 getBytes(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のbyte 配列で取得します。値を取得する列は，引数で指定します。

バイトは JDBC ドライバによって返された行の値を表します。

(2) 形式

```
public synchronized byte[] getBytes(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-29 検索結果と戻り値の関係（getBytes メソッドの場合)」を参照してください。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.23　getCharacterStream(int columnIndex)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.io.Reader オブジェクトに取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized Reader getCharacterStream(int columnIndex) throws SQLException
```

(3)　引数

int columnIndex：

　列番号を指定します。

(4)　戻り値

列値を格納したjava.io.Reader オブジェクトが返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-30　検索結果と戻り値の関係（getCharacterStream メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR BINARY VARBINARY	ナル値	null
	上記以外	検索結果を格納したjava.io.Reader オブジェクト
そのほか	該当しません	SQLException が投入される

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合

- このメソッドでは取得できないデータ型の場合
- エンコードに失敗した場合
- JDBC ドライバ内でエラーが発生した場合

8.6.24 getCharacterStream(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値をjava.io.Reader オブジェクトに取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized Reader getCharacterStream(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

列値を格納したjava.io.Reader オブジェクトが返却されます。検索結果と戻り値の関係については，「表 8-30 検索結果と戻り値の関係（getCharacterStream メソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- エンコードに失敗した場合
- JDBC ドライバ内でエラーが発生した場合

8.6.25 getConcurrency()

(1) 機能

このResultSet オブジェクトの並行処理モードを取得します。

(2) 形式

```
public synchronized int getConcurrency() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にResultSet.CONCUR_READ_ONLY が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.26 getCursorName()

(1) 機能

このResultSet オブジェクトが使用する SQL カーソルの名前を取得します。

(2) 形式

```
public synchronized String getCursorName() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に空の文字列が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.27 getDate(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値をjava.sql.Date オブジェクトに取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized java.sql.Date getDate(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：
　列番号を指定します。

(4) 戻り値

列値を格納したjava.sql.Date オブジェクトが返却されます。検索結果と戻り値の関係を次の表に示します。

DATE 型，TIME 型，TIMESTAMP 型，または文字列型（CHAR，VARCHAR）の変換については，「7.6.2　データの変換処理」の「(2)　getXXX メソッド実行時のデータ変換処理（DATE 型，TIME 型，TIMESTAMP 型，または文字列型の場合)」を参照してください。

表 8-31　検索結果と戻り値の関係（getDate メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	null
	[*半角空白*]*日付形式*※[*半角空白*]	検索結果の前後の半角空白を取り除きjava.sql.Date オブジェクトにしたもの
	上記以外	SQLException が投入される
DATE	ナル値	null
	上記以外	検索結果をjava.sql.Date オブジェクトにしたもの
TIME	ナル値	null
	上記以外	検索結果をjava.sql.Date オブジェクトにしたもの
TIMESTAMP	ナル値	null
	上記以外	検索結果をjava.sql.Date オブジェクトにしたもの
そのほか	該当しません	SQLException が投入される

注※

*日付形式*とは，'*YYYY-MM-DD*'で表される文字列表現のことです。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Date として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.28　getDate(int columnIndex, Calendar cal)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.sql.Date オブジェクトに取得します。値を取得する列は，引数で指定します。

このメソッドは，指定されたカレンダーを使って日付に適切なミリ秒値を作成します。

(2)　形式

```
public synchronized java.sql.Date getDate(int columnIndex, Calendar cal) throws SQLException
```

(3)　引数

int columnIndex：

　列番号を指定します。

Calendar cal：

　データベースに格納されている値のタイムゾーンが設定されたカレンダーを指定します。

(4)　戻り値

列値を格納したjava.sql.Date オブジェクトが返却されます。検索結果と戻り値の関係については，「表 8-31　検索結果と戻り値の関係（getDate メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Date として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.29　getDate(String columnName)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.sql.Date オブジェクトに取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized java.sql.Date getDate(String columnName) throws SQLException
```

(3)　引数

String columnName：

　列名を指定します。

(4)　戻り値

列値を格納したjava.sql.Date オブジェクトが返却されます。検索結果と戻り値の関係については，「表 8-31　検索結果と戻り値の関係（getDate メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Date として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.30　getDate(String columnName, Calendar cal)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.sql.Date オブジェクトに取得します。値を取得する列は，引数で指定します。

このメソッドは，指定されたカレンダーを使って日付に適切なミリ秒値を作成します。

(2)　形式

```
public synchronized java.sql.Date getDate(String columnName, Calendar cal) throws SQLExcepti
on
```

(3)　引数

String columnName：

　列名を指定します。

Calendar cal：

　データベースに格納されている値のタイムゾーンが設定されたカレンダーを指定します。

(4)　戻り値

列値を格納したjava.sql.Date オブジェクトが返却されます。検索結果と戻り値の関係については，「表 8-31　検索結果と戻り値の関係（getDate メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Date として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.31　getDouble(int columnIndex)

(1)　機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のdouble で取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized double getDouble(int columnIndex) throws SQLException
```

(3)　引数

int columnIndex：

　列番号を指定します。

(4)　戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-32　検索結果と戻り値の関係（getDouble メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	0.0
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]であり，かつ-Double.MAX_VALUE 以上，かつDouble.MIN_VALUE 以下，かつDouble.MIN_VALUE 以上かつDouble.MAX_VALUE 以下	検索結果をdouble 値にしたもの
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]かつDouble.MAX_VALUE より大きい	Infinity
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]かつ-Double.MAX_VALUE より小さい	-Infinity
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]かつDouble.MIN_VALUE より小さく 0 より大きい	0.0
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]かつ-Double.MIN_VALUE より大きく 0 より小さい	-0.0
	[*半角空白*]-Infinity[*半角空白*]	-Infinity
	[*半角空白*][+]Infinity[*半角空白*]	Infinity

HADB のデータ型	検索結果	戻り値
	[*半角空白*][+|-]NaN[*半角空白*]	NaN
	上記以外（double 値にできない）	SQLException が投入される
SMALLINT	ナル値	0.0
	上記以外	検索結果をdouble 値にしたもの
INTEGER	ナル値	0.0
	上記以外	検索結果をdouble 値にしたもの
DECIMAL NUMERIC	ナル値	0.0
	上記以外	検索結果をdouble 値にしたもの
DOUBLE PRECISION FLOAT	ナル値	0.0
	上記以外	検索結果をdouble 値にしたもの
BOOLEAN※	ナル値	0.0
	true	1.0
	false	0.0
そのほか	該当しません	SQLException が投入される

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がdouble として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.32　getDouble(String columnName)

(1)　機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のdouble で取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized double getDouble(String columnName) throws SQLException
```

(3)　引数

String columnName：

　列名を指定します。

(4)　戻り値

列値が返却されます。

検索結果と戻り値の関係については，「表 8-32　検索結果と戻り値の関係（getDouble メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がdouble として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.33 getFetchDirection()

(1) 機能

このResultSet オブジェクトのフェッチ方向を取得します。

(2) 形式

```
public synchronized int getFetchDirection() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にResultSet.FETCH_FORWARD が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.34 getFetchSize()

(1) 機能

ResultSet オブジェクトのフェッチサイズを取得します。

(2) 形式

```
public synchronized int getFetchSize() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このResultSet オブジェクトの現在のフェッチサイズが返却されます。setFetchSize メソッドで設定した値が返却されます。setFetchSize メソッドで値を設定していない場合は0 が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.35 getFloat(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のfloat で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized float getFloat(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-33　検索結果と戻り値の関係（getFloat メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	0.0
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]であり，かつ次のどちらか • -Float.MAX_VALUE 以上-Float.MIN_VALUE 以下 • Float.MIN_VALUE 以上Float.MAX_VALUE 以下	検索結果をfloat 値にしたもの
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]かつFloat.MAX_VALUE より大きい	Infinity
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]かつ-Float.MAX_VALUE より小さい	-Infinity
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]かつFloat.MIN_VALUE より小さく 0 より大きい	0.0
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]かつ-Float.MIN_VALUE より大きく 0 より小さい	-0.0
	[*半角空白*]-Infinity[*半角空白*]	-Infinity
	[*半角空白*][+]Infinity[*半角空白*]	Infinity
	[*半角空白*][+|-]NaN[*半角空白*]	NaN
	上記以外（float 値にできない）	SQLException が投入される
SMALLINT	ナル値	0.0
	上記以外	検索結果をfloat 値にしたもの
INTEGER	ナル値	0.0
	上記以外	検索結果をfloat 値にしたもの
DECIMAL NUMERIC	ナル値	0.0
	上記以外	検索結果をfloat 値にしたもの
DOUBLE PRECISION FLOAT	ナル値	0.0
	次のどちらか • -Float.MAX_VALUE 以上-Float.MIN_VALUE 以下 • Float.MIN_VALUE 以上Float.MAX_VALUE 以下	検索結果をfloat 値にしたもの
	Float.MAX_VALUE より大きい	Infinity
	-Float.MAX_VALUE より小さい	-Infinity
	Float.MIN_VALUE より小さく 0 より大きい	0.0

HADB のデータ型	検索結果	戻り値
	-Float.MIN_VALUE より大きく 0 より小さい	-0.0
BOOLEAN※	ナル値	0.0
	true	1.0
	false	0.0
そのほか	該当しません	SQLException が投入される

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がfloat として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.36 getFloat(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のfloat で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized float getFloat(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

列値が返却されます。

検索結果と戻り値の関係については，「表 8-33 検索結果と戻り値の関係（getFloat メソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がfloat として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.37 getHoldability()

(1) 機能

このResultSet オブジェクトの保持機能の状態を表す値を取得します。

(2) 形式

```
public synchronized int getHoldability() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にResultSet.HOLD_CURSORS_OVER_COMMIT が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
- このResultSet オブジェクトを生成したStatement オブジェクトがクローズされている場合
- このResultSet オブジェクトを生成したStatement オブジェクトを生成したConnection オブジェクトがクローズされている場合

8.6.38 getInt(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のint で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized int getInt(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-34 検索結果と戻り値の関係（getInt メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	0
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]であり，かつ Integer.MIN_VALUE 以上，Integer.MAX_VALUE 以下	検索結果の整数部分の値をint 値にしたもの

HADB のデータ型	検索結果	戻り値
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]であり，かつInteger.MAX_VALUE より大きいかInteger.MIN_VALUE より小さい	SQLException が投入される
	[*半角空白*]-Infinity[*半角空白*]	
	[*半角空白*][+]Infinity[*半角空白*]	
	[*半角空白*][+|-]NaN[*半角空白*]	
	上記以外（double 値にできない）	
SMALLINT	ナル値	0
	上記以外	検索結果をint 値にしたもの
INTEGER	ナル値	0
	Integer.MIN_VALUE 以上かつInteger.MAX_VALUE 以下	検索結果をint 値にしたもの
	上記以外	SQLException が投入される
DECIMAL NUMERIC	ナル値	0
	Integer.MIN_VALUE 以上かつInteger.MAX_VALUE 以下	検索結果の整数部分の値をint 値にしたもの
	上記以外	SQLException が投入される
DOUBLE PRECISION FLOAT	ナル値	0
	Integer.MIN_VALUE 以上かつInteger.MAX_VALUE 以下	検索結果の整数部分の値をint 値にしたもの
	上記以外	SQLException が投入される
BOOLEAN※	ナル値	0
	true	1
	false	0
そのほか	該当しません	SQLException が投入される

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。

- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がint として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.39　getInt(String columnName)

(1)　機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のint で取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized int getInt(String columnName) throws SQLException
```

(3)　引数

String columnName：

　列名を指定します。

(4)　戻り値

列値が返却されます。

検索結果と戻り値の関係については，「表 8-34　検索結果と戻り値の関係（getInt メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合

- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がint として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.40 getLong(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のlong で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized long getLong(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-35 検索結果と戻り値の関係（getLong メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	0
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点の文字列表現*[*半角空白*]であり，かつ Long.MIN_VALUE 以上，Long.MAX_VALUE 以下	検索結果の整数部分の値をlong 値にしたもの
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点の文字列表現*[*半角空白*]であり，かつ Long.MAX_VALUE より大きいかLong.MIN_VALUE より小さい	SQLException が投入される
	[*半角空白*]-Infinity[*半角空白*]	
	[*半角空白*][+]Infinity[*半角空白*]	
	[*半角空白*][+|-]NaN[*半角空白*]	

HADB のデータ型	検索結果	戻り値
	上記以外（double 値，またはBigDecimal オブジェクトにできない）	
SMALLINT	ナル値	0
	上記以外	検索結果をlong 値にしたもの
INTEGER	ナル値	0
	上記以外	検索結果をlong 値にしたもの
DECIMAL NUMERIC	ナル値	0
	Long.MIN_VALUE 以上かつLong.MAX_VALUE 以下	検索結果の整数部分の値をlong 値にしたもの
DOUBLE PRECISION FLOAT	ナル値	0
	Long.MIN_VALUE 以上かつLong.MAX_VALUE 以下	検索結果の整数部分の値をlong 値にしたもの
	上記以外	SQLException が投入される
BOOLEAN※	ナル値	0
	true	1
	false	0
そのほか	該当しません	SQLException が投入される

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がlong として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.41　getLong(String columnName)

(1)　機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のlong で取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized long getLong(String columnName) throws SQLException
```

(3)　引数

String columnName：

　列名を指定します。

(4)　戻り値

列値が返却されます。

検索結果と戻り値の関係については，「表 8-35　検索結果と戻り値の関係（getLong メソッドの場合）」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がlong として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.42 getMetaData()

(1) 機能

このResultSet オブジェクトのメタ情報を取得します。

(2) 形式

```
public synchronized ResultSetMetaData getMetaData() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このResultSet オブジェクトのメタ情報が，ResultSetMetaData オブジェクトに格納されて返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.43 getObject(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のObject で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized Object getObject(int columnIndex) throws SQLException
```

(3) 引数

`int columnIndex`：

列番号を指定します。

(4) 戻り値

列値が Java オブジェクトとして返却されます。

Java オブジェクトの型は，JDBC 仕様で指定されている組み込み型のマッピングに従って，列の SQL 型に対応するデフォルトの Java オブジェクトの型になります。

検索結果と戻り値の関係を次の表に示します。

表 8-36　検索結果と戻り値の関係（getObject メソッドの場合）

HADB のデータ型	検索結果	戻り値
ARRAY	ナル値	null
	上記以外	検索結果を格納したArray オブジェクト
CHAR VARCHAR	ナル値	null
	上記以外	*検索結果*
SMALLINT	ナル値	null
	上記以外	検索結果で生成したInteger オブジェクト
INTEGER	ナル値	null
	上記以外	検索結果で生成したLong オブジェクト
DECIMAL NUMERIC	ナル値	null
	上記以外	*検索結果*
DOUBLE PRECISION FLOAT	ナル値	null
	上記以外	検索結果で生成したDouble オブジェクト
DATE	ナル値	null
	上記以外	検索結果で生成したjava.sql.Date オブジェクト
TIME	ナル値	null
	上記以外	検索結果で生成したjava.sql.Time オブジェクト
TIMESTAMP	ナル値	null
	上記以外	検索結果で生成したjava.sql.Timestamp オブジェクト
BINARY VARBINARY	ナル値	null
	上記以外	*検索結果*

HADB のデータ型	検索結果	戻り値
ROW	ナル値以外※1	*検索結果*
BOOLEAN※2	ナル値	null
	ナル値以外	検索結果で生成したBoolean オブジェクト

注※1

検索結果がナル値になることはありません。

注※2

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- JDBC ドライバ内でエラーが発生した場合

8.6.44 getObject(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のObject で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized Object getObject(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

列値がJavaオブジェクトとして返却されます。検索結果と戻り値の関係については，「表 8-36 検索結果と戻り値の関係（getObject メソッドの場合）」を参照してください。

Javaオブジェクトの型は，JDBC仕様で指定されている組み込み型のマッピングに従って，列のSQL型に対応するデフォルトのJavaオブジェクトの型になります。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- このResultSetオブジェクトがクローズされている場合

  このResultSetオブジェクトを生成したStatementオブジェクトがクローズされたことによって，ResultSetオブジェクトがクローズされた場合も含みます。
- このResultSetオブジェクトを生成したStatementオブジェクトを作成したConnectionがクローズされている場合
- トランザクションの決着によってResultSetオブジェクトが無効になった場合
- 存在しない列名を指定した場合
- JDBCドライバ内でエラーが発生した場合

8.6.45 getObject(int columnIndex,Class<T> type)

(1) 機能

ResultSetオブジェクトの現在行の列の値を取得し，指定されたクラスのJavaデータ型に変換します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized <T> T getObject(int columnIndex,Class<T> type) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

Class<T> type：

変換後のJavaデータ型を表すクラスを指定します。columnIndexで指定した列の値が，指定されたクラスのJavaデータ型に変換されます。

変換の組み合わせを次の表に示します。次の表に示す組み合わせ以外を指定した場合は，エラーになります。

表 8-37　HADB のデータ型と Java データ型の組み合わせ

HADB のデータ型	Java データ型(<T>の指定値)
ARRAY	Array
CHAR	String
VARCHAR	
SMALLINT	Integer
INTEGER	Long
DECIMAL	java.math.BigDecimal
NUMERIC	
DOUBLE PRECISION	Double
FLOAT	
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp
BINARY	byte[]
VARBINARY	
ROW	
BOOLEAN※	Boolean

注※

DatabaseMetadata から生成されたResultset オブジェクトの場合は，BOOLEAN 型データが存在します。

(4)　戻り値

列の値が，「指定されたクラスのオブジェクト」として返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-38　検索結果と戻り値の関係（getObject メソッドの場合）

HADB のデータ型	検索結果	戻り値
ARRAY	ナル値	null
	上記以外	検索結果を格納したArray オブジェクト
CHAR VARCHAR	ナル値	null
	上記以外	検索結果で生成したString オブジェクト

HADB のデータ型	検索結果	戻り値
SMALLINT	ナル値	null
	上記以外	検索結果で生成したInteger オブジェクト
INTEGER	ナル値	null
	上記以外	検索結果で生成したLong オブジェクト
DECIMAL NUMERIC	ナル値	null
	上記以外	検索結果で生成したjava.math.BigDecimal オブジェクト
DOUBLE PRECISION FLOAT	ナル値	null
	上記以外	検索結果で生成したDouble オブジェクト
DATE	ナル値	null
	上記以外	検索結果で生成したjava.sql.Date オブジェクト
TIME	ナル値	null
	上記以外	検索結果で生成したjava.sql.Time オブジェクト
TIMESTAMP	ナル値	null
	上記以外	検索結果で生成したjava.sql.Timestamp オブジェクト
BINARY VARBINARY	ナル値	null
	上記以外	*検索結果*
ROW	ナル値以外※1	*検索結果*
BOOLEAN※2	ナル値	null
	上記以外	検索結果で生成したBoolean オブジェクト

注※1

検索結果がナル値になることはありません。

注※2

DatabaseMetadata から生成されたResultset オブジェクトの場合は，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。

- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合

- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- JDBC ドライバ内でエラーが発生した場合
- type にnull を指定した場合
- type に指定できる組み合わせ以外の値を指定した場合

8.6.46　getObject(String columnLabel,Class<T> type)

(1)　機能

ResultSet オブジェクトの現在行の列の値を取得し，指定されたクラスの Java データ型に変換します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized <T> T getObject(String columnLabel,Class<T> type) throws SQLException
```

(3)　引数

String columnLabel：

　列名を指定します。

Class<T> type：

　変換後の Java データ型を表すクラスを指定します。columnLabel で指定した列の値が，指定されたクラスの Java データ型に変換されます。

　変換の組み合わせを「表 8-37　HADB のデータ型と Java データ型の組み合わせ」に示します。「表 8-37　HADB のデータ型と Java データ型の組み合わせ」に示す組み合わせ以外を指定した場合は，エラーになります。

(4)　戻り値

列の値が，「指定されたクラスのオブジェクト」として返却されます。検索結果と戻り値の関係については，「表 8-38　検索結果と戻り値の関係（getObject メソッドの場合）」を参照してください。

(5)　発生する例外

発生する例外については，「8.6.45　getObject(int columnIndex,Class<T> type)」の「(5)　発生する例外」を参照してください。

8.6.47　getRow()

(1)　機能

現在の行の番号を取得します。最初の行は1 になり，2 番目は2 になります。先頭行の前または最終行の後ろの場合は0 になります。

(2)　形式

```
public synchronized int getRow() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

現在の行番号が返却されます。現在の行がInteger.MAX_VALUE より大きい場合は，Integer.MAX_VALUE が返却されます。

また，最大検索行数が 2,147,483,647 を超える場合は，2,147,483,647 が返却されます。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.48　getShort(int columnIndex)

(1)　機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のshort で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized short getShort(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-39 検索結果と戻り値の関係（getShort メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	0
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]であり，かつ Short.MIN_VALUE 以上，Short.MAX_VALUE 以下	検索結果の整数部分の値をshort 値にしたもの
	[*半角空白*]*整数の文字列表現，10 進数の文字列表現*，または*浮動小数点数の文字列表現*[*半角空白*]であり，かつ Short.MAX_VALUE より大きいかShort.MIN_VALUE より小さい	SQLException が投入される
	[*半角空白*]-Infinity[*半角空白*]	
	[*半角空白*][+]Infinity[*半角空白*]	
	[*半角空白*][+|-]NaN[*半角空白*]	
	上記以外（double 値にできない）	
SMALLINT	ナル値	0
	Short.MIN_VALUE 以上かつShort.MAX_VALUE 以下	検索結果をshort 値にしたもの
	上記以外	SQLException が投入される
INTEGER	ナル値	0
	Short.MIN_VALUE 以上かつShort.MAX_VALUE 以下	検索結果をshort 値にしたもの
	上記以外	SQLException が投入される
DECIMAL NUMERIC	ナル値	0
	Short.MIN_VALUE 以上かつShort.MAX_VALUE 以下	検索結果の整数部分の値をshort 値にしたもの
	上記以外	SQLException が投入される
DOUBLE PRECISION	ナル値	0

HADB のデータ型	検索結果	戻り値
FLOAT	Short.MIN_VALUE 以上かつShort.MAX_VALUE 以下	検索結果の整数部分の値をshort 値にしたもの
	上記以外	SQLException が投入される
BOOLEAN※	ナル値	0
	true	1
	false	0
そのほか	該当しません	SQLException が投入される

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がshort として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.49 getShort(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のshort で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized short getShort(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-39 検索結果と戻り値の関係（getShortメソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がshort として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.50 getStatement()

(1) 機能

このResultSet オブジェクトを生成したStatement オブジェクトを取得します。

(2) 形式

```
public synchronized Statement getStatement() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

ResultSet オブジェクトを生成したStatement オブジェクトが返却されます。結果集合がDatabaseMetaData インタフェースのメソッドで生成された場合は，null が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.51 getString(int columnIndex)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のString で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized String getString(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

表 8-40 検索結果と戻り値の関係（getString メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	null
	上記以外	*検索結果*

HADB のデータ型	検索結果	戻り値
SMALLINT	ナル値	null
	上記以外	検索結果を文字列表現にしたString オブジェクト
INTEGER	ナル値	null
	上記以外	検索結果を文字列表現にしたString オブジェクト
DECIMAL NUMERIC	ナル値	null
	上記以外	検索結果を文字列表現にしたString オブジェクト
DOUBLE PRECISION FLOAT	ナル値	null
	上記以外	検索結果を文字列表現にしたString オブジェクト
DATE	ナル値	null
	上記以外	*YYYY-MM-DD* 形式の文字列のString オブジェクト
TIME	ナル値	null
	上記以外	*hh*:*mm*:*ss*[.*f*...]形式の文字列のString オブジェクト
TIMESTAMP	ナル値	null
	上記以外	*YYYY-MM-DD*△*hh*:*mm*:*ss*[.*f*...]形式（△は半角空白）の文字列のString オブジェクト
BINARY VARBINARY	ナル値	null
	上記以外	*検索結果*
BOOLEAN※	ナル値	null
	true	文字列"true"のString オブジェクト
	false	文字列"false"のString オブジェクト

注※

DatabaseMetadata から生成したResultset オブジェクトの場合，BOOLEAN 型データが存在します。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

- 存在しない列番号を指定した場合
- エンコードに失敗した場合
- このメソッドでは取得できないデータ型の場合
- JDBC ドライバ内でエラーが発生した場合

8.6.52 getString(String columnName)

(1) 機能

ResultSet オブジェクトの現在行の列の値を，Java プログラミング言語のString で取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized String getString(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-40 検索結果と戻り値の関係（getString メソッドの場合)」を参照してください。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- エンコードに失敗した場合
- このメソッドでは取得できないデータ型の場合

- JDBC ドライバ内でエラーが発生した場合

8.6.53　getTime(int columnIndex)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.sql.Time オブジェクトに取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized java.sql.Time getTime(int columnIndex) throws SQLException
```

(3)　引数

int columnIndex：

　列番号を指定します。

(4)　戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

DATE 型，TIME 型，TIMESTAMP 型，または文字列型（CHAR，VARCHAR）の変換については，「7.6.2　データの変換処理」の「(2)　getXXX メソッド実行時のデータ変換処理（DATE 型，TIME 型，TIMESTAMP 型，または文字列型の場合）」を参照してください。

表 8-41　検索結果と戻り値の関係（getTime メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	null
	[*半角空白*]*時刻形式*※[*半角空白*]	検索結果の前後の半角空白を取り除きjava.sql.Time オブジェクトにしたもの
	上記以外	SQLException が投入される
TIME	ナル値	null
	上記以外	検索結果をjava.sql.Time オブジェクトにしたもの
TIMESTAMP	ナル値	null
	上記以外	検索結果をjava.sql.Time オブジェクトにしたもの
そのほか	該当しません	SQLException が投入される

注※

*時刻形式*とは，'*hh*:*mm*:*ss*[.*f*...]'で表される文字列表現のことです。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- このResultSetオブジェクトがクローズされている場合

  このResultSetオブジェクトを生成したStatementオブジェクトがクローズされたことによって，ResultSetオブジェクトがクローズされた場合も含みます。
- このResultSetオブジェクトを生成したStatementオブジェクトを作成したConnectionがクローズされている場合
- トランザクションの決着によってResultSetオブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Timeとして取得できない場合
- JDBCドライバ内でエラーが発生した場合

8.6.54 getTime(int columnIndex, Calendar cal)

(1) 機能

ResultSetオブジェクトの現在行の列の値をjava.sql.Timeオブジェクトに取得します。値を取得する列は，引数で指定します。

このメソッドは，指定されたカレンダーを使って時刻に適切なミリ秒値を作成します。

(2) 形式

```
public synchronized java.sql.Time getTime(int columnIndex, Calendar cal) throws SQLException
```

(3) 引数

int columnIndex：

列番号を指定します。

Calendar cal：

データベースに格納されている値のタイムゾーンが設定されたカレンダーを指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-41 検索結果と戻り値の関係（getTimeメソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- このResultSetオブジェクトがクローズされている場合

  このResultSetオブジェクトを生成したStatementオブジェクトがクローズされたことによって，ResultSetオブジェクトがクローズされた場合も含みます。
- このResultSetオブジェクトを生成したStatementオブジェクトを作成したConnectionがクローズされている場合
- トランザクションの決着によってResultSetオブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Timeとして取得できない場合
- JDBCドライバ内でエラーが発生した場合

8.6.55 getTime(String columnName)

(1) 機能

ResultSetオブジェクトの現在行の列の値をjava.sql.Timeオブジェクトに取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized java.sql.Time getTime(String columnName) throws SQLException
```

(3) 引数

String columnName：

列名を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-41 検索結果と戻り値の関係（getTimeメソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- このResultSetオブジェクトがクローズされている場合
  このResultSetオブジェクトを生成したStatementオブジェクトがクローズされたことによって，ResultSetオブジェクトがクローズされた場合も含みます。
- このResultSetオブジェクトを生成したStatementオブジェクトを作成したConnectionがクローズされている場合
- トランザクションの決着によってResultSetオブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Timeとして取得できない場合
- JDBCドライバ内でエラーが発生した場合

8.6.56 getTime(String columnName, Calendar cal)

(1) 機能

ResultSetオブジェクトの現在行の列の値をjava.sql.Timeオブジェクトに取得します。値を取得する列は，引数で指定します。

このメソッドは，指定されたカレンダーを使って時刻に適切なミリ秒値を作成します。

(2) 形式

```
public synchronized java.sql.Time getTime(String columnName, Calendar cal) throws SQLExcepti
on
```

(3) 引数

String columnName：

列名を指定します。

Calendar cal：

データベースに格納されている値のタイムゾーンが設定されたカレンダーを指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-41 検索結果と戻り値の関係（getTimeメソッドの場合）」を参照してください。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- このResultSetオブジェクトがクローズされている場合
  このResultSetオブジェクトを生成したStatementオブジェクトがクローズされたことによって，ResultSetオブジェクトがクローズされた場合も含みます。
- このResultSetオブジェクトを生成したStatementオブジェクトを作成したConnectionがクローズされている場合
- トランザクションの決着によってResultSetオブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Timeとして取得できない場合
- JDBCドライバ内でエラーが発生した場合

8.6.57 getTimestamp(int columnIndex)

(1) 機能

ResultSetオブジェクトの現在行の列の値をjava.sql.Timestampオブジェクトに取得します。値を取得する列は，引数で指定します。

(2) 形式

```
public synchronized java.sql.Timestamp getTimestamp(int columnIndex) throws SQLException
```

(3) 引数

int columnIndex：
　列番号を指定します。

(4) 戻り値

列値が返却されます。検索結果と戻り値の関係を次の表に示します。

DATE型，TIME型，TIMESTAMP型または文字列型（CHAR，VARCHAR）の変換については，「7.6.2 データの変換処理」の「(2) getXXXメソッド実行時のデータ変換処理（DATE型，TIME型，TIMESTAMP型，または文字列型の場合）」を参照してください。

表 8-42　検索結果と戻り値の関係（getTimestamp メソッドの場合）

HADB のデータ型	検索結果	戻り値
CHAR VARCHAR	ナル値	null
	[*半角空白*]*時刻印形式*※[*半角空白*]	検索結果の前後の半角空白を取り除きjava.sql.Timestamp オブジェクトにしたもの
	上記以外	SQLException が投入される
DATE	ナル値	null
	上記以外	検索結果をjava.sql.Timestamp オブジェクトにしたもの
TIME	ナル値	null
	上記以外	検索結果をjava.sql.Timestamp オブジェクトにしたもの
TIMESTAMP	ナル値	null
	上記以外	検索結果をjava.sql.Timestamp オブジェクトにしたもの
そのほか	該当しません	SQLException が投入される

注※

時刻印形式とは，'*YYYY-MM-DD hh*:*mm*:*ss*[.*f*...]'で表される文字列表現のことです。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Timestamp として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.58　getTimestamp(int columnIndex, Calendar cal)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.sql.Timestamp オブジェクトに取得します。値を取得する列は，引数で指定します。

このメソッドは，指定されたカレンダーを使ってタイムスタンプに適切なミリ秒値を作成します。

(2)　形式

```
public synchronized java.sql.Timestamp getTimestamp(int columnIndex, Calendar cal) throws SQLException
```

(3)　引数

int columnIndex：

　列番号を指定します。

Calendar cal：

　データベースに格納されている値のタイムゾーンが設定されたカレンダーを指定します。

(4)　戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-42　検索結果と戻り値の関係(getTimestamp メソッドの場合)」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列番号を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Timestamp として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.59　getTimestamp(String columnName)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.sql.Timestamp オブジェクトに取得します。値を取得する列は，引数で指定します。

(2)　形式

```
public synchronized java.sql.Timestamp getTimestamp(String columnName) throws SQLException
```

(3)　引数

String columnName：

　列名を指定します。

(4)　戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-42　検索結果と戻り値の関係(getTimestamp メソッドの場合)」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Timestamp として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.60　getTimestamp(String columnName, Calendar cal)

(1)　機能

ResultSet オブジェクトの現在行の列の値をjava.sql.Timestamp オブジェクトに取得します。値を取得する列は，引数で指定します。

このメソッドは，指定されたカレンダーを使ってタイムスタンプに適切なミリ秒値を作成します。

(2)　形式

```
public synchronized java.sql.Timestamp getTimestamp(String columnName, Calendar cal) throws
SQLException
```

(3)　引数

String columnName：

　列名を指定します。

Calendar cal：

　データベースに格納されている値のタイムゾーンが設定されたカレンダーを指定します。

(4)　戻り値

列値が返却されます。検索結果と戻り値の関係については，「表 8-42　検索結果と戻り値の関係(getTimestamp メソッドの場合)」を参照してください。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- 存在しない列名を指定した場合
- このメソッドでは取得できないデータ型の場合
- 列の値がjava.sql.Timestamp として取得できない場合
- JDBC ドライバ内でエラーが発生した場合

8.6.61　getType()

(1)　機能

ResultSet オブジェクトの型を返します。

(2)　形式

```
public synchronized int getType() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

次に示すどちらかの値が返却されます。

ResultSet.TYPE_FORWARD_ONLY：

　カーソルが順方向にだけ移動できます。

ResultSet.TYPE_SCROLL_INSENSITIVE：

　カーソルがスクロールできますが，値の変更は反映されません。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.62　getWarnings()

(1)　機能

このResultSet オブジェクトに関する呼び出しによって報告される，最初の警告を取得します。2 つ以上の警告がある場合，後続の警告は最初の警告にチェーンされ，直前に取得された警告のSQLWarning オブジェクトのgetNextWarning メソッドを呼び出すことによって取得されます。

(2) 形式

```
public synchronized SQLWarning getWarnings() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

最初のSQLWarning オブジェクトが返却されます。SQLWarning オブジェクトがない場合はnull が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.63 isAfterLast()

(1) 機能

ResultSet オブジェクトの最終行の後ろに，カーソルがあるかどうかの情報を取得します。

(2) 形式

```
public synchronized boolean isAfterLast() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

カーソルが最終行の後ろにある場合は，true が返却されます。最終行の後ろにない場合，または結果集合の行数が 0 行の場合は，false が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.64 isBeforeFirst()

(1) 機能

ResultSet オブジェクトの先頭行の前に，カーソルがあるかどうかの情報を取得します。

(2) 形式

```
public synchronized boolean isBeforeFirst() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

カーソルが先頭行の前にある場合は，true が返却されます。先頭行の前にない場合，または結果集合の行数が 0 行の場合は，false が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。

- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.65 isClosed()

(1) 機能

このResultSet オブジェクトがクローズされているかどうかを示す値を取得します。

close メソッドを実行したあとに，このメソッドが実行された場合にだけ，true を返却することを保証します。

(2) 形式

```
public synchronized boolean isClosed() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このResultSet オブジェクトがクローズされている場合は，true が返却されます。ResultSet オブジェクトがクローズされていない場合は，false が返却されます。

(5) 発生する例外

なし。

8.6.66 isFirst()

(1) 機能

ResultSet オブジェクトの先頭行に，カーソルがあるかどうかの情報を取得します。

(2) 形式

```
public synchronized boolean isFirst() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

カーソルが先頭行にある場合はtrue が，そうでない場合はfalse が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.67 isLast()

(1) 機能

ResultSet オブジェクトの最終行に，カーソルがあるかどうかの情報を取得します。

(2) 形式

```
public synchronized boolean isLast() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

カーソルが最終行にある場合はtrue が，そうでない場合はfalse が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.68 last()

(1) 機能

ResultSet オブジェクトの最終行に，カーソルを移動します。

(2) 形式

```
public synchronized boolean last() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

カーソルが最終行に移動した場合はtrue が，結果集合の行数が 0 の場合はfalse が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- このResultSet オブジェクトの型がResultSet.TYPE_FORWARD_ONLY の場合

- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.69 next()

(1) 機能

カーソルを次の行に移動します。カーソルが先頭行の前にある場合は先頭行に，最終行にある場合は最終行の後ろに移動します。

(2) 形式

```
public synchronized boolean next() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

このメソッドを実行したことによって，カーソルの位置が先頭行の前または最終行の後ろに移動した場合はfalse が，そうでない場合はtrue が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.70　previous()

(1)　機能

カーソルを 1 つ前の行に移動します。

(2)　形式

```
public synchronized boolean previous() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

このメソッドを実行したことによって，カーソルの位置が先頭行の前に移動した場合はfalse が，そうでない場合はtrue が返却されます。

(5)　発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- このResultSet オブジェクトの型がResultSet.TYPE_FORWARD_ONLY の場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.71　relative(int rows)

(1)　機能

カーソルを移動します。

(2)　形式

```
public synchronized boolean relative(int rows) throws SQLException
```

(3) 引数

int rows：

カーソルの移動行数（現在の行から移動する行数）を指定します。

正の値を指定した場合，カーソルは順方向に移動します。負の値を指定した場合は逆方向に移動します。

(4) 戻り値

このメソッドを実行したことによって，カーソルの位置が先頭行の前または最終行の後ろに移動した場合はfalse が，そうでない場合はtrue が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- このResultSet オブジェクトの型がResultSet.TYPE_FORWARD_ONLY の場合
- 現在の位置が取得できない場合
- 現在のカーソル位置が有効な行にない場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- データベースのアクセスエラーが発生した場合

8.6.72 setFetchDirection(int direction)

(1) 機能

ResultSet オブジェクトのフェッチ方向を設定します。

(2) 形式

```
public synchronized void setFetchDirection(int direction) throws SQLException
```

(3) 引数

int direction：

フェッチ方向を指定します。

ResultSet.FETCH_FORWARD だけを指定できます。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- direction にResultSet.FETCH_FORWARD 以外が指定された場合

8.6.73 setFetchSize(int rows)

(1) 機能

ResultSet オブジェクトを検索する際のフェッチサイズ（フェッチする行数）を設定します。

(2) 形式

```
public synchronized void setFetchSize(int rows) throws SQLException
```

(3) 引数

int rows：

フェッチする行数を 0～65,535 の範囲で指定します。

0 を指定した場合は，システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティの adb_clt_fetch_size の値を基に検索が実施されます。

このメソッドを実行しない場合，Statement オブジェクトに指定した行数値を基に検索が実施されます。Statement オブジェクトに行数値を指定していない，またはStatement オブジェクトから生成した

ResultSet オブジェクトではない場合は，プロパティのadb_clt_fetch_size の値を基に検索が実施されます。

(4) 戻り値

なし。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合
  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合
- rows に 0～65,535 以外の値を指定した場合
- rows に指定した値が，最大格納行数（このResultSet オブジェクトを生成したStatement オブジェクトのsetMaxRows メソッドの設定値）よりも大きい場合
- rows に指定した値が，最大格納行数（このResultSet オブジェクトを生成したStatement オブジェクトのsetMaxLargeRows メソッドの設定値）よりも大きい場合

(6) 留意事項

留意事項については，「8.4.37　setFetchSize(int rows)」の「(6)　留意事項」を参照してください。

8.6.74 wasNull()

(1) 機能

最後に取得した列の値がナル値かどうかの情報を返します。

(2) 形式

```
public synchronized boolean wasNull() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

最後に取得した列の値がナル値の場合はtrue が，そうでない場合はfalse が返却されます。

get*XXX* メソッドによって列の値を取得する前は，false が返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- このResultSet オブジェクトがクローズされている場合

  このResultSet オブジェクトを生成したStatement オブジェクトがクローズされたことによって，ResultSet オブジェクトがクローズされた場合も含みます。
- このResultSet オブジェクトを生成したStatement オブジェクトを作成したConnection がクローズされている場合
- トランザクションの決着によってResultSet オブジェクトが無効になった場合

8.6.75 ResultSet インタフェースでサポートしているフィールド

ResultSet インタフェースでサポートしているフィールドを次の表に示します。

表 8-43 ResultSet インタフェースでサポートしているフィールド

フィールド	備考
public static final int FETCH_FORWARD	−
public static final int FETCH_REVERSE	−
public static final int FETCH_UNKNOWN	−
public static final int TYPE_FORWARD_ONLY	−
public static final int TYPE_SCROLL_INSENSITIVE	−
public static final int TYPE_SCROLL_SENSITIVE	この値が指定された場合，TYPE_SCROLL_INSENSITIVE が指定されたと仮定されます。
public static final int CONCUR_READ_ONLY	−
public static final int CONCUR_UPDATABLE	この値が指定された場合，CONCUR_READ_ONLY が指定されたと仮定されます。
public static final int HOLD_CURSORS_OVER_COMMIT	この値が指定された場合でも，トランザクションが決着すると，ResultSet オブジェクトが無効になることがあります。
public static final int CLOSE_CURSORS_AT_COMMIT	この値が指定された場合，HOLD_CURSORS_OVER_COMMIT が指定されたと仮定されます。

（凡例）−：特にありません。

8.6.76　ResultSet インタフェースに関する注意事項

(1)　getXXX メソッドによる値の取得

- get*XXX* メソッドによってマッピングできるかどうかについては，「7.6.1　データ型のマッピング」の「(2)　検索データ取得時のマッピング」を参照してください。
- get*XXX* メソッドに指定した列番号または列名称が存在しない場合は，SQLException が投入されます。
- get*XXX* メソッドに指定する値が，実際の値を表現できない場合（例：INTEGER 型の40,000 という値を getShort で取得する場合），オーバフローによってSQLException が投入されます。オーバフローが発生するおそれがあるget*XXX* メソッドと HADB のデータ型の組み合わせについては，「7.6.3　オーバフローが発生したときの処理」を参照してください。

(2)　データマッピング（変換）

検索データ取得時に使用するget*XXX* メソッドによってマッピングできるかどうかについては，「7.6.1 データ型のマッピング」の「(2)　検索データ取得時のマッピング」を参照してください。マッピングできない JDBC の SQL データ型に対してget*XXX* メソッドが呼び出された場合，SQLException が投入されます。

(3)　結果セットタイプが ResultSet.TYPE_SCROLL_INSENSITIVE，または ResultSet.TYPE_SCROLL_SENSITIVE の場合のメモリ使用量

結果セットタイプがResultSet.TYPE_SCROLL_INSENSITIVE またはResultSet.TYPE_SCROLL_SENSITIVE のときに，ResultSet インタフェースの次に示すメソッドを実行すると，検索結果蓄積用のメモリを JDBC ドライバが確保します。

- ResultSet.next メソッド
- ResultSet.last メソッド
- ResultSet.absolute メソッド
- ResultSet.relative メソッド
- ResultSet.afterLast メソッド

JDBC ドライバは，検索結果中の値ごとにメモリオブジェクトを割り当てて蓄積します。値が可変長である場合，メモリオブジェクトは検索データの実サイズに合わせた大きさとなります。

(4)　next，absolute，relative，last，および afterLast メソッド

next メソッドを実行すると，JDBC ドライバは次の表に示すようにデータベースからデータを取得，蓄積します。

表 8-44　next メソッド実行時の，データベースからのデータの取得，蓄積

状態	結果セットタイプ	
	ResultSet.TYPE_FORWARD_ONLY	ResultSet.TYPE_SCROLL_INSENSITIVE，または ResultSet.TYPE_SCROLL_SENSITIVE
next メソッドで移動した現在行のデータを JDBC ドライバ内に読み込んでいない	遷移した現在行を接続先のデータベースから取得します。	遷移した現在行をデータベースから取得し，JDBC ドライバのメモリに読み込み，蓄積します。
next メソッドで移動した現在行のデータを JDBC ドライバ内に読み込んでいる		データベースからデータを取得しません。

absolute，relative，last，およびafterLast メソッドを実行すると，JDBC ドライバは次の表に示すようにデータベースからデータを取得，蓄積します。

表 8-45　absolute，relative，last，および afterLast メソッド実行時の，データベースからのデータの取得，蓄積

状態	結果セットタイプが ResultSet.TYPE_SCROLL_INSENSITIVE，または ResultSet.TYPE_SCROLL_SENSITIVE
検索結果の先頭行から指定行※までに，JDBC ドライバが読み込んでいないデータがある	読み込んでいない行をデータベースから取得し，JDBC ドライバのメモリに蓄積します。
検索結果の先頭行から指定行※まで，JDBC ドライバはデータを読み込んでいる	データベースからデータを取得しません。

注

結果セットタイプがResultSet.TYPE_FORWARD_ONLY の場合，SQLException が投入されます。

注※

last メソッドおよびafterLast メソッドの場合，先頭行から最終行になります。

(5)　getAsciiStream および getCharacterStream メソッドに関する注意事項

getAsciiStream およびgetCharacterStream メソッドによって返却されたオブジェクトは，JDBC ドライバが暗黙的にクローズすることはありません。メソッドの呼び出し側でclose メソッドを実行してください。

(6)　検索結果行数の上限

ResultSet オブジェクトが HADB サーバから取得できる検索結果の行数を次の表に示します。次の表に示す行以上の検索結果は，JDBC ドライバが破棄します。

表 8-46　ResultSet オブジェクトが HADB サーバから取得できる検索結果の行数

ResultSet オブジェクト	結果セットタイプ	
	ResultSet.TYPE_SCROLL_INSENSITIVE	左記以外
次のメソッドを実行したStatementオブジェクトから生成したResultSetオブジェクト • setMaxRows • setLargeMaxRows	検索結果行数は，次のメソッドで指定した行数です。 • setMaxRows • setLargeMaxRows setLargeMaxRows メソッドでInteger.MAX_VALUE を超える値を指定した場合でも，上限値はInteger.MAX_VALUEです。	検索結果行数は，次のメソッドで指定した行数です。 • setMaxRows • setLargeMaxRows
上記以外のResultSet オブジェクト	上限値はInteger.MAX_VALUE です。	上限はありません。

8.7 DatabaseMetaData インタフェース

ここでは，DatabaseMetaData インタフェースで提供されているメソッドについて説明します。

8.7.1 DatabaseMetaData インタフェースのメソッド一覧

(1) DatabaseMetaData インタフェースの主な機能

DatabaseMetaData インタフェースでは，主に次の機能が提供されています。

- 接続先のデータベースに関する各種情報の返却
- 表一覧，列一覧などの一覧情報の返却（ResultSet（結果セット）に格納）

(2) HADB でサポートしている DatabaseMetaData インタフェースのメソッド

HADB でサポートしているDatabaseMetaData インタフェースのメソッドの一覧を次の表に示します。

表 8-47 DatabaseMetaData インタフェースのメソッドの一覧

項番	DatabaseMetaData インタフェースのメソッド	機能
1	allProceduresAreCallable()	getProcedures メソッドによって返されるすべてのプロシジャが，現在の HADB ユーザから呼び出せるかどうかを取得します。
2	allTablesAreSelectable()	getTables メソッドによって返されるすべての表が，現在の HADB ユーザによって使用できるかどうかを取得します。
3	autoCommitFailureClosesAllResultSets()	自動コミットモード有効時にSQLException が発生した場合，すべてのオープンされたResultSet がクローズされるかどうかを返却します。
4	dataDefinitionCausesTransactionCommit()	トランザクションのデータ定義文が，トランザクションを強制的にコミットさせるかどうかを取得します。
5	dataDefinitionIgnoredInTransactions()	トランザクションでデータ定義文が無視されるかどうかを取得します。
6	deletesAreDetected(int type)	ResultSet クラスのrowDeleted メソッドを呼び出すことによって可視の行が削除されたことを検出できるかどうかを取得します。
7	doesMaxRowSizeIncludeBlobs()	getMaxRowSize メソッドの戻り値が SQL データの型の LONGVARCHAR およびLONGVARBINARY を含むかどうかを取得します。

項番	DatabaseMetaData インタフェースのメソッド	機能
8	generatedKeyAlwaysReturned()	自動生成キーの列として指定された列名またはインデクスが有効であり，かつ文が成功した場合に，生成されたキーが常に返されるかどうかを取得します。
9	getAttributes(String catalog, String schemaPattern, String typeNamePattern, String attributeNamePattern)	指定されたスキーマおよびカタログで使用可能なユーザ定義の型 (UDT)のための指定された型の指定された属性に関する記述を取得します。
10	getBestRowIdentifier(String catalog, String schema, String table, int scope, boolean nullable)	行を一意に識別する表の最適な列セットに関する記述を取得します。
11	getCatalogs()	カタログ名を取得します。
12	getCatalogSeparator()	カタログ名と表名のセパレータを取得します。
13	getCatalogTerm()	catalog に対する推奨用語を取得します。
14	getClientInfoProperties()	ドライバがサポートするクライアント情報プロパティのリストを返却します。
15	getColumnPrivileges(String catalog, String schema, String table, String columnNamePattern)	表の列へのアクセス権に関する記述を取得します。
16	getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)	表の列情報を取得します。
17	getConnection()	このDatabaseMetaData インスタンスを生成したConnection インスタンスを取得します。
18	getCrossReference(String parentCatalog, String parentSchema, String parentTable, String foreignCatalog, String foreignSchema, String foreignTable)	指定された参照表と指定された被参照表とのクロスリファレンス情報を取得します。
19	getDatabaseMajorVersion()	データベース（HADB サーバ）のメジャーバージョンを取得します。
20	getDatabaseMinorVersion()	データベース（HADB サーバ）のマイナーバージョンを取得します。
21	getDatabaseProductName()	接続するデータベース（HADB サーバ）の製品名称を取得します。
22	getDatabaseProductVersion()	接続するデータベース（HADB サーバ）のバージョンを取得します。
23	getDefaultTransactionIsolation()	このデータベースのデフォルトのトランザクション隔離性水準を取得します。
24	getDriverMajorVersion()	JDBC ドライバのメジャーバージョンを取得します。
25	getDriverMinorVersion()	JDBC ドライバのマイナーバージョンを取得します。
26	getDriverName()	JDBC ドライバの名前を取得します。

項番	DatabaseMetaData インタフェースのメソッド	機能
27	getDriverVersion()	JDBC ドライバのバージョンを取得します。
28	getExportedKeys(String catalog, String schema, String table)	参照表の外部キーに関する情報を取得します。
29	getExtraNameCharacters()	二重引用符（”）で囲まれていない識別名に使用できるすべての特殊文字を取得します。
30	getFunctionColumns(String catalog, String schemaPattern, String functionNamePattern, String columnNamePattern)	関数のパラメタおよび返される型に関する情報を返却します。
31	getFunctions(String catalog, String schemaPattern, String functionNamePattern)	関数に関する情報を返却します。
32	getIdentifierQuoteString()	SQL 識別子を引用するのに使用する文字列を取得します。
33	getImportedKeys(String catalog, String schema, String table)	被参照表の主キーに関する情報を取得します。
34	getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate)	インデクスに関する情報を取得します。
35	getJDBCMajorVersion()	ドライバの JDBC メジャーバージョンを取得します。
36	getJDBCMinorVersion()	ドライバの JDBC マイナーバージョンを取得します。
37	getMaxBinaryLiteralLength()	バイナリリテラル中に入れることができる 16 進数の最大文字数を取得します。
38	getMaxCatalogNameLength()	カタログ名の最大文字数を取得します。
39	getMaxCharLiteralLength()	文字データとして指定できる最大文字数を取得します。
40	getMaxColumnNameLength()	列名に指定できる最大文字数を取得します。
41	getMaxColumnsInGroupBy()	GROUP BY 句中に指定できるグループ化列の数の最大値を取得します。
42	getMaxColumnsInIndex()	インデクス構成列数の最大値を取得します。
43	getMaxColumnsInOrderBy()	ORDER BY 句中に指定できる列数の最大値を取得します。
44	getMaxColumnsInSelect()	選択リストに指定できる選択式の最大数を取得します。
45	getMaxColumnsInTable()	表に定義できる列数の最大値を取得します。
46	getMaxConnections()	同時に HADB サーバに接続できる HADB クライアントの最大数を取得します。
47	getMaxCursorNameLength()	カーソル名の最大文字数を取得します。
48	getMaxIndexLength()	インデクスのキー長の最大値を取得します。
49	getMaxLogicalLobSize()	このデータベースで，LOB の論理サイズとして許容される最大バイト数を取得します。

項番	DatabaseMetaData インタフェースのメソッド	機能
50	getMaxProcedureNameLength()	プロシジャ名の最大文字数を取得します。
51	getMaxRowSize()	1 行の最大バイト数を取得します。
52	getMaxSchemaNameLength()	スキーマ名の最大文字数を取得します。
53	getMaxStatementLength()	SQL 文に指定できる文字列長の最大値を取得します。
54	getMaxStatements()	同時実行できる SQL 文の最大数を取得します。
55	getMaxTableNameLength()	表名に指定できる最大文字数を取得します。
56	getMaxTablesInSelect()	SELECT 文中に指定できる表数の最大値を取得します。
57	getMaxUserNameLength()	認可識別子の最大文字数を取得します。
58	getNumericFunctions()	使用できる数学関数をコンマで区切ったリストで取得します。
59	getPrimaryKeys(String catalog, String schema, String table)	指定された表の主キー列の記述を取得します。
60	getProcedureColumns(String catalog, String schemaPattern, String procedureNamePattern, String columnNamePattern)	ストアドプロシジャパラメタに関する記述を取得します。
61	getProcedures(String catalog, String schemaPattern, String procedureNamePattern)	ストアドプロシジャに関する記述を取得します。
62	getProcedureTerm()	procedure に対する推奨用語を取得します。
63	getPseudoColumns(String catalog,String schemaPattern,String tableNamePattern,String columnNamePattern)	指定されたカタログおよびスキーマ内の特定の表で使用できる擬似列または隠し列の説明を取得します。
64	getResultSetHoldability()	ResultSet オブジェクトのデフォルトの保持機能を取得します。
65	getRowIdLifetime()	RowId 型をサポートしているかどうかを示します。また，サポートしている場合は，RowId オブジェクトが有効である期間も示します。
66	getSchemas()	スキーマ名を取得します。
67	getSchemas(String catalog, String schemaPattern)	スキーマ名を取得します。
68	getSchemaTerm()	schema に対する推奨用語を取得します。
69	getSearchStringEscape()	ワイルドカード文字をエスケープするために使用する文字列を取得します。
70	getSQLKeywords()	データベース固有の SQL キーワードであり，かつ SQL:2003 のキーワードではないすべてのキーワードを，コンマで区切ったリストで取得します。
71	getSQLStateType()	SQLException クラスのgetSQLState メソッドによって返される SQLSTATE が X/Open（現在は Open Group）の SQL CLI であるか SQL:2003 であるかを取得します。

項番	DatabaseMetaData インタフェースのメソッド	機能
72	getStringFunctions()	使用できる文字列関数をコンマで区切ったリストで取得します。
73	getSuperTables(String catalog, String schemaPattern, String tableNamePattern)	特定のスキーマで定義されているテーブル階層の情報を取得します。
74	getSuperTypes(String catalog, String schemaPattern, String typeNamePattern)	特定のスキーマで定義されているユーザ定義型（UDT）階層の情報を取得します。
75	getSystemFunctions()	使用できるシステム関数をコンマで区切ったリストで取得します。
76	getTablePrivileges(String catalog, String schemaPattern, String tableNamePattern)	表に対するアクセス権限に関する情報を取得します。
77	getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)	表に関する情報を取得します。
78	getTableTypes()	表の種類を取得します。
79	getTimeDateFunctions()	使用できる時間関数と日付関数をコンマで区切ったリストで取得します。
80	getTypeInfo()	標準 SQL の型に関する情報を取得します。
81	getUDTs(String catalog, String schemaPattern, String typeNamePattern, int[] types)	ユーザ定義型（UDT）に関する情報を取得します。
82	getURL()	HADB サーバの接続先情報を指定した URL を取得します。
83	getUserName()	HADB サーバに接続した認可識別子を取得します。
84	getVersionColumns(String catalog, String schema, String table)	行の任意の値が変更された場合に，自動的に更新される表の列に関する記述を取得します。
85	insertsAreDetected(int type)	ResultSet クラスのrowInserted メソッドを呼び出すことによって可視の行が挿入されたことを検出できるかどうかを取得します。
86	isCatalogAtStart()	完全修飾された表名の開始部分（または終了部分）にカタログが現れるかどうかを取得します。
87	isReadOnly()	読み取り専用モードかどうかを取得します。
88	locatorsUpdateCopy()	LOB への変更が，コピーに対して行われたのか，LOB に直接行われたのかを示します。
89	nullPlusNonNullIsNull()	ナル値と非ナル値の連結をナル値とするかどうかを取得します。
90	nullsAreSortedAtEnd()	ナル値が，終了時にソート順に関係なくソートされるかどうかを取得します。
91	nullsAreSortedAtStart()	ナル値が，開始時にソート順に関係なくソートされるかどうかを取得します。
92	nullsAreSortedHigh()	ナル値が高位にソートされるかどうかを取得します。
93	nullsAreSortedLow()	ナル値が下位にソートされるかどうかを取得します。

項番	DatabaseMetaData インタフェースのメソッド	機能
94	othersDeletesAreVisible(int type)	ほかで行われた削除が可視かどうかを取得します。
95	othersInsertsAreVisible(int type)	ほかで行われた挿入が可視かどうかを取得します。
96	othersUpdatesAreVisible(int type)	ほかで行われた更新が可視かどうかを取得します。
97	ownDeletesAreVisible(int type)	結果セット自身の削除が可視かどうかを取得します。
98	ownInsertsAreVisible(int type)	結果セット自身の挿入が可視かどうかを取得します。
99	ownUpdatesAreVisible(int type)	結果セット自身の更新が可視かどうかを取得します。
100	storesLowerCaseIdentifiers()	大文字と小文字が混在する二重引用符なしの SQL 識別子を，大文字と小文字を区別しないで処理し，小文字で格納するかどうかを取得します。
101	storesLowerCaseQuotedIdentifiers()	大文字と小文字が混在する二重引用符付きの SQL 識別子を，大文字と小文字を区別しないで処理し，小文字で格納するかどうかを取得します。
102	storesMixedCaseIdentifiers()	大文字と小文字が混在する二重引用符なしの SQL 識別子を，大文字と小文字を区別しないで処理し，大文字と小文字混在で格納するかどうかを取得します。
103	storesMixedCaseQuotedIdentifiers()	大文字と小文字が混在する二重引用符付きの SQL 識別子を，大文字と小文字を区別しないで処理し，大文字と小文字混在で格納するかどうかを取得します。
104	storesUpperCaseIdentifiers()	大文字と小文字が混在する二重引用符なしの SQL 識別子を，大文字と小文字を区別しないで処理し，大文字で格納するかどうかを取得します。
105	storesUpperCaseQuotedIdentifiers()	大文字と小文字が混在する二重引用符付きの SQL 識別子を，大文字と小文字を区別しないで処理し，大文字で格納するかどうかを取得します。
106	supportsAlterTableWithAddColumn()	列追加のあるALTER TABLE がサポートされているかどうかを取得します。
107	supportsAlterTableWithDropColumn()	列削除のあるALTER TABLE がサポートされているかどうかを取得します。
108	supportsANSI92EntryLevelSQL()	ANSI92 エントリレベルの SQL 文法がサポートされているかどうかを取得します。
109	supportsANSI92FullSQL()	ANSI92 完全レベルの SQL 文法がサポートされているかどうかを取得します。
110	supportsANSI92IntermediateSQL()	ANSI92 中間レベルの SQL 文法がサポートされているかどうかを取得します。
111	supportsBatchUpdates()	バッチ更新がサポートされているかどうかを取得します。
112	supportsCatalogsInDataManipulation()	データ操作文でカタログ名を使用できるかどうかを取得します。
113	supportsCatalogsInIndexDefinitions()	インデクス定義文でカタログ名を使用できるかどうかを取得します。

項番	DatabaseMetaData インタフェースのメソッド	機能
114	supportsCatalogsInPrivilegeDefinitions()	権限付与の定義文（GRANT 文），または権限取り消しの定義文（REVOKE 文）で，カタログ名を使用できるかどうかを取得します。
115	supportsCatalogsInProcedureCalls()	プロシジャ呼び出し文でカタログ名を使用できるかどうかを取得します。
116	supportsCatalogsInTableDefinitions()	表定義文でカタログ名を使用できるかどうかを取得します。
117	supportsColumnAliasing()	列の別名がサポートされているかどうかを取得します。
118	supportsConvert()	SQL の型間のCONVERT 関数がサポートされているかどうかを取得します。
119	supportsConvert(int fromType, int toType)	指定された SQL の型間のCONVERT 関数がサポートされているかどうかを取得します。
120	supportsCoreSQLGrammar()	ODBC Core SQL 文法がサポートされているかどうかを取得します。
121	supportsCorrelatedSubqueries()	外への参照列を含む副問合せがサポートされているかどうかを取得します。
122	supportsDataDefinitionAndDataManipulationTransactions()	トランザクションで，データ定義文とデータ操作文の両方がサポートされているかどうかを取得します。
123	supportsDataManipulationTransactionsOnly()	トランザクションでデータ操作文だけがサポートされているかどうかを取得します。
124	supportsDifferentTableCorrelationNames()	表相関名がサポートされている場合，表名と異なる名前であるという制限を付けるかどうかを取得します。
125	supportsExpressionsInOrderBy()	ORDER BY リスト中で値式がサポートされているかどうかを取得します。
126	supportsExtendedSQLGrammar()	ODBC Extended SQL 文法がサポートされているかどうかを取得します。
127	supportsFullOuterJoins()	完全な入れ子の外結合がサポートされているかどうかを取得します。
128	supportsGetGeneratedKeys()	文が実行されたあとに自動生成キーを取得できるかどうかを取得します。
129	supportsGroupBy()	GROUP BY 句がサポートされているかどうかを取得します。
130	supportsGroupByBeyondSelect()	SELECT 文中のすべての列がGROUP BY 句に含まれるという条件で，このデータベースによって，GROUP BY 句でSELECT 文中にない列の使用がサポートされているかどうかを取得します。
131	supportsGroupByUnrelated()	GROUP BY 句でSELECT 文中にない列の使用がサポートされているかどうかを取得します。
132	supportsIntegrityEnhancementFacility()	SQL Integrity Enhancement Facility がサポートされているかどうかを取得します。

項番	DatabaseMetaData インタフェースのメソッド	機能
133	supportsLikeEscapeClause()	LIKE 述語のエスケープ句がサポートされているかどうかを取得します。
134	supportsLimitedOuterJoins()	外結合に関し，制限されたサポートが提供されるかどうかを取得します。
135	supportsMinimumSQLGrammar()	ODBC Minimum SQL 文法がサポートされているかどうかを取得します。
136	supportsMixedCaseIdentifiers()	大文字と小文字が混在する二重引用符なしの SQL 識別子を，大文字と小文字を区別して処理し，大文字と小文字混在で格納するかどうかを取得します。
137	supportsMixedCaseQuotedIdentifiers()	大文字と小文字が混在する二重引用符付きの SQL 識別子を，大文字と小文字を区別して処理し，結果として大文字と小文字混在で格納するかどうかを取得します。
138	supportsMultipleOpenResults()	CallableStatement オブジェクトから同時に返された複数の ResultSet オブジェクトを持つことが可能かどうかを取得します。
139	supportsMultipleResultSets()	execute メソッドの単一の呼び出しからの複数のResultSet オブジェクトの取得がサポートされているかどうかを取得します。
140	supportsMultipleTransactions()	一度に複数のトランザクションを（異なった接続で）オープンできるかどうかを取得します。
141	supportsNamedParameters()	CALL 文への名前付きパラメタがサポートされているかどうかを取得します。
142	supportsNonNullableColumns()	列を非ナル値として定義できるかどうかを取得します。
143	supportsOpenCursorsAcrossCommit()	コミット間でカーソルがオープンされたままの状態がサポートされているかどうかを取得します。
144	supportsOpenCursorsAcrossRollback()	ロールバック間でカーソルがオープンされたままの状態がサポートされているかどうかを取得します。
145	supportsOpenStatementsAcrossCommit()	コミット間で文ハンドルがオープンされたままの状態がサポートされているかどうかを取得します。
146	supportsOpenStatementsAcrossRollback()	ロールバック間で文ハンドルがオープンされたままの状態がサポートされているかどうかを取得します。
147	supportsOrderByUnrelated()	ORDER BY 句でSELECT 文中にない列の使用がサポートされているかどうかを取得します。
148	supportsOuterJoins()	外結合が何らかの形式でサポートされているかどうかを取得します。
149	supportsPositionedDelete()	位置指定されたDELETE 文がサポートされているかどうかを取得します。
150	supportsPositionedUpdate()	位置指定されたUPDATE 文がサポートされているかどうかを取得します。

項番	DatabaseMetaData インタフェースのメソッド	機能
151	supportsRefCursors()	データベースがREF CURSOR をサポートしているかどうかを取得します。
152	supportsResultSetConcurrency(int type, int concurrency)	指定した結果セットタイプと並行処理タイプの組み合わせをサポートしているかどうかを返します。
153	supportsResultSetHoldability(int holdability)	指定されたResultSet オブジェクトの保持機能がサポートされているかどうかを返します。
154	supportsResultSetType(int type)	指定した結果セットタイプがサポートされているかどうかを返します。
155	supportsSavepoints()	セーブポイントがサポートされているかどうかを取得します。
156	supportsSchemasInDataManipulation()	データ操作文でスキーマ名を使用できるかどうかを取得します。
157	supportsSchemasInIndexDefinitions()	インデクス定義文でスキーマ名を使用できるかどうかを取得します。
158	supportsSchemasInPrivilegeDefinitions()	権限付与の定義文（GRANT 文），または権限取り消しの定義文（REVOKE 文）で，スキーマ名を使用できるかどうかを取得します。
159	supportsSchemasInProcedureCalls()	プロシジャ呼び出し文でスキーマ名を使用できるかどうかを取得します。
160	supportsSchemasInTableDefinitions()	表定義文でスキーマ名を使用できるかどうかを取得します。
161	supportsSelectForUpdate()	SELECT FOR UPDATE 文がサポートされているかどうかを取得します。
162	supportsStatementPooling()	文ハンドルのプールがサポートされているかどうかを取得します。
163	supportsStoredFunctionsUsingCallSyntax()	ストアドプロシジャエスケープ構文を使用したユーザ定義関数，またはベンダ関数の呼び出しをサポートするかどうかを取得します。
164	supportsStoredProcedures()	ストアドプロシジャエスケープ構文を使用するストアドプロシジャ呼び出しがサポートされているかどうかを判定します。
165	supportsSubqueriesInComparisons()	比較述語で副問合せがサポートされているかどうかを取得します。
166	supportsSubqueriesInExists()	EXISTS 述語で副問合せがサポートされているかどうかを取得します。
167	supportsSubqueriesInIns()	IN 述語で副問合せがサポートされているかどうかを取得します。
168	supportsSubqueriesInQuantifieds()	限定述語で副問合せがサポートされているかどうかを取得します。
169	supportsTableCorrelationNames()	表の相関名がサポートされているかどうかを取得します。
170	supportsTransactionIsolationLevel(int level)	指定したトランザクション隔離性水準がサポートされているかどうかを返します。

項番	DatabaseMetaData インタフェースのメソッド	機能
171	supportsTransactions()	トランザクションがサポートされているかどうかを取得します。
172	supportsUnion()	SQL UNION がサポートされているかどうかを取得します。
173	supportsUnionAll()	SQL UNION ALL がサポートされているかどうかを取得します。
174	updatesAreDetected(int type)	ResultSet クラスのrowUpdated メソッドを呼び出すことによって可視の行が更新されたことを検出できるかどうかを取得します。
175	usesLocalFilePerTable()	各表にファイルを使用するかどうかを取得します。
176	usesLocalFiles()	ローカルファイルに表を格納するかどうかを取得します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されることがあります。

(3) 必要なパッケージ名称とクラス名称

DatabaseMetaData インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbDatabaseMetaData

(4) パターン文字列中に指定できる特殊文字

DatabaseMetaData クラスのメソッドには，String のパターン文字列を指定する引数があります。パターン文字列中には，次の表に示す特殊文字を指定できます。

表 8-48 パターン文字列中に指定できる特殊文字

指定できる特殊文字	意味
_（下線）	任意の 1 文字です。
%	0 文字以上の任意の長さの文字列です。
¥	エスケープ文字です。パターン文字列中に指定したエスケープ文字の直後の特殊文字を通常の文字として扱います。 ¥は，Shift-JIS の 0x5c（UTF-16LE の 0x5c00）で示される文字です。UTF-8 の場合は，¥（バックスラッシュ）で表示される文字を指定してください。

8.7.2 allProceduresAreCallable()

(1) 機能

getProcedures メソッドによって返されるすべてのプロシジャが，現在の HADB ユーザから呼び出せるかどうかを取得します。

(2) 形式

```
public synchronized boolean allProceduresAreCallable() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.3 allTablesAreSelectable()

(1) 機能

getTables メソッドによって返されるすべての表が，現在の HADB ユーザによって使用できるかどうかを取得します。

(2) 形式

```
public synchronized boolean allTablesAreSelectable() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.4 autoCommitFailureClosesAllResultSets()

(1) 機能

自動コミットモード有効時にSQLException が発生した場合，すべてのオープンされたResultSet がクローズされるかどうかを返却します。

(2) 形式

```
public synchronized boolean autoCommitFailureClosesAllResultSets() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.5 dataDefinitionCausesTransactionCommit()

(1) 機能

トランザクションのデータ定義文が，トランザクションを強制的にコミットさせるかどうかを取得します。

(2) 形式

```
public synchronized boolean dataDefinitionCausesTransactionCommit() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にture が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.6 dataDefinitionIgnoredInTransactions()

(1) 機能

トランザクションでデータ定義文が無視されるかどうかを取得します。

(2) 形式

```
public synchronized boolean dataDefinitionIgnoredInTransactions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.7 deletesAreDetected(int type)

(1) 機能

ResultSet クラスのrowDeleted メソッドを呼び出すことによって可視の行が削除されたことを検出できるかどうかを取得します。

(2) 形式

```
public synchronized boolean deletesAreDetected(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.8 doesMaxRowSizeIncludeBlobs()

(1) 機能

getMaxRowSize メソッドの戻り値が SQL データの型のLONGVARCHAR およびLONGVARBINARY を含むかどうかを取得します。

(2) 形式

```
public synchronized boolean doesMaxRowSizeIncludeBlobs() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.9 generatedKeyAlwaysReturned()

(1) 機能

自動生成キーの列として指定された列名またはインデクスが有効であり，かつ文が成功した場合に，生成されたキーが常に返されるかどうかを取得します。

(2) 形式

```
public synchronized boolean generatedKeyAlwaysReturned() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.10 getAttributes(String catalog, String schemaPattern, String typeNamePattern, String attributeNamePattern)

(1) 機能

指定されたスキーマおよびカタログで使用可能なユーザ定義の型（UDT）のための指定された型の指定された属性に関する記述を取得します。

(2) 形式

```
public synchronized ResultSet getAttributes(String catalog, String schemaPattern, String typeNamePattern, String attributeNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。ただし，指定しても無視されます。

String typeNamePattern：

型名パターンを指定します。ただし，指定しても無視されます。

String attributeNamePattern：

属性名パターンを指定します。ただし，指定しても無視されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-49 返却される ResultSet オブジェクトの形式（getAttributes メソッドの場合）

列番号	型	列名	列の意味
1	String	TYPE_CAT	カタログ名
2	String	TYPE_SCHEM	スキーマ名
3	String	TYPE_NAME	型名
4	String	ATTR_NAME	属性名
5	int	DATA_TYPE	SQL の型の属性の型
6	String	ATTR_TYPE_NAME	型名
7	int	ATTR_SIZE	列サイズ

列番号	型	列名	列の意味
8	int	DECIMAL_DIGITS	小数点以下の桁数
9	int	NUM_PREC_RADIX	基数
10	int	NULLABLE	この型にナル値を使用できるかどうか
11	String	REMARKS	コメント記述列
12	String	ATTR_DEF	デフォルト値
13	int	SQL_DATA_TYPE	未使用
14	int	SQL_DATETIME_SUB	未使用
15	int	CHAR_OCTET_LENGTH	CHAR 型の列の最大バイト数
16	int	ORDINAL_POSITION	列番号
17	String	IS_NULLABLE	この型にナル値が使用できるかどうか
18	String	SCOPE_CATALOG	参照属性のスコープである表のカタログ名
19	String	SCOPE_SCHEMA	参照属性のスコープである表のスキーマ名
20	String	SCOPE_TABLE	参照属性のスコープである表の名称
21	short	SOURCE_DATA_TYPE	個別の型またはユーザ生成Ref 型，java.sql.Types の SQL 型のソースの型

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.11 getBestRowIdentifier(String catalog, String schema, String table, int scope, boolean nullable)

(1) 機能

行を一意に識別する表の最適な列セットに関する記述を取得します。

(2) 形式

```
public synchronized ResultSet getBestRowIdentifier(String catalog, String schema, String tab
le, int scope, boolean nullable) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schema：

スキーマ名を指定します。ただし，指定しても無視されます。

String table：

表名を指定します。ただし，指定しても無視されます。

int scope：

対象のスケールを指定します。ただし，指定しても無視されます。

boolean nullable：

ナル値を指定可能な列を含むかを指定します。ただし，指定しても無視されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-50 返却される ResultSet オブジェクトの形式（getBestRowIdentifier メソッドの場合）

列番号	型	列名	列の意味
1	short	SCOPE	結果の実際のスケール
2	String	COLUMN_NAME	列名
3	int	DATA_TYPE	SQL の型
4	String	TYPE_NAME	型名
5	int	COLUMN_SIZE	精度
6	int	BUFFER_LENGTH	未使用
7	short	DECIMAL_DIGITS	小数点以下の桁数
8	short	PSEUDO_COLUMN	擬似列かどうか

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.12　getCatalogs()

(1)　機能

カタログ名を取得します。

(2)　形式

```
public synchronized ResultSet getCatalogs() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-51　返却される ResultSet オブジェクトの形式（getCatalogs メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	カタログ名

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.13　getCatalogSeparator()

(1)　機能

カタログ名と表名のセパレータを取得します。

(2)　形式

```
public synchronized String getCatalogSeparator() throws SQLException
```

(3)　引数

なし。

(4) 戻り値

常に空の文字列が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.14 getCatalogTerm()

(1) 機能

catalog に対する推奨用語を取得します。

(2) 形式

```
public synchronized String getCatalogTerm() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に空の文字列が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.15 getClientInfoProperties()

(1) 機能

ドライバがサポートするクライアント情報プロパティのリストを返却します。

(2) 形式

```
public synchronized ResultSet getClientInfoProperties() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-52 返却される ResultSet オブジェクトの形式

列番号	型	列名	列の意味
1	String	NAME	クライアント情報プロパティの名前
2	int	MAX_LEN	プロパティ値の最大長
3	String	DEFAULT_VALUE	プロパティのデフォルト値
4	String	DESCRIPTION	プロパティの記述

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.16 getColumnPrivileges(String catalog, String schema, String table, String columnNamePattern)

(1) 機能

表の列へのアクセス権に関する記述を取得します。

(2) 形式

```
public synchronized ResultSet getColumnPrivileges(String catalog, String schema, String tabl
e, String columnNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schema：

スキーマ名を指定します。ただし，指定しても無視されます。

String table：

表名を指定します。ただし，指定しても無視されます。

String columnNamePattern：

列名パターンを指定します。ただし，指定しても無視されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-53 返却される ResultSet オブジェクトの形式（getColumnPrivileges メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	カタログ名
2	String	TABLE_SCHEM	スキーマ名
3	String	TABLE_NAME	表名
4	String	COLUMN_NAME	列名
5	String	GRANTOR	アクセス権の付与者
6	String	GRANTEE	アクセス権の被付与者
7	String	PRIVILEGE	アクセス権限名
8	String	IS_GRANTABLE	アクセス権の被付与者が，別の HADB ユーザにアクセス権を与えることができるかどうか

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.17 getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)

(1) 機能

表の列情報を取得します。

このメソッドを実行した HADB ユーザが持っている権限によって，取得できる表の列情報が変わります。権限と取得できる表の列情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2) 形式

```
public synchronized ResultSet getColumns(String catalog, String schemaPattern, String tableN
amePattern, String columnNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターン※を指定します。大文字と小文字を区別します。

String tableNamePattern：

表名パターン※を指定します。大文字と小文字を区別します。

String columnNamePattern：

列名パターン※を指定します。大文字と小文字を区別します。

注※

各パターンに指定できる特殊文字については，「8.7.1 DatabaseMetaData インタフェースのメソッド一覧」の「(4) パターン文字列中に指定できる特殊文字」を参照してください。

(4) 戻り値

ResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-54 返却される ResultSet オブジェクトの形式（getColumns メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	常に空の文字列を返します。
2	String	TABLE_SCHEM	スキーマ名
3	String	TABLE_NAME	表名
4	String	COLUMN_NAME	列名
5	int	DATA_TYPE	SQL の型
6	String	TYPE_NAME	型名
7	int	COLUMN_SIZE	列サイズ
8	int	BUFFER_LENGTH	常にナル値を返します。
9	int	DECIMAL_DIGITS	小数点以下の桁数
10	int	NUM_PREC_RADIX	基数 • 概数値：2 • 真数値：10

列番号	型	列名	列の意味
			• 数値以外：0
11	int	NULLABLE	この型にナル値を使用できるかどうかを返します。 • columnNoNulls：ナル値を使用できない可能性があります。 • columnNullable：ナル値を使用できます。
12	String	REMARKS	コメント記述列 常に空の文字列を返します。
13	String	COLUMN_DEF	列の既定値を返します。 • 返却された値がアポストロフィ(')で囲まれている場合は，列の既定値が文字列であることを意味しています。 • 列の既定値にNULL が指定された場合は，アポストロフィ(')で囲まれていないNULL という文字列を返します。 • 列の既定値が指定されていない場合は，ナル値を返します。 • そのほかの返却値については，マニュアル『HADB　システム構築・運用ガイド』の『SQL_COLUMNS の内容』のDEFAULT_VALUE 列を参照してください。
14	int	SQL_DATA_TYPE	常にナル値を返します。
15	int	SQL_DATETIME_SUB	常にナル値を返します。
16	int	CHAR_OCTET_LENGTH	CHAR 型の列の最大バイト数
17	int	ORDINAL_POSITION	列番号 1 から始まります。
18	String	IS_NULLABLE	この型にナル値が使用できるかどうかを返します。 • "NO"：ナル値を使用できません。 • "YES"：ナル値を使用できる可能性があります。
19	String	SCOPE_CATALOG	常に空の文字列を返します。
20	String	SCOPE_SCHEMA	常に空の文字列を返します。
21	String	SCOPE_TABLE	常に空の文字列を返します。
22	short	SOURCE_DATA_TYPE	常にナル値を返します。
23	String	IS_AUTOINCREMENT	列が自動インクリメントされるかどうかを示します。 • "NO"：列が自動インクリメントされません。 常にNO を返します。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6) 留意事項

次に示す 2 つのメソッドで取得した，ビュー表の列情報の値が異なることがあります。

1. このメソッドで取得したResultSet オブジェクトのDECIMAL_DIGITS の値
2. ResultSetMetaData インタフェースのgetScale メソッドで取得した値

上記の 1.では，ビュー表を定義したときの位取りの値を取得します。一方，上記の 2.では，ビュー表を検索したときの位取りの値を取得します。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，ビュー表を定義したときとビュー表を検索したときで異なる場合は，上記の 2 つの値が異なります。

8.7.18 getConnection()

(1) 機能

このDatabaseMetaData インスタンスを生成したConnection インスタンスを取得します。

(2) 形式

```
public synchronized Connection getConnection() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Connection オブジェクトが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.19 getCrossReference(String parentCatalog, String parentSchema, String parentTable, String foreignCatalog, String foreignSchema, String foreignTable)

(1) 機能

指定された参照表と指定された被参照表とのクロスリファレンス情報を取得します。

このメソッドを実行したHADBユーザが持っている権限によって，取得できるクロスリファレンス情報が変わります。権限と取得できる情報の対応については，マニュアル『HADB　システム構築・運用ガイド』の『HADBユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2) 形式

```
public synchronized ResultSet getCrossReference
       (String parentCatalog,String parentSchema, String parentTable,
        String foreignCatalog, String foreignSchema, String foreignTable)
       throws SQLException
```

(3) 引数

String parentCatalog：

被参照表のカタログ名を指定します。ただし，指定しても無視されます。

String parentSchema：

被参照表のスキーマ名パターン※を指定します。

String parentTable：

被参照表の表名パターン※を指定します。

String foreignCatalog：

参照表のカタログ名を指定します。ただし，指定しても無視されます。

String foreignSchema：

参照表のスキーマ名パターン※を指定します。

String foreignTable：

参照表の表名パターン※を指定します。

注※

各パターンに指定できる特殊文字については，「8.7.1　DatabaseMetaDataインタフェースのメソッド一覧」の「(4)　パターン文字列中に指定できる特殊文字」を参照してください。

(4) 戻り値

返却されるResultSetオブジェクトの形式を次の表に示します。

表 8-55　返却される ResultSet オブジェクトの形式（getCrossReference メソッドの場合）

列番号	型	列名	列の意味
1	String	PKTABLE_CAT	被参照表のカタログ名 常に空文字が返却されます。
2	String	PKTABLE_SCHEM	被参照表のスキーマ名
3	String	PKTABLE_NAME	被参照表の表名
4	String	PKCOLUMN_NAME	主キーの列名
5	String	FKTABLE_CAT	参照表のカタログ名 常に空文字が返却されます。
6	String	FKTABLE_SCHEM	参照表のスキーマ名
7	String	FKTABLE_NAME	参照表の表名
8	String	FKCOLUMN_NAME	外部キーの列名
9	short	KEY_SEQ	外部キーのシーケンス番号
10	short	UPDATE_RULE	主キー更新時の外部キーに適用される動作 • importedKeyNoAction：主キーは更新できません。ただし，CREATE TABLE 文の参照制約定義で参照制約チェック抑止指定（DISABLE）を指定した場合は，主キーを更新できます。
11	short	DELETE_RULE	主キー削除時の外部キーに適用される動作 • importedKeyNoAction：主キーは削除できません。ただし，CREATE TABLE 文の参照制約定義で参照制約チェック抑止指定（DISABLE）を指定した場合は，主キーを削除できます。
12	String	FK_NAME	参照制約の制約名
13	String	PK_NAME	主キーのインデクス名
14	short	DEFERRABILITY	外部キーに対する制約の評価を，コミットまで延期できるかどうか • importedKeyNotDeferrable：延期できません。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.20　getDatabaseMajorVersion()

(1)　機能

データベース（HADB サーバ）のメジャーバージョンを取得します。

(2)　形式

```
public synchronized int getDatabaseMajorVersion() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

HADB サーバのメジャーバージョンが返却されます。

例えば，HADB サーバのバージョンが 01-00 の場合，int 型の1 が返却されます。

JDBC ドライバが出力する SQL トレースのヘッダ部に表示されるサーバのメジャーバージョンと同じ値が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.21　getDatabaseMinorVersion()

(1)　機能

データベース（HADB サーバ）のマイナーバージョンを取得します。

(2)　形式

```
public synchronized int getDatabaseMinorVersion() throws SQLException
```

(3)　引数

なし。

(4) 戻り値

HADB サーバのマイナーバージョンが返却されます。

例えば，HADB サーバのバージョンが 01-00 の場合，int 型の0 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.22 getDatabaseProductName()

(1) 機能

接続するデータベース（HADB サーバ）の製品名称を取得します。

(2) 形式

```
public synchronized String getDatabaseProductName() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

”Hitachi Advanced Data Binder”が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.23 getDatabaseProductVersion()

(1) 機能

接続するデータベース（HADB サーバ）のバージョンを取得します。

(2) 形式

```
public synchronized String getDatabaseProductVersion() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

"*vv-rr*"の形式（例："01-00"）で HADB サーバのバージョンが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.24 getDefaultTransactionIsolation()

(1) 機能

このデータベースのデフォルトのトランザクション隔離性水準を取得します。

(2) 形式

```
public synchronized int getDefaultTransactionIsolation() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にConnection.TRANSACTION_READ_COMMITTED（トランザクション隔離性水準のデフォルトがREAD COMMITTED）が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.25　getDriverMajorVersion()

(1)　機能

JDBC ドライバのメジャーバージョンを取得します。

(2)　形式

```
public synchronized int getDriverMajorVersion()
```

(3)　引数

なし。

(4)　戻り値

JDBC ドライバのメジャーバージョンが返却されます。

例えば，JDBC ドライバのバージョンが 01-00 の場合，1 が返却されます。

(5)　発生する例外

なし。

8.7.26　getDriverMinorVersion()

(1)　機能

JDBC ドライバのマイナーバージョンを取得します。

(2)　形式

```
public synchronized int getDriverMinorVersion()
```

(3)　引数

なし。

(4)　戻り値

JDBC ドライバのマイナーバージョンが返却されます。

例えば，JDBC ドライバのバージョンが 01-00 の場合，0 が返却されます。

(5) 発生する例外

なし。

8.7.27 getDriverName()

(1) 機能

JDBC ドライバの名前を取得します。

(2) 形式

```
public synchronized String getDriverName() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

”Hitachi Advanced Data Binder JDBC Driver”が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.28 getDriverVersion()

(1) 機能

JDBC ドライバのバージョンを取得します。

(2) 形式

```
public synchronized String getDriverVersion() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

"*vv-rr*"の形式（例："01-00"）で JDBC ドライバのバージョンが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.29 getExportedKeys(String catalog, String schema, String table)

(1) 機能

参照表の外部キーに関する情報を取得します。

このメソッドを実行した HADB ユーザが持っている権限によって，取得できる外部キーに関する情報が変わります。権限と取得できる情報の対応については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2) 形式

```
public synchronized ResultSet getExportedKeys(String catalog, String schema, String table) t
hrows SQLException
```

(3) 引数

String catalog：

被参照表のカタログ名を指定します。ただし，指定しても無視されます。

String schema：

被参照表のスキーマ名パターン※を指定します。

String table：

被参照表の表名パターン※を指定します。

注※

各パターンに指定できる特殊文字については，「8.7.1　DatabaseMetaData インタフェースのメソッド一覧」の「(4)　パターン文字列中に指定できる特殊文字」を参照してください。

(4) 戻り値

返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-56 返却される ResultSet オブジェクトの形式（getExportedKeys メソッドの場合）

列番号	型	列名	列の意味
1	String	PKTABLE_CAT	被参照表のカタログ名 常に空文字が返却されます。
2	String	PKTABLE_SCHEM	被参照表のスキーマ名
3	String	PKTABLE_NAME	被参照表の表名
4	String	PKCOLUMN_NAME	主キーの列名
5	String	FKTABLE_CAT	参照表のカタログ名 常に空文字が返却されます。
6	String	FKTABLE_SCHEM	参照表のスキーマ名
7	String	FKTABLE_NAME	参照表の表名
8	String	FKCOLUMN_NAME	外部キーの列名
9	short	KEY_SEQ	外部キーのシーケンス番号
10	short	UPDATE_RULE	主キー更新時の外部キーに適用される動作 • importedKeyNoAction：主キーは更新できません。ただし，CREATE TABLE 文の参照制約定義で参照制約チェック抑止指定（DISABLE）を指定した場合は，主キーを更新できます。
11	short	DELETE_RULE	主キー削除時の外部キーに適用される動作 • importedKeyNoAction：主キーは削除できません。ただし，CREATE TABLE 文の参照制約定義で参照制約チェック抑止指定（DISABLE）を指定した場合は，主キーを削除できます。
12	String	FK_NAME	参照制約の制約名
13	String	PK_NAME	主キーのインデクス名
14	short	DEFERRABILITY	外部キーに対する制約の評価を，コミットまで延期できるかどうか • importedKeyNotDeferrable：延期できません。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.30 getExtraNameCharacters()

(1) 機能

二重引用符（”）で囲まれていない識別名に使用できるすべての特殊文字を取得します。

(2) 形式

```
public synchronized String getExtraNameCharacters() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

”¥@#”が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.31 getFunctionColumns(String catalog, String schemaPattern, String functionNamePattern, String columnNamePattern)

(1) 機能

関数のパラメタと返される型に関する情報を返却します。

(2) 形式

```
public synchronized ResultSet getFunctionColumns(String catalog, String schemaPattern, Strin
g functionNamePattern, String columnNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。大文字と小文字を区別します。

String functionNamePattern：

関数名パターンを指定します。大文字と小文字を区別します。

String columnNamePattern：

パラメタ名パターンを指定します。大文字と小文字を区別します。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-57 返却される ResultSet オブジェクトの形式

列番号	型	列名	列の意味
1	String	FUNCTION_CAT	カタログ名
2	String	FUNCTION_SCHEM	認可識別子名
3	String	FUNCTION_NAME	関数名
4	String	COLUMN_NAME	列/パラメタ名
5	short	COLUMN_TYPE	列の種類/パラメタ
6	int	DATA_TYPE	パラメタの SQL 型
7	String	TYPE_NAME	パラメタの SQL 型名
8	int	PRECISION	パラメタの精度
9	int	LENGTH	パラメタのサイズ
10	short	SCALE	パラメタの位取り（小数部分の桁数）
11	short	RADIX	パラメタの基数
12	short	NULLABLE	ナル値可否
13	String	REMARKS	パラメタに関するコメント
14	int	CHAR_OCTET_LENGTH	バイナリと文字ベースのパラメタまたは列の最大長
15	int	ORDINAL_POSITION	入力および出力パラメタの 1 から始まる順番 • 関数の戻り値の場合は 0 を返す
16	String	IS_NULLABLE	パラメタまたは列でナル値を許可するかどうか
17	String	SPECIFIC_NAME	この関数をスキーマ内で一意に識別する名前

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.32 getFunctions(String catalog, String schemaPattern, String functionNamePattern)

(1) 機能

関数に関する情報を返却します。

(2) 形式

```
public synchronized ResultSet getFunctions(String catalog, String schemaPattern, String functionNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。大文字と小文字を区別します。

String functionNamePattern：

関数名パターンを指定します。大文字と小文字を区別します。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-58 返却される ResultSet オブジェクトの形式

列番号	型	列名	列の意味
1	String	FUNCTION_CAT	カタログ名
2	String	FUNCTION_SCHEM	認可識別子名
3	String	FUNCTION_NAME	関数名
4	String	REMARKS	関数に関する説明
5	short	FUNCTION_TYPE	関数の種類

列番号	型	列名	列の意味
6	String	SPECIFIC_NAME	この関数をそのスキーマ内で一意に識別する名前

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.33 getIdentifierQuoteString()

(1) 機能

SQL 識別子を引用するのに使用する文字列を取得します。

(2) 形式

```
public synchronized String getIdentifierQuoteString() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

二重引用符（”）が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.34 getImportedKeys(String catalog, String schema, String table)

(1) 機能

被参照表の主キーに関する情報を取得します。

このメソッドを実行したHADBユーザが持っている権限によって，取得できる主キーに関する情報が変わります。権限と取得できる情報の対応については，マニュアル『HADB　システム構築・運用ガイド』の『HADBユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2)　形式

```
public synchronized ResultSet getImportedKeys(String catalog, String schema, String table) t
hrows SQLException
```

(3)　引数

String catalog：

参照表のカタログ名を指定します。ただし，指定しても無視されます。

String schema：

参照表のスキーマ名パターン※を指定します。

String table：

参照表の表名パターン※を指定します。

注※

各パターンに指定できる特殊文字については，「8.7.1　DatabaseMetaDataインタフェースのメソッド一覧」の「(4)　パターン文字列中に指定できる特殊文字」を参照してください。

(4)　戻り値

返却されるResultSetオブジェクトの形式を次の表に示します。

表 8-59　返却されるResultSetオブジェクトの形式（getImportedKeysメソッドの場合）

列番号	型	列名	列の意味
1	String	PKTABLE_CAT	被参照表のカタログ名 常に空文字が返却されます。
2	String	PKTABLE_SCHEM	被参照表のスキーマ名
3	String	PKTABLE_NAME	被参照表の表名
4	String	PKCOLUMN_NAME	主キーの列名
5	String	FKTABLE_CAT	参照表のカタログ名 常に空文字が返却されます。
6	String	FKTABLE_SCHEM	参照表のスキーマ名
7	String	FKTABLE_NAME	参照表の表名
8	String	FKCOLUMN_NAME	外部キーの列名
9	short	KEY_SEQ	外部キーのシーケンス番号

列番号	型	列名	列の意味
10	short	UPDATE_RULE	主キー更新時の外部キーに適用される動作 • importedKeyNoAction：主キーは更新できません。ただし，CREATE TABLE 文の参照制約定義で参照制約チェック抑止指定（DISABLE）を指定した場合は，主キーを更新できます。
11	short	DELETE_RULE	主キー削除時の外部キーに適用される動作 • importedKeyNoAction：主キーは削除できません。ただし，CREATE TABLE 文の参照制約定義で参照制約チェック抑止指定（DISABLE）を指定した場合は，主キーを削除できます。
12	String	FK_NAME	参照制約の制約名
13	String	PK_NAME	主キーのインデクス名
14	short	DEFERRABILITY	外部キーに対する制約の評価を，コミットまで延期できるかどうか • importedKeyNotDeferrable：延期できません。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.35 getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate)

(1) 機能

インデクスに関する情報を取得します。

このメソッドを実行した HADB ユーザが持っている権限によって，取得できるインデクス情報が変わります。権限と取得できるインデクス情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2) 形式

```
public synchronized ResultSet getIndexInfo(String catalog, String schema, String table, bool
ean unique, boolean approximate) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schema：

スキーマ名パターン※を指定します。大文字と小文字を区別します。

String table：

表名パターン※を指定します。大文字と小文字を区別します。

boolean unique：

ユニーク属性を指定します。次に示すどちらかの値を指定してください。

true：ユニークインデクスの情報だけを取得します。

false：すべてのインデクスの情報を取得します。

boolean approximate：

指定する必要はありません。指定しても無視されます。

注※

各パターンに指定できる特殊文字については，「8.7.1 DatabaseMetaData インタフェースのメソッド一覧」の「(4) パターン文字列中に指定できる特殊文字」を参照してください。

(4) 戻り値

ResultSet オブジェクトが返却されます。

返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-60 返却される ResultSet オブジェクトの形式（getIndexInfo メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	カタログ名 常に空の文字列を返します。
2	String	TABLE_SCHEM	スキーマ名
3	String	TABLE_NAME	表名
4	boolean	NON_UNIQUE	インデクスを定義するキー値（インデクスとして定義する 1 つまたは複数の列全体の値）が，すべての行で異なっている場合にfalse，そうでない場合にtrueを返します。
5	String	INDEX_QUALIFIER	インデクスのカタログ名 常に空の文字列を返します。
6	String	INDEX_NAME	インデクス識別子
7	short	TYPE	インデクス種別を返します。

列番号	型	列名	列の意味
			常にDatabaseMetaData.tableIndexOther を返します。
8	short	ORDINAL_POSITION	インデクスを構成する列の順序（インデクスを構成する列名順を識別する番号で，1 から始まる整数）を返します。
9	String	COLUMN_NAME	列名を返します。
10	String	ASC_OR_DESC	B-tree インデクスのキー値の並び順を返します。 • 'A'：ASC（昇順） • 'D'：DESC（降順） B-tree インデクス以外のインデクスの場合は，ナル値を返します。
11	int	CARDINALITY	インデクス内のユニークな値の数 常にナル値を返します。
12	int	PAGES	インデクスで使用しているページ数 常にナル値を返します。
13	String	FILTER_CONDITION	フィルタ条件 常に空の文字列を返します。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.36 getJDBCMajorVersion()

(1) 機能

ドライバの JDBC メジャーバージョンを取得します。

(2) 形式

```
public synchronized int getJDBCMajorVersion() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

JDBC メジャーバージョンが返却されます。4 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.37 getJDBCMinorVersion()

(1) 機能

ドライバの JDBC マイナーバージョンを取得します。

(2) 形式

```
public synchronized int getJDBCMinorVersion() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

JDBC マイナーバージョンが返却されます。

JRE8 版の場合は2 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.38 getMaxBinaryLiteralLength()

(1) 機能

バイナリリテラル中に入れることができる 16 進数の最大文字数を取得します。

(2) 形式

```
public synchronized int getMaxBinaryLiteralLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に64,000 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.39 getMaxCatalogNameLength()

(1) 機能

カタログ名の最大文字数を取得します。

(2) 形式

```
public synchronized int getMaxCatalogNameLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に0 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.40 getMaxCharLiteralLength()

(1) 機能

文字データとして指定できる最大文字数を取得します。

(2) 形式

```
public synchronized int getMaxCharLiteralLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

文字データとして指定できる最大文字数が返却されます。常に32,000 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.41 getMaxColumnNameLength()

(1) 機能

列名に指定できる最大文字数を取得します。

(2) 形式

```
public synchronized int getMaxColumnNameLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

列名に指定できる最大文字数が返却されます。常に100 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.42 getMaxColumnsInGroupBy()

(1) 機能

GROUP BY 句中に指定できるグループ化列の数の最大値を取得します。

(2) 形式

```
public synchronized int getMaxColumnsInGroupBy() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

GROUP BY 句中に指定できるグループ化列の数の最大値が返却されます。常に64 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.43 getMaxColumnsInIndex()

(1) 機能

インデクス構成列数の最大値を取得します。

(2) 形式

```
public synchronized int getMaxColumnsInIndex() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

インデクス構成列数の最大値が返却されます。常に16 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.44 getMaxColumnsInOrderBy()

(1) 機能

ORDER BY 句中に指定できる列数の最大値を取得します。

(2) 形式

```
public synchronized int getMaxColumnsInOrderBy() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

ORDER BY 句中に指定できる列数の最大値が返却されます。常に16 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.45 getMaxColumnsInSelect()

(1) 機能

選択リストに指定できる選択式の最大数を取得します。

(2) 形式

```
public synchronized int getMaxColumnsInSelect() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

選択リストに指定できる選択式の最大数が返却されます。常に4,000 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.46 getMaxColumnsInTable()

(1) 機能

表に定義できる列数の最大値を取得します。

(2) 形式

```
public synchronized int getMaxColumnsInTable() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

表に定義できる列数の最大値が返却されます。常に4,000 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.47 getMaxConnections()

(1) 機能

同時に HADB サーバに接続できる HADB クライアントの最大数を取得します。

(2) 形式

```
public synchronized int getMaxConnections() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

同時に接続できる HADB ユーザの最大数が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.48 getMaxCursorNameLength()

(1) 機能

カーソル名の最大文字数を取得します。

(2) 形式

```
public synchronized int getMaxCursorNameLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

カーソル名の最大文字数が返却されます。常に0 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.49 getMaxIndexLength()

(1) 機能

インデクスのキー長の最大値を取得します。

(2) 形式

```
public synchronized int getMaxIndexLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

インデクスのキー長の最大値が返却されます。常に4,036 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.50 getMaxLogicalLobSize()

(1) 機能

このデータベースで，LOB の論理サイズとして許容される最大バイト数を取得します。

(2) 形式

```
public synchronized long getMaxLogicalLobSize() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に0 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.51　getMaxProcedureNameLength()

(1)　機能

プロシジャ名の最大文字数を取得します。

(2)　形式

```
public synchronized int getMaxProcedureNameLength() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

プロシジャ名の最大文字数が返却されます。常に0 が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.52　getMaxRowSize()

(1)　機能

1 行の最大バイト数を取得します。

(2)　形式

```
public synchronized int getMaxRowSize() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常に0 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.53 getMaxSchemaNameLength()

(1) 機能

スキーマ名の最大文字数を取得します。

(2) 形式

```
public synchronized int getMaxSchemaNameLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

スキーマ名の最大文字数が返却されます。常に100 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.54 getMaxStatementLength()

(1) 機能

SQL 文に指定できる文字列長の最大値を取得します。

(2) 形式

```
public synchronized int getMaxStatementLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

SQL 文に指定できる文字列長の最大値が返却されます。常に16,000,000（単位：文字）が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.55 getMaxStatements()

(1) 機能

同時実行できる SQL 文の最大数を取得します。

lConnection オブジェクトで作成できるStatement 系オブジェクトの最大数を取得します。

(2) 形式

```
public synchronized int getMaxStatements() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

同時実行できる SQL 文の最大数が返却されます。常に4,095 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.56 getMaxTableNameLength()

(1) 機能

表名に指定できる最大文字数を取得します。

(2) 形式

```
public synchronized int getMaxTableNameLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

表名に指定できる最大文字数が返却されます。常に100 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.57 getMaxTablesInSelect()

(1) 機能

SELECT 文中に指定できる表数の最大値を取得します。

(2) 形式

```
public synchronized int getMaxTablesInSelect() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

SELECT 文中に指定できる表数の最大値が返却されます。常に64 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.58 getMaxUserNameLength()

(1) 機能

認可識別子の最大文字数を取得します。

(2) 形式

```
public synchronized int getMaxUserNameLength() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

認可識別子の最大文字数が返却されます。常に100 が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.59 getNumericFunctions()

(1) 機能

使用できる数学関数をコンマで区切ったリストで取得します。

(2) 形式

```
public synchronized String getNumericFunctions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.60 getPrimaryKeys(String catalog, String schema, String table)

(1) 機能

指定された表の主キー列の記述を取得します。

このメソッドを実行した HADB ユーザが持っている権限によって，取得できる主キーの情報が変わります。権限と取得できる主キーの情報については，マニュアル『HADB システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2) 形式

```
public synchronized ResultSet getPrimaryKeys(String catalog, String schema, String table) th
rows SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schema：

スキーマ名パターン※を指定します。

String table：

表名パターン※を指定します。

注※

各パターンに指定できる特殊文字については，「8.7.1 DatabaseMetaData インタフェースのメソッド一覧」の「(4) パターン文字列中に指定できる特殊文字」を参照してください。

(4) 戻り値

ResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-61 返却される ResultSet オブジェクトの形式（getPrimaryKeys メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	カタログ名 常に空の文字列を返します。

列番号	型	列名	列の意味
2	String	TABLE_SCHEM	スキーマ名
3	String	TABLE_NAME	表名
4	String	COLUMN_NAME	列名
5	short	KEY_SEQ	主キー内で付けられる列順の番号
6	String	PK_NAME	主キー名

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.61 getProcedureColumns(String catalog, String schemaPattern, String procedureNamePattern, String columnNamePattern)

(1) 機能

ストアドプロシジャパラメタに関する記述を取得します。

(2) 形式

```
public synchronized ResultSet getProcedureColumns(String catalog, String schemaPattern, Stri
ng procedureNamePattern, String columnNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。ただし，指定しても無視されます。

String procedureNamePattern：

プロシジャ名パターンを指定します。ただし，指定しても無視されます。

String columnNamePattern：

パラメタ名パターンを指定します。ただし，指定しても無視されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-62 返却される ResultSet オブジェクトの形式（getProcedureColumns メソッドの場合）

列番号	型	列名	列の意味
1	String	PROCEDURE_CAT	カタログ名
2	String	PROCEDURE_SCHEM	スキーマ名
3	String	PROCEDURE_NAME	プロシジャ名
4	String	COLUMN_NAME	パラメタ名
5	short	COLUMN_TYPE	パラメタの種類
6	int	DATA_TYPE	パラメタの SQL 型
7	String	TYPE_NAME	パラメタの SQL 型名
8	int	PRECISION	パラメタの精度
9	int	LENGTH	パラメタのサイズ
10	short	SCALE	パラメタの位取り
11	short	RADIX	パラメタの基数
12	short	NULLABLE	ナル値を使用できるかどうかを返します。
13	String	REMARKS	パラメタに関するコメント
14	String	COLUMN_DEF	列のデフォルト値
15	int	SQL_DATA_TYPE	将来使用するための予約
16	int	SQL_DATETIME_SUB	将来使用するための予約
17	int	CHAR_OCTET_LENGTH	バイナリおよび文字ベースのパラメタ，または列の最大長
18	int	ORDINAL_POSITION	入力および出力パラメタの 1 から始まる順番
19	String	IS_NULLABLE	パラメタまたは列でナル値を許可するかどうか
20	String	SPECIFIC_NAME	このプロシジャを一意に識別する名前

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.62 getProcedures(String catalog, String schemaPattern, String procedureNamePattern)

(1) 機能

ストアドプロシジャに関する記述を取得します。

(2) 形式

```
public synchronized ResultSet getProcedures(String catalog, String schemaPattern, String pro
cedureNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。ただし，指定しても無視されます。

String procedureNamePattern：

プロシジャ名パターンを指定します。ただし，指定しても無視されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-63 返却される ResultSet オブジェクトの形式（getProcedures メソッドの場合）

列番号	型	列名	列の意味
1	String	PROCEDURE_CAT	カタログ名
2	String	PROCEDURE_SCHEM	スキーマ名
3	String	PROCEDURE_NAME	プロシジャ名
4	String	RESERVE1	将来使用するための予約
5	String	RESERVE2	将来使用するための予約
6	String	RESERVE3	将来使用するための予約
7	String	REMARKS	プロシジャの説明文
8	short	PROCEDURE_TYPE	プロシジャの種類
9	String	SPECIFIC_NAME	このプロシジャを一意に識別する名前

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.63 getProcedureTerm()

(1) 機能

procedure に対する推奨用語を取得します。

(2) 形式

```
public synchronized String getProcedureTerm() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に空の文字列が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.64 getPseudoColumns(String catalog,String schemaPattern,String tableNamePattern,String columnNamePattern)

(1) 機能

指定されたカタログおよびスキーマ内の特定の表で使用できる擬似列または隠し列の説明を取得します。ただし，このメソッドでは，引数の指定はすべて無視され，常に空の結果セットが返却されます。

(2) 形式

```
public synchronized ResultSet getPseudoColumns(String catalog,String schemaPattern,String ta
bleNamePattern,String columnNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。ただし，指定しても無視されます。

String tableNamePattern：

表名パターンを指定します。ただし，指定しても無視されます。

String columnNamePattern：

列名パターンを指定します。ただし，指定しても無視されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-64 返却される ResultSet オブジェクトの形式（getPseudoColumns メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	カタログ名
2	String	TABLE_SCHEM	スキーマ名
3	String	TABLE_NAME	表名
4	String	COLUMN_NAME	列名
5	int	DATA_TYPE	java.sql.Types からの SQL 型
6	int	COLUMN_SIZE	列サイズ
7	int	DECIMAL_DIGITS	小数点以下の桁数
8	int	NUM_PREC_RADIX	基数
9	String	COLUMN_USAGE	列の許可された使用方法
10	String	REMARKS	コメント記述列
11	int	CHAR_OCTET_LENGTH	char 型の列の最大バイト数
12	String	IS_NULLABLE	ナル値が使用できるかどうか

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.65 getResultSetHoldability()

(1) 機能

ResultSet オブジェクトのデフォルトの保持機能を取得します。

(2) 形式

```
public synchronized int getResultSetHoldability() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にResultSet.HOLD_CURSORS_OVER_COMMIT が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.66 getRowIdLifetime()

(1) 機能

RowId 型をサポートしているかどうかを示します。また，サポートしている場合はRowId オブジェクトが有効である期間も示します。

(2) 形式

```
public synchronized RowIdLifetime getRowIdLifetime() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にRowIdLifetime.ROWID_UNSUPPORTED を返却します。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.67 getSchemas()

(1) 機能

スキーマ名を取得します。

このメソッドを実行した HADB ユーザが持っている権限によって，取得できるスキーマの情報が変わります。権限と取得できるスキーマの情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2) 形式

```
public synchronized ResultSet getSchemas() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

ResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-65　返却される ResultSet オブジェクトの形式（getSchemas メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_SCHEM	スキーマ名
2	String	TABLE_CATALOG	常に空の文字列を返します。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.68　getSchemas(String catalog, String schemaPattern)

(1)　機能

スキーマ名を取得します。

このメソッドを実行した HADB ユーザが持っている権限によって，取得できるスキーマの情報が変わります。権限と取得できるスキーマの情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2)　形式

```
public synchronized ResultSet getSchemas(String catalog, String schemaPattern) throws SQLExc
eption
```

(3)　引数

String catalog：

　カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

　スキーマ名パターンを指定します。大文字と小文字は区別されます。

　パターンに指定できる特殊文字については，「8.7.1　DatabaseMetaData インタフェースのメソッド一覧」の「(4)　パターン文字列中に指定できる特殊文字」を参照してください。

(4)　戻り値

ResultSet オブジェクトが返却されます。

返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-66　返却される ResultSet オブジェクトの形式

列番号	型	列名	列の意味
1	String	TABLE_SCHEM	スキーマ名
2	String	TABLE_CATALOG	常に空の文字列を返します。

8.7.69 getSchemaTerm()

(1) 機能

schema に対する推奨用語を取得します。

(2) 形式

```
public synchronized String getSchemaTerm() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

”schema”が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.70 getSearchStringEscape()

(1) 機能

ワイルドカード文字をエスケープするために使用する文字列を取得します。

このメソッドを実行すると，表一覧や，列一覧などの一覧情報を取得するDatabaseMetaData インタフェースのメソッドで，パターン文字列を指定する引数に指定するワイルドカード文字をエスケープするために使用する文字列を取得できます。

(2) 形式

```
public synchronized String getSearchStringEscape() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常に"¥"が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.71 getSQLKeywords()

(1) 機能

データベース固有の SQL キーワードであり，かつ SQL:2003 のキーワードではないすべてのキーワードを，コンマで区切ったリストで取得します。

(2) 形式

```
public synchronized String getSQLKeywords() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

予約語については，マニュアル『HADB　SQL リファレンス』の『予約語』を参照してください。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

(6) 留意事項

このメソッドで返却する予約語には，削除した予約語も含まれるため，注意してください。

8.7.72 getSQLStateType()

(1) 機能

SQLException クラスのgetSQLState メソッドによって返されるSQLSTATE が X/Open（現在は Open Group）の SQL CLI であるか SQL:2003 であるかを取得します。

(2) 形式

```
public synchronized int getSQLStateType() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にsqlStateSQL が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.73 getStringFunctions()

(1) 機能

使用できる文字列関数をコンマで区切ったリストで取得します。

(2) 形式

```
public synchronized String getStringFunctions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.74 getSuperTables(String catalog, String schemaPattern, String tableNamePattern)

(1) 機能

特定のスキーマで定義されているテーブル階層の情報を取得します。

(2) 形式

```
public synchronized ResultSet getSuperTables(String catalog, String schemaPattern, String ta
bleNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。ただし，指定しても無視されます。

String tableNamePattern：

表名パターンを指定します。ただし，指定しても無視されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-67 返却される ResultSet オブジェクトの形式（getSuperTables メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	カタログ名
2	String	TABLE_SCHEM	スキーマ名
3	String	TABLE_NAME	型名
4	String	SUPERTABLE_NAME	スーパタイプ名

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.75 getSuperTypes(String catalog, String schemaPattern, String typeNamePattern)

(1) 機能

特定のスキーマで定義されているユーザ定義型（UDT）階層の情報を取得します。

(2) 形式

```
public synchronized ResultSet getSuperTypes(String catalog, String schemaPattern, String typ
eNamePattern) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。ただし，指定しても無視されます。

String typeNamePattern：

UDT 名パターンを指定します。ただし，指定しても無視されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-68 返却される ResultSet オブジェクトの形式（getSuperTypes メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	UDT のカタログ名
2	String	TABLE_SCHEM	UDT のスキーマ名
3	String	TABLE_NAME	UDT の型名
4	String	SUPERTYPE_CAT	直接のスーパタイプのカタログ名
5	String	SUPERTYPE_SCHEM	直接のスーパタイプのスキーマ名

列番号	型	列名	列の意味
6	String	SUPERTYPE_NAME	直接のスーパタイプのスーパタイプ名

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.76 getSystemFunctions()

(1) 機能

使用できるシステム関数をコンマで区切ったリストで取得します。

(2) 形式

```
public synchronized String getSystemFunctions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.77 getTablePrivileges(String catalog, String schemaPattern, String tableNamePattern)

(1) 機能

表に対するアクセス権限に関する情報を取得します。

このメソッドを実行したHADBユーザが持っている権限によって，取得できるアクセス権限の情報が変わります。権限と取得できる表の情報の対応については，マニュアル『HADB　システム構築・運用ガイド』の『HADBユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2)　形式

```
public synchronized ResultSet getTablePrivileges(String catalog, String schemaPattern, Strin
g tableNamePattern) throws SQLException
```

(3)　引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターン※を指定します。

String tableNamePattern：

表名パターン※を指定します。

注※

各パターンに指定できる特殊文字については，「8.7.1　DatabaseMetaDataインタフェースのメソッド一覧」の「(4)　パターン文字列中に指定できる特殊文字」を参照してください。

(4)　戻り値

返却されるResultSetオブジェクトの形式を次の表に示します。

表8-69　返却されるResultSetオブジェクトの形式（getTablePrivilegesメソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	カタログ名 常に空文字を返します。
2	String	TABLE_SCHEM	スキーマ名
3	String	TABLE_NAME	表名
4	String	GRANTOR	アクセス権限の付与者
5	String	GRANTEE	アクセス権限の被付与者
6	String	PRIVILEGE	許可されているアクセス権限名を返します。 • "SELECT"：SELECT権限 • "INSERT"：INSERT権限 • "UPDATE"：UPDATE権限 • "DELETE"：DELETE権限 • "TRUNCATE"：TRUNCATE権限

列番号	型	列名	列の意味
			• ”REFERENCES”：REFERENCES 権限 • ”IMPORT TABLE”：IMPORT TABLE 権限 • ”REBUILD INDEX”：REBUILD INDEX 権限 • ”GET COSTINFO”：GET COSTINFO 権限 • ”EXPORT TABLE”：EXPORT TABLE 権限 • ”MERGE CHUNK”：MERGE CHUNK 権限 • ”CHANGE CHUNK COMMENT”：CHANGE CHUNK COMMENT 権限 • ”CHANGE CHUNK STATUS”：CHANGE CHUNK STATUS 権限 • ”ARCHIVE CHUNK”：ARCHIVE CHUNK 権限 • ”UNARCHIVE CHUNK”：UNARCHIVE CHUNK 権限
7	String	IS_GRANTABLE	アクセス権限の被付与者が，ほかの HADB ユーザにアクセス権限を与えることができるかどうかを返します。 • "YES"：ほかの HADB ユーザにアクセス権限を与えられます。 • "NO"：ほかの HADB ユーザにアクセス権限を与えられません。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.78 getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)

(1) 機能

表に関する情報を取得します。

このメソッドを実行した HADB ユーザが持っている権限によって，取得できる表の情報が変わります。権限と取得できる表の情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

(2) 形式

```
public synchronized ResultSet getTables(String catalog, String schemaPattern, String tableNa
mePattern, String[] types) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターン※を指定します。大文字と小文字は区別されます。

String tableNamePattern：

表名パターン※を指定します。大文字と小文字は区別されます。

String[] types：

表の型のリストを指定します。getTableTypes メソッドによって返却される表の型を指定します。大文字と小文字は区別されます。

null を指定した場合，すべての表の種類が指定されたと仮定されます。

注※

各パターンに指定できる特殊文字については，「8.7.1 DatabaseMetaData インタフェースのメソッド一覧」の「(4) パターン文字列中に指定できる特殊文字」を参照してください。

(4) 戻り値

ResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-70 返却される ResultSet オブジェクトの形式（getTables メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_CAT	常に空の文字列を返します。
2	String	TABLE_SCHEM	スキーマ名
3	String	TABLE_NAME	表名
4	String	TABLE_TYPE	表の種類 • TABLE：実表 • VIEW：ビュー表 • SYSTEM TABLE：ディクショナリ表，またはシステム表
5	String	REMARKS	常に空の文字列を返します。
6	String	TYPE_CAT	常に空の文字列を返します。
7	String	TYPE_SCHEM	常に空の文字列を返します。
8	String	TYPE_NAME	常に空の文字列を返します。
9	String	SELF_REFERENCING_COL_NAME	常に空の文字列を返します。
10	String	REF_GENERATION	常に空の文字列を返します。

(5) 発生する例外

次に示す場合にSQLExceptionが投入されます。

- このメソッドを実行する前に，Connectionオブジェクトがクローズされている場合
- 引数String[] typesの1つ以上の要素がnullである場合
- 引数String[] typesの1つ以上の要素が，次のすべての文字列に該当しない場合
  "TABLE"，"VIEW"，"SYSTEM TABLE"

8.7.79 getTableTypes()

(1) 機能

表の種類を取得します。

(2) 形式

```
public synchronized ResultSet getTableTypes() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

ResultSetオブジェクトが返却されます。返却されるResultSetオブジェクトの形式を次の表に示します。

表 8-71 返却される ResultSet オブジェクトの形式（getTableTypes メソッドの場合）

列番号	型	列名	列の意味
1	String	TABLE_TYPE	表の種類 • TABLE：実表 • VIEW：ビュー表 • SYSTEM TABLE：ディクショナリ表，およびシステム表

(5) 発生する例外

このメソッドを実行する前に，Connectionオブジェクトがクローズされている場合，SQLExceptionが投入されます。

8.7.80　getTimeDateFunctions()

(1)　機能

使用できる時間関数と日付関数をコンマで区切ったリストで取得します。

(2)　形式

```
public synchronized String getTimeDateFunctions() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

String オブジェクトが返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.81　getTypeInfo()

(1)　機能

標準 SQL の型に関する情報を取得します。

(2)　形式

```
public synchronized ResultSet getTypeInfo() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

ResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-72　返却される ResultSet オブジェクトの形式（getTypeInfo メソッドの場合）

列番号	型	列名	列の意味
1	String	TYPE_NAME	型名
2	short	DATA_TYPE	java.sql.Types の SQL データ型
3	int	PRECISION	最大の精度
4	String	LITERAL_PREFIX	リテラルを引用するのに使用する接頭辞
5	String	LITERAL_SUFFIX	リテラルを引用するのに使用する接尾辞
6	String	CREATE_PARAMS	型の作成に使用するパラメタ
7	short	NULLABLE	この型にナル値を使用できるかどうかを返します。 常にtypeNullableUnknown を返します。
8	boolean	CASE_SENSITIVE	大文字と小文字を区別するかどうかを返します。 • true：文字列系データ型 • false：そのほかのデータ型
9	short	SEARCHABLE	この型に"WHERE"を使用できるかどうかを返します。 常にtypeSearchable を返します。
10	boolean	UNSIGNED_ATTRIBUTE	符号なし属性かどうかを返します。 数値データは符号ありのためfalse，そのほかのデータ型は符号の有無に関係ないためfalse を返します。
11	boolean	FIXED_PREC_SCALE	通貨の値になれるかどうかを返します。 常にfalse を返します。
12	boolean	AUTO_INCREMENT	自動インクリメントの値に使用できるかどうかを返します。 常にfalse を返します。
13	String	LOCAL_TYPE_NAME	型名の地域対応されたバージョン 型名と同じものを返します。
14	short	MINIMUM_SCALE	サポートされる最小スケール
15	short	MAXIMUM_SCALE	サポートされる最大スケール
16	int	SQL_DATA_TYPE	常にナル値を返します。
17	int	SQL_DATETIME_SUB	常にナル値を返します。
18	int	NUM_PREC_RADIX	数値データは10，それ以外は0

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.82 getUDTs(String catalog, String schemaPattern, String typeNamePattern, int[] types)

(1) 機能

ユーザ定義型（UDT）に関する情報を取得します。

(2) 形式

```
public synchronized ResultSet getUDTs(String catalog, String schemaPattern, String typeNameP
attern, int[] types) throws SQLException
```

(3) 引数

String catalog：

カタログ名を指定します。ただし，指定しても無視されます。

String schemaPattern：

スキーマ名パターンを指定します。大文字と小文字は区別されます。

String tableNamePattern：

表名パターンを指定します。大文字と小文字は区別されます。

int[] types：

ユーザ定義型のリストを指定します。大文字と小文字は区別されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-73 返却される ResultSet オブジェクトの形式（getUDTs メソッドの場合）

列番号	型	列名	列の意味
1	String	TYPE_CAT	カタログ名
2	String	TYPE_SCHEM	スキーマ名
3	String	TYPE_NAME	型名
4	String	CLASS_NAME	Java のクラス名
5	int	DATA_TYPE	java.sql.Types で定義されている型値
6	String	REMARKS	型に関する説明
7	short	BASE_TYPE	DISTINCT 型のソースの型の型コード，または java.sql.Types で定義される構造型の

列番号	型	列名	列の意味
			SELF_REFERENCING_COLUMN のユーザ生成参照型を実装する型の型コード

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.83 getURL()

(1) 機能

HADB サーバの接続先情報を指定した URL を取得します。

(2) 形式

```
public synchronized String getURL() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

接続先情報を指定した URL がない場合は，null が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.84 getUserName()

(1) 機能

HADB サーバに接続した認可識別子を取得します。

(2) 形式

```
public synchronized String getUserName() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

String オブジェクトが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.85 getVersionColumns(String catalog, String schema, String table)

(1) 機能

行の任意の値が変更された場合に，自動的に更新される表の列に関する記述を取得します。

(2) 形式

```
public synchronized ResultSet getVersionColumns(String catalog, String schema, String table)
 throws SQLException
```

(3) 引数

String catalog：

　カタログ名を指定します。ただし，指定しても無視されます。

String schema：

　スキーマ名を指定します。大文字と小文字は区別されます。

String table：

　表名を指定します。大文字と小文字は区別されます。

(4) 戻り値

常に検索結果行数 0 のResultSet オブジェクトが返却されます。返却されるResultSet オブジェクトの形式を次の表に示します。

表 8-74 返却される ResultSet オブジェクトの形式（getVersionColumns メソッドの場合）

列番号	型	列名	列の意味
1	short	SCOPE	未使用
2	String	COLUMN_NAME	列名
3	int	DATA_TYPE	SQL 型
4	String	TYPE_NAME	型名
5	int	COLUMN_SIZE	精度
6	int	BUFFER_LENGTH	列値のバイト数
7	short	DECIMAL_DIGITS	小数点以下の桁数
8	short	PSEUDO_COLUMN	擬似列かどうか

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.86 insertsAreDetected(int type)

(1) 機能

ResultSet クラスのrowInserted メソッドを呼び出すことによって可視の行が挿入されたことを検出できるかどうかを取得します。

(2) 形式

```
public synchronized boolean insertsAreDetected(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE

- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.87 isCatalogAtStart()

(1) 機能

完全修飾された表名の開始部分（または終了部分）にカタログが現れるかどうかを取得します。

(2) 形式

```
public synchronized boolean isCatalogAtStart() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.88 isReadOnly()

(1) 機能

読み取り専用モードかどうかを取得します。

(2) 形式

```
public synchronized boolean isReadOnly() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.89 locatorsUpdateCopy()

(1) 機能

LOB への変更が，コピーに対して行われたのか，LOB に直接行われたのかを示します。

(2) 形式

```
public synchronized boolean locatorsUpdateCopy() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.90　nullPlusNonNullIsNull()

(1)　機能

ナル値と非ナル値の連結をナル値とするかどうかを取得します。

(2)　形式

```
public synchronized boolean nullPlusNonNullIsNull() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.91　nullsAreSortedAtEnd()

(1)　機能

ナル値が，終了時にソート順に関係なくソートされるかどうかを取得します。

(2)　形式

```
public synchronized boolean nullsAreSortedAtEnd() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.92 nullsAreSortedAtStart()

(1) 機能

ナル値が，開始時にソート順に関係なくソートされるかどうかを取得します。

(2) 形式

```
public synchronized boolean nullsAreSortedAtStart() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.93 nullsAreSortedHigh()

(1) 機能

ナル値が高位にソートされるかどうかを取得します。

(2) 形式

```
public synchronized boolean nullsAreSortedHigh() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.94 nullsAreSortedLow()

(1) 機能

ナル値が下位にソートされるかどうかを取得します。

(2) 形式

```
public synchronized boolean nullsAreSortedLow() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.95 othersDeletesAreVisible(int type)

(1) 機能

ほかで行われた削除が可視かどうかを取得します。

(2) 形式

```
public synchronized boolean othersDeletesAreVisible(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.96 othersInsertsAreVisible(int type)

(1) 機能

ほかで行われた挿入が可視かどうかを取得します。

(2) 形式

```
public synchronized boolean othersInsertsAreVisible(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.97 othersUpdatesAreVisible(int type)

(1) 機能

ほかで行われた更新が可視かどうかを取得します。

(2) 形式

```
public synchronized boolean othersUpdatesAreVisible(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.98 ownDeletesAreVisible(int type)

(1) 機能

結果セット自身の削除が可視かどうかを取得します。

(2) 形式

```
public synchronized boolean ownDeletesAreVisible(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.99 ownInsertsAreVisible(int type)

(1) 機能

結果セット自身の挿入が可視かどうかを取得します。

(2) 形式

```
public synchronized boolean ownInsertsAreVisible(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.100 ownUpdatesAreVisible(int type)

(1) 機能

結果セット自身の更新が可視かどうかを取得します。

(2) 形式

```
public synchronized boolean ownUpdatesAreVisible(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.101　storesLowerCaseIdentifiers()

(1)　機能

大文字と小文字が混在する二重引用符なしの SQL 識別子を，大文字と小文字を区別しないで処理し，小文字で格納するかどうかを取得します。

(2)　形式

```
public synchronized boolean storesLowerCaseIdentifiers() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.102　storesLowerCaseQuotedIdentifiers()

(1)　機能

大文字と小文字が混在する二重引用符付きの SQL 識別子を，大文字と小文字を区別しないで処理し，小文字で格納するかどうかを取得します。

(2)　形式

```
public synchronized boolean storesLowerCaseQuotedIdentifiers() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.103 storesMixedCaseIdentifiers()

(1) 機能

大文字と小文字が混在する二重引用符なしの SQL 識別子を，大文字と小文字を区別しないで処理し，大文字と小文字混在で格納するかどうかを取得します。

(2) 形式

```
public synchronized boolean storesMixedCaseIdentifiers() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.104 storesMixedCaseQuotedIdentifiers()

(1) 機能

大文字と小文字が混在する二重引用符付きの SQL 識別子を，大文字と小文字を区別しないで処理し，大文字と小文字混在で格納するかどうかを取得します。

(2) 形式

```
public synchronized boolean storesMixedCaseQuotedIdentifiers() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.105 storesUpperCaseIdentifiers()

(1) 機能

大文字と小文字が混在する二重引用符なしの SQL 識別子を，大文字と小文字を区別しないで処理し，大文字で格納するかどうかを取得します。

(2) 形式

```
public synchronized boolean storesUpperCaseIdentifiers() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.106　storesUpperCaseQuotedIdentifiers()

(1)　機能

大文字と小文字が混在する二重引用符付きの SQL 識別子を，大文字と小文字を区別しないで処理し，大文字で格納するかどうかを取得します。

(2)　形式

```
public synchronized boolean storesUpperCaseQuotedIdentifiers() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.107　supportsAlterTableWithAddColumn()

(1)　機能

列追加のあるALTER TABLE がサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsAlterTableWithAddColumn() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.108 supportsAlterTableWithDropColumn()

(1) 機能

列削除のあるALTER TABLE がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsAlterTableWithDropColumn() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.109 supportsANSI92EntryLevelSQL()

(1) 機能

ANSI92 エントリレベルの SQL 文法がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsANSI92EntryLevelSQL() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.110 supportsANSI92FullSQL()

(1) 機能

ANSI92 完全レベルの SQL 文法がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsANSI92FullSQL() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.111 supportsANSI92IntermediateSQL()

(1) 機能

ANSI92 中間レベルの SQL 文法がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsANSI92IntermediateSQL() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.112 supportsBatchUpdates()

(1) 機能

バッチ更新がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsBatchUpdates() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.113 supportsCatalogsInDataManipulation()

(1) 機能

データ操作文でカタログ名を使用できるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsCatalogsInDataManipulation() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.114 supportsCatalogsInIndexDefinitions()

(1) 機能

インデクス定義文でカタログ名を使用できるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsCatalogsInIndexDefinitions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.115　supportsCatalogsInPrivilegeDefinitions()

(1)　機能

権限付与の定義文（GRANT 文），または権限取り消しの定義文（REVOKE 文）で，カタログ名を使用できるかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsCatalogsInPrivilegeDefinitions() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.116　supportsCatalogsInProcedureCalls()

(1)　機能

プロシジャ呼び出し文でカタログ名を使用できるかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsCatalogsInProcedureCalls() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.117 supportsCatalogsInTableDefinitions()

(1) 機能

表定義文でカタログ名を使用できるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsCatalogsInTableDefinitions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.118 supportsColumnAliasing()

(1) 機能

列の別名がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsColumnAliasing() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.119 supportsConvert()

(1) 機能

SQL の型間のCONVERT 関数がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsConvert() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.120 supportsConvert(int fromType, int toType)

(1) 機能

指定された SQL の型間のCONVERT 関数がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsConvert(int fromType, int toType) throws SQLException
```

(3) 引数

int fromType：

変換元の SQL 型（java.sql.Types の SQL 型）

int toType：

変換先の SQL 型（java.sql.Types の SQL 型）

(4) 戻り値

次に示すどちらかの値が返却されます。

- true：サポートしています。
- false：サポートしていません。

変換元の型と変換先の型の組み合わせによる戻り値を次の表に示します。

表 8-75 変換元の型と変換先の型の組み合わせによる戻り値

変換元の型 java.sql.Typesの型	変換先の型 java.sql.Typesの型																									
	BIT ※1	TINYINT ※1	SMALLINT ※1	INTEGER ※2	BIGINT ※2	FLOAT ※2	REAL ※1	DOUBLE ※2	NUMERIC ※2	DECIMAL※2	CHAR ※2	VARCHAR ※2	LONGVARCHAR ※1	DATE ※2	TIME ※2	TIMESTAMP ※2	BINARY ※2	VARBINARY ※2	LONGVARBINARY ※1	JAVA_OBJECT ※1	STRUCT ※1	ARRAY ※1	BLOB ※1	CLOB ※1	REF ※1
BIT ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
TINYINT ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
SMALLINT ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
INTEGER ※2	－	－	－	○	○	○	－	○	○	○	○	○	－	△	－	△	－	－	－	－	－	－	－	－	－
BIGINT ※2	－	－	－	○	○	○	－	○	○	○	○	○	－	△	－	△	－	－	－	－	－	－	－	－	－
FLOAT ※2	－	－	－	○	○	○	－	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－
REAL ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
DOUBLE ※2	－	－	－	○	○	○	－	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－
NUMERIC ※2	－	－	－	○	○	○	－	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－
DECIMAL ※2	－	－	－	○	○	○	－	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－
CHAR ※2	－	－	－	○	○	○	－	○	○	○	○	○	－	○	○	○	○	○	－	－	－	－	－	－	－
VARCHAR ※2	－	－	－	○	○	○	－	○	○	○	○	○	－	○	○	○	○	○	－	－	－	－	－	－	－
LONGVARCHAR ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
DATE ※2	－	－	－	△	△	－	－	－	－	－	○	○	－	○	－	○	－	－	－	－	－	－	－	－	－
TIME ※2	－	－	－	－	－	－	－	－	－	－	○	○	－	－	○	－	－	－	－	－	－	－	－	－	－
TIMESTAMP ※2	－	－	－	△	△	－	－	－	－	－	○	○	－	○	－	○	－	－	－	－	－	－	－	－	－
BINARY ※2	－	－	－	－	－	－	－	－	－	－	○	○	－	－	－	－	○	○	－	－	－	－	－	－	－
VARBINARY ※2	－	－	－	－	－	－	－	－	－	－	○	○	－	－	－	－	○	○	－	－	－	－	－	－	－
LONGVARBINARY ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
JAVA_OBJECT ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
STRUCT ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
ARRAY ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
BLOB ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
CLOB ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
REF ※1	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－

（凡例）

○：true(サポート)

△：true(通算日変換としてサポート)

－：false(未サポート)

注※1

HADB サーバに対応するデータ型がありません。

注※2

対応する HADB サーバのデータ型については，「7.6.1　データ型のマッピング」の「(1)　HADB のデータ型と JDBC の SQL データ型の対応」を参照してください。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.121　supportsCoreSQLGrammar()

(1)　機能

ODBC Core SQL 文法がサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsCoreSQLGrammar() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.122　supportsCorrelatedSubqueries()

(1)　機能

外への参照列を含む副問合せがサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsCorrelatedSubqueries() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.123　supportsDataDefinitionAndDataManipulationTransactions()

(1)　機能

トランザクションで，データ定義文とデータ操作文の両方がサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsDataDefinitionAndDataManipulationTransactions() throws S
QLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.124 supportsDataManipulationTransactionsOnly()

(1) 機能

トランザクションでデータ操作文だけがサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsDataManipulationTransactionsOnly() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.125 supportsDifferentTableCorrelationNames()

(1) 機能

表相関名がサポートされている場合，表名と異なる名前であるという制限を付けるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsDifferentTableCorrelationNames() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.126 supportsExpressionsInOrderBy()

(1) 機能

ORDER BY リスト中で値式がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsExpressionsInOrderBy() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.127 supportsExtendedSQLGrammar()

(1) 機能

ODBC Extended SQL 文法がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsExtendedSQLGrammar() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.128 supportsFullOuterJoins()

(1) 機能

完全な入れ子の外結合がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsFullOuterJoins() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.129 supportsGetGeneratedKeys()

(1) 機能

文が実行されたあとに自動生成キーを取得できるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsGetGeneratedKeys() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.130 supportsGroupBy()

(1) 機能

GROUP BY 句がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsGroupBy() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.131　supportsGroupByBeyondSelect()

(1)　機能

SELECT 文中のすべての列がGROUP BY 句に含まれるという条件で，このデータベースによって，GROUP BY 句でSELECT 文中にない列の使用がサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsGroupByBeyondSelect() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.132　supportsGroupByUnrelated()

(1)　機能

GROUP BY 句でSELECT 文中にない列の使用がサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsGroupByUnrelated() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.133 supportsIntegrityEnhancementFacility()

(1) 機能

SQL Integrity Enhancement Facility がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsIntegrityEnhancementFacility() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.134 supportsLikeEscapeClause()

(1) 機能

LIKE 述語のエスケープ句がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsLikeEscapeClause() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.135 supportsLimitedOuterJoins()

(1) 機能

外結合に関し，制限されたサポートが提供されるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsLimitedOuterJoins() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.136 supportsMinimumSQLGrammar()

(1) 機能

ODBC Minimum SQL 文法がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsMinimumSQLGrammar() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.137 supportsMixedCaseIdentifiers()

(1) 機能

大文字と小文字が混在する二重引用符なしの SQL 識別子を，大文字と小文字を区別して処理し，大文字と小文字混在で格納するかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsMixedCaseIdentifiers() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.138　supportsMixedCaseQuotedIdentifiers()

(1)　機能

大文字と小文字が混在する二重引用符付きの SQL 識別子を，大文字と小文字を区別して処理し，結果として大文字と小文字混在で格納するかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsMixedCaseQuotedIdentifiers() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.139　supportsMultipleOpenResults()

(1)　機能

CallableStatement オブジェクトから同時に返された複数のResultSet オブジェクトを持つことが可能かどうかを取得します。

(2)　形式

```
public synchronized boolean supportsMultipleOpenResults() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.140 supportsMultipleResultSets()

(1) 機能

execute メソッドの単一の呼び出しからの複数のResultSet オブジェクトの取得がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsMultipleResultSets() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.141 supportsMultipleTransactions()

(1) 機能

一度に複数のトランザクションを（異なった接続で）オープンできるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsMultipleTransactions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.142 supportsNamedParameters()

(1) 機能

CALL 文への名前付きパラメタがサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsNamedParameters() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.143 supportsNonNullableColumns()

(1) 機能

列を非ナル値として定義できるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsNonNullableColumns() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.144 supportsOpenCursorsAcrossCommit()

(1) 機能

コミット間でカーソルがオープンされたままの状態がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsOpenCursorsAcrossCommit() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.145　supportsOpenCursorsAcrossRollback()

(1)　機能

ロールバック間でカーソルがオープンされたままの状態がサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsOpenCursorsAcrossRollback() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.146　supportsOpenStatementsAcrossCommit()

(1)　機能

コミット間で文ハンドルがオープンされたままの状態がサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsOpenStatementsAcrossCommit() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.147 supportsOpenStatementsAcrossRollback()

(1) 機能

ロールバック間で文ハンドルがオープンされたままの状態がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsOpenStatementsAcrossRollback() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.148 supportsOrderByUnrelated()

(1) 機能

ORDER BY 句でSELECT 文中にない列の使用がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsOrderByUnrelated() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.149 supportsOuterJoins()

(1) 機能

外結合が何らかの形式でサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsOuterJoins() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.150 supportsPositionedDelete()

(1) 機能

位置指定されたDELETE 文がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsPositionedDelete() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.151 supportsPositionedUpdate()

(1) 機能

位置指定されたUPDATE 文がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsPositionedUpdate() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.152 supportsRefCursors()

(1) 機能

データベースがREF CURSOR をサポートしているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsRefCursors() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.153 supportsResultSetConcurrency(int type, int concurrency)

(1) 機能

指定した結果セットタイプと並行処理タイプの組み合わせをサポートしているかどうかを返します。

(2) 形式

```
public synchronized boolean supportsResultSetConcurrency(int type, int concurrency) throws S
QLException
```

(3) 引数

int type：

結果セットタイプを指定します。

int concurrency：

並行処理タイプを指定します。

(4) 戻り値

boolean 型：

次に示すどちらかの値が返却されます。

true：サポートしています。

false：サポートしていません。

typeがResultSet.TYPE_FORWARD_ONLYまたはResultSet.TYPE_SCROLL_INSENSITIVE，かつconcurrencyがResultSet.CONCUR_READ_ONLYの場合，trueが返却されます。それ以外の場合はfalseが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connectionオブジェクトがクローズされている場合，SQLExceptionが投入されます。

8.7.154 supportsResultSetHoldability(int holdability)

(1) 機能

指定されたResultSetオブジェクトの保持機能がサポートされているかどうかを返します。

(2) 形式

```
public synchronized boolean supportsResultSetHoldability(int holdability) throws SQLExceptio
n
```

(3) 引数

int holdability：

次に示すどちらかの値を指定します。

- ResultSet.HOLD_CURSORS_OVER_COMMIT

  Connection.commitメソッドが呼び出されたときにResultSetオブジェクトがクローズされません。

- ResultSet.CLOSE_CURSORS_AT_COMMIT

  Connection.commitメソッドが呼び出されたときにResultSetオブジェクトがクローズされます。

(4) 戻り値

boolean型：

true：サポートしています。

false：サポートしていません。

holdabilityにResultSet.HOLD_CURSORS_OVER_COMMITが指定された場合は，trueが返却されます。それ以外の場合は，falseが返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connectionオブジェクトがクローズされている場合，SQLExceptionが投入されます。

8.7.155　supportsResultSetType(int type)

(1)　機能

指定した結果セットタイプがサポートされているかどうかを返します。

(2)　形式

```
public synchronized boolean supportsResultSetType(int type) throws SQLException
```

(3)　引数

int type：

結果セットタイプを指定します。

(4)　戻り値

boolean 型：

次に示すどちらかの値が返却されます。

true：サポートしています。

false：サポートしていません。

type にResultSet.TYPE_FORWARD_ONLY またはResultSet.TYPE_SCROLL_INSENSITIVE を指定した場合は，true が返却されます。それ以外の場合は，false が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.156　supportsSavepoints()

(1)　機能

セーブポイントがサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsSavepoints() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.157 supportsSchemasInDataManipulation()

(1) 機能

データ操作文でスキーマ名を使用できるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsSchemasInDataManipulation() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.158 supportsSchemasInIndexDefinitions()

(1) 機能

インデクス定義文でスキーマ名を使用できるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsSchemasInIndexDefinitions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.159 supportsSchemasInPrivilegeDefinitions()

(1) 機能

権限付与の定義文（GRANT 文），または権限取り消しの定義文（REVOKE 文）で，スキーマ名を使用できるかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsSchemasInPrivilegeDefinitions() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.160　supportsSchemasInProcedureCalls()

(1)　機能

プロシジャ呼び出し文でスキーマ名を使用できるかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsSchemasInProcedureCalls() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.161　supportsSchemasInTableDefinitions()

(1)　機能

表定義文でスキーマ名を使用できるかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsSchemasInTableDefinitions() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.162 supportsSelectForUpdate()

(1) 機能

SELECT FOR UPDATE 文がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsSelectForUpdate() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.163 supportsStatementPooling()

(1) 機能

文ハンドルのプールがサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsStatementPooling() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.164 supportsStoredFunctionsUsingCallSyntax()

(1) 機能

ストアドプロシジャエスケープ構文を使用したユーザ定義関数，またはベンダ関数の呼び出しをサポートするかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsStoredFunctionsUsingCallSyntax() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.165 supportsStoredProcedures()

(1) 機能

ストアドプロシジャエスケープ構文を使用するストアドプロシジャ呼び出しがサポートされているかどうかを判定します。

(2) 形式

```
public synchronized boolean supportsStoredProcedures() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.166 supportsSubqueriesInComparisons()

(1) 機能

比較述語で副問合せがサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsSubqueriesInComparisons() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.167　supportsSubqueriesInExists()

(1)　機能

EXISTS 述語で副問合せがサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsSubqueriesInExists() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.168　supportsSubqueriesInIns()

(1)　機能

IN 述語で副問合せがサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsSubqueriesInIns() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.169 supportsSubqueriesInQuantifieds()

(1) 機能

限定述語で副問合せがサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsSubqueriesInQuantifieds() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.170 supportsTableCorrelationNames()

(1) 機能

表の相関名がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsTableCorrelationNames() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.171 supportsTransactionIsolationLevel(int level)

(1) 機能

指定したトランザクション隔離性水準がサポートされているかどうかを返します。

(2) 形式

```
public synchronized boolean supportsTransactionIsolationLevel(int level) throws SQLException
```

(3) 引数

int level：

トランザクション隔離性水準を指定します。

(4) 戻り値

boolean 型：

true：サポートしています。

false：サポートしていません。

level にConnection.TRANSACTION_READ_COMMITTED またはConnection.TRANSACTION_REPEATABLE_READ を指定した場合は，true が返却されます。それ以外の場合は，false が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.172　supportsTransactions()

(1)　機能

トランザクションがサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsTransactions() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.173　supportsUnion()

(1)　機能

SQL UNION がサポートされているかどうかを取得します。

(2)　形式

```
public synchronized boolean supportsUnion() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.174 supportsUnionAll()

(1) 機能

SQL UNION ALL がサポートされているかどうかを取得します。

(2) 形式

```
public synchronized boolean supportsUnionAll() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にtrue が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.175 updatesAreDetected(int type)

(1) 機能

ResultSet クラスのrowUpdated メソッドを呼び出すことによって可視の行が更新されたことを検出できるかどうかを取得します。

(2) 形式

```
public synchronized boolean updatesAreDetected(int type) throws SQLException
```

(3) 引数

int type：

結果セットタイプを指定します。

- ResultSet.TYPE_FORWARD_ONLY
- ResultSet.TYPE_SCROLL_INSENSITIVE
- ResultSet.TYPE_SCROLL_SENSITIVE

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.176 usesLocalFilePerTable()

(1) 機能

各表にファイルを使用するかどうかを取得します。

(2) 形式

```
public synchronized boolean usesLocalFilePerTable() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.7.177　usesLocalFiles()

(1)　機能

ローカルファイルに表を格納するかどうかを取得します。

(2)　形式

```
public synchronized boolean usesLocalFiles() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

このメソッドを実行する前に，Connection オブジェクトがクローズされている場合，SQLException が投入されます。

8.8 ResultSetMetaData インタフェース

ここでは，ResultSetMetaData インタフェースで提供されているメソッドについて説明します。

8.8.1 ResultSetMetaData インタフェースのメソッド一覧

(1) ResultSetMetaData インタフェースの主な機能

ResultSetMetaData インタフェースでは，主に次の機能が提供されています。

- ResultSet（結果セット）の各列に対する，データ型およびデータ長などのメタ情報の返却

(2) HADB でサポートしている ResultSetMetaData インタフェースのメソッド

HADB でサポートしているResultSetMetaData インタフェースのメソッドの一覧を次の表に示します。

表 8-76 ResultSetMetaData インタフェースのメソッドの一覧

項番	ResultSetMetaData インタフェースのメソッド	機能
1	getCatalogName(int column)	指定した列の表のカタログ名を取得します。
2	getColumnClassName(int column)	列のデータ型に対する Java クラスの完全指定された名前を取得します。
3	getColumnCount()	ResultSet オブジェクトの列数を取得します。
4	getColumnDisplaySize(int column)	指定した列の通常の最大幅を文字数で返します。
5	getColumnLabel(int column)	印刷や表示に使用する列の推奨タイトルを取得します。
6	getColumnName(int column)	指定した列の名称を取得します。
7	getColumnType(int column)	指定した列の SQL データ型を取得します。
8	getColumnTypeName(int column)	指定した列のデータ型を取得します。
9	getPrecision(int column)	指定した列の桁数を取得します。
10	getScale(int column)	指定した列の小数点以下の桁数を取得します。
11	getSchemaName(int column)	指定した列のスキーマ名を取得します。
12	getTableName(int column)	指定した列の表名を取得します。
13	isAutoIncrement(int column)	指定した列が自動的に番号付けされて読み取り専用として扱われるかどうかを返します。
14	isCaseSensitive(int column)	指定した列が大文字と小文字を区別するかどうかを返します。
15	isCurrency(int column)	指定した列が通貨の値かどうかを返します。

項番	ResultSetMetaData インタフェースのメソッド	機能
16	isDefinitelyWritable(int column)	指定した列への書き込みが必ず成功するかどうかを返します。
17	isNullable(int column)	指定した列にナル値をセットできるかどうかを返します。
18	isReadOnly(int column)	指定した列が読み取り専用でないかどうかを返します。
19	isSearchable(int column)	指定した列がwhere 句に指定できるかどうかを返します。
20	isSigned(int column)	指定した列の値が，符号付き数値かどうかを返します。
21	isWritable(int column)	指定した列への書き込みを成功させることができるかどうかを返します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されることがあります。

(3) 必要なパッケージ名称とクラス名称

ResultSetMetaData インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbResultSetMetaData

8.8.2 getCatalogName(int column)

(1) 機能

指定した列の表のカタログ名を取得します。

(2) 形式

```
public synchronized String getCatalogName(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

String オブジェクトが返却されます。

常に空の文字列が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.3 getColumnClassName(int column)

(1) 機能

列のデータ型に対する Java クラスの完全指定された名前を取得します。

(2) 形式

```
public synchronized String getColumnClassName(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

String オブジェクトが返却されます。

列に対してResultSet オブジェクトのgetObject メソッドを実行した結果，返却する Java クラスの型を String 型で返します。列のデータ型と返却値を次の表に示します。

表 8-77 getColumnClassName メソッドを実行して返却される文字列

列のデータ型（HADB のデータ型）	返却される文字列
INTEGER	”java.lang.Long”
SMALLINT	”java.lang.Integer”
	”java.lang.Short”※
DOUBLE PRECISION，FLOAT	”java.lang.Double”
DECIMAL，NUMERIC	”java.math.BigDecimal”
CHAR	”java.lang.String”
VARCHAR	”java.lang.String”
DATE	”java.sql.Date”

列のデータ型（HADB のデータ型）	返却される文字列
TIME	"java.sql.Time"
TIMESTAMP	"java.sql.Timestamp"
BINARY	"java.lang.Object"
VARBINARY	"java.lang.Object"
ROW	"java.lang.Object"
BOOLEAN（DatabaseMetaData から生成したResultSet にだけ存在する列）	"java.lang.Boolean"

注※

DatabaseMetaData から生成したResultSet にだけ存在する列で，データ型がshort で規定されている列の場合，この値が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.4 getColumnCount()

(1) 機能

ResultSet オブジェクトの列数を取得します。

(2) 形式

```
public synchronized int getColumnCount() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

ResultSet オブジェクトの列数が返却されます。

(5) 発生する例外

なし。

8.8.5 getColumnDisplaySize(int column)

(1) 機能

指定した列の通常の最大幅を文字数で返します。

(2) 形式

```
public synchronized int getColumnDisplaySize(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

最大文字数が返却されます。getColumnDisplaySize メソッドの戻り値を次の表に示します。

表 8-78 getColumnDisplaySize メソッドの戻り値

列のデータ型（HADB のデータ型）	戻り値（最大文字数）
INTEGER	20
SMALLINT	11
	6※1
DOUBLE PRECISION FLOAT	23
DECIMAL(*m*,*n*) NUMERIC(*m*,*n*)	*m* ＋ 2
CHAR(*n*) VARCHAR(*n*)	*n*
DATE	10
TIME(*p*)	*p* ＝ 0 の場合：8 *p* ＞ 0 の場合：8 ＋（*n* ＋ 1）
TIMESTAMP(*p*)	*p* ＝ 0 の場合：19 *p* ＞ 0 の場合：19 ＋（*n* ＋ 1）
BINARY(*n*) VARBINARY(*n*)	*n*×2
ROW	*行長*※2

列のデータ型（HADB のデータ型）	戻り値（最大文字数）
BOOLEAN（DatabaseMetaData から生成したResultSet にだけ存在する列）	5

注※1

DatabaseMetaData から生成したResultSet にだけ存在する列で，データ型がshort で規定されている列の場合，この値が返却されます。

注※2

各列のデータ長の総和になります。各列のデータ長の求め方については，マニュアル『HADB　SQL リファレンス』の『データ型の種類』を参照してください。

(5)　発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.6　getColumnLabel(int column)

(1)　機能

印刷や表示に使用する列の推奨タイトルを取得します。

(2)　形式

```
public synchronized String getColumnLabel(int column) throws SQLException
```

(3)　引数

int column：

1 から始まる列番号を指定します。

(4)　戻り値

String オブジェクトが返却されます。

ResultSetMetaData オブジェクトのgetColumnName での返却値と同じ値（列名）が返却されます。

(5)　発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.7　getColumnName(int column)

(1)　機能

指定した列の名称を取得します。

(2)　形式

```
public synchronized String getColumnName(int column) throws SQLException
```

(3)　引数

int column：

1 から始まる列番号を指定します。

(4)　戻り値

String オブジェクトが返却されます。

HADB サーバから送られる列名情報から，検索結果列の名称を取得して返します。検索結果列の名称については，マニュアル『HADB　SQL リファレンス』の『SELECT 文の指定形式および規則』の『規則』を参照してください。

Array オブジェクトから取得したResultSet クラスの 1 列目は，JDBC ドライバが生成する列であり，列名として"JDBC_Array_Index"を返します。

(5)　発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.8　getColumnType(int column)

(1)　機能

指定した列の SQL データ型を取得します。

(2)　形式

```
public synchronized int getColumnType(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

java.sql.Types からの SQL 型が返却されます。

列のデータ型と返却値の対応については，「7.6.1 データ型のマッピング」の「(1) HADB のデータ型と JDBC の SQL データ型の対応」を参照してください。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.9 getColumnTypeName(int column)

(1) 機能

指定した列のデータ型を取得します。

(2) 形式

```
public synchronized String getColumnTypeName(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

String オブジェクトが返却されます。getColumnTypeName メソッドの戻り値を次の表に示します。

表 8-79 getColumnTypeName メソッドの戻り値

列のデータ型（HADB のデータ型）	戻り値（返却される文字列）
INTEGER	”INTEGER”
SMALLINT	”SMALLINT”※
DECIMAL, NUMERIC	”DECIMAL”

列のデータ型（HADB のデータ型）	戻り値（返却される文字列）
CHAR	”CHAR”
DOUBLE PRECISION，FLOAT	”DOUBLE PRECISION”
VARCHAR	”VARCHAR”
DATE	”DATE”
TIME	”TIME”
TIMESTAMP	”TIMESTAMP”
BINARY	”BINARY”
VARBINARY	”VARBINARY”
ROW	”ROW”
BOOLEAN（DatabaseMetaData から生成したResultSet にだけ存在する列）	”BOOLEAN”

注※

DatabaseMetaData から生成したResultSet にだけ存在する列で，データ型がshort で規定されている列の場合，この値が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.10 getPrecision(int column)

(1) 機能

指定した列の桁数を取得します。

(2) 形式

```
public synchronized int getPrecision(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

列の桁数が 10 進数で返却されます。指定した列が数データ型の場合は，桁数が返却されます。指定した列が数データ型でない場合は，列長がバイト単位で返却されます。getPrecision メソッドの戻り値を次の表に示します。

表 8-80　getPrecision メソッドの戻り値

列のデータ型（HADB のデータ型）	戻り値（列の桁数）
INTEGER	19
SMALLINT	10
	5※1
DOUBLE PRECISION FLOAT	17
DECIMAL(*m*,*n*) NUMERIC(*m*,*n*)	*m*
CHAR(*n*) VARCHAR(*n*)	*n*
DATE	10
TIME(*p*)	*p* ＝ 0 の場合：8 *p* ＞ 0 の場合：8 ＋（*n* ＋ 1）
TIMESTAMP(*p*)	*p* ＝ 0 の場合：19 *p* ＞ 0 の場合：19 ＋（*n* ＋ 1）
BINARY(*n*) VARBINARY(*n*)	*n*
ROW	*行長*※2
BOOLEAN（DatabaseMetaData から生成したResultSet にだけ存在する列）	1

注※1

DatabaseMetaData から生成したResultSet にだけ存在する列で，データ型がshort で規定されている列の場合，この値が返却されます。

注※2

各列のデータ長の総和になります。各列のデータ長の求め方については，マニュアル『HADB　SQL リファレンス』の『データ型の種類』を参照してください。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.11　getScale(int column)

(1)　機能

指定した列の小数点以下の桁数を取得します。

(2)　形式

```
public synchronized int getScale(int column) throws SQLException
```

(3)　引数

int column：

1 から始まる列番号を指定します。

(4)　戻り値

小数点以下の桁数が 10 進数で返却されます。getScale メソッドの戻り値を次の表に示します。

表 8-81　getScale メソッドの戻り値

列のデータ型（HADB のデータ型）	戻り値（小数点以下の桁数）
DECIMAL(*m*,*n*) NUMERIC(*m*,*n*)	*n*
TIME(*p*) TIMESTAMP(*p*)	*p*
上記以外	0

(5)　発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.12　getSchemaName(int column)

(1)　機能

指定した列のスキーマ名を取得します。

(2)　形式

```
public synchronized String getSchemaName(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

String オブジェクトが返却されます。

常に空の文字列が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.13 getTableName(int column)

(1) 機能

指定した列の表名を取得します。

(2) 形式

```
public synchronized String getTableName(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

String オブジェクトが返却されます。

常に空の文字列が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.14　isAutoIncrement(int column)

(1)　機能

指定した列が自動的に番号付けされて読み取り専用として扱われるかどうかを返します。

(2)　形式

```
public synchronized boolean isAutoIncrement(int column) throws SQLException
```

(3)　引数

int column：

1 から始まる列番号を指定します。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.15　isCaseSensitive(int column)

(1)　機能

指定した列が大文字と小文字を区別するかどうかを返します。

(2)　形式

```
public synchronized boolean isCaseSensitive(int column) throws SQLException
```

(3)　引数

int column：

1 から始まる列番号を指定します。

(4)　戻り値

常にfalse が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.16 isCurrency(int column)

(1) 機能

指定した列が通貨の値かどうかを返します。

(2) 形式

```
public synchronized boolean isCurrency(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.17 isDefinitelyWritable(int column)

(1) 機能

指定した列への書き込みが必ず成功するかどうかを返します。

(2) 形式

```
public synchronized boolean isDefinitelyWritable(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

常にfalse が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.18 isNullable(int column)

(1) 機能

指定した列にナル値をセットできるかどうかを返します。

(2) 形式

```
public synchronized int isNullable(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

次に示すどちらかの値が返却されます。

- ResultSetMetaData.columnNoNulls：ナル値をセットできません。
- ResultSetMetaData.columnNullable：ナル値をセットできます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.19　isReadOnly(int column)

(1)　機能

指定した列が読み取り専用でないかどうかを返します。

(2)　形式

```
public synchronized boolean isReadOnly(int column) throws SQLException
```

(3)　引数

int column：

1 から始まる列番号を指定します。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.20　isSearchable(int column)

(1)　機能

指定した列がwhere 句に指定できるかどうかを返します。

(2)　形式

```
public synchronized boolean isSearchable(int column) throws SQLException
```

(3)　引数

int column：

1 から始まる列番号を指定します。

(4)　戻り値

次に示すどちらかの値が返却されます。

true：

where 句に指定できます。

false：

where 句に指定できません。

DatabaseMetaData インタフェースで生成したResultSet オブジェクトの場合は，false が返却されます。そうでない場合は，true が返却されます。

(5) 発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.21 isSigned(int column)

(1) 機能

指定した列の値が，符号付き数値かどうかを返します。

(2) 形式

```
public synchronized boolean isSigned(int column) throws SQLException
```

(3) 引数

int column：

1 から始まる列番号を指定します。

(4) 戻り値

次に示すどちらかの値が返却されます。

true：

符号付き数値です。

false：

符号付き数値ではありません。

パラメタのデータ型と戻り値の関係を次の表に示します。

表 8-82　パラメタのデータ型と戻り値の関係

パラメタのデータ型	戻り値
INTEGER，SMALLINT，DOUBLE PRECISION，FLOAT，DECIMAL，NUMERIC	true
上記以外	false

(5)　発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.8.22　isWritable(int column)

(1)　機能

指定した列への書き込みを成功させることができるかどうかを返します。

(2)　形式

```
public synchronized boolean isWritable(int column) throws SQLException
```

(3)　引数

int column：

1 から始まる列番号を指定します。

(4)　戻り値

常にfalse が返却されます。

(5)　発生する例外

column に指定した値が0 以下，または列数より大きい場合，SQLException が投入されます。

8.9 SQLException インタフェース

SQLException は，java.sql パッケージのSQLException クラスを直接利用します。SQLException インタフェースが提供する各メソッドの詳細，使用方法については，JavaSoft が提供する『JDBC 関連ドキュメント』を参照してください。

8.10　SQLWarning インタフェース

SQLWarning インタフェースでは，データベースアクセスの警告に関する情報を提供します。

SQLWarning オブジェクトは，警告が報告される原因となったメソッドのオブジェクトに，例外での通知なしで蓄積されます。

8.10.1　SQLWarning オブジェクトの生成

SQL の実行で発生した警告を，JDBC ドライバ内で保持するよう指定している場合，SQLWarning オブジェクトを生成し，警告情報を保持します。

警告保持指定は，URL またはユーザプロパティのsqlwarningkeep，setSQLWarningKeep メソッドで指定できます。

8.10.2　SQLWarning オブジェクトの解放

SQLWarning オブジェクトは，警告が報告される原因となったメソッドのオブジェクト（Connection，Statement，PreparedStatement，およびResultSet）から，チェーンによって蓄積されます。

蓄積されたSQLWarning オブジェクトを明示的に解放するには，警告が報告される原因となったメソッドのオブジェクトに対してclearWarnings メソッドを実行してください。

8.11　サポートしていないインタフェース

HADB では，次に示すインタフェースはサポートしていません。

- Array
- Blob
- CallableStatement
- Clob
- Savepoint
- SQLData
- SQLInput
- SQLOutput

9 JDBC 2.1 コア API

この章では，JDBC 2.1 コア API で追加された機能の HADB でのサポート範囲について説明します。

9.1　結果セットの拡張機能のサポート範囲

JDBC 2.1 コア API の結果セット（ResultSet クラス）の拡張機能に対する HADB でのサポート範囲を次の表に示します。

表 9-1　JDBC 2.1 コア API の結果セット（ResultSet クラス）の拡張機能に対する HADB でのサポート範囲

結果セット（ResultSet クラス）の拡張機能名		HADB でのサポート範囲
スクロールタイプ	順方向専用型	○
	スクロール非反映型	○
	スクロール反映型	×
並行処理タイプ	読み取り専用型	○
	更新可能型	×

（凡例）

○：HADB でサポートしています。

×：HADB ではサポートしていません。

重要

- サポート対象外のスクロールタイプや並行処理タイプが指定されても，エラーにはなりません。指定されたスクロールタイプや並行処理タイプに最も近い結果セットを仮定して，Statement クラスまたはそのサブクラスのインスタンスを生成します。この場合，警告（SQLWarning オブジェクト）を生成して，Connection クラスのインスタンスに関連づけます。
- スクロール型結果セットでは，すべての検索データを JDBC ドライバ内でキャッシングするため，データ量が多いとメモリ不足や性能低下が起こるおそれがあります。したがって，スクロール型結果セットを使用する場合は，SQL 文に条件を付加するなどして，検索データ量を抑えるようにしてください。

9.2　バッチ更新機能のサポート範囲

Statement クラスおよびPreparedStatement クラスのバッチ更新機能の HADB でのサポート範囲について説明します。

9.2.1　バッチ更新機能を使用できる SQL 文

バッチ更新機能を使用できる SQL 文を次に示します。

- 定義系 SQL
- DELETE 文
- INSERT 文
- PURGE CHUNK 文
- TRUNCATE TABLE 文
- UPDATE 文

なお，次の SQL 文を指定した場合，executeBatch メソッドまたはexecuteLargeBatch メソッドの実行時に BatchUpdateException が投入されます。

- SELECT 文
- ？パラメタを指定したPURGE CHUNK 文

9.2.2　Statement クラスでのバッチ更新機能

Statement クラスでのバッチ更新の留意点を次に示します。

- 複数の SQL 文を，addBatch メソッドで登録します。
- 登録した SQL 文を，executeBatch メソッドまたはexecuteLargeBatch メソッドで一括して実行します。
- 一括実行の結果として，それぞれの SQL 文で更新された行数の配列を返却します。
- 一括実行の途中でエラーが発生した場合，BatchUpdateException が投入されます。

9.2.3　PreparedStatement クラスでのバッチ更新機能

PreparedStatement クラスでのバッチ更新の留意点を次に示します。

- PreparedStatement インスタンス生成時に指定した SQL 文に対する？パラメタを，通常の手順（set*XXX* メソッド）で設定します。

- addBatch メソッドで？パラメタのセットを登録します。
- 登録した複数セットの？パラメタを，executeBatch メソッドまたはexecuteLargeBatch メソッドで一括して実行します。
- 一括実行の結果として，それぞれの？パラメタのセットで更新した行数の配列が返却されます。
- 一括実行の途中でエラーが発生した場合，BatchUpdateException が投入されます。

9.2.4 注意事項

(1) 暗黙的コミットの実行

SQL 文のバッチ更新機能を使用する際，次に示す SQL 文をaddBatch すると，その SQL 文が実行されたときに HADB サーバが暗黙的にコミットを実行するため，注意が必要です。

- 定義系 SQL
- PURGE CHUNK 文
- TRUNCATE TABLE 文

(2) パラメタと SQL 文の addBatch の混在時でのバッチ更新機能

パラメタと SQL 文のaddBatch が混在している場合，一括更新をしないで逐次実行します。例を次に示します。

```
PreparedStatement pstmt = con.prepareStatement("UPDATE T1 SET C1=? WHERE C2=?");
pstmt.setInt(1, 1);
pstmt.setInt(2, 1);
pstmt.addBatch();
pstmt.setInt(1, 2);
pstmt.setInt(2, 2);
pstmt.addBatch();
pstmt.addBatch("INSERT INTO T2 VALUES(1,2,3)");
pstmt.setInt(1, 3);
pstmt.setInt(2, 4);
pstmt.addBatch();
pstmt.setInt(1, 4);
pstmt.setInt(2, 4);
pstmt.addBatch();
pstmt.executeBatch();
```

この AP を実行すると，パラメタと SQL 文のaddBatch が混在しているため，各addBatch 単位での SQL 実行となります。そのため，次の AP を実行した場合と同じ結果となります。

```
PreparedStatement pstmt = con.prepareStatement("UPDATE T1 SET C1=? WHERE C2=?");
pstmt.setInt(1, 1);
pstmt.setInt(2, 1);
pstmt.executeUpdate();
```

```
pstmt.setInt(1, 2);
pstmt.setInt(2, 2);
pstmt.executeUpdate();
pstmt.executeUpdate("INSERT INTO T2 VALUES(1,2,3)");
pstmt.setInt(1, 3);
pstmt.setInt(2, 4);
pstmt.executeUpdate();
pstmt.setInt(1, 4);
pstmt.setInt(2, 4);
pstmt.executeUpdate();
```

なお，パラメタと SQL 文のaddBatch とが混在するバッチ更新機能を使用する場合，Connection クラスの自動コミットモードを無効にすることを推奨します。

メモ

パラメタと SQL 文の混在時は逐次実行になりますが，自動コミットモードが有効の場合は，実行単位ごとにコミットが暗黙的に実行されます。そのため，バッチ更新の途中でエラーが発生すると，エラーが発生した直前までがコミットされた状態になり，どの時点までコミットされたかを認識できないので，自動コミットモードを無効にすることを推奨しています。

(3) addBatch メソッドを使用して多数のパラメタを登録する場合

addBatch メソッドを使用して登録したすべてのパラメタは，executeBatch メソッドまたは executeLargeBatch メソッドが実行されるまで JDBC ドライバ内に保存されます。そのため，多数のパラメタを登録する際は，メモリ使用量に注意してください。

(4) 例外 BatchUpdateException で通知する更新カウント

バッチ更新実行時に発生する例外BatchUpdateException の，getUpdateCounts メソッドの戻り値で通知する更新カウントの内容を次に示します。

- 実行した SQL 数と要素数が等しい配列
- 各配列要素には更新行数を設定

ただし，例外発生時に内部的にロールバックが行われた場合，要素数 0 の配列を返却します。

更新カウントの例を次に示します。

■JDBC ドライバによる逐次実行のプログラム例

```
Statement stmt = con.createStatement();
stmt.addBatch("INSERT INTO T1 VALUES(1,'aaaa')");
stmt.addBatch("INSERT INTO T1 VALUES(2,'bbbbbbbb')");...[A]
stmt.addBatch("INSERT INTO T1 VALUES(3,'cccc')");
stmt.executeBatch();
```

プログラム例を実行し，[A]で登録したパラメタ，またはSQLの処理でエラーになった場合，getUpdateCountsメソッドで返却する更新カウントの内容を次に示します。

- 要素数1の配列
- 要素0の値：更新行数

更新行数がInteger.MAX_VALUEを超える可能性がある場合は，executeBatchメソッドではなく，executeLargeBatchメソッドを使用してください。また，getUpdateCountsメソッドではなく，getLargeUpdateCountsメソッドを使用してください。

(5) バッチ更新時のキャンセル

一括実行中にキャンセルを受け付けると，そのときまでに実行されたSQL文の結果は基本的にロールバックされます。ただし，次に示す範囲のSQL文はコミットされた状態になります。

- 自動コミットモードを有効にしている場合：最後に正常終了したSQL文まで
- 自動コミットモードを無効にしている場合：最後に正常終了した，暗黙的コミットを行うSQL文まで

9.3 追加されたデータ型

JDBC 2.1 コア API では，幾つかの新たな JDBC SQL タイプが追加されました。次の JDBC SQL タイプが追加されましたが，JDBC ドライバでは使用できません。

- BLOB
- CLOB
- ARRAY
- REF
- DISTINCT
- STRUCT
- JAVA OBJECT

9.4 サポートしていないインタフェース

HADB では，次に示すインタフェースはサポートしていません。

- Array
- Blob
- Clob
- Ref
- SQLData
- SQLInput
- SQLOutput
- Struct

10 JDBC 2.0 Optional Package

この章では，JDBC 2.0 Optional Package の各インタフェースとメソッドについて説明します。

10.1　JDBC 2.0 Optional Package の追加機能に対する HADB でのサポート範囲

JDBC 2.0 Optional Package の追加機能に対する HADB でのサポート範囲を次の表に示します。

表 10-1　JDBC 2.0 Optional Package の追加機能に対する HADB でのサポート範囲

JDBC 2.0 Optional Package の追加機能	対応するインタフェース	HADB でのサポート範囲
JNDI 対応	DataSource	○
接続プール	ConnectionPoolDataSource	○
	PooledConnection	○
RowSets	RowSet	×
	RowSetInternal	×
	RowSetListner	×
	RowSetMetaData	×
	RowSetReader	×

（凡例）

○：HADB でサポートしています。

×：HADB ではサポートしていません。

なお，上記の表に記載しているインタフェースのほかに，HADB 独自の接続情報設定および取得に関するメソッドがあります。詳細については，「10.5　接続情報設定および取得インタフェース」を参照してください。

10.2　DataSource インタフェース

ここでは，DataSource インタフェースで提供されているメソッドについて説明します。

10.2.1　DataSource インタフェースのメソッド一覧

HADB でサポートしているDataSource インタフェースのメソッドの一覧を次の表に示します。

表 10-2　DataSource インタフェースのメソッドの一覧

項番	DataSource インタフェースのメソッド	機能
1	getConnection()	HADB サーバへの接続を行います。
2	getConnection(String username, String password)	
3	getLoginTimeout()	setLoginTimeout メソッドで設定した値（HADB サーバへの接続処理のタイムアウト時間）を取得します。
4	getLogWriter()	DataSource オブジェクトのログライターを取得します。
5	setLoginTimeout(int seconds)	HADB サーバへの接続処理のタイムアウト時間（単位：秒）を設定します。
6	setLogWriter(PrintWriter out)	DataSource オブジェクトのログライターを設定します。

DataSource インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbDataSource

10.2.2　getConnection()

(1)　機能

HADB サーバへの接続を行います。

DataSource オブジェクトに設定した HADB サーバへの接続情報に従って，HADB サーバへの接続を行い，Connection オブジェクトを返却します。

HADB サーバへの接続情報の指定には優先順位があります。優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

なお，getConnection メソッドを実行するにはCONNECT 権限が必要です。

(2) 形式

```
public synchronized Connection getConnection() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Connection オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- データベースのアクセスエラーが発生した場合
- HADB サーバへの接続情報の指定内容が不正な場合

10.2.3 getConnection(String username, String password)

(1) 機能

HADB サーバへの接続を行います。

DataSource オブジェクトに設定した HADB サーバへの接続情報と，引数に指定した接続情報に従って，HADB サーバへの接続を行い，Connection オブジェクトを返却します。

HADB サーバへの接続情報の指定には優先順位があります。優先順位については，「7.3.3 接続情報の優先順位」を参照してください。

なお，getConnection メソッドを実行するにはCONNECT 権限が必要です。

(2) 形式

```
public synchronized Connection getConnection(String username, String password) throws SQLExc
eption
```

(3) 引数

String username：

HADB サーバに接続する認可識別子を指定します。

String password：

HADB サーバに接続する認可識別子のパスワードを指定します。

username またはpassword にnull を指定した場合，認可識別子またはパスワードが指定されていないと見なされます。また，password に長さ 0 の文字列を指定した場合も，パスワードが指定されていないと見なされます。

(4) 戻り値

Connection オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- データベースのアクセスエラーが発生した場合
- HADB サーバへの接続情報の指定内容が不正な場合
- username に指定した認可識別子が長さ 0 の文字列の場合

10.2.4 getLoginTimeout()

(1) 機能

setLoginTimeout メソッドで設定した値（HADB サーバへの接続処理のタイムアウト時間）を取得します。

(2) 形式

```
public synchronized int getLoginTimeout()
```

(3) 引数

なし。

(4) 戻り値

setLoginTimeout メソッドで設定した値（HADB サーバへの接続処理のタイムアウト時間）が返却されます。setLoginTimeout メソッドでタイムアウト時間を設定していない場合は，0 が返却されます。

(5) 発生する例外

なし。

10.2.5 getLogWriter()

(1) 機能

DataSource オブジェクトのログライターを取得します。

(2) 形式

```
public synchronized PrintWriter getLogWriter() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

DataSource オブジェクトのログライターが返却されます。ログライターが設定されていない場合は，null が返却されます。

(5) 発生する例外

なし。

10.2.6 setLoginTimeout(int seconds)

(1) 機能

HADB サーバへの接続処理のタイムアウト時間（単位：秒）を設定します。

getConnection メソッドを実行して HADB サーバに接続する際に，ここで設定したタイムアウト時間が適用されます。

(2) 形式

```
public synchronized void setLoginTimeout(int seconds) throws SQLException
```

(3) 引数

int seconds：

HADB サーバへの接続処理のタイムアウト時間（単位：秒）を 0～300 の範囲で指定します。

0 を指定した場合，システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティの adb_clt_rpc_con_wait_time の値が仮定されます。adb_clt_rpc_con_wait_time を指定していない場合は，adb_clt_rpc_con_wait_time のデフォルト値が仮定されます。

(4) 戻り値

なし。

(5) 発生する例外

seconds に不正な値（0 未満または 301 以上の値）を指定した場合，SQLException が投入されます。

(6) 留意事項

HADB サーバへの接続処理のタイムアウト時間は，各プロパティやメソッドなど複数の個所で設定できます。実際に適用される HADB サーバへの接続処理のタイムアウト時間の優先順位については，「7.3.3 接続情報の優先順位」の「(1) HADB サーバへの接続時に必要となる接続情報」を参照してください。

10.2.7 setLogWriter(PrintWriter out)

(1) 機能

DataSource オブジェクトのログライターを設定します。

(2) 形式

```
public synchronized void setLogWriter(PrintWriter out) throws SQLException
```

(3) 引数

PrintWriter out：

ログライターを指定します。

(4) 戻り値

なし。

(5) 発生する例外

なし。

10.3　ConnectionPoolDataSource インタフェース

ここでは，ConnectionPoolDataSource インタフェースで提供されているメソッドについて説明します。

10.3.1　ConnectionPoolDataSource インタフェースのメソッド一覧

HADB でサポートしているConnectionPoolDataSource インタフェースのメソッドの一覧を次の表に示します。

表 10-3　ConnectionPoolDataSource インタフェースのメソッドの一覧

項番	ConnectionPoolDataSource インタフェースのメソッド	機能
1	getLoginTimeout()	setLoginTimeout メソッドで設定した値（HADB サーバへの接続処理のタイムアウト時間）を取得します。
2	getLogWriter()	ConnectionPoolDataSource オブジェクトのログライターを取得します。
3	getPooledConnection()	DataSource オブジェクトに設定されている接続情報から，PooledConnection オブジェクトを生成します。
4	getPooledConnection(String user, String password)	引数で指定した接続情報と，DataSource オブジェクトに設定されている接続情報から，PooledConnection オブジェクトを生成します。
5	setLoginTimeout(int seconds)	HADB サーバへの接続処理のタイムアウト時間（単位：秒）を設定します。
6	setLogWriter(PrintWriter out)	ConnectionPoolDataSource オブジェクトのログライターを設定します。

ConnectionPoolDataSource インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbConnectionPoolDataSource

10.3.2　getLoginTimeout()

(1)　機能

setLoginTimeout メソッドで設定した値（HADB サーバへの接続処理のタイムアウト時間）を取得します。

(2) 形式

```
public synchronized int getLoginTimeout() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

setLoginTimeout メソッドで設定した値（HADB サーバへの接続処理のタイムアウト時間）が返却されます。setLoginTimeout メソッドでタイムアウト時間を設定していない場合は，0 が返却されます。

(5) 発生する例外

なし。

10.3.3 getLogWriter()

(1) 機能

ConnectionPoolDataSource オブジェクトのログライターを取得します。

(2) 形式

```
public synchronized PrintWriter getLogWriter() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

ConnectionPoolDataSource オブジェクトのログライターが返却されます。ログライターが設定されていない場合は，null が返却されます。

(5) 発生する例外

なし。

10.3.4　getPooledConnection()

(1)　機能

DataSource オブジェクトに設定されている接続情報から，PooledConnection オブジェクトを生成します。

認可識別子とパスワードについては，指定の優先順位があります。優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

なお，getPooledConnection メソッドを実行するにはCONNECT 権限が必要です。

(2)　形式

```
public synchronized PooledConnection getPooledConnection() throws SQLException
```

(3)　引数

なし。

(4)　戻り値

PooledConnection オブジェクトが返却されます。

(5)　発生する例外

なし。

10.3.5　getPooledConnection(String user, String password)

(1)　機能

引数で指定した接続情報と，DataSource オブジェクトに設定されている接続情報から，PooledConnection オブジェクトを生成します。

認可識別子とパスワードについては，指定の優先順位があります。優先順位については，「7.3.3　接続情報の優先順位」を参照してください。

なお，getPooledConnection メソッドを実行するにはCONNECT 権限が必要です。

(2)　形式

```
public synchronized PooledConnection getPooledConnection(String user, String password) throw
s SQLException
```

(3) 引数

String user：

HADB サーバに接続する認可識別子を指定します。

String password：

HADB サーバに接続する認可識別子のパスワードを指定します。

user またはpassword にnull を指定した場合，認可識別子またはパスワードが指定されていないと仮定されます。また，password が長さ 0 の文字列の場合も，パスワードが指定されていないと仮定されます。

user に指定した認可識別子は，setUser メソッドで指定した認可識別子より優先されます。同様に，password に指定したパスワードも，setPassword メソッドで指定したパスワードより優先されます。

(4) 戻り値

PooledConnection オブジェクトが返却されます。

(5) 発生する例外

次に示す場合にSQLException が投入されます。

- user に指定した認可識別子が長さ 0 の文字列の場合

10.3.6 setLoginTimeout(int seconds)

(1) 機能

HADB サーバへの接続処理のタイムアウト時間（単位：秒）を設定します。

getConnection メソッドを実行して HADB サーバに接続する際に，ここで設定したタイムアウト時間が適用されます。

(2) 形式

```
public synchronized void setLoginTimeout(int seconds) throws SQLException
```

(3) 引数

int seconds：

HADB サーバへの接続処理のタイムアウト時間（単位：秒）を 0～300 の範囲で指定します。

0 を指定した場合，システムプロパティ，ユーザプロパティ，または接続用の URL のプロパティの adb_clt_rpc_con_wait_time の値が仮定されます。adb_clt_rpc_con_wait_time を指定していない場合は，adb_clt_rpc_con_wait_time のデフォルト値が仮定されます。

(4) 戻り値

なし。

(5) 発生する例外

seconds に不正な値（0 未満または 301 以上の値）を指定した場合，SQLException が投入されます。

(6) 留意事項

HADB サーバへの接続処理のタイムアウト時間は，各プロパティやメソッドなど複数の個所で設定できます。実際に適用される HADB サーバへの接続処理のタイムアウト時間の優先順位については，「7.3.3 接続情報の優先順位」の「(1) HADB サーバへの接続時に必要となる接続情報」を参照してください。

10.3.7 setLogWriter(PrintWriter out)

(1) 機能

ConnectionPoolDataSource オブジェクトのログライターを設定します。

(2) 形式

```
public synchronized void setLogWriter(PrintWriter out)
```

(3) 引数

PrintWriter out：

ログライターを指定します。

(4) 戻り値

なし。

(5) 発生する例外

なし。

10.4　PooledConnection インタフェース

ここでは，PooledConnection インタフェースで提供されているメソッドについて説明します。

10.4.1　PooledConnection インタフェースのメソッド一覧

HADB でサポートしているPooledConnection インタフェースのメソッドの一覧を次の表に示します。

表 10-4　PooledConnection インタフェースのメソッドの一覧

項番	PooledConnection インタフェースのメソッド	機能
1	addConnectionEventListener(ConnectionEventListener listener)	イベントリスナを登録し，このPooledConnection オブジェクトでイベントが発生したときに通知されるようにします。
2	close()	HADB サーバとの物理的な接続を切断します。接続プールにプールされているすべてのコネクションの物理的な切断を行います。
3	getConnection()	接続プールにプールされているコネクションを使用して HADB サーバに接続します。
4	removeConnectionEventListener(ConnectionEventListener listener)	指定したイベントリスナを，このPooledConnection オブジェクトでイベントが発生したときに通知されるコンポーネントリストから削除します。

PooledConnection インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbPooledConnection

10.4.2　addConnectionEventListener(ConnectionEventListener listener)

(1)　機能

イベントリスナを登録し，このPooledConnection オブジェクトでイベントが発生したときに通知されるようにします。

このメソッドで登録したイベントリスナから，ほかのメソッドを呼び出すことはできません。呼び出した場合，デッドロックなどが発生して応答がなくなるおそれがあります。

(2)　形式

```
public synchronized void addConnectionEventListener(ConnectionEventListener listener)
```

(3) 引数

ConnectionEventListener listener：

ConnectionEventListener インタフェースを実装し，接続が閉じたかエラーが発生したときに通知されるようにするコンポーネントです。通常は接続プール管理プログラムです。

null を指定した場合，何も登録しません。

(4) 戻り値

なし。

(5) 発生する例外

なし。

10.4.3 close()

(1) 機能

HADB サーバとの物理的な接続を切断します。接続プールにプールされているすべてのコネクションの物理的な切断を行います。データベースにアクセス中のコネクションであっても切断します。

(2) 形式

```
public synchronized void close()
```

(3) 引数

なし。

(4) 戻り値

なし。

(5) 発生する例外

なし。

10.4.4 getConnection()

(1) 機能

接続プールにプールされているコネクションを使用して HADB サーバに接続します。接続プールにプールされているコネクションがすべて使用中の場合，HADB サーバとの物理的な接続を新規に確立し，HADB サーバに接続します。

なお，getConnection メソッドを実行するにはCONNECT 権限が必要です。

メモ

- HADB サーバとの物理的な接続は，このクラスオブジェクトがクローズされるまで切断されません。Connection オブジェクトに対してclose メソッドを実行しても，HADB サーバとの物理的な接続は切断されません（このクラスオブジェクトによってコネクションが保持されます)。保持されたコネクションは，次にgetConnection メソッドを実行する際に使用されます。
- 接続プールにプールされているコネクションを使用して HADB サーバに接続する場合，setLoginTimeout メソッドで設定した HADB サーバへの接続処理のタイムアウト時間は適用されません。setLoginTimeout メソッドで設定したタイムアウト時間は，HADB サーバへの物理接続を確立する際の通信処理に掛かる時間を監視しています。接続プールにプールされているコネクションを使用して HADB サーバに接続する場合は，物理接続が発生しないため（通信処理に掛かる時間が発生しないため)，タイムアウト時間が適用されません。

(2) 形式

```
public synchronized Connection getConnection() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

Connection オブジェクトが返却されます。

(5) 発生する例外

データベースのアクセスエラーが発生した場合に，SQLException が投入されます。

10.4.5 removeConnectionEventListener(ConnectionEventListener listener)

(1) 機能

指定したイベントリスナを，このPooledConnection オブジェクトでイベントが発生したときに通知されるコンポーネントリストから削除します。

(2) 形式

```
public synchronized void removeConnectionEventListener(ConnectionEventListener listener)
```

(3) 引数

ConnectionEventListener listener：

ConnectionEventListener インタフェースを実装し，イベントリスナとして登録されたコンポーネントです。通常は接続プール管理プログラムです。

(4) 戻り値

なし。

(5) 発生する例外

なし。

10.5　接続情報設定および取得インタフェース

ここでは，接続情報設定および取得インタフェースで提供されているメソッドについて説明します。

10.5.1　接続情報設定および取得インタフェースのメソッド一覧

DataSource およびConnectionPoolDataSource のクラスでは，JDBC 2.0 Optional Package 規格で定められたメソッドのほかに，次の表に示す HADB 独自のメソッドを提供しています。

表 10-5　接続情報設定および取得インタフェースのメソッドの一覧

項番	メソッド	機能
1	getApName()	setApName メソッドで設定した AP 識別子を取得します。
2	getEncodeLang()	setEncodeLang メソッドで設定した変換文字セット名称を取得します。
3	getInterfaceMethodTrace()	setInterfaceMethodTrace メソッドで設定した JDBC インタフェースメソッドトレースの取得状況を取得します。
4	getNotErrorOccurred()	ConnectionEventListener.connectionErrorOccurred の発生を抑止するかどうかの設定情報を取得します。
5	getPassword()	setPassword メソッドで設定したパスワードを取得します。
6	getSQLWarningKeep()	SQL 実行時に発生した警告情報を保持するかどうかの設定情報を取得します。
7	getTraceNumber()	setTraceNumber メソッドで設定した JDBC インタフェースメソッドトレースのエントリ数を取得します。
8	getUser()	setUser メソッドで設定した認可識別子を取得します。
9	getHostName()	setHostName メソッドで設定した HADB サーバのホスト名を取得します。
10	getPort()	setPort メソッドで設定した HADB サーバのポート番号を取得します。
11	setApName(String name)	HADB サーバに接続する AP の識別子を設定します。
12	setEncodeLang(String lang)	文字コード変換時の変換文字セット名称を設定します。
13	setInterfaceMethodTrace(boolean flag)	JDBC インタフェースメソッドトレースを取得するかどうかを設定します。
14	setNotErrorOccurred(boolean mode)	ConnectionEventListener.connectionErrorOccurred の発生を抑止するかどうかを設定します。
15	setPassword(String password)	HADB サーバに接続する認可識別子のパスワードを設定します。
16	setSQLWarningKeep(boolean mode)	SQL 実行時に発生した警告情報を保持するかどうかを設定します。
17	setTraceNumber(int num)	JDBC インタフェースメソッドトレースのエントリ数を設定します。
18	setUser(String user)	HADB サーバに接続する認可識別子を設定します。

項番	メソッド	機能
19	setHostName(String name)	接続先の HADB サーバのホスト名を設定します。
20	setPort(int port)	接続先の HADB サーバのポート番号を設定します。

10.5.2 getApName()

(1) 機能

setApName メソッドで設定した AP 識別子を取得します。

(2) 形式

```
public synchronized String getApName() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

AP 識別子が返却されます。setApName メソッドを実行していない場合は，”********”が返却されます。

(5) 発生する例外

なし。

10.5.3 getEncodeLang()

(1) 機能

setEncodeLang メソッドで設定した変換文字セット名称を取得します。

(2) 形式

```
public synchronized String getEncodeLang() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

変換文字セット名称が返却されます。setEncodeLang メソッドを実行していない場合は，null が返却されます。

(5) 発生する例外

なし。

10.5.4 getInterfaceMethodTrace()

(1) 機能

setInterfaceMethodTrace メソッドで設定した JDBC インタフェースメソッドトレースの取得状況を取得します。JDBC インタフェースメソッドトレースについては，「7.7.1 JDBC インタフェースメソッドトレース」を参照してください。

(2) 形式

```
public boolean getInterfaceMethodTrace() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

JDBC インタフェースメソッドトレースの取得状況が返却されます。

- true：JDBC インタフェースメソッドトレースを取得しています。
- false：JDBC インタフェースメソッドトレースを取得していません。

(5) 発生する例外

なし。

10.5.5 getNotErrorOccurred()

(1) 機能

ConnectionEventListener.connectionErrorOccurred の発生を抑止するかどうかの設定情報を取得します。setNotErrorOccurred メソッドで設定した情報を取得します。

(2) 形式

```
public boolean getNotErrorOccurred() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

ConnectionEventListener.connectionErrorOccurred を発生させるかどうかの設定情報が返却されます。

- true：connectionErrorOccurred は発生しません。
- false：connectionErrorOccurred が発生します。

setNotErrorOccurred メソッドを実行していない場合は，デフォルト値のfalse が返却されます。

(5) 発生する例外

なし。

10.5.6 getPassword()

(1) 機能

setPassword メソッドで設定したパスワードを取得します。

(2) 形式

```
public synchronized String getPassword() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

setPassword メソッドで設定したパスワードが返却されます。

(5) 発生する例外

なし。

10.5.7 getSQLWarningKeep()

(1) 機能

SQL 実行時に発生した警告情報を保持するかどうかの設定情報を取得します。setSQLWarningKeep メソッドで設定した情報を取得します。

(2) 形式

```
public synchronized boolean getSQLWarningKeep() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

発生した警告情報をConnection クラスで保持するかどうかの設定情報が返却されます。

- true：警告情報を保持します。
- false：警告情報を保持しません。

setSQLWarningKeep メソッドを実行していない場合は，デフォルト値のtrue が返却されます。

(5) 発生する例外

なし。

10.5.8 getTraceNumber()

(1) 機能

setTraceNumber メソッドで設定した JDBC インタフェースメソッドトレースのエントリ数を取得します。

(2) 形式

```
public synchronized int getTraceNumber() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

setTraceNumber メソッドで設定した JDBC インタフェースメソッドトレースのエントリ数が返却されます。setTraceNumber メソッドを実行していない場合は，デフォルト値の500 が返却されます。

(5) 発生する例外

なし。

10.5.9 getUser()

(1) 機能

setUser メソッドで設定した認可識別子を取得します。

(2) 形式

```
public synchronized String getUser() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

setUser メソッドで設定した認可識別子が返却されます。setUser メソッドを実行していない場合は，null が返却されます。

(5) 発生する例外

なし。

10.5.10 getHostName()

(1) 機能

setHostName メソッドで設定した HADB サーバのホスト名を取得します。

(2) 形式

```
public synchronized String getHostName() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

setHostName メソッドで設定した HADB サーバのホスト名が返却されます。setHostName メソッドを実行していない場合は，null が返却されます。

(5) 発生する例外

なし。

10.5.11 getPort()

(1) 機能

setPort メソッドで設定した HADB サーバのポート番号を取得します。

(2) 形式

```
public synchronized int getPort() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

setPort メソッドで設定した HADB サーバのポート番号が返却されます。setPort メソッドを実行していない場合は，-1 が返却されます。

(5) 発生する例外

なし。

10.5.12 setApName(String name)

(1) 機能

HADB サーバに接続する AP 識別子を設定します。

(2) 形式

```
public synchronized void setApName(String name) throws SQLException
```

(3) 引数

String name：

AP 識別子を 1～30 バイトの文字列で指定します。空白，文字列の長さが 0，null を指定した場合は，このメソッドによる AP 識別子の設定がないものと見なされます。

なお，AP 識別子をどこにも指定しないで HADB サーバに接続した場合，AP 識別子には”********”が設定されます。

(4) 戻り値

なし。

(5) 発生する例外

name に不正な値（31 バイト以上の文字列）を指定した場合，SQLException が投入されます。

(6) 留意事項

このメソッドで指定した AP 識別子は，Java 仮想マシンのデフォルトの変換文字セットで変換されます。そのため，AP 識別子には変換文字セットに依存しない半角英数字だけで構成される名称を指定することを推奨します。

10.5.13　setEncodeLang(String lang)

(1)　機能

文字コード変換時の変換文字セット名称を設定します。

(2)　形式

```
public synchronized void setEncodeLang(String lang) throws SQLException
```

(3)　引数

String lang：

変換文字セットを指定します。指定できる変換文字セットについては，『JavaTM Platform, Standard Edition JDK ドキュメント』の『国際化サポート』で示される『サポートされているエンコーディング』の一覧から選択してください。

(4)　戻り値

なし。

(5)　発生する例外

Java 仮想マシンがサポートしない変換文字セットを指定した場合，SQLException を投入します。

(6)　留意事項

このメソッドは，「表 7-15　HADB サーバの文字コードに対応する文字セット名称」の組み合わせの変換文字セット以外の文字セットで文字コード変換したい場合だけ使用してください。「表 7-15　HADB サーバの文字コードに対応する文字セット名称」の組み合わせの変換文字セットで文字コード変換する場合は，このメソッドを使用する必要はありません。

10.5.14　setInterfaceMethodTrace(boolean flag)

(1)　機能

JDBC インタフェースメソッドトレースを取得するかどうかを設定します。JDBC インタフェースメソッドトレースについては，「7.7.1　JDBC インタフェースメソッドトレース」を参照してください。

JDBC インタフェースメソッドトレースを取得する場合は，setLogWriter メソッドで出力先を設定する必要があります。

(2) 形式

```
public synchronized void setInterfaceMethodTrace(boolean flag) throws SQLException
```

(3) 引数

boolean flag：

JDBC インタフェースメソッドトレースを取得するかどうかを指定します。

- true：取得します。
- false：取得しません。

このメソッドを実行しない場合は，false が仮定されます。

(4) 戻り値

なし。

(5) 発生する例外

なし。

(6) 留意事項

JDBC インタフェースメソッドトレースの取得有無は，インスタンス単位で設定できません。このメソッドで設定した JDBC インタフェースメソッドトレースの取得有無は，設定時点および設定以降に存在するすべてのDataSource，およびConnectionPoolDataSource のインスタンスに影響します。

10.5.15 setNotErrorOccurred(boolean mode)

(1) 機能

ConnectionEventListener.connectionErrorOccurred の発生を抑止するかどうかを設定します。

(2) 形式

```
public synchronized void setNotErrorOccurred(boolean mode) throws SQLException
```

(3) 引数

boolean mode：

ConnectionEventListener.connectionErrorOccurred の発生を抑止するかどうかを指定します。

次に示すどちらかの値を指定します。

- true：connectionErrorOccurred の発生を抑止します。
- false：connectionErrorOccurred の発生を抑止しません。

このメソッドを実行しない場合は，false が仮定されます。

ConnectionPoolDataSource を使用している場合，致命的な接続エラーが発生したときに呼ばれる ConnectionEventListener.connectionErrorOccurred の，呼び出しを抑止するための設定をします。通常は未設定にするか，またはfalse を設定します。

(4) 戻り値

なし。

(5) 発生する例外

なし。

10.5.16 setPassword(String password)

(1) 機能

HADB サーバに接続する認可識別子のパスワードを設定します。

次に示すメソッドを実行する際，setUser およびsetPassword メソッドで設定した認可識別子とパスワードを使って HADB サーバに接続します。

- DataSource インタフェースのgetConnection メソッド（引数なしの場合）
- ConnectionPoolDataSource インタフェースのgetPooledConnection メソッド

なお，パスワードの指定には，優先順位があります。詳細については，「7.3.3 接続情報の優先順位」を参照してください。

(2) 形式

```
public synchronized void setPassword(String password) throws SQLException
```

(3) 引数

String password：

HADB サーバに接続する認可識別子のパスワードを指定します。null を指定した場合は，このメソッドによるパスワードの設定がないものと見なされます。

(4) 戻り値

なし。

(5) 発生する例外

なし。

10.5.17 setSQLWarningKeep(boolean mode)

(1) 機能

SQL 実行時に発生した警告情報を保持するかどうかを設定します。

(2) 形式

```
public synchronized void setSQLWarningKeep(boolean mode) throws SQLException
```

(3) 引数

boolean mode：

警告情報を保持するかどうかを指定します。次に示すどちらかの値を指定します。

- true：警告情報を保持します。
- false：警告情報を保持しません。

このメソッドを実行しない場合は，true が仮定されます。

(4) 戻り値

なし。

(5) 発生する例外

なし。

10.5.18 setTraceNumber(int num)

(1) 機能

JDBC インタフェースメソッドトレースのエントリ数を設定します。

(2) 形式

```
public synchronized void setTraceNumber(int num) throws SQLException
```

(3) 引数

int num：

JDBC インタフェースメソッドトレースのエントリ数を 10～1,000 の範囲で指定します。このメソッドを実行しない場合，JDBC インタフェースメソッドトレースのエントリ数は 500 になります。

(4) 戻り値

なし。

(5) 発生する例外

エントリ数に 10～1,000 以外の値を設定した場合は，SQLException が投入されます。

10.5.19 setUser(String user)

(1) 機能

HADB サーバに接続する認可識別子を設定します。

次に示すメソッドを実行する際，setUser およびsetPassword メソッドで設定した認可識別子とパスワードを使って HADB サーバに接続します。

- DataSource インタフェースのgetConnection メソッド（引数なしの場合）
- ConnectionPoolDataSource インタフェースのgetPooledConnection メソッド

なお，認可識別子の指定には，優先順位があります。詳細については，「7.3.3 接続情報の優先順位」を参照してください。

(2) 形式

```
public synchronized void setUser(String user) throws SQLException
```

(3) 引数

String user：

HADB サーバに接続する認可識別子を指定します。null を指定した場合は，このメソッドによる認可識別子の設定がないものと見なされます。

(4) 戻り値

なし。

(5) 発生する例外

user に指定した文字列の長さが 0 の場合は，SQLException が投入されます。

10.5.20 setHostName(String name)

(1) 機能

接続先の HADB サーバのホスト名を設定します。

HADB サーバのホスト名の指定には，優先順位があります。詳細については，「7.3.3 接続情報の優先順位」を参照してください。

(2) 形式

```
public synchronized void setHostName(String name) throws SQLException
```

(3) 引数

String name：

接続先の HADB サーバのホスト名を指定します。null を指定した場合は，このメソッドによるホスト名の設定がないものと見なされます。

(4) 戻り値

なし。

(5) 発生する例外

name に不正な値（0 バイト以下または 256 バイト以上の文字列）を指定した場合，SQLException が投入されます。

10.5.21 setPort(int port)

(1) 機能

接続先の HADB サーバのポート番号を設定します。

HADB サーバのポート番号の指定には，優先順位があります。詳細については，「7.3.3　接続情報の優先順位」を参照してください。

(2)　形式

```
public synchronized void setPort(int port) throws SQLException
```

(3)　引数

int port：

　接続先の HADB サーバのポート番号を5001～65535 の範囲で指定します。

(4)　戻り値

なし。

(5)　発生する例外

引数port の指定値が，5001～65535 以外の値を指定した場合は，SQLException が投入されます。

11 JDBC 3.0 API

この章では，JDBC 3.0 API の各インタフェースとメソッドについて説明します。

11.1　JDBC 3.0 API の追加機能に対する HADB でのサポート範囲

JDBC 3.0 API の追加機能に対する HADB でのサポート範囲を次の表に示します。

表 11-1　JDBC 3.0 API の追加機能に対する HADB でのサポート範囲

追加機能	対応するインタフェース	HADB でのサポート範囲
セーブポイント	Connection	×
	Savepoint	×
接続プール機能強化	PreparedStatement	×
パラメタメタデータ	ParameterMetaData	○
	PreparedStatement	○
自動生成キー	Connection	×
	DatabaseMetaData	×
	Statement	×
ホールダブルカーソル	Connection	○
	DatabaseMetaData	○
	Statement	○
	ResultSet	○
データベースメタデータ追加 API	DatabaseMetaData	○

（凡例）

○：HADB でサポートしています。

×：HADB ではサポートしていません。

11.2 ParameterMetaData インタフェース

ここでは，ParameterMetaData インタフェースで提供されているメソッドについて説明します。

11.2.1 ParameterMetaData インタフェースのメソッド一覧

(1) ParameterMetaData インタフェースの主な機能

ParameterMetaData インタフェースでは，主に次の機能が提供されています。

- PreparedStatement オブジェクト内のパラメタのデータ型およびデータ長などのメタ情報の返却

(2) HADB でサポートしている ParameterMetaData インタフェースのメソッド

HADB でサポートしているParameterMetaData インタフェースのメソッドの一覧を次の表に示します。

表 11-2 ParameterMetaData インタフェースのメソッドの一覧

項番	ParameterMetaData インタフェースのメソッド	機能
1	getParameterClassName(int param)	パラメタのデータ型に対する Java クラスの完全指定された名前を取得します。
2	getParameterCount()	PreparedStatement オブジェクトのパラメタ数を取得します。
3	getParameterMode(int param)	指定したパラメタのモードを取得します。
4	getParameterType(int param)	指定したパラメタの SQL データ型を取得します。
5	getParameterTypeName(int param)	指定したパラメタのデータ型を取得します。
6	getPrecision(int param)	指定したパラメタの桁数を取得します。
7	getScale(int param)	指定したパラメタの小数点以下の桁数を取得します。
8	isNullable(int param)	指定したパラメタにナル値をセットできるかどうかを返します。
9	isSigned(int param)	指定したパラメタの値が，符号付き数値かどうかを返します。

重要

この表に記載されていないメソッドは，HADB ではサポートしていません。サポートしていないメソッドを実行すると，SQLException が投入されることがあります。

(3) 必要なパッケージ名称とクラス名称

ParameterMetaData インタフェースを使用する場合に必要なパッケージ名称とクラス名称を次に示します。

- パッケージ名称：com.hitachi.hadb.jdbc
- クラス名称：AdbParameterMetaData

11.2.2 getParameterClassName(int param)

(1) 機能

パラメタのデータ型に対する Java クラスの完全指定された名前を取得します。

(2) 形式

```
public synchronized String getParameterClassName(int param) throws SQLException
```

(3) 引数

int param：

1 から始まるパラメタ番号を指定します。

(4) 戻り値

String オブジェクトが返却されます。

パラメタに対してPreparedStatement オブジェクトのsetObject メソッドによって使用される Java クラスの型をString 型で返します。パラメタのデータ型と返却値を次の表に示します。

表 11-3 getParameterClassName メソッドを実行して返却される文字列

パラメタのデータ型（HADB のデータ型）	返却される文字列
INTEGER	”java.lang.Long”
SMALLINT	”java.lang.Integer”
DOUBLE PRECISION, FLOAT	”java.lang.Double”
DECIMAL, NUMERIC	”java.math.BigDecimal”
CHAR	”java.lang.String”
VARCHAR	”java.lang.String”
DATE	”java.sql.Date”
TIME	”java.sql.Time”
TIMESTAMP	”java.sql.Timestamp”
BINARY	”java.lang.Object”

パラメタのデータ型（HADB のデータ型）	返却される文字列
VARBINARY	”java.lang.Object”
ROW	”java.sql.Object”

(5) 発生する例外

param に指定した値が0 以下，またはパラメタ数より大きい場合，SQLException が投入されます。

11.2.3 getParameterCount()

(1) 機能

PreparedStatement オブジェクトのパラメタ数を取得します。

(2) 形式

```
public synchronized int getParameterCount() throws SQLException
```

(3) 引数

なし。

(4) 戻り値

PreparedStatement オブジェクトのパラメタ数が返却されます。

(5) 発生する例外

なし。

11.2.4 getParameterMode(int param)

(1) 機能

指定したパラメタのモードを取得します。

(2) 形式

```
public synchronized int getParameterMode(int param) throws SQLException
```

(3) 引数

int param：

1 から始まるパラメタ番号を指定します。

(4) 戻り値

常にParameterMetaData.parameterModeIn が返却されます。

(5) 発生する例外

param に指定した値が0 以下，またはパラメタ数より大きい場合，SQLException が投入されます。

11.2.5 getParameterType(int param)

(1) 機能

指定したパラメタの SQL データ型を取得します。

(2) 形式

```
public synchronized int getParameterType(int param) throws SQLException
```

(3) 引数

int param：

1 から始まるパラメタ番号を指定します。

(4) 戻り値

java.sql.Types からの SQL 型が返却されます。

列のデータ型と返却値の対応については，「7.6.1 データ型のマッピング」の「(1) HADB のデータ型と JDBC の SQL データ型の対応」を参照してください。

(5) 発生する例外

param に指定した値が0 以下，またはパラメタ数より大きい場合，SQLException が投入されます。

11.2.6　getParameterTypeName(int param)

(1)　機能

指定したパラメタのデータ型を取得します。

(2)　形式

```
public synchronized String getParameterTypeName(int param) throws SQLException
```

(3)　引数

int param：

1 から始まるパラメタ番号を指定します。

(4)　戻り値

String オブジェクトが返却されます。getParameterTypeName メソッドの戻り値を次の表に示します。

表 11-4　getParameterTypeName メソッドの戻り値

パラメタのデータ型（HADB のデータ型）	戻り値（返却される文字列）
INTEGER	”INTEGER”
SMALLINT	”SMALLINT”
DECIMAL, NUMERIC	”DECIMAL”
DOUBLE PRECISION, FLOAT	”DOUBLE PRECISION”
CHAR	”CHAR”
VARCHAR	”VARCHAR”
DATE	”DATE”
TIME	”TIME”
TIMESTAMP	”TIMESTAMP”
BINARY	”BINARY”
VARBINARY	”VARBINARY”
ROW	”ROW”

(5)　発生する例外

param に指定した値が0 以下，またはパラメタ数より大きい場合，SQLException が投入されます。

11.2.7　getPrecision(int param)

(1)　機能

パラメタの桁数を取得します。

(2)　形式

```
public synchronized int getPrecision(int param) throws SQLException
```

(3)　引数

int param：

1 から始まるパラメタ番号を指定します。

(4)　戻り値

パラメタの桁数が 10 進数で返却されます。

指定したパラメタが数データ型の場合は，桁数が返却されます。指定したパラメタが数データ型でない場合は，パラメタ長がバイト単位で返却されます。getPrecision メソッドの戻り値を次の表に示します。

表 11-5　getPrecision メソッドの戻り値

パラメタのデータ型（HADB のデータ型）	戻り値（列の桁数）
INTEGER	19
SMALLINT	10
DOUBLE PRECISION FLOAT	17
DECIMAL(m,n) NUMERIC(m,n)	m
CHAR(n) VARCHAR(n)	n
DATE	10
TIME(p)	$p = 0$ の場合：8 $p > 0$ の場合：$8 + (p + 1)$
TIMESTAMP(p)	$p = 0$ の場合：19 $p > 0$ の場合：$19 + (p + 1)$
BINARY(n) VARBINARY(n)	n

パラメタのデータ型（HADB のデータ型）	戻り値（列の桁数）
ROW	*行長*※

注※

各列のデータ長の総和になります。各列のデータ長の求め方については，マニュアル『HADB　SQL リファレンス』の『データ型の種類』の『データ格納長』を参照してください。

(5)　発生する例外

param に指定した値が0 以下，またはパラメタ数より大きい場合，SQLException が投入されます。

11.2.8　getScale(int param)

(1)　機能

パラメタの小数点以下の桁数を取得します。

(2)　形式

```
public synchronized int getScale(int param) throws SQLException
```

(3)　引数

int param：

1 から始まるパラメタ番号を指定します。

(4)　戻り値

指定した列の小数点以下の桁数が 10 進数で返却されます。getScale メソッドの戻り値を次の表に示します。

表 11-6　getScale メソッドの戻り値

パラメタのデータ型（HADB のデータ型）	戻り値（小数点以下の桁数）
DECIMAL(*m*,*n*) NUMERIC(*m*,*n*)	*n*
TIME(*p*) TIMESTAMP(*p*)	*p*
上記以外	0

(5) 発生する例外

param に指定した値が0 以下，またはパラメタ数より大きい場合，SQLException が投入されます。

11.2.9 isNullable(int param)

(1) 機能

指定したパラメタにナル値をセットできるかどうかを返します。

(2) 形式

```
public synchronized int isNullable(int param) throws SQLException
```

(3) 引数

int param：

1 から始まるパラメタ番号を指定します。

(4) 戻り値

次に示すどちらかの値が返却されます。

- ParameterMetaData.parameterNoNulls：ナル値をセットできません。
- ParameterMetaData.parameterNullable：ナル値をセットできます。

(5) 発生する例外

param に指定した値が0 以下，またはパラメタ数より大きい場合，SQLException が投入されます。

11.2.10 isSigned(int param)

(1) 機能

指定したパラメタの値が，符号付き数値かどうかを返します。

(2) 形式

```
public synchronized boolean isSigned(int param) throws SQLException
```

(3) 引数

int param：

1 から始まるパラメタ番号を指定します。

(4) 戻り値

次に示すどちらかの値が返却されます。

- true：符号付き数値です。
- false：符号付き数値ではありません。

パラメタのデータ型と戻り値の関係を次の表に示します。

表 11-7 パラメタのデータ型と戻り値の関係

パラメタのデータ型	戻り値
INTEGER, SMALLINT, DOUBLE PRECISION, FLOAT, DECIMAL, NUMERIC	true
上記以外	false

(5) 発生する例外

param に指定した値が0 以下，またはパラメタ数より大きい場合，SQLException が投入されます。

11.3　サポートしていないインタフェース

HADB では，次に示すインタフェースはサポートしていません。

- Savepoint

12 JDBC 4.0 API

この章では，JDBC 4.0 API の各インタフェースとメソッドについて説明します。

12.1　JDBC 4.0 API の追加機能に対する HADB でのサポート範囲

JDBC 4.0 API の追加機能に対する HADB でのサポート範囲を次の表に示します。

表 12-1　JDBC 4.0 API の追加機能に対する HADB でのサポート範囲

追加機能	対応するインタフェース	HADB でのサポート範囲
java.sql.Driver 自動ローディング	－	○
ROWID データ型	CallableStatement	×
	PreparedStatement	×
	RowId	×
	DatabaseMetaData	×
各国文字データ型	PreparedStatement	×
	ResultSet	×
XML サポート	SQLXML	×
ラッパーパターン	Connection	○
	DatabaseMetaData	○
	DataSource	○
	ResultSet	○
	ResultSetMetaData	○
	Statement	○
	PreparedStatement	○
	ParameterMetaData	○
SQL 例外拡張	すべてのインタフェース	○
接続管理	Connection	○
	Statement	○
スカラ関数追加	Connection	○
	DatabaseMetaData	○

（凡例）

－：対応するインタフェースはありません。

○：HADB でサポートしています。

×：HADB ではサポートしていません。

12.1.1　java.sql.Driver 自動ローディング

JDBC 4.0 API では，Driver クラスの登録は不要です。ただし，Driver クラスを明示的に登録する処理があっても問題ありません。

12.1.2　ラッパーパターン

JDBC 4.0 API では，次に示すインタフェースがWrapper インタフェースを継承します。

- Connection
- DatabaseMetaData
- DataSource
- ResultSet
- ResultSetMetaData
- Statement
- PreparedStatement
- ParameterMetaData

12.1.3　SQL 例外拡張

SQLException のサブクラスとして複数の例外クラスが追加されています。HADB でのサポート可否を次の表に示します。

表 12-2　追加された例外クラスと HADB でのサポート可否

項番	クラス名	HADB でのサポート可否
1	SQLNonTransientException	○
2	SQLFeatureNotSupportedException	○
3	SQLNonTransientConnectionException	○
4	SQLDataException	○
5	SQLIntegrityConstraintViolationException	○
6	SQLInvalidAuthorizationSpecException	○
7	SQLSyntaxErrorException	○
8	SQLTransientException	○
9	SQLTransientConnectionException	○
10	SQLTransactionRollbackException	○

項番	クラス名	HADB でのサポート可否
11	SQLTimeoutException	○
12	SQLRecoverableException	○
13	SQLClientInfoException	○

（凡例）

○：HADB でサポートしています。

SQL 例外拡張の詳細については，「12.3　SQL 例外拡張機能」を参照してください。

12.1.4　接続管理

このドライバでは次に示すインタフェースのメソッドをサポートしています。

- Connection インタフェースの isValid メソッド
- Statement インタフェースの isPoolable メソッド

12.1.5　スカラ関数追加

JDBC 4.0 API で追加されたスカラ関数と，HADB でのサポート範囲を次の表に示します。

表 12-3　HADB がサポートするスカラ関数

項番	スカラ関数	HADB でのサポート範囲
1	CHAR_LENGTH	×
2	CHARACTER_LENGTH	×
3	CURRENT_DATE	○
4	CURRENT_TIME	○
5	CURRENT_TIMESTAMP	○
6	EXTRACT	○
7	OCTET_LENGTH	×
8	POSITION	×

（凡例）

○：HADB でサポートしています。

×：HADB ではサポートしていません。

12.2 Wrapper インタフェース

ここでは，Wrapper インタフェースで提供されているメソッドについて説明します。

12.2.1 Wrapper インタフェースのメソッド一覧

(1) Wrapper インタフェースの主な機能

Wrapper インタフェースでは，JDBC で規定された以外のメソッドを呼び出すための標準化された仕組みを提供します。

(2) HADB でサポートしている Wrapper インタフェースのメソッド

HADB でサポートしているWrapper インタフェースのメソッドの一覧を次の表に示します。

表 12-4 Wrapper インタフェースのメソッド一覧

項番	Wrapper インタフェースのメソッド	機能
1	isWrapperFor(Class<?> iface)	指定されたクラスのオブジェクトをunwrap メソッドで返却できるかどうかを返します。
2	unwrap(Class<T> iface)	指定されたクラスのオブジェクトを返します。

(3) 指定できるクラスの一覧

Wrapper を継承したインタフェースに対し，unwrap で指定できるクラスの一覧を次の表に示します。

表 12-5 unwrap で指定できるクラス

項番	インタフェース	unwrap で指定できるクラス
1	java.sql.Connection	AdbConnection
2	java.sql.DatabaseMetaData	AdbDatabaseMetaData
3	javax.sql.DataSource	AdbDataSource
4	java.sql.ResultSet	AdbResultSet
5	java.sql.ResultSetMetaData	AdbResultSetMetaData
6	java.sql.Statement	AdbStatement
7	java.sql.PreparedStatement	AdbPreparedStatement
8	java.sql.ParameterMetaData	AdbParameterMetaData

(4) コーディング例

Wrapper インタフェースのコーディング例を次に示します。

```
Connection con = DriverManager.getConnection(url,info);
Class<?> clazz = Class.forName("com.hitachi.hadb.jdbc.AdbConnection");
if(con.isWrapperFor(clazz)){
  AdbConnection acon = (AdbConnection)con.unwrap(clazz);
  acon.xxxx();
}
```

12.2.2 isWrapperFor(Class<?> iface)

(1) 機能

指定されたクラスのオブジェクトをunwrap メソッドで返却できるかどうかを返します。

(2) 形式

```
public synchronized boolean isWrapperFor(Class<?> iface) throws SQLException
```

(3) 引数

Class<?> iface：

チェック対象のクラス

(4) 戻り値

指定されたクラスのオブジェクトをunwrap で返却できる場合は，true が返却されます。それ以外の場合は，false が返却されます。

(5) 発生する例外

なし。

12.2.3 unwrap(Class<T> iface)

(1) 機能

指定されたクラスのオブジェクトを返却します。

(2) 形式

```
public synchronized <T> T unwrap(Class<T> iface) throws SQLException
```

(3) 引数

Class<T> iface：

チェック対象のクラス

(4) 戻り値

指定されたクラスのオブジェクトが返却されます。

(5) 発生する例外

指定されたクラスのオブジェクトを返却できない場合，SQLException が投入されます。

12.3 SQL 例外拡張機能

SQLException のサブクラスとして複数の例外クラスが追加されています。JDBC 4.0 API で返却される例外クラスの一覧と各クラスの説明を次の表に示します。

表 12-6 JDBC 4.0 API で返却される例外クラスの一覧と各クラスの説明

項番	クラス名	説明	接続状態中にエラーが発生した場合の接続状態
1	SQLNonTransientException	一時的ではないエラーを示しています。失敗した SQL 文を再実行しても，正常に実行できない場合に投入されます。	有効
2	SQLFeatureNotSupportedException	SQLSTATE のクラス値が0A（サポートされていない機能）の場合に投入されます。	有効
3	SQLNonTransientConnectionException	SQLSTATE のクラス値が08（コネクション違反）の場合に投入されます。	－
4	SQLDataException	SQLSTATE のクラス値が22（データ例外）の場合に投入されます。	有効
5	SQLIntegrityConstraintViolationException	SQLSTATE のクラス値が23（整合性制約違反）の場合に投入されます。	有効
6	SQLInvalidAuthorizationSpecException	SQLSTATE のクラス値が28（認可識別子の指定が正しくない）の場合に投入されます。	－
7	SQLSyntaxErrorException	SQLSTATE のクラス値が42（構文誤りまたはアクセス規則違反）の場合に投入されます。	有効
8	SQLTransientException	一時的なエラーを示しています。失敗した SQL 文を再実行した場合，成功する可能性があるときに投入されます。	有効
9	SQLTransientConnectionException	SQLSTATE のクラス値が08（コネクション違反）の場合に投入されます。HADB サーバが開始または終了中であるときなどが該当します。	－
10	SQLTransactionRollbackException	SQLSTATE のクラス値が40（トランザクションがロールバックした）の場合に投入されます。	有効
11	SQLTimeoutException	タイムアウトが発生した場合に投入されます。	有効
12	SQLRecoverableException	接続を再確保したあとに，失敗したトランザクションを再実行した場合，成功する可能性があるときに投入されます。	無効
13	SQLClientInfoException	設定できないクライアントのプロパティが 1 つ以上ある場合に，Connection.setClientInfo メソッドによって投入されます。	－

（凡例）

－：該当しません。

注

SQLSTATE のクラス値については，マニュアル『HADB　メッセージ』の『SQLSTATE の出力形式』を参照してください。

JDBC 4.0 API で追加された例外クラスの継承関係を次に示します。

```
java.sql.SQLException
  |
  ├ java.sql.SQLNonTransientException
  |     ├ java.sql.SQLFeatureNotSupportedException
  |     ├ java.sql.SQLNonTransientConnectionException
  |     ├ java.sql.SQLDataException
  |     ├ java.sql.SQLIntegrityConstraintViolationException
  |     ├ java.sql.SQLInvalidAuthorizationSpecException
  |     └ java.sql.SQLSyntaxErrorException
  |
  ├ java.sql.SQLTransientException
  |     ├ java.sql.SQLTransientConnectionException
  |     ├ java.sql.SQLTransactionRollbackException
  |     └ java.sql.SQLTimeoutException
  |
  ├ java.sql.SQLRecoverableException
  └ java.sql.SQLClientInfoException
```

各例外クラスは，java.sql パッケージのクラスを直接利用します。例外クラスが提供する各メソッドの詳細，使用方法については，JDBC 規格の関連ドキュメントを参照してください。

重要

executeQuery メソッドなどの SQL 文を実行するメソッドで，setQueryTimeout メソッドまたは adb_clt_rpc_sql_wait_time に指定した時間を超えたためにタイムアウトが発生した場合，JDBC 規格ではSQLTimeoutException が投入されます。ただし，HADB クライアントから HADB サーバに処理要求をしてから，応答が戻ってくるまでの待ち時間でタイムアウトが発生した場合は，HADB サーバと必ず切断されるため，SQLRecoverableException が投入されます。

12.4　サポートしていないインタフェース

HADB では，次に示すインタフェースはサポートしていません。

- NClob
- RowId
- SQLXML

13 JDBC 4.1 API

この章では，JDBC 4.1 API で追加された機能に対する HADB でのサポート範囲について説明します。

13.1　JDBC 4.1 API の追加機能に対する HADB でのサポート範囲

JDBC 4.1 API の追加機能に対する HADB でのサポート範囲を次の表に示します。

表 13-1　JDBC 4.1 API の追加機能に対する HADB でのサポート範囲

追加機能	対応するインタフェース	HADB でのサポート範囲
try-with-resources 文	Connection	○
	ResultSet	
	Statement	
getObject メソッドの変換先Java 型指定	CallableStatement	○※
	ResultSet	
親ロガーの取得	Driver	×
	DataSource	
	ConnectionPoolDataSource	
スキーマ指定	Connection	×
物理接続の中止およびタイムアウト	Connection	×
依存オブジェクトクローズ時のStatement オブジェクトクローズ	Statement	○
データベースメタデータの追加 API	DatabaseMetaData	○

（凡例）

○：HADB でサポートしています。

×：HADB ではサポートしていません。

注※

ResultSet インタフェースのgetObject メソッドです。また，一部のJava 型への変換だけをサポートしています。詳細については，「8.6.45　getObject(int columnIndex,Class<T> type)」を参照してください。

13.1.1　try-with-resources 文

リソースを記述する try 文です。try-with-resources 文によって，文の終わりで確実に各リソースが閉じられるようにできます。

Connection インタフェース，ResultSet インタフェース，およびStatement インタフェースでtry-with-resources 文を使用できます。使用方法については，JDBC 規格の関連ドキュメントを参照してください。

13.1.2　依存オブジェクトクローズ時の Statement オブジェクトクローズ

Statement インタフェースで，closeOnCompletion メソッドおよびisCloseOnCompletion メソッドをサポートしています。

14 JDBC 4.2 API

この章では，JDBC 4.2 API で追加された機能に対する HADB でのサポート範囲について説明します。

14.1　JDBC 4.2 API の追加機能に対する HADB でのサポート範囲

JDBC 4.2 API の追加機能に対する HADB でのサポート範囲を次の表に示します。

表 14-1　JDBC 4.2 API の追加機能に対する HADB でのサポート範囲

追加機能	対応するインタフェース	HADB でのサポート範囲
REF CURSOR	CallableStatement	×
SQLType インタフェース	SQLType	×
	CallableStatement	
	PreparedStatement	
	ResultSet	
	SQLOutput	
JDBCType Enum	SQLType	×
大きい更新カウント	PreparedStatement	○
	Statement	
データベースメタデータの追加 API	DatabaseMetaData	○

（凡例）

○：HADB でサポートしています。

×：HADB ではサポートしていません。

14.1.1　大きい更新カウント

更新系 SQL の更新行数を long 型で扱える次のメソッドをサポートしています。

PreparedStatement インタフェース

- executeLargeUpdate

Statement インタフェース

- getLargeUpdateCount
- setLargeMaxRows
- getLargeMaxRows
- executeLargeBatch
- executeLargeUpdate

次の SQL 文を実行する際に，返される行数がInteger.MAX_VALUE を超える可能性がある場合は，これらのメソッドを使用してください。

- UPDATE 文
- INSERT 文
- DELETE 文

15 APの作成

この章では，HADB ODBC ドライバの環境設定，および ODBC 対応の AP 作成時の留意事項について説明します。

15.1　HADB が提供している ODBC ドライバ

HADB が提供する ODBC ドライバ（以降，HADB ODBC ドライバと表記）を使用して HADB のデータベースにアクセスできます。ここでは，HADB ODBC ドライバの準拠範囲とシステム構成について説明します。

15.1.1　HADB ODBC が準拠している ODBC ドライバのバージョン

HADB ODBC ドライバは ODBC3.5 に準拠しています。ODBC インタフェースを使用して HADB のデータベースにアクセスでき，BI ツールとの連携ができます。

15.1.2　システム構成

(1)　前提プログラム（Windows 版の HADB クライアントの場合）

HADB ODBC ドライバの前提プログラムを次に示します。

■HADB ODBC ドライバ（64 ビットモード）の場合

前提となるオペレーティングシステム

- Windows Server 2012
- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022
- Windows 10(x64)
- Windows 11

前提となるソフトウェア

- Microsoft Data Access Components
- HADB クライアント

■HADB ODBC ドライバ（32 ビットモード）の場合

前提となるオペレーティングシステム

- Windows 10
- Windows 10(x64)

前提となるソフトウェア

- Microsoft Data Access Components

- HADB クライアント

(2) 前提プログラム（Linux 版の HADB クライアントの場合）

HADB ODBC ドライバの前提プログラムを次に示します。

前提となるオペレーティングシステム

- RHEL 7 バージョン 7.1 以降

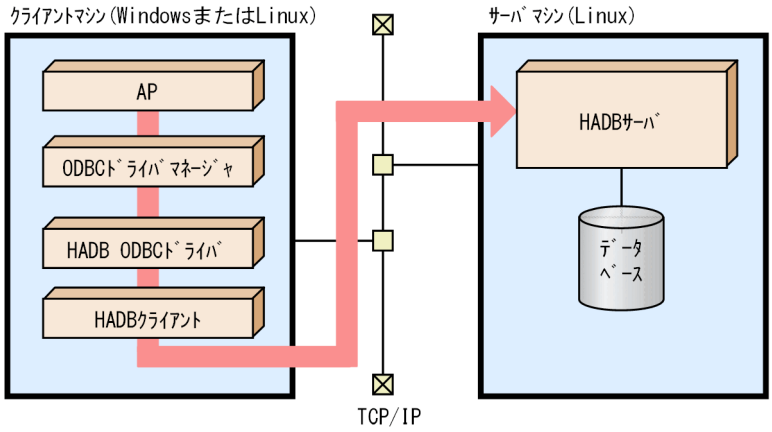
前提となるソフトウェア

- unixODBC バージョン 2.3.9（ドライバマネージャ）
- HADB クライアント

(3) 機器構成

HADB ODBC ドライバを使用する際のシステム構成を次の図に示します。

図 15-1 HADB ODBC ドライバを使用する際のシステム構成



15.1.3 文字コード変換について

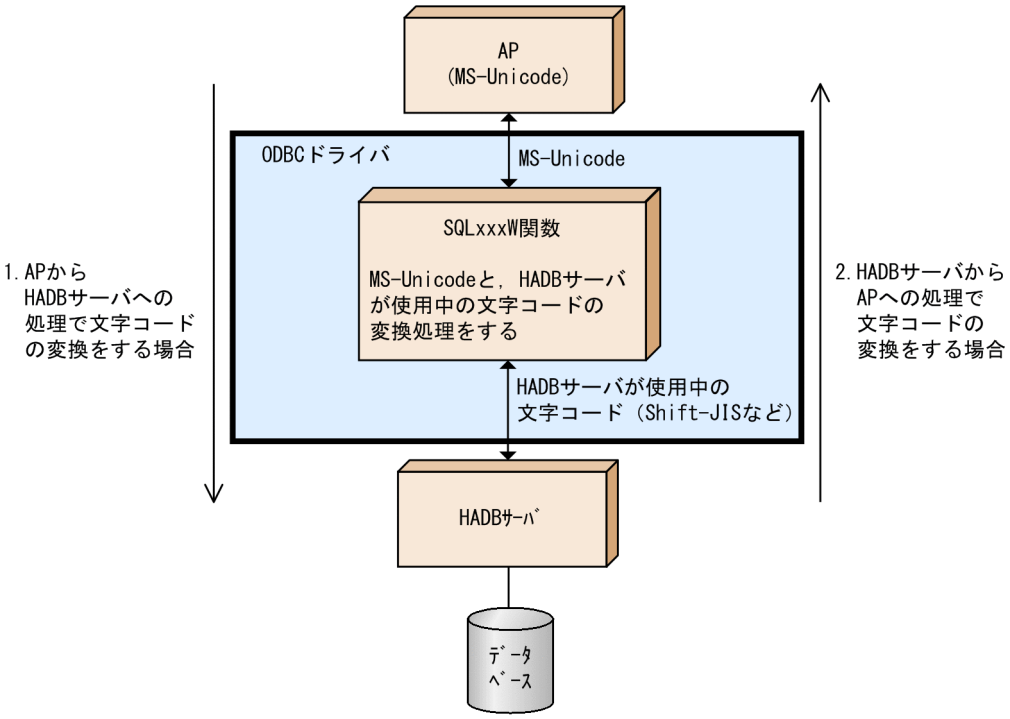
(1) 文字コード変換処理

HADB が提供する ODBC ドライバは，MS-Unicode インタフェースを含む ODBC3.5 をサポートします。

MS-Unicode インタフェースである`SQL`*`xxx`*`W` 関数を使用した場合，ODBC ドライバ内で MS-Unicode の文字列データと HADB が対応している文字コードを変換する処理を行います。

次の図に，MS-Unicode の文字列データと HADB が対応している文字コードを変換する処理を示します。

図 15-2　MS-Unicode の文字列データと HADB が対応している文字コードを変換する処理



[説明]

1. AP から HADB サーバへの処理で文字コード変換する場合，SQL*xxx*W 関数で MS-Unicode の文字列データを HADB が対応している文字コードに変換したあと，HADB サーバに処理を引き渡します。
2. HADB サーバから AP への処理で文字コード変換する場合，SQL*xxx*W 関数で HADB が対応している文字コードを MS-Unicode の文字列データに変換したあと，AP に出力します。

HADB では，次に示す文字コードをサポートしています。

- Unicode（UTF-8）
- Shift-JIS

(2)　留意事項

- ダイアクリティカルマーク（区分符号）付き文字について

  ウムラウト（ä，ö，ü）やトレマ（ë，ï，ü，ÿ）などのダイアクリティカルマーク付き文字を，比較，変数に代入，または検索する場合，ダイアクリティカルマークなし文字として扱われることがあります。これは，SQL*xxx*W 関数を使用して文字コード変換がされたとき，ダイアクリティカルマークが失われるためです。

- サロゲートペアなど，文字コード変換できない文字について

  ODBC3.5 インタフェースのSQL*xxx*W 関数を使用した場合，文字コード変換が行われます。その際，サロゲートペアの文字など，文字コード変換できない文字は，比較，格納代入時にはエラーになります。また，検索時には#に置き換えられることがあります。さらに，#が出現した位置以降の文字については，文字化けが発生することがあります。

(3) Linux 版の HADB クライアントの場合の留意事項

- Linux 版の HADB ODBC ドライバでは，ワイド文字を扱う際，1 文字を 2 バイトの UTF-16LE 形式で処理します。そのため，ワイド文字のデータを AP から HADB ODBC ドライバに渡す際には，データは UTF-16LE 形式にする必要があります。

15.1.4 ODBC カーソルライブラリの使用について

マイクロソフト社提供の ODBC カーソルライブラリを使用すると，次に示す機能が利用できます。

(1) スクロール可能なカーソル

SQL_ATTR_CURSOR_SCROLLABLE 属性にSQL_SCROLLABLE を指定すると，次のことができます。

- SQLFetchScroll による行セットデータのフェッチ
- SQLSetPos を使用した行セット内の任意の行へのカーソルの位置づけ

ただし，SQLSetPos を使用して，行セットデータを最新の状態に変更することはできません。また，結果セットのデータを更新または削除することもできません。

(2) ブックマーク機能

カーソル行のブックマークは，次に示す手順で取得できます。

1. SQL_ATTR_USE_BOOKMARKS 属性にSQL_UB_VARIABLE を指定する
2. SQLSetPos を使用して行セット内の任意の行にカーソルを位置づける
3. 第0 列のデータを取得する

また，SQL_ATTR_FETCH_BOOKMARK_PTR 属性にブックマークを指定すると，SQLFetchScroll の実行時の開始行にブックマークを指定できます。

15.1.5 配列型の列を定義した表へのアクセスについて

ODBC ドライバを使用して配列型の列にアクセスする場合，添え字に符号なし整数定数を指定した配列要素参照を指定する必要があります。例を次に示します。表T1 のC1 列が配列型の列とします。

(例)

- 実行できる SQL 文の例

```
SELECT "C1"[1],"C1"[2] FROM "T1"
```

```
SELECT "C1"[1] FROM "T1" WHERE "C1"[2]=?
```

添え字に符号なし整数定数を指定した配列要素参照を指定しているため，上記の SQL 文は実行できます。

- 実行できない SQL 文の例

```
SELECT "C1" FROM "T1"
```

```
SELECT "C1[1]" FROM T1 WHERE "C1"=?
```

添え字に符号なし整数定数を指定した配列要素参照の形式で指定していないため，上記の SQL 文は実行できません。

添え字および配列要素参照については，マニュアル『HADB　SQL リファレンス』の『配列要素参照』を参照してください。

15.2 HADB ODBC ドライバの環境設定（Windows 版の HADB クライアントの場合）

ここでは，HADB ODBC ドライバの環境設定について説明します。ここで説明する作業は，管理者特権を持つ OS ユーザが実施してください。

15.2.1 データソースの設定

データソースの設定手順を次に示します。

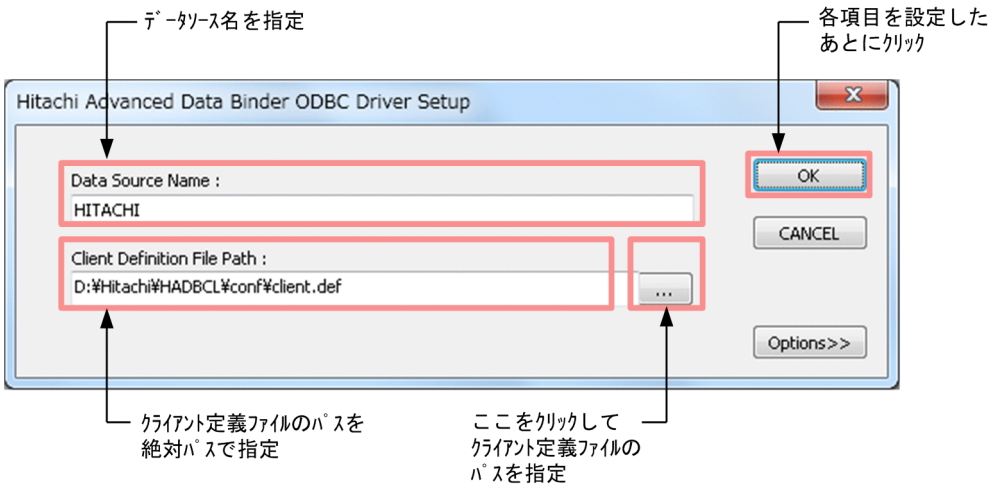
手順

1. [ODBC データソースアドミニストレーター（odbcad32.exe)］を実行します。
   実行する ODBC データソースアドミニストレーターは，HADB クライアントをインストールしたクライアントマシンの OS 種別，および BI ツールの動作モードによって異なります。ODBC データソースアドミニストレーターが格納されているフォルダを次の表に示します。

   表 15-1 ODBC データソースアドミニストレーターが格納されているフォルダ

項番	クライアントマシンの OS 種別	BI ツールの動作モード	HADB ODBC ドライバの種別	ODBC データソースアドミニストレーターが格納されているフォルダ
1	32bit	32bit	32bit	%windir%¥system32¥odbcad32.exe
2	64bit	64bit	64bit	
3		32bit(WOW64)	32bit	%windir%¥SysWOW64¥odbcad32.exe
4			64bit	－

   （凡例）
   －：この組み合わせでは使用できません。
2. 登録または修正するデータソースの種別をタブ項目から選択し，［追加］ボタンをクリックします。
3. [データソースの新規作成］ダイアログボックスが表示されます。[Hitachi Advanced Data Binder ODBC Driver］を選択してください。
4. [Hitachi Advanced Data Binder ODBC Driver Setup］ダイアログが表示されます。各項目を設定して，［OK］をクリックします。



データソース名は 32 バイト以内で入力してください。全角文字を入力する場合は，表示文字数＝バイト数ではありません。

クライアント定義ファイルのパス名は，255 バイト以内で入力してください。

> **メモ**
>
> - HADB ODBC ドライバトレース情報を出力する場合に，［Options>>］ボタンをクリックします。設定方法については，「17.3.1　ODBC データソースアドミニストレーターで設定する方法」を参照してください。
> - データソースの設定を中止する場合は，［CANCEL］をクリックします。

5. 登録したデータソースが表示されます。登録内容を確認してください。

15.2.2　レジストリキーの登録

HADB クライアントの環境設定時に，レジストリ登録コマンドを使用してレジストリキーを登録したかどうかを確認してください。レジストリキーの登録操作については，「4.2.1　Windows 版の HADB クライアントの場合」の「(1)　HADB クライアントのインストール」を参照してください。

15.2.3　データソースの削除

データソースの削除手順を次に示します。

手順

1. ［ODBC データソースアドミニストレーター（odbcad32.exe)］を実行します。

   実行する ODBC データソースアドミニストレーターは，HADB クライアントをインストールしたクライアントマシンの OS 種別，および BI ツールの動作モードによって異なります。ODBC データソースアドミニストレーターが格納されているフォルダを次の表に示します。

表 15-2　ODBC データソースアドミニストレーターが格納されているフォルダ

項番	クライアントマシンの OS 種別	BI ツールの動作モード	HADB ODBC ドライバの種別	ODBC データソースアドミニストレーターが格納されているフォルダ
1	32bit	32bit	32bit	%windir%¥system32¥odbcad32.exe
2	64bit	64bit	64bit	
3		32bit(WOW64)	32bit	%windir%¥SysWOW64¥odbcad32.exe
4			64bit	－

（凡例）

－：この組み合わせでは使用できません。

2. 削除するデータソースの種別をタブ項目から選択し，削除するデータソース名を選択します。
3. [削除] ボタンをクリックします。

15.3 HADB ODBC ドライバの環境設定（Linux 版の HADB クライアントの場合）

HADB ODBC ドライバは，HADB クライアントをインストールしたときにクライアントライブラリと同時にインストールされます。HADB クライアントのインストールについては，「4.2.2 Linux 版の HADB クライアントの場合」の「(2) HADB クライアントのインストール」を参照してください。

ここでは，HADB ODBC ドライバの環境設定について説明します。

15.3.1 データソースの設定

オープンソースで提供されている，unixODBC をドライバマネージャとした場合のデータソースの登録手順を次に示します。なお，手順で示す指定例は，HADB クライアントのインストール先ディレクトリを/home/osuser01/clientdir としています。

> **メモ**
>
> HADB ODBC ドライバは，unixODBC が提供する GUI ツールの ODBC Config を使用したデータソースの登録をサポートしていません。

手順

1. odbcinst.ini を設定する

   HADB ODBC ドライバの情報を登録するために，/usr/local/etc フォルダに格納されている odbcinst.ini を編集します。ODBC トレースを出力する場合は，ODBC トレースの開始と停止の指示，および ODBC トレースの出力先の情報をodbcinst.ini に記述します。編集したodbcinst.ini の設定内容は，すべてのユーザに反映されます。

   なお，odbcinst.ini はroot が所有しているため，root で編集する必要があります。

   指定例

```
[Hitachi Advanced Data Binder]                                  ...a
Driver = /home/osuser01/clientdir/client/lib/libadbodbc.so      ...b

[ODBC]                                                          ...c
Trace = YES                                                     ...d
TraceFile = /home/osuser01/clientdir/spool/odbctrc_DM.log       ...e
```

   [説明]

   a. データソースと対応するドライバ名称を指定します。「Hitachi Advanced Data Binder」を指定してください。

   b. HADB ODBC ドライバを絶対パスで指定します。

   c. ODBC トレースを出力する場合に，[ODBC]タグを指定します。

d. ODBC トレースを出力する場合に指定します。

ODBC トレースの出力の開始と停止を，「YES」または「NO」で指定できます。

e. ODBC トレースを出力する場合に，ODBC トレースを出力するファイルの絶対パスを指定します。ファイル名は任意の名称を指定してください。

なお，ODBC トレースの出力先フォルダは，AP を実行するユーザにアクセス権限があるフォルダを指定してください。

2..odbc.ini を設定する

HADB ODBC ドライバの接続設定をするために，.odbc.ini を編集します。.odbc.ini は，アプリケーションの実行ユーザのホームディレクトリに格納してください。

指定例

```
[HADB]                                                  ...a
Driver = Hitachi Advanced Data Binder                   ...b
CLTPATH = /home/osuser01/clientdir/conf/client.def      ...c
```

[説明]

a. データソース名を識別するための任意の名称を指定します。指定値の制限については，ドライバマネージャの仕様に依存します。

b. ドライバ名称を指定します。

手順 1.でodbcinst.ini に指定したドライバ名称の「Hitachi Advanced Data Binder」を指定してください。

c. このデータソースで使用するクライアント定義ファイル（client.def）の絶対パスを指定します。

この指定は任意です。指定を省略した場合，環境変数ADBCLTDIR に指定したディレクトリ下のクライアント定義ファイル（$ADBCLTDIR/conf/client.def）が使用されます。

15.4　HADB ODBC ドライバを使用した HADB サーバへの接続

HADB ODBC ドライバを使用して HADB サーバに接続するには，次の 2 つの ODBC 関数を使用します。

1. ハンドルを割り当てる ODBC 関数
   - SQLAllocHandle
2. HADB ODBC ドライバとデータソース（HADB サーバ）の接続を確立させる ODBC 関数
   次に示す ODBC 関数のどれか 1 つを実行します。
   - SQLConnect またはSQLConnectW
   - SQLDriverConnect またはSQLDriverConnectW
   - SQLBrowseConnect またはSQLBrowseConnectW

上記の ODBC 関数を使用したときの留意事項を説明します。

■SQLAllocHandle

最初に環境ハンドルを割り当てます。そのあと，割り当てられた環境ハンドルを使用して，接続ハンドルを割り当てます。割り当てられた接続ハンドルは，SQLConnect 関数などを実行して，HADB ODBC ドライバとデータソース（HADB サーバ）の接続を確立する際に使用します。

SQLAllocHandle の詳細については，「16.3.1　SQLAllocHandle」を参照してください。

■SQLConnect またはSQLConnectW

接続の確立に必要なデータソース名，認可識別子，およびパスワードを関数の引数として指定します。引数すべての指定が必要です。

SQLConnect またはSQLConnectW の詳細については，「16.3.2　SQLConnect，SQLConnectW」を参照してください。

■SQLDriverConnect またはSQLDriverConnectW，およびSQLBrowseConnect またはSQLBrowseConnectW

接続の際，接続文字列には次の指定が必要です。

- 接続文字列にデータソース名称（DSN）または ODBC ドライバ名称（DRIVER）
- 認可識別子（UID）
- パスワード（PWD）

接続文字列の指定例を次に示します。

- データソース名称を指定した接続の場合
  "DSN=*XXXXX*;UID=*YYYYY*;PWD=*ZZZZZ*"
- ODBC ドライバ名称を指定した接続の場合
  "DRIVER=Hitachi Advanced Data Binder ODBC Driver;UID=*YYYYY*;PWD=*ZZZZZ*"

SQLDriverConnect またはSQLDriverConnectW の詳細については，「16.3.3　SQLDriverConnect，SQLDriverConnectW」を参照してください。SQLBrowseConnect またはSQLBrowseConnectW の詳細については，「16.3.4　SQLBrowseConnect，SQLBrowseConnectW」を参照してください。

15.5　データ型の対応

ここでは，データ型の対応関係について説明します。

15.5.1　ODBC の SQL データ型と HADB のデータ型の対応

ODBC の SQL データ型と HADB のデータ型の対応を次の表に示します。

表 15-3　ODBC の SQL データ型と HADB のデータ型の対応

分類	ODBC の SQL データ型	対応する HADB のデータ型	説明	使用可否
文字データ	SQL_CHAR	CHARACTER	固定長文字列	○
	SQL_VARCHAR	VARCHAR	可変長文字列	○
	SQL_LONGVARCHAR	－	可変長文字列	×
	SQL_WCHAR	CHARACTER	固定長文字列（Unicode 用）	○
	SQL_WVARCHAR	VARCHAR	可変長文字列（Unicode 用）	○
	SQL_WLONGVARCHAR	－	可変長文字列（Unicode 用）	×
数データ	SQL_DECIMAL	DECIMAL，NUMERIC	固定小数点数	○
	SQL_NUMERIC	DECIMAL，NUMERIC	固定小数点数	○
	SQL_TINYINT	－	1 バイトの整数	×
	SQL_SMALLINT	－	2 バイトの整数	○※1
	SQL_INTEGER	SMALLINT	4 バイトの整数	○
	SQL_BIGINT	INTEGER	8 バイトの整数	○
	SQL_REAL	－	単精度浮動小数点数	×
	SQL_FLOAT	DOUBLE PRECISION，FLOAT	倍精度浮動小数点数	○
	SQL_DOUBLE	DOUBLE PRECISION，FLOAT	倍精度浮動小数点数	○
	SQL_BIT	－	ビット	×
	SQL_BINARY	BINARY	固定長バイナリデータ	○
	SQL_VARBINARY	VARBINARY	可変長バイナリデータ	○
	SQL_LONGVARBINARY	－	可変長バイナリデータ	×

分類	ODBC の SQL データ型	対応する HADB のデータ型	説明	使用可否
日付，時刻データ	SQL_TYPE_DATE (, SQL_DATE) ※2	DATE	日付	○
	SQL_TYPE_TIME (, SQL_TIME) ※2	TIME	時刻	○
	SQL_TYPE_TIMESTAMP (, SQL_TIMESTAMP) ※2	TIMESTAMP	時刻印	○
	SQL_INTERVAL_MONTH	－	月間隔	×
	SQL_INTERVAL_YEAR	－	年間隔	×
	SQL_INTERVAL_YEAR_TO_MONTH	－	－	×
	SQL_INTERVAL_DAY	－	日間隔	×
	SQL_INTERVAL_HOUR	－	時間隔	×
	SQL_INTERVAL_MINUTE	－	分間隔	×
	SQL_INTERVAL_SECOND	－	秒間隔	×
	SQL_INTERVAL_DAY_TO_HOUR	－	－	×
	SQL_INTERVAL_DAY_TO_MINUTE	－	－	×
	SQL_INTERVAL_DAY_TO_SECOND	－	－	×
	SQL_INTERVAL_HOUR_TO_MINUTE	－	－	×
	SQL_INTERVAL_HOUR_TO_SECOND	－	－	×
	SQL_INTERVAL_MINUTE_TO_SECOND	－	－	×
そのほか	SQL_GUID	－	－	×

(凡例)

－：対応するデータ型なし

○：使用できるデータ型

×：使用できないデータ型

注※1

SQL_SMALLINT に対応する HADB のデータ型はありません。そのため，SQL_SMALLINT を使用して HADB のデータベースにアクセスできません。SQL_SMALLINT は，HADB ODBC ドライバが提供する一部のカタログ関数のインタフェースとしてだけ使用できます。例えば，SQLColumns の結果セット列の SQL_DATA_TYPE の値などで使用できます。

注※2

SQL_TYPE_DATE，SQL_TYPE_TIME，SQL_TYPE_TIMESTAMP は，ODBC3.0 の日付時刻データ型です。通常はこちらを使用してください。

( ) 内は ODBC2.0 の日付時刻データ型です。ODBC3.0 の日付時刻データ型と同等に扱われます。環境変数ADBODBAPMODE の指定によっては，メタデータの返却値として，( ) 内の識別子が返却されることがあります。

15.5.2　ODBC の SQL データ型と C データ型の対応

ODBC の SQL データ型と C データ型の対応について，比較・格納代入できるデータ型の組み合わせ，および検索できるデータ型の組み合わせを次の表に示します。

表 15-4　比較・格納代入できるデータ型の組み合わせ（C データ型から ODBC の SQL データ型への変換）

ODBCのSQLデータ型	Cデータ型																			
	SQL_C_CHAR	SQL_C_WCHAR	SQL_C_BIT	SQL_C_NUMERIC	SQL_C_STINYINT	SQL_C_UTINYINT	SQL_C_TINYINT	SQL_C_SBIGINT	SQL_C_UBIGINT	SQL_C_SSHORT	SQL_C_USHORT	SQL_C_SHORT	SQL_C_SLONG	SQL_C_ULONG	SQL_C_LONG	SQL_C_FLOAT	SQL_C_DOUBLE	SQL_C_BINARY	SQL_C_TYPE_DATE (, SQL_C_DATE)	SQL_C_TYPE_TIME (, SQL_C_TIME)	SQL_C_TYPE_TIMESTAMP (, SQL_C_TIMESTAMP)
SQL_CHAR	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SQL_WCHAR	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SQL_VARCHAR	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SQL_WVARCHAR	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SQL_DECIMAL	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－
SQL_NUMERIC	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－
SQL_INTEGER	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	－	－	－
SQL_SMALLINT	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－
SQL_BIGINT	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	－	－	－
SQL_FLOAT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○	－	－	－
SQL_DOUBLE	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○	－	－	－
SQL_BINARY	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	●	－	－	－
SQL_VARBINARY	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	●	－	－	－
SQL_TYPE_DATE (, SQL_DATE)	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	●	－	○
SQL_TYPE_TIME (, SQL_TIME)	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	●	○
SQL_TYPE_TIMESTAMP (, SQL_TIMESTAMP)	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	○	●

（凡例）

●：デフォルトの変換，または推奨する組み合わせです。

○：変換できます。ただし，変換時にデータの切り捨てや精度が失われることがあります。

－：変換できません。

表 15-5　検索できるデータ型の組み合わせ（ODBC の SQL データ型から C データ型への変換）

ODBCのSQLデータ型	Cデータ型																			
	SQL_C_CHAR	SQL_C_WCHAR	SQL_C_BIT	SQL_C_NUMERIC	SQL_C_STINYINT	SQL_C_UTINYINT	SQL_C_TINYINT	SQL_C_SBIGINT	SQL_C_UBIGINT	SQL_C_SSHORT	SQL_C_USHORT	SQL_C_SHORT	SQL_C_SLONG	SQL_C_ULONG	SQL_C_LONG	SQL_C_FLOAT	SQL_C_DOUBLE	SQL_C_BINARY	SQL_C_TYPE_DATE (, SQL_C_DATE)	SQL_C_TYPE_TIME (, SQL_C_TIME)	SQL_C_TYPE_TIMESTAMP (, SQL_C_TIMESTAMP)
SQL_CHAR	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SQL_WCHAR	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SQL_VARCHAR	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SQL_WVARCHAR	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SQL_DECIMAL	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－
SQL_NUMERIC	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－
SQL_INTEGER	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	－	－	－
SQL_SMALLINT	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	－	－	－
SQL_BIGINT	○※	○	○	○	○	○	○	●※	○	○	○	○	○	○	○	○	○	○	－	－	－
SQL_FLOAT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○	－	－	－
SQL_DOUBLE	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○	－	－	－
SQL_BINARY	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	●	－	－	－
SQL_VARBINARY	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	●	－	－	－
SQL_TYPE_DATE (, SQL_DATE)	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	●	－	○
SQL_TYPE_TIME (, SQL_TIME)	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	●	○
SQL_TYPE_TIMESTAMP (, SQL_TIMESTAMP)	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	○	●

（凡例）

●：デフォルトの変換，または推奨する組み合わせです。

○：変換できます。ただし，変換時にデータの切り捨てや精度が失われることがあります。

－：変換できません。

注※

環境変数ADBODBAPMODE にACCESS を指定している場合，SQL_C_CHAR がデフォルトの変換になります。

15.5.3　データ型変換時の注意事項

(1)　BINARY 型および VARBINARY 型に関する規則

■比較・格納代入時の規則

- HADB のBINARY 型またはVARBINARY 型に対してデータ入力する際の規則を次に示します。
  - 入力データがSQL_C_WCHAR の場合，CHAR 型への変換後に，SQL_C_CHAR からバイナリ型への変換と同等の処理が行われます。
  - 入力データが文字データ（SQL_C_WCHAR の場合はSQL_C_CHAR への変換後のデータ）の場合，2 バイトごとに 1 バイト（8 ビット）のバイナリデータに変換されます。文字データは，2 バイトで 1 つの 16 進数を表しています。例えば，01 はバイナリの00000001 に変換され，FF はバイナリの11111111 に変換されます。したがって，「入力データのバイト長÷2」が「入力先のデータの定義長」を超える場合，エラーとなります。
- SQL_C_BINARY 型のデータを，HADB の各データ型に対してデータ入力する際の規則を次に示します。
  - 入力先が文字列系データ（SQL_CHAR など）の場合，入力データの実長が入力先のデータの定義長を超えるとエラーになります。

■検索時の規則

- HADB のBINARY 型またはVARBINARY 型から，データを出力する際の規則を次に示します。
  - 文字列データで受け取る場合，変換元のデータの各バイト（8 ビット）は，2 文字の ASCII 文字として表示されます。この文字は，16 進数を ASCII 表現したものです。例えば，バイナリの00000001 は01 に変換され，バイナリの11111111 はFF に変換されます。BuffeLength が，「変換元のデータのバイト長×2（SQL_C_WCHAR の場合はさらに×2)」以下の場合，切り捨て（警告）が発生します。

(2)　DATE 型，TIME 型，および TIMESTAMP 型に関する規則

- DATE 型の日付データまたはCHAR 型の日付データをTIMESTAMP 型の時刻印データに変換する場合，時刻印データの時刻部分には 0 が設定されます。
- TIME 型の時刻データまたはCHAR 型の日付データをTIMESTAMP 型の時刻印データに変換する場合，時刻印データの日付部分には現在の日付が設定されます。
- TIME 型の時刻データまたはCHAR 型の時刻データを，TIME 型の時刻データまたはTIMESTAMP 型の時刻印データに変換する際，変換前データの小数秒の桁数が，変換後データの小数秒の桁数を超える場合は，超える部分については切り捨てられます。また，このときの戻り値はSQL_SUCCESS になります。
- TIMESTAMP 型の時刻印データまたはCHAR 型の時刻印データをDATE 型の日付データに変換する場合，変換前の時刻印データの日付部分だけが変換されます。また，このときの戻り値はSQL_SUCCESS になります。
- TIMESTAMP 型の時刻印データまたはCHAR 型の時刻印データをTIME 型の時刻データに変換する場合，変換前の時刻印データの時刻部分だけが変換されます。また，このときの戻り値はSQL_SUCCESS になります。

- TIMESTAMP 型の時刻印データまたはCHAR 型の時刻印データを，TIME 型の時刻データまたはTIMESTAMP 型の時刻印データに変換する際，変換前データの小数秒の桁数が，変換後データの小数秒の桁数を超える場合は，超える部分については切り捨てられます。また，このときの戻り値はSQL_SUCCESS になります。
- VARCHAR 型についての扱いは，CHAR 型と同じになります。

(3) SQL_DECIMAL 型および SQL_NUMERIC 型に関する規則

HADB では，SQL_DECIMAL 型とSQL_NUMERIC 型についての扱いは同じです。どちらでも使用できます。

「精度の値＝スケールの値」となる桁数で，かつ 1 未満の小数点を持つ値をSQL_DECIMAL 型またはSQL_NUMERIC 型に対してデータ型を変換する場合，小数点の前に 0 を入力するとエラーになります。これは 0 も 1 桁としてカウントするためです。0 を付けないで，小数点から記述を始めると正常に変換できます。

(例)

テーブルに定義されたデータ型が DECIMAL(3,3)の値を検索，またはDECIMAL(3,3)の値に比較・代入する場合

- 0.123：エラーになります。
- .123：正常に変換されます。

(4) SQL_DOUBLE 型および SQL_FLOAT 型に関する規則

HADB では，SQL_DOUBLE 型とSQL_FLOAT 型についての扱いは同じです。どちらでも使用できます。

(5) SQL_C_NUMERIC 型に関する規則

- HADB のSQL_C_NUMERIC 型で使用できる最大精度は 38 になります。
- SQL_DOUBLE 型，SQL_FLOAT 型からSQL_C_NUMERIC 型にデータ型を変換する場合，またはSQL_CHAR 型，SQL_VARCHAR 型の数値文字列からSQL_C_NUMERIC 型にデータ型を変換する場合，変換元のデータ形式が浮動小数点数の形式のときは，いったん固定小数点数の形式に変換し，そのあとにSQL_C_NUMERIC 型に変換します。
- 浮動小数点数から固定小数点数への変換，または固定小数点数からSQL_C_NUMERIC 型への変換規則を次に示します。
  - 浮動小数点数の形式は，浮動小数点数定数の形式に準拠します。ただし，仮数部は 38 桁，指数部は 3 桁まで有効とします。浮動小数点数定数の形式については，マニュアル『HADB　SQL リファレンス』の『定数の記述形式』を参照してください。
  - 指数として指定できる最大値は 38，最小値は-38 になります。
  - 指数が正の値の場合，仮数部の整数の桁数と，指数の値の合計が 38 を超えると，固定小数点数の形式に変換する際に，整数の切り捨てが発生するため，SQL_ERROR を返します。
  - 指数が負の値の場合，仮数部の小数の桁数と，指数の絶対値の合計が 38 を超えると，固定小数点数の形式に変換する際に，小数の桁あふれが発生します。桁があふれた部分に 0 以外の数字がある場合は，SQL_SUCCESS_WITH_INFO を返します。そうでない場合は，SQL_SUCCESS を返します。

15.6　エラー発生時に返却される情報

ここでは，HADB ODBC ドライバまたは HADB（HADB サーバおよび HADB クライアント）でエラーが発生したときに返却される情報について説明します。

エラー情報の取得方法

エラーが発生したときに最後に使用したハンドルを，SQLGetDiagField およびSQLGetDiagRec，または SQLGetDiagFieldW およびSQLGetDiagRecW に指定してエラー情報を取得します。

SQLGetDiagField およびSQLGetDiagRec，またはSQLGetDiagFieldW およびSQLGetDiagRecW で取得できるエラー情報の内容は，エラーが発生した場所によって異なります。

■HADB ODBC ドライバでエラーが発生した場合

- SQLSTATE は ODBC の規約に従い，エラーごとに設定されます。SQLSTATE については，『MSDN ライブラリ』の『ODBC プログラマーズリファレンス』，およびマニュアル『HADB　メッセージ』の『SQLSTATE』を参照してください。
- NativeCode には常に0 が返却されます。
- メッセージテキストには，次の文字列の後ろにKFAA72 で始まるメッセージが設定されます。

  [Hitachi Advanced Data Binder][ODBC Driver]

■HADB でエラーが発生した場合

- SQLSTATE には HADB で発生したエラーに対応するSQLSTATE が設定されます。SQLSTATE については，マニュアル『HADB　メッセージ』の『SQLSTATE の一覧』を参照してください。
- NativeCode にはSQLCODE が設定されます。
- メッセージテキストには，HADB から返却されるメッセージが次の文字列の後ろに設定されます。

  [Hitachi Advanced Data Binder][ODBC Driver]

  HADB クライアントから詳細情報のメッセージが取得できない場合は，上記の文字列だけが設定されます。

15.7　制限事項

ODBC ドライバを経由して HADB にアクセスする場合の制限事項について説明します。

15.7.1　ROW の指定

ROW 指定の問合せは実行できません。また，ROW を指定したUPDATE 文またはINSERT 文は実行できません。

15.7.2　AUTOCOMMIT の仕様

AUTOCOMMIT がON の場合，SQLExecute，SQLExecDirect，SQLExecDirectW，またはSQLCloseCursor の実行後に自動的に発行されるコミットによって，実行済みの前処理情報（SQLPrepare を実行した内容）は削除されないで，コミットが実行されます。

（例）

1. ステートメントA でSQLPrepare を実行
2. ステートメントB でSQLPrepare を実行
3. ステートメントA でSQLExecute を実行（コミットが自動的に発行される）
4. ステートメントB でSQLExecute を実行（3.で前処理情報は削除されないで，コミットが実行される）

15.7.3　最大 SQL 処理リアルスレッド数に関する注意事項

同一の接続ハンドルで割り当てた複数のステートメントハンドルで SQL 文を同時に実行した場合に，最大 SQL 処理リアルスレッド数分の処理リアルスレッドが使用できないときは，SQL 文の実行要求がエラーとなります（待ち状態にはなりません）。

なお，最大 SQL 処理リアルスレッド数は，次のオペランドで指定しています。

- サーバ定義のadb_sql_exe_max_rthd_num
- クライアント定義のadb_sql_exe_max_rthd_num

15.8　注意事項

15.8.1　更新操作によるカーソルを使用した検索への影響

カーソルを使用した検索中に更新操作を行うと，タイミングによっては更新操作の結果が，検索の結果に反映されることがあります。更新操作の結果を検索の結果に反映させないようにするには，次のように運用してください。

- カーソルを閉じたあとに，行の追加または更新を行う
- 追加または更新する行が，検索結果と一致しないようにデータや探索条件などを工夫する

15.8.2　HADB ODBC ドライバを ODBC2.x アプリケーションで使用する場合の注意事項

環境ハンドルのSQL_ATTR_ODBC_VERSION 属性にSQL_OV_ODBC2 を指定した場合，ここで説明する項目については，HADB ODBC ドライバは ODBC2.x 規格に従って動作します。

(1)　SQLSTATE について

オープングループおよび ISO の規格に合わせて，ODBC3.x で幾つかのSQLSTATE が変更されています。HADB ODBC ドライバは，次のマッピング表に従って ODBC2.x のSQLSTATE を返却します。SQLSTATE については，マニュアル『HADB　メッセージ』の『SQLSTATE の一覧』を参照してください。

表 15-6　SQLSTATE のマッピング表

項番	ODBC3.x	ODBC2.x
1	07005	24000
2	07009※	S1002
3		S1093
4	22007	22008
5	22018	22005
6	HY001	S1001
7	HY003	S1003
8	HY004	S1004
9	HY007	S1010
10	HY009	S1009
11	HY010	S1010

項番	ODBC3.x	ODBC2.x
12	HY011	S1011
13	HY024	S1009
14	HY090	S1090
15	HY091	S1091
16	HY092	S1092
17	HY096	S1096
18	HY100	S1100
19	HY104	S1104
20	HY105	S1105
21	HYC00	S1C00

注※

SQLSTATE が07009 の場合，1 対 1 のマッピングになりません。マッピングは次の表に従ってください。

表 15-7　SQLSTATE 07009 のマッピング

項番	SQLSTATE 07009 を返却する関数	SQL_OV_ODBC2 指定時の SQLSTATE 07009 のマッピング先の SQLSTATE
1	SQLBindParameter	S1093
2	SQLColAttribute	S1002
3	SQLDescribeCol	07009
4	SQLDescribeParam	S1093
5	SQLFetch	S1002
6	SQLGetData	S1002
7	SQLGetDescField	07009
8	SQLGetDescRec	07009
9	SQLSetDescRec	07009
10	SQLSetDescField	07009

(2)　SQL_NO_DATA の戻り値について

次のどれかの ODBC 関数を呼び出して，DELETE 文またはUPDATE 文を実行した際に，対象となる行がないときは，SQL_SUCCESS が返却されます。SQL_NO_DATA は返却されません。

- SQLExecDirect
- SQLExecute

- SQLParamData

(3) SQLGetInfo の返却値について

環境ハンドルのSQL_ATTR_ODBC_VERSION の指定値によって，SQL_ALTER_TABLE への返却値が次のように異なります。

- SQL_ATTR_ODBC_VERSION にSQL_OV_ODBC3 を指定した場合

  SQL_ALTER_TABLE には，SQL_AT_ADD_COLUMN_SINGLE，SQL_AT_DROP_COLUMN_CASCADE，または SQL_AT_DROP_COLUMN_RESTRICT が返却されます。
- SQL_ATTR_ODBC_VERSION にSQL_OV_ODBC2 を指定した場合

  SQL_ALTER_TABLE には，SQL_AT_ADD_COLUMN またはSQL_AT_DROP_COLUMN が返却されます。

(4) 制限事項

Linux 版 HADB ODBC ドライバを使用する場合，ODBC2.0 関数（SQLSetScrollOptions 関数など）は実行できません。

16 ODBC 関数

この章では，HADB が提供する ODBC 関数について説明します。

16.1 ODBC 関数の一覧

ODBC 関数の一覧を次の表に示します。

表 16-1 ODBC 関数の一覧

項番	分類	API 名称	HADB ODBC3.0 でのサポート有無	HADB ODBC3.5 でのサポート有無	備考
1	データソースとの接続	SQLAllocHandle	○	○	−
2		SQLConnect	○	○	
3		SQLConnectW※	×	○	
4		SQLDriverConnect	○	○	
5		SQLDriverConnectW※	×	○	
6		SQLBrowseConnect	○	○	
7		SQLBrowseConnectW※	×	○	
8	ドライバおよびデータソースの情報取得	SQLDataSources	△	△	
9		SQLDataSourcesW※	×	△	
10		SQLDrivers	△	△	
11		SQLDriversW※	×	△	
12		SQLGetInfo	○	○	
13		SQLGetInfoW※	×	○	
14		SQLGetFunctions	○	○	
15		SQLGetTypeInfo	○	○	
16		SQLGetTypeInfoW※	×	○	
17	ドライバオプションの設定および取得	SQLSetConnectAttr	○	○	
18		SQLSetConnectAttrW※	×	○	
19		SQLGetConnectAttr	○	○	
20		SQLGetConnectAttrW※	×	○	
21		SQLSetEnvAttr	○	○	
22		SQLGetEnvAttr	○	○	
23		SQLSetStmtAttr	○	○	
24		SQLSetStmtAttrW※	×	○	
25		SQLGetStmtAttr	○	○	

項番	分類	API 名称	HADB ODBC3.0 でのサポート有無	HADB ODBC3.5 でのサポート有無	備考
26		SQLGetStmtAttrW※	×	○	
27	ディスクリプタ値の設定	SQLGetDescField	○	○	
28		SQLGetDescFieldW※	×	○	
29		SQLGetDescRec	○	○	
30		SQLGetDescRecW※	×	○	
31		SQLSetDescField	○	○	
32		SQLSetDescFieldW※	×	○	
33		SQLSetDescRec	○	○	
34		SQLCopyDesc	○	○	
35	SQL 要求の作成	SQLPrepare	○	○	
36		SQLPrepareW※	×	○	
37		SQLBindParameter	○	○	
38		SQLGetCursorName	○	○	
39		SQLGetCursorNameW※	×	○	
40		SQLSetCursorName	○	○	
41		SQLSetCursorNameW※	×	○	
42		SQLDescribeParam	○	○	
43		SQLNumParams	○	○	
44	SQL の実行	SQLExecute	○	○	
45		SQLExecDirect	○	○	
46		SQLExecDirectW※	×	○	
47		SQLNativeSql	○	○	
48		SQLNativeSqlW※	×	○	
49		SQLParamData	○	○	
50		SQLPutData	○	○	
51	実行結果および実行結果情報の取得	SQLRowCount	○	○	
52		SQLNumResultCols	○	○	
53		SQLDescribeCol	○	○	
54		SQLDescribeColW※	×	○	

項番	分類	API 名称	HADB ODBC3.0 でのサポート有無	HADB ODBC3.5 でのサポート有無	備考
55		SQLColAttribute	○	○	
56		SQLColAttributeW※	×	○	
57		SQLBindCol	○	○	
58		SQLFetch	○	○	
59		SQLFetchScroll	×	×	HADB には，逆方向のカーソルなどの機能がないため，無条件にエラーになります。ただし，マイクロソフト社提供のカーソルライブラリを使用すると，この関数を使用できます。 詳細については，「15.1.4　ODBC カーソルライブラリの使用について」を参照してください。
60		SQLGetData	○	○	－
61		SQLSetPos	×	×	無条件にエラーになります。ただし，マイクロソフト社提供のカーソルライブラリを使用すると，この関数を使用できます。 詳細については，「15.1.4　ODBC カーソルライブラリの使用について」を参照してください。
62		SQLBulkOperations	×	×	HADB はブックマークをサポートしていないため，無条件にエラーになります。
63		SQLMoreResults	○	○	常にSQL_NO_DATA が返却されます。
64		SQLGetDiagField	○	○	－
65		SQLGetDiagFieldW※	×	○	
66		SQLGetDiagRec	○	○	
67		SQLGetDiagRecW※	×	○	

項番	分類	API 名称	HADB ODBC3.0 でのサポート有無	HADB ODBC3.5 でのサポート有無	備考
68	データソースのシステム情報の取得	SQLColumnPrivileges	○	○	
69		SQLColumnPrivilegesW※	×	○	
70		SQLColumns	○	○	
71		SQLColumnsW※	×	○	
72		SQLForeignKeys	○	○	
73		SQLForeignKeysW※	×	○	
74		SQLPrimaryKeys	○	○	
75		SQLPrimaryKeysW※	×	○	
76		SQLProcedureColumns	○	○	検索結果セットの行数は，常に 0 行になります。
77		SQLProcedureColumnsW※	×	○	
78		SQLProcedures	○	○	
79		SQLProceduresW※	×	○	
80		SQLSpecialColumns	○	○	
81		SQLSpecialColumnsW※	×	○	
82		SQLStatistics	○	○	－
83		SQLStatisticsW※	×	○	
84		SQLTablePrivileges	○	○	
85		SQLTablePrivilegesW※	×	○	
86		SQLTables	○	○	
87		SQLTablesW※	×	○	
88	SQL 実行の終了	SQLFreeStmt	○	○	
89		SQLCloseCursor	○	○	
90		SQLCancel	○	○	
91		SQLEndTran	○	○	
92	データソースとの切断	SQLDisconnect	○	○	
93		SQLFreeHandle	○	○	

（凡例）

○：サポート

×：未サポート

△：ドライバマネージャが提供

－：なし

注※

SQL*xxx*W 関数では，MS-Unicode の文字列データと HADB が対応している文字コードを変換する処理を行います。文字コード変換処理については，「15.1.3　文字コード変換について」を参照してください。

16.2 SQLxxx 関数および SQLxxxW 関数の留意事項

この節では，この章で記載している次の関数の留意事項について説明します。

- ODBC3.0 で使用するSQL*xxx* 関数
- ODBC3.5 で使用するSQL*xxx*W 関数

■Windows 版および Linux 版の HADB クライアント共通の留意事項

ODBC3.0 のSQL*xxx* 関数でSQLCHAR 型を定義している引数については，ODBC3.5 のSQL*xxx*W 関数ではSQLWCHAR 型を定義しています。

また，長さを指定するパラメタについては，次のように指定します。

- ODBC3.0 のSQL*xxx* 関数のパラメタ：バイト長
- ODBC3.5 のSQL*xxx*W 関数
  - パラメタがSQLPOINTER 型の場合：バイト長
  - パラメタがSQLWCHAR 型の場合：文字数

■Linux 版の HADB クライアントの場合の留意事項

ODBC3.5 のSQL*xxx*W 関数で，長さを指定する引数に文字数ではなくバイト長を指定する場合があります。その際，奇数バイトを指定したときには，内部で有効な文字数（$\downarrow$*指定したバイト長* ÷ 2$\downarrow$ × 2）に変換して処理されます。

なお，ODBC3.0 と ODBC3.5 で使用する関数名が異なる場合は，ODBC3.0 で使用するSQL*xxx* 関数名，ODBC3.5 で使用するSQL*xxx*W 関数名の順に記載しています。

16.3　データソースとの接続

ここでは，データソースとの接続時に使用する ODBC 関数について説明します。

16.3.1　SQLAllocHandle

(1)　機能

環境，接続，ステートメント，およびディスクリプタのハンドルを割り当てます。

(2)　形式

```
SQLRETURN SQLAllocHandle
(
  SQLSMALLINT     HandleType,         /* In */
  SQLHANDLE       InputHandle,        /* In */
  SQLHANDLE     * OutputHandlePtr     /* In/Out */
)
```

(3)　引数

HandleType：

次のどれかのハンドルの種類を指定します。

- SQL_HANDLE_ENV：環境ハンドル
- SQL_HANDLE_DBC：接続ハンドル
- SQL_HANDLE_STMT：ステートメントハンドル
- SQL_HANDLE_DESC：ディスクリプタハンドル

InputHandle：

HandleType パラメタの値に応じて次の入力ハンドルを指定します。

HandleType	InputHandle に指定する値
SQL_HANDLE_ENV	SQL_NULL_HANDLE
SQL_HANDLE_DBC	環境ハンドル
SQL_HANDLE_STMT	接続ハンドル
SQL_HANDLE_DESC	接続ハンドル

OutputHandlePtr：

新しく割り当てられたハンドルを返すバッファへのポインタを指定します。

戻り値がSQL_SUCCESS の場合は，ハンドルの値が設定されます。SQL_ERROR の場合は，次の値が設定されます。

HandleType	OutputHandlePtr に設定される値
SQL_HANDLE_ENV	NULL
SQL_HANDLE_DBC	SQL_NULL_HDBC
SQL_HANDLE_STMT	SQL_NULL_HSTMT
SQL_HANDLE_DESC	SQL_NULL_HDESC

(4) 戻り値

SQL_SUCCESS, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08003	接続が存在しない	接続が完了していません。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		○
HY009	NULL ポインタの不正使用		○
HY010	関数シーケンスエラー	接続ハンドル取得前に ODBC バージョンの設定が行われていません。	○
HY013	メモリ管理エラー	HandleType がSQL_HANDLE_DBC, SQL_HANDLE_STMT, またはSQL_HANDLE_DESC であるが，メモリオブジェクトにアクセスできません（メモリ不足)。	×
HY014	ハンドル数の上限を超過	－	×
HY092	無効な属性識別子，または無効なオプション識別子	HandleType に指定できない値が指定されました。	○
HYC00	オプション機能は実装されていない	－	×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.3.2 SQLConnect，SQLConnectW

(1) 機能

HADB ODBC ドライバとデータソース（HADB サーバ）の接続を確立します。

なお，SQLConnect またはSQLConnectW を実行するにはCONNECT 権限が必要です。

(2) 形式

- SQLConnect の場合

```
SQLRETURN SQLConnect
(
  SQLHDBC           ConnectionHandle,  /* In */
  SQLCHAR         * ServerName,        /* In */
  SQLSMALLINT       NameLength1,       /* In */
  SQLCHAR         * UserName,          /* In */
  SQLSMALLINT       NameLength2,       /* In */
  SQLCHAR         * Authentication,    /* In */
  SQLSMALLINT       NameLength3        /* In */
)
```

- SQLConnectW の場合

```
SQLRETURN SQLConnectW
(
  SQLHDBC           ConnectionHandle,  /* In */
  SQLWCHAR        * ServerName,        /* In */
  SQLSMALLINT       NameLength1,       /* In */
  SQLWCHAR        * UserName,          /* In */
  SQLSMALLINT       NameLength2,       /* In */
  SQLWCHAR        * Authentication,    /* In */
  SQLSMALLINT       NameLength3        /* In */
)
```

(3) 引数

ConnectionHandle：

コネクションハンドルを指定します。

ServerName：

データソース名を指定します。

NameLength1：

データソース名の長さ※を指定します。

データソース名が NULL 終端文字で終わる場合は，SQL_NTS を指定する必要があります。

また，0 や負の値を指定した場合は，エラーとなります。

UserName：

HADB サーバに接続するユーザ ID（認可識別子）を指定します。

NameLength2：

ユーザ ID の長さ※を指定します。

ユーザ ID が NULL 終端文字で終わる場合は，SQL_NTS を指定する必要があります。

また，0 や負の値を指定した場合は，エラーとなります。

Authentication：

HADB サーバに接続する認可識別子のパスワードを指定します。NULL を指定した場合は，エラーとなります。

ODBC 実装規約によって，パスワードには，[,]，{,}，(,)，,，;，?，*，=，!，@の 13 種類の文字は含めないことを推奨します。HADB のパスワードについては，マニュアル『HADB　システム構築・運用ガイド』の『パスワードの指定規則』を参照してください。

NameLength3：

パスワードの長さ※を指定します。

パスワードが NULL 終端文字で終わる場合は，SQL_NTS を指定する必要があります。

負の値または 256 以上の値を指定した場合は，エラーとなります。

また，Authentication に 256 バイト以上の文字列を指定し，かつ，この引数にSQL_NTS を指定した場合は，エラーとなります。

注※

長さの単位は，SQLConnect の場合はバイト長，SQLConnectW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01S02	オプション値の変更		×
08001	クライアントが接続を確立できない		×
08002	接続名が使用中である		○
08004	サーバが接続を拒否した		×
08S01	通信リンク失敗		×

SQLSTATE	説明	備考	返却
28000	無効な認証指定		○
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C052	バージョン不一致エラー	ODBC ドライバと HADB クライアントのバージョンが異なります。	○
5D001	HADB 固有のエラー	HADB でエラーが発生しましたが，固有の SQLSTATE またはエラーメッセージが取得できません。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		○
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM002	データソースが見つからない かつ デフォルトのドライバが指定されていない		×
IM003	指定されたドライバがロードできない		×
IM004	ドライバがSQL_HANDLE_ENV に対する SQLAllocHandle に失敗した		×
IM005	ドライバがSQL_HANDLE_DBC に対する SQLAllocHandle に失敗した		×
IM006	ドライバがSQLSetConnectAttr または SQLSetConnectAttrW に失敗した		×
IM009	トランスレータ DLL をロードできない		×
IM010	データソース名が長過ぎる		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.3.3 SQLDriverConnect，SQLDriverConnectW

(1) 機能

次の接続属性のどれかを使用して，データソース（HADB サーバ）との接続を確立します。

- データソース名称，1 つ以上の認可識別子，1 つ以上のパスワード，およびデータソースとの接続に必要なほかの情報を含む接続文字列を使用して接続を確立します。
- 部分的な接続文字列および追加情報を使用しないで接続を確立します。この場合，ドライバマネージャと HADB ODBC ドライバは，AP に接続情報を要求します。
- システム定義で定義されていないデータソースとの接続を確立します。アプリケーションが部分的な接続文字列を提供した場合，HADB ODBC ドライバはユーザに接続情報を要求します。
- .dsn ファイルの情報から作成された接続文字列を使用して，データソースとの接続を確立します。

接続が確立すると，SQLDriverConnect またはSQLDriverConnectW は完全な接続文字列を返します。

なお，SQLDriverConnect またはSQLDriverConnectW を実行するにはCONNECT 権限が必要です。

(2) 形式

- SQLDriverConnect の場合

```
SQLRETURN SQLDriverConnect
(
  SQLHDBC         ConnectionHandle,        /* In */
  SQLHWND         WindowHandle,            /* In */
  SQLCHAR       * InConnectionString,      /* In */
  SQLSMALLINT     StringLength1,           /* In */
  SQLCHAR       * OutConnectionString,     /* Out */
  SQLSMALLINT     BufferLength,            /* In */
  SQLSMALLINT   * StringLength2Ptr,        /* Out */
  SQLUSMALLINT    DriverCompletion         /* In */
)
```

- SQLDriverConnectW の場合

```
SQLRETURN SQLDriverConnectW
(
  SQLHDBC         ConnectionHandle,        /* In */
  SQLHWND         WindowHandle,            /* In */
  SQLWCHAR      * InConnectionString,      /* In */
  SQLSMALLINT     StringLength1,           /* In */
  SQLWCHAR      * OutConnectionString,     /* Out */
  SQLSMALLINT     BufferLength,            /* In */
  SQLSMALLINT   * StringLength2Ptr,        /* Out */
  SQLUSMALLINT    DriverCompletion         /* In */
)
```

(3) 引数

ConnectionHandle：

接続ハンドルを指定します。

WindowHandle：

親ウィンドウのハンドルを指定します。

ウィンドウを適用できない，またはダイアログボックスを表示しない場合は，NULL ポインタを指定してください。

InConnectionString：

接続文字列を指定します。

接続文字列に指定できる接続属性は次のとおりです。

接続属性	説明
DSN	データソース名称
DRIVER	ODBC ドライバ名称："Hitachi Advanced Data Binder ODBC Driver"
UID	認可識別子
PWD	パスワード
CLTPATH	クライアント定義ファイルの絶対パス

StringLength1：

InConnectionString に指定した接続文字列の長さ※を指定します。

InConnectionString に指定した接続文字列が NULL 終端文字で終わる場合は，SQL_NTS を指定する必要があります。

また，0 や負の値を指定した場合は，エラーとなります。

OutConnectionString：

完全な接続文字列を格納するバッファへのポインタを指定します。

HADB サーバへの接続に成功した場合の接続文字列を完全な接続文字列として返します。

BufferLength：

OutConnectionString を格納するバッファの長さ※を指定します。

この長さに NULL 終端文字は含まれます。SQL_NTS は指定できません。

StringLength2Ptr：

完全な接続文字列の有効な長さ※を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

重要

ここに格納された接続文字列の長さ※が，BufferLength から NULL 終端文字分を引いた長さ※より大きい場合，OutConnectionString に格納される文字列はBufferLength から NULL 終端文字分を引いた長さ※に切り捨てられ，末尾に NULL 終端文字が付加されます。

DriverCompletion：

ドライバマネージャまたは HADB ODBC ドライバがさらに接続情報を必要とするかどうかをフラグで指定します。次のフラグが指定できます。

■Windows 版の HADB クライアントの場合

- SQL_DRIVER_PROMPT
- SQL_DRIVER_COMPLETE
- SQL_DRIVER_COMPLETE_REQUIRED
- SQL_DRIVER_NOPROMPT

■Linux 版の HADB クライアントの場合

- SQL_DRIVER_NOPROMPT

  Linux 版の HADB ODBC ドライバでは，ダイアログボックスの表示をサポートしていません。そのため，SQL_DRIVER_NOPROMPT 以外を指定した場合，SQL_ERROR が返されます。

注※

長さの単位は，SQLDriverConnect の場合はバイト長，SQLDriverConnectW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_NEED_DATA，SQL_NO_DATA，SQL_ERROR，または SQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	*OutConnectionString バッファの大きさが不足しているため，完全な接続文字列を格納できませんでした（情報が切り捨てられました）。切り捨てられる前の完全な接続文字列の長さが，*StringLength2Ptr バッファに格納されます。このとき，SQL_SUCCESS_WITH_INFO を返します。	○

SQLSTATE	説明	備考	返却
01S00	無効な接続文字列属性	接続文字列（InConnectionString）に無効な属性キーワードが含まれています。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S02	オプション値の変更	HADB ODBC ドライバは SQLSetConnectAttr または SQLSetConnectAttrW のValuePtr に指定された値をサポートしていないため，類似の値で置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	×
01S08	ファイル DSN の保存エラー	*InConnectionString の接続文字列に FILEDSN キーワードが含まれていますが，.dsn ファイルが保存されませんでした。このとき，SQL_SUCCESS_WITH_INFO を返します。	×
01S09	無効なキーワード	－	×
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
08001	クライアントが接続を確立できない	• HADB ODBC ドライバは，データソースに接続できません。 • DriverCompletion に SQL_DRIVER_NOPROMPT が指定されていますが，接続に必要な属性文字列が InConnectionString に指定されていません。	○
08002	接続名が使用中である	－	×
08004	サーバが接続を拒否した	データソースは実装時に定義された理由で接続の確立を拒否しました。	×
08S01	通信リンク失敗	－	×
28000	無効な認証指定	接続文字列で指定された認可識別子またはパスワードが，データソース定義の制限に違反しています。	○
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C052	バージョン不一致エラー	ODBC ドライバと HADB クライアントのバージョンが異なります。	○
5D001	HADB 固有のエラー	HADB でエラーが発生しましたが，固有の SQLSTATE またはエラーメッセージが取得できません。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		×

SQLSTATE	説明	備考	返却
HY013	メモリ管理エラー	メモリオブジェクトにアクセスできないため，関数の呼び出しを処理できません。	×
HY090	無効な文字列長または無効なバッファ長	• InConnectionString に指定された接続属性に対する属性値が無効な長さです。 • BufferLength に無効な値（SQL_NTS）が指定されました。	○
HY092	無効な属性識別子，または無効なオプション識別子	－	×
HY110	DriverCompletion が無効なオプション識別子		×
HYC00	オプション機能は実装されていない	HADB ODBC ドライバはアプリケーションが要求した ODBC の動作をサポートしていません。	×
HYT00	タイムアウト終了	データソースの接続が完了する前にログインタイムアウト時間が経過しました。ログインタイムアウト時間は，SQLSetConnectAttr または SQLSetConnectAttrW の SQL_ATTR_LOGIN_TIMEOUT で設定できます。	×
HYT01	接続タイムアウト終了	データソースが要求に応答する前に接続タイムアウト時間が経過しました。接続タイムアウト時間は，SQLSetConnectAttr またはSQLSetConnectAttrW の SQL_ATTR_CONNECTION_TIMEOUT で設定できます。	×
IM001	ドライバはこの関数をサポートしていない	－	×
IM002	データソースが見つからない かつ デフォルトのドライバが指定されていない		×
IM003	指定されたドライバがロードできない		×
IM004	ドライバがSQL_HANDLE_ENV に対する SQLAllocHandle に失敗した		×
IM005	ドライバがSQL_HANDLE_DBC に対する SQLAllocHandle に失敗した		×
IM006	ドライバがSQLSetConnectAttr または SQLSetConnectAttrW に失敗した		×
IM007	データソースまたはドライバが指定されていない かつ ダイアログが禁止された	接続文字列にデータソース名称またはドライバ名称が指定されていませんが，DriverCompletion にSQL_DRIVER_NOPROMPT が指定されています。	×

SQLSTATE	説明	備考	返却
IM008	ダイアログの失敗	ドライバはログインダイアログボックスを表示しようとしましたが失敗しました。 WindowHandle に NULL ポインタが指定されていますが，DriverCompletion に SQL_DRIVER_NOPROMPT が指定されていません。	×
IM009	トランスレータ DLL をロードできない	－	×
IM010	データソース名が長過ぎる		×
IM011	ドライバ名が長過ぎる		×
IM012	DRIVER キーワードの構文エラー		×
IM014	ファイル DSN の名前が無効である		×
IM015	ファイルデータソースが壊れている		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

- 接続文字列に指定する接続属性を次の表に示します。

表 16-2 接続文字列に指定する接続属性

項番	接続属性	接続属性値	指定要否	
			DSN 接続	DRIVER 接続
1	DSN	データソース名称を指定してください。	○	－
2	DRIVER	ODBC ドライバ名称を指定してください。ODBC ドライバの名称は"Hitachi Advanced Data Binder ODBC Driver"です。	－	○
3	UID	認可識別子を指定してください。	○	○
4	PWD	パスワードを指定します。 255 バイト以下の文字列を指定してください。	○	○
5	CLTPATH	クライアント定義ファイルを絶対パスで指定してください。クライアント定義ファイルを絶対パスで指定する際，255 バイト以下の文	－	△

項番	接続属性	接続属性値	指定要否	
			DSN 接続	DRIVER 接続
		字列を指定してください。この指定を省略した場合は，環境定義 ADBCLTDIR に指定したフォルダ下にあるファイルを使用します。		

（凡例）

○：必ず指定する接続属性です。

△：任意で指定する接続属性です。

－：指定する必要がない接続属性です。

- 接続文字列の指定例を次に示します。
  - "DSN=*XXXXX*;UID=*YYYYY*;PWD=*ZZZZZ*"
  - "DRIVER=Hitachi Advanced Data Binder ODBC Driver;CLTPATH=*XXXXX*;UID=*YYYYY*;PWD=*ZZZZZ*"
- DSN 接続属性およびDRIVER 接続属性を接続文字列中に両方指定した場合，先に指定した接続属性が有効になります。
- 同じ接続属性が接続文字列中に重複して指定されている場合，あとに指定した接続属性の接続属性値が有効になります。
- 接続文字列に指定する接続属性は順不同に指定できますが，最初の要求時に"DSN=*XXXXX*"または"DRIVER=Hitachi Advanced Data Binder ODBC Driver"の指定がない場合，ODBC ドライバマネージャでエラーになります。ただし，FILEDSN 指定の場合は該当しません。
- 接続属性は大文字，小文字を区別しません。
- 接続属性値は大文字，小文字を区別します。
- ;（セミコロン）は，区切り文字として扱います。このため，接続属性値のパスワード文字列中に;（セミコロン）を含めることはできません。また，ODBC 実装規約によって，パスワードには，[,]，{,}，(,)，,，?，*，=，!，@の 12 種類の文字は含めないことを推奨します。HADB のパスワードについては，マニュアル『HADB　システム構築・運用ガイド』の『パスワードの指定規則』を参照してください。

16.3.4　SQLBrowseConnect，SQLBrowseConnectW

(1)　機能

データソース（HADB サーバ）との接続に必要な属性と属性値を 1 つずつ参照する方法をサポートします。SQLBrowseConnect またはSQLBrowseConnectW を呼び出すたびに，連続するレベルの属性と属性値を返します。すべてのレベルの属性を指定すると，データソースへの接続が完了し，完全な接続文字列を返します。

なお，SQLBrowseConnect またはSQLBrowseConnectW を実行するにはCONNECT 権限が必要です。

(2) 形式

- SQLBrowseConnect の場合

```
SQLRETURN SQLBrowseConnect
(
  SQLHDBC         ConnectionHandle,        /* In */
  SQLCHAR       * InConnectionString,      /* In */
  SQLSMALLINT     StringLength1,           /* In */
  SQLCHAR       * OutConnectionString,     /* Out */
  SQLSMALLINT     BufferLength,            /* In */
  SQLSMALLINT   * StringLength2Ptr         /* Out */
)
```

- SQLBrowseConnectW の場合

```
SQLRETURN SQLBrowseConnectW
(
  SQLHDBC         ConnectionHandle,        /* In */
  SQLWCHAR      * InConnectionString,      /* In */
  SQLSMALLINT     StringLength1,           /* In */
  SQLWCHAR      * OutConnectionString,     /* Out */
  SQLSMALLINT     BufferLength,            /* In */
  SQLSMALLINT   * StringLength2Ptr         /* Out */
)
```

(3) 引数

ConnectionHandle：

接続ハンドルを指定します。

InConnectionString：

ブラウズ要求接続文字列を指定します。

ブラウズ要求接続文字列に指定できる接続属性は次のとおりです。

接続属性	説明
DSN	データソース名称
DRIVER	ODBC ドライバ名称："Hitachi Advanced Data Binder ODBC Driver"
UID	認可識別子
PWD	パスワード
CLTPATH	クライアント定義ファイルの絶対パス

StringLength1：

InConnectionString に指定したブラウズ要求接続文字列の長さ※を指定します。

InConnectionString に指定したブラウズ要求接続文字列が NULL 終端文字で終わる場合は，SQL_NTS を指定する必要があります。

また，0 や負の値を指定した場合は，エラーとなります。

OutConnectionString：

ブラウズ結果接続文字列を格納するバッファへのポインタを指定します。

HADB サーバへの接続に成功した場合の接続文字列を完全な接続文字列として返します。

SQL_NEED_DATA を返す場合，HADB サーバとの接続時に不足した接続属性を返します。

BufferLength：

OutConnectionString を格納するバッファの長さ※を指定します。

この長さに NULL 終端文字は含まれます。SQL_NTS は指定できません。

StringLength2Ptr：

ブラウズ結果接続文字列の有効な長さ※を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

OutConnectionString に返される文字列の有効な長さ※を返します。

> **重要**
>
> ここに格納された接続文字列の長さ※が，BufferLength から NULL 終端文字分を引いた長さ※より大きい場合，OutConnectionString に格納される文字列はBufferLength から NULL 終端文字分を引いた長さ※に切り捨てられ，末尾に NULL 終端文字が付加されます。

注※

長さの単位は，SQLBrowseConnect の場合はバイト長，SQLBrowseConnectW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_NEED_DATA，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	*OutConnectionString バッファの大きさが不足しているため，ブラウズ結果接続文字列を格納できませんでした（情報が切り捨てられました）。切り捨てられる前のブラウズ結果接続文字列の長さが，*StringLength2Ptr バッファに格納されます。このとき，SQL_NEED_DATA を返します。	○
01S00	無効な接続文字列属性	ブラウズ要求接続文字列（InConnectionString）に無効な属性キー	○

SQLSTATE	説明	備考	返却
		ワードが含まれています。このとき，SQL_NEED_DATA を返します。 ブラウズ要求接続文字列（InConnectionString）に指定された属性キーワードは現在の接続レベルに適用できません。このとき，SQL_NEED_DATA を返します。	
01S02	オプション値の変更	HADB ODBC ドライバは SQLSetConnectAttr または SQLSetConnectAttrW のValuePtr に指定された値をサポートしていないため，類似の値で置き換えられました。このとき，SQL_SUCCESS_WITH_INFO を返します。	×
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
08001	クライアントが接続を確立できない	HADB ODBC ドライバは，データソースに接続できません。	×
08002	接続名が使用中である	－	×
08S01	通信リンク失敗		×
28000	無効な認証指定	ブラウズ要求接続文字列で指定された認可識別子またはパスワードが，データソースの制限に違反しています。	○
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C052	バージョン不一致エラー	ODBC ドライバと HADB クライアントのバージョンが異なります。	○
5D001	HADB 固有のエラー	HADB でエラーが発生しましたが，固有の SQLSTATE またはエラーメッセージが取得できません。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		×
HY013	メモリ管理エラー	メモリオブジェクトにアクセスできないため，関数の呼び出しを処理できません。	×
HY090	無効な文字列長または無効なバッファ長	• InConnectionString に指定された接続属性に対する属性値が無効な長さです。 • BufferLength に無効な値（SQL_NTS）が指定されました。	○
HYT00	タイムアウト終了	データソースの接続が完了する前にログインタイムアウト時間が経過しました。ログインタイムアウト時間は，	×

SQLSTATE	説明	備考	返却
		SQLSetConnectAttr または SQLSetConnectAttrW の SQL_ATTR_LOGIN_TIMEOUT で設定できます。	
HYT01	接続タイムアウト終了	データソースが要求に応答する前に接続タイムアウト時間が経過しました。接続タイムアウト時間は，SQLSetConnectAttr またはSQLSetConnectAttrW の SQL_ATTR_CONNECTION_TIMEOUT で設定できます。	×
IM001	ドライバはこの関数をサポートしていない	－	×
IM002	データソースが見つからない かつ デフォルトのドライバが指定されていない		×
IM003	指定されたドライバがロードできない		×
IM004	ドライバがSQL_HANDLE_ENV に対する SQLAllocHandle に失敗した		×
IM005	ドライバがSQL_HANDLE_DBC に対する SQLAllocHandle に失敗した		×
IM006	ドライバがSQLSetConnectAttr または SQLSetConnectAttrW に失敗した		×
IM009	トランスレータ DLL をロードできない		×
IM010	データソース名が長過ぎる		×
IM011	ドライバ名が長過ぎる		×
IM012	DRIVER キーワードの構文エラー		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

- ブラウズ要求接続文字列に指定する接続属性を次の表に示します。

表 16-3 ブラウズ要求接続文字列に指定する接続属性

項番	接続属性	接続属性値	指定要否	
			DSN 接続	DRIVER 接続
1	DSN	データソース名称を指定してください。	○	－

項番	接続属性	接続属性値	指定要否	
			DSN 接続	DRIVER 接続
2	DRIVER	ODBC ドライバ名称を指定してください。ODBC ドライバの名称は"Hitachi Advanced Data Binder ODBC Driver"です。	－	○
3	UID	認可識別子を指定してください。	○	○
4	PWD	パスワードを指定します。 255 バイト以下の文字列を指定してください。	○	○
5	CLTPATH	クライアント定義ファイルを絶対パスで指定してください。クライアント定義ファイルを絶対パスで指定する際，255 バイト以下の文字列を指定してください。この指定を省略した場合は，環境定義 ADBCLTDIR に指定したフォルダ下にあるファイルを使用します。	－	△

（凡例）

○：必ず指定する接続属性です。

△：任意で指定する接続属性です。

－：指定する必要がない接続属性です。

- ブラウズ要求接続文字列の指定例を次に示します。
  - ”DSN=*XXXXX*;UID=*YYYYY*;PWD=*ZZZZZ*”
  - ”DRIVER=Hitachi Advanced Data Binder ODBC Driver;CLTPATH=*XXXXX*;UID=*YYYYY*;PWD=*ZZZZZ*”
- DSN 接続属性およびDRIVER 接続属性をブラウズ要求接続文字列中に両方指定した場合，あとに指定した接続属性が有効になります。
- 同じ接続属性がブラウズ要求接続文字列中に重複して指定されている場合，先に指定した接続属性の接続属性値が有効になります。
- ブラウズ要求接続文字列に指定する接続属性は順不同に指定できますが，最初の要求時に"DSN=*XXXXX*"または"DRIVER=Hitachi Advanced Data Binder ODBC Driver"の指定がない場合，ODBC ドライバマネージャでエラーになります。
- 接続属性は大文字，小文字を区別しません。
- 接続属性値は大文字，小文字を区別します。
- ;（セミコロン）は，区切り文字として扱います。このため，接続属性値のパスワード文字列中に;（セミコロン）を含めることはできません。また，ODBC 実装規約によって，パスワードには，[,]，{,}，(,)，,，?，*，=，!，@の 12 種類の文字は含めないことを推奨します。HADB のパスワードについては，マニュアル『HADB　システム構築・運用ガイド』の『パスワードの指定規則』を参照してください。

- この関数の引数に指定したパラメタの値不正によってSQL_NEED_DATA が返却された場合にこの関数を再実行し，パスワードの文字列などの不正によってSQL_ERROR が返却されたときは，そのままSQLFreeHandle を実行しても ODBC ドライバマネージャからSQL_ERROR（SQLSTATE：HY010）が返却されて，ハンドルが解放されないおそれがあります。この場合，正しいパラメタを設定して関数を再実行したあとに，SQLFreeHandle を実行してください。または，アプリケーションを強制終了してください。

16.4　ドライバおよびデータソースの情報取得

ここでは，ドライバおよびデータソースの情報取得時に使用する ODBC 関数について説明します。

16.4.1　SQLDataSources，SQLDataSourcesW

(1)　機能

次の情報を返します。

- データソース名
- データソースに関連づけられたドライバの記述

この関数は，ドライバマネージャで実行できます。HADB ODBC ドライバでは，常にSQL_SUCCESS を返す関数としてSQLDataSources またはSQLDataSourcesW を提供しています。

(2)　形式

- SQLDataSources の場合

```
SQLRETURN SQLDataSources
(
  SQLHENV           EnvironmentHandle,  /* In */
  SQLUSMALLINT      Direction,          /* In */
  SQLCHAR         * ServerName,         /* Out */
  SQLSMALLINT       BufferLength1,      /* In */
  SQLSMALLINT     * NameLength1Ptr,     /* Out */
  SQLCHAR         * Description,        /* Out */
  SQLSMALLINT       BufferLength2,      /* In */
  SQLSMALLINT     * NameLength2Ptr      /* Out */
)
```

- SQLDataSourcesW の場合

```
SQLRETURN SQLDataSourcesW
(
  SQLHENV           EnvironmentHandle,  /* In */
  SQLUSMALLINT      Direction,          /* In */
  SQLWCHAR        * ServerName,         /* Out */
  SQLSMALLINT       BufferLength1,      /* In */
  SQLSMALLINT     * NameLength1Ptr,     /* Out */
  SQLWCHAR        * Description,        /* Out */
  SQLSMALLINT       BufferLength2,      /* In */
  SQLSMALLINT     * NameLength2Ptr      /* Out */
)
```

(3) 引数

EnvironmentHandle：

環境ハンドルを指定します。

Direction：

ドライバマネージャがデータソースの情報を取得するときの読み込み方式を指定します。次の値を指定できます。

- SQL_FETCH_FIRST：一覧の先頭からデータソース名（以降 DSN）をフェッチします（ユーザ DSN，システム DSN の両方）。
- SQL_FETCH_FIRST_USER：最初のユーザ DSN をフェッチします。
- SQL_FETCH_FIRST_SYSTEM：最初のシステム DSN をフェッチします。
- SQL_FETCH_NEXT：一覧の中の次の DSN をフェッチします。対象となるのはFIRST でフェッチ対象とした DSN と同じ種類のものです（ユーザ DSN とシステム DSN の両方，ユーザ DSN だけ，またはシステム DSN だけ）。

ServerName：

データソース名を返すバッファへのポインタを指定します。

BufferLength1：

*ServerName バッファの長さ※を指定します。指定が有効になる最大値はSQL_MAX_DSN_LENGTH に NULL 終端文字分を足した値です。それより大きい値を指定しても前述の最大値が指定されます。この長さに NULL 終端文字は含まれます。

NameLength1Ptr：

*ServerName の有効な長さ※の合計を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

> **重要**
>
> ここに格納されたデータソース名の長さ※が，BufferLength1 から NULL 終端文字分を引いた長さ※より大きい場合，*ServerName に格納される文字列はBufferLength1 から NULL 終端文字分を引いた長さ※に切り捨てられ，末尾に NULL 終端文字が付加されます。

Description：

データソースに関連づけられたドライバの記述（HADB サーバなど）を返すバッファへのポインタを指定します。

BufferLength2：

* Description バッファの長さ※を指定します。

この長さに NULL 終端文字は含まれます。

NameLength2Ptr：

*Description に返す有効な長さ※の合計を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

> **重要**
>
> ここに格納されたドライバの記述の長さ※が，BufferLength2 から NULL 終端文字分を引いた長さ※より大きい場合，* Description に格納される文字列はBufferLength2 から NULL 終端文字分を引いた長さ※に切り捨てられ，末尾に NULL 終端文字が付加されます。

注※

長さの単位は，SQLDataSources の場合はバイト長，SQLDataSourcesW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY103	無効な取得コード		×

(凡例)

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.4.2 SQLDrivers，SQLDriversW

(1) 機能

ドライバ記述とドライバ属性キーワードの一覧を返します。

この関数は，ドライバマネージャで実行できます。HADB ODBC ドライバでは，常にSQL_SUCCESS を返す関数としてSQLDrivers またはSQLDriversW を提供しています。

(2) 形式

- SQLDrivers の場合

```
SQLRETURN SQLDrivers
(
  SQLHENV            EnvironmentHandle,     /* In */
  SQLUSMALLINT       Direction,             /* In */
  SQLCHAR          * DriverDescription,     /* Out */
  SQLSMALLINT        BufferLength1,         /* In */
  SQLSMALLINT      * DescriptionLengthPtr,  /* Out */
  SQLCHAR          * DriverAttributes,      /* Out */
  SQLSMALLINT        BufferLength2,         /* In */
  SQLSMALLINT      * AttributesLengthPtr    /* Out */
)
```

- SQLDriversW の場合

```
SQLRETURN SQLDriversW
(
  SQLHENV            EnvironmentHandle,     /* In */
  SQLUSMALLINT       Direction,             /* In */
  SQLWCHAR         * DriverDescription,     /* Out */
  SQLSMALLINT        BufferLength1,         /* In */
  SQLSMALLINT      * DescriptionLengthPtr,  /* Out */
  SQLWCHAR         * DriverAttributes,      /* Out */
  SQLSMALLINT        BufferLength2,         /* In */
  SQLSMALLINT      * AttributesLengthPtr    /* Out */
)
```

(3) 引数

EnvironmentHandle：

環境ハンドルを指定します。

Direction：

ドライバマネージャがデータソースの情報を取得するときの読み込み方式を指定します。次の値を指定できます。

- SQL_FETCH_FIRST：一覧の先頭からドライバ記述をフェッチします（ユーザ DSN，システム DSN の両方）。
- SQL_FETCH_NEXT：一覧の中の次のドライバ記述をフェッチします。

DriverDescription：

ドライバ記述を返すバッファのポインタを指定します。

BufferLength1：

ドライバ記述を返すバッファの長さ※を指定します。

DescriptionLengthPtr：

*DriverDescription の有効な長さ※の合計を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

DriverAttributes：

ドライバの属性値を格納したバッファのポインタを指定します。

BufferLength2：

ドライバの属性値を返すバッファの長さ※を指定します。

AttributesLengthPtr：

*AttributesLengthPtr の有効な長さ※の合計を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

注※

長さの単位は，SQLDrivers の場合はバイト長，SQLDriversW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY103	無効な取得コード		×

(凡例)

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.4.3　SQLGetInfo，SQLGetInfoW

(1)　機能

接続に関連づけられているドライバとデータソースの一般的な情報を返します。

(2)　形式

- SQLGetInfo の場合

```
SQLRETURN SQLGetInfo
(
  SQLHDBC           ConnectionHandle,   /* In */
  SQLUSMALLINT      InfoType,           /* In */
  SQLPOINTER        InfoValuePtr,       /* In/Out */
  SQLSMALLINT       BufferLength,       /* In */
  SQLSMALLINT     * StringLengthPtr     /* Out */
)
```

- SQLGetInfoW の場合

```
SQLRETURN SQLGetInfoW
(
  SQLHDBC           ConnectionHandle,   /* In */
  SQLUSMALLINT      InfoType,           /* In */
  SQLPOINTER        InfoValuePtr,       /* In/Out */
  SQLSMALLINT       BufferLength,       /* In */
  SQLSMALLINT     * StringLengthPtr     /* Out */
)
```

(3)　引数

ConnectionHandle：

接続ハンドルを指定します。

InfoType：

情報型を指定します。

指定できる情報型については，「16.13　SQLGetInfo および SQLGetInfoW の InfoType に指定できる情報型」を参照してください。

InfoValuePtr：

情報を返すバッファのポインタを指定します。要求されたInfoType に応じた情報が返されます。指定できる情報型をInfoType に指定した場合でも，その情報型が未サポートのときは，空の文字列や，0 が返されることがあります。

BufferLength：

*InfoValuePtr バッファの長さ（単位：バイト長）を指定します。この長さに NULL 終端文字は含まれます。SQL_NTS は指定できません。

*InfoValuePtr の値が文字列以外，またはInfoValuePtr が NULL ポインタの場合，この引数の指定は無視されます。

InfoValuPtr の値が文字列以外の場合，HADB ODBC ドライバはInfoType に基づき，*InfoValuePtr のサイズをSQLUSMALLINT またはSQLUINTEGER と見なします。バッファサイズが不足している場合の動作は保証しません。

StringLengthPtr：

*InfoValuePtr に返す，有効な長さ（単位：バイト長）の合計を返すバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

重要

文字データの場合，この長さ（単位：バイト長）がBufferLength から NULL 終端文字分を引いた長さより大きい場合，*InfoValuePtr に格納される文字列はBufferLength から NULL 終端文字分を引いた長さ（単位：バイト長）に切り捨てられ，末尾に NULL 終端文字が付加されます。そのほかのデータ型の場合，BufferLength の値は無視され，ドライバはInfoType に基づき，*InfoValuePtr のサイズをSQLUSMALLINT またはSQLUINTEGER と見なします。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	*InfoValuePtr バッファの大きさが不足しているため，要求されたすべての情報を格納できませんでした（情報が切り捨てられました）。切り捨てられる前の要求情報の長さが，*StringLengthPtr に格納されます。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
08003	接続が存在しない	－	×
08S01	通信リンク失敗		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×

SQLSTATE	説明	備考	返却
HY009	NULL ポインタの不正使用		×
HY013	メモリ管理エラー		×
HY024	無効な属性値		×
HY090	無効な文字列長または無効なバッファ長	BufferLength に無効な値（SQL_NTS）が指定されました。	○
HY096	情報型が範囲外である	InfoType に指定された値はサポートしていません。	○
HY117	接続がサスペンド中	－	×
HYC00	オプション機能は実装されていない		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

InfoType にSQL_DRIVER_ODBC_VER 情報型以外を指定する場合，接続が確立している必要があります。

16.4.4 SQLGetFunctions

(1) 機能

指定した ODBC 関数がサポートされているかどうかの情報，または各 ODBC 関数のサポート情報の一覧を返します。

(2) 形式

```
SQLRETURN SQLGetFunctions
(
  SQLHDBC           ConnectionHandle,    /* In */
  SQLUSMALLINT      FunctionId,          /* In */
  SQLUSMALLINT    * SupportedPtr         /* Out */
)
```

(3) 引数

ConnectionHandle：

接続ハンドルを指定します。

FunctionId：

全関数のサポート情報リストを取得したい場合は，SQL_API_ODBC3_ALL_FUNCTIONS を指定します。

関数単体のサポート情報を取得したい場合は，調査対象の ODBC 関数の#define 値を指定します。

SupportedPtr：

取得したい関数のサポート情報を返すバッファのポインタを指定します。

FunctionId に ODBC 関数の#define 値を指定した場合，SQL_TRUE（サポートされている）または SQL_FALSE（サポートされていない）が返されます。

SupportedPtr にはNULL も指定できます。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY095	関数型が範囲外である		○
HYT01	接続タイムアウト終了		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.4.5　SQLGetTypeInfo，SQLGetTypeInfoW

(1)　機能

HADB ODBC ドライバがサポートしているデータ型に関する情報を SQL 文の結果セットの形式で返します。

(2)　形式

- SQLGetTypeInfo の場合

```
SQLRETURN SQLGetTypeInfo
(
  SQLHSTMT       StatementHandle,      /* In */
  SQLSMALLINT    DataType              /* In */
)
```

- SQLGetTypeInfoW の場合

```
SQLRETURN SQLGetTypeInfoW
(
  SQLHSTMT       StatementHandle,      /* In */
  SQLSMALLINT    DataType              /* In */
)
```

(3)　引数

StatementHandle：

ステートメントハンドルを指定します。

DataType：

SQL データ型識別子を指定します。取得したいデータによって次の値を指定します。

- 個別のデータ型に関する情報を取得する場合

  「15.5.1　ODBC の SQL データ型と HADB のデータ型の対応」に記載されている ODBC SQL データ型識別子を指定します。

- サポートしているすべてのデータ型に関する情報を取得する場合

  SQL_ALL_TYPES を指定します。

(4)　戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

SQLGetTypeInfo またはSQLGetTypeInfoW を実行すると結果セットが作成されます。返却される結果セットの形式を次に示します。

表 16-4　返却される ResultSet オブジェクトの形式

列番号	型	列名	列の意味
1	Varchar	TYPE_NAME	HADB データ型名
2	Smallint	DATA_TYPE	ODBC SQL データ型
3	Integer	COLUMN_SIZE	列サイズ
4	Varchar	LITERAL_PREFIX	リテラルを引用するために使用する接頭辞
5	Varchar	LITERAL_SUFFIX	リテラルを引用するために使用する接尾辞
6	Varchar	CREATE_PARAMS	型の作成に使用するパラメタ
7	Smallint	NULLABLE	ナル値を使用できるかどうかを返します。 常にSQL_NULLABLE_UNKNOWN を返します。
8	Smallint	CASE_SENSITIVE	大文字と小文字を区別するかどうかを返します。 • SQL_TRUE：文字列系データ型 • SQL_FALSE：そのほかのデータ型
9	Smallint	SEARCHABLE	WHERE でデータ型がどのように使われるかを返します。 常にSQL_SEARCHABLE を返します。
10	Smallint	UNSIGNED_ATTRIBUTE	符号なし属性かどうかを返します。 • SQL_FALSE：数値データ型（符号ありのため） • ナル値：そのほかのデータ型
11	Smallint	FIXED_PREC_SCALE	通貨の値になれるかどうかを返します。 常にSQL_FALSE を返します。
12	Smallint	AUTO_UNIQUE_VALUE	自動インクリメントの値に使用できるかどうかを返します。 常にナル値を返します。
13	Varchar	LOCAL_TYPE_NAME	型名の地域対応されたバージョン 型名と同じものを返します。
14	Smallint	MINIMUM_SCALE	サポートされる最小スケール スケールが適用されない場合はナル値を返します。
15	Smallint	MAXIMUM_SCALE	サポートされる最大スケール スケールが適用されない場合はナル値を返します。
16	Smallint	SQL_DATA_TYPE	SQL_DESC_TYPE フィールドに設定される ODBC SQL データ型
17	Smallint	SQL_DATETIME_SUB	日付時刻のサブコード 日付時刻のデータ型以外の場合はナル値を返します。
18	Integer	NUM_PREC_RADIX	• 10：数値データ型 • ナル値：そのほかのデータ型

列番号	型	列名	列の意味
19	Smallint	INTERVAL_PRECISION	常にナル値を返します。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01S02	オプション値の変更		×
08S01	通信リンク失敗		×
24000	無効なカーソル状態		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY004	無効な SQL データ型		○
HY008	動作がキャンセルされた		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY117	接続がサスペンド中		×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.5　ドライバオプションの設定および取得

ここでは，ドライバオプションの設定，および取得時に使用する ODBC 関数について説明します。

16.5.1　SQLSetConnectAttr，SQLSetConnectAttrW

(1)　機能

接続属性を設定します。

(2)　形式

- SQLSetConnectAttr の場合

```
SQLRETURN SQLSetConnectAttr
(
  SQLHDBC           ConnectionHandle,   /* In */
  SQLINTEGER        Attribute,          /* In */
  SQLPOINTER        ValuePtr,           /* In */
  SQLINTEGER        StringLength        /* In */
)
```

- SQLSetConnectAttrW の場合

```
SQLRETURN SQLSetConnectAttrW
(
  SQLHDBC           ConnectionHandle,   /* In */
  SQLINTEGER        Attribute,          /* In */
  SQLPOINTER        ValuePtr,           /* In */
  SQLINTEGER        StringLength        /* In */
)
```

(3)　引数

ConnectionHandle：

属性を設定する接続の接続ハンドルを指定します。

Attribute：

設定する接続属性を指定します。指定できる属性については，「16.14　SQLSetConnectAttr，SQLSetConnectAttrW，SQLGetConnectAttr および SQLGetConnectAttrW に指定できる属性」を参照してください。

ValuePtr：

Attribute で関連づけられた値を指すポインタまたは 32 ビット整数値を指定します。指定できる値については，「16.14　SQLSetConnectAttr，SQLSetConnectAttrW，SQLGetConnectAttr および SQLGetConnectAttrW に指定できる属性」を参照してください。

StringLength：

*ValuePtr の長さを指定します（単位：バイト長）。ValuePtr が整数の場合，この引数の指定は無視されます。

ValuePtr が文字列へのポインタである場合，文字列の長さ（単位：バイト長）またはSQL_NTS を指定します。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01S02	オプション値の変更		○
08002	接続名が使用中である		×
08003	接続が存在しない		×
08S01	通信リンク失敗		×
24000	無効なカーソル状態		×
25000	ローカルトランザクション中の不正な操作		×
3D000	無効なカタログ名		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		×
HY011	ここでは属性を設定できない		○
HY013	メモリ管理エラー		×
HY024	無効な属性値		○
HY090	無効な文字列長または無効なバッファ長	次の条件をすべて満たしています。 • *ValuePtr が文字列です。 • StringLength が 0 より小さい値です。 • StringLength がSQL_NTS 以外の値です。	○
HY092	無効な属性識別子，または無効なオプション識別子	－	○

SQLSTATE	説明	備考	返却
HY114	ドライバは接続レベルの非同期実行をサポートしていない		×
HY117	接続がサスペンド中		×
HY121	カーソルライブラリとドライバ依存のコネクションプーリングは同時実行できない		×
HYC00	オプション機能は実装されていない	指定されたAttributeは，規格にはありますがサポートしていません。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×
IM009	トランスレータDLLをロードできない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×
S1118	ドライバは非同期通知をサポートしていない		×

（凡例）

○：HADB ODBCドライバが返すことがあるSQLSTATEです。

×：HADB ODBCドライバが返さないSQLSTATEです。

－：なし。

(6) 注意事項

- ODBC規約に存在しないHADB独自のAttributeを指定する場合は，adbodb.hをincludeしてください。
- この関数で指定するトランザクションアクセスモード（属性はSQL_ATTR_ACCESS_MODE）は，次の表に示す優先順位に従って決まります（優先順位の番号が小さいほど，優先順位が高くなります）。

表16-5　トランザクションアクセスモードの優先順位

優先順位	トランザクションアクセスモードの指定個所
1	SQLSetConnectAttrまたはSQLSetConnectAttrWの指定
2	クライアント定義のadb_clt_trn_access_modeオペランドの指定

16.5.2　SQLGetConnectAttr，SQLGetConnectAttrW

(1)　機能

接続ハンドルの属性に設定されている値を返します。

(2)　形式

- SQLGetConnectAttr の場合

```
SQLRETURN SQLGetConnectAttr
(
  SQLHDBC           ConnectionHandle,    /* In */
  SQLINTEGER        Attribute,           /* In */
  SQLPOINTER        ValuePtr,            /* Out */
  SQLINTEGER        BufferLength,        /* In */
  SQLINTEGER      * StringLengthPtr      /* Out */
)
```

- SQLGetConnectAttrW の場合

```
SQLRETURN SQLGetConnectAttrW
(
  SQLHDBC           ConnectionHandle,    /* In */
  SQLINTEGER        Attribute,           /* In */
  SQLPOINTER        ValuePtr,            /* Out */
  SQLINTEGER        BufferLength,        /* In */
  SQLINTEGER      * StringLengthPtr      /* Out */
)
```

(3)　引数

ConnectionHandle：

接続ハンドルを指定します。

Attribute：

取得する属性を指定します。指定できる属性については，「16.14　SQLSetConnectAttr，SQLSetConnectAttrW，SQLGetConnectAttr および SQLGetConnectAttrW に指定できる属性」を参照してください。

ValuePtr：

Attribute で指定された属性の値を返すバッファのポインタを指定します。ValuePtr にはNULL も指定できます。

BufferLength：

Attribute に指定された属性の値の型が，文字列またはバイナリの場合，*ValuePtr の長さ（単位：バイト長）を指定します。それ以外のデータ型の場合，この引数の指定は無視されます。

StringLengthPtr：

返した属性の値の有効な長さ（単位：バイト長）の合計を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

ValuePtr がNULL の場合，*StringLengthPtr に値は返されません。

(4) 戻り値

SQL_SUCCESS，SQL_NO_DATA，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		×
08003	接続が存在しない		○
08S01	通信リンク失敗		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY092	無効な属性識別子，または無効なオプション識別子		○
HYC00	オプション機能は実装されていない		○
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

ODBC 規約に存在しない HADB 独自のAttribute を指定する場合は，adbodb.h をinclude してください。

16.5.3　SQLSetEnvAttr

(1)　機能

環境属性を設定します。

(2)　形式

```
SQLRETURN SQLSetEnvAttr
(
  SQLHENV           EnvironmentHandle,    /* In */
  SQLINTEGER        Attribute,            /* In */
  SQLPOINTER        ValuePtr,             /* In */
  SQLINTEGER        StringLength          /* In */
)
```

(3)　引数

EnvironmentHandle：

属性を設定する接続の環境ハンドルを指定します。

Attribute：

設定する環境属性を指定します。指定できる属性については，「16.15　SQLSetEnvAttr および SQLGetEnvAttr に指定できる属性」を参照してください。

ValuePtr：

Attribute で関連づけられた値を指すポインタまたは 32 ビット整数値を指定します。指定できる値については，「16.15　SQLSetEnvAttr および SQLGetEnvAttr に指定できる属性」を参照してください。

StringLength：

ValuePtr が文字列またはバイナリを指すポインタの場合，*ValuePtr の長さを指定します（単位：バイト長）。ValuePtr が整数の場合，この引数の指定は無視されます。

(4)　戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5)　SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01S02	オプション値の変更		×
HY000	一般エラー		×

SQLSTATE	説明	備考	返却
HY001	メモリ割り当てエラー		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○
HY011	ここでは属性を設定できない	EnvironmentHandle に接続ハンドルが割り当てられています。	×
HY013	メモリ管理エラー	－	×
HY024	無効な属性値	Attribute に指定された値に対して，ValuePtr に無効な値が指定されています。	○
HY090	無効な文字列長または無効なバッファ長	StringLength に 0 より小さく，SQL_NTS 以外の値が指定されています。	○
HY092	無効な属性識別子，または無効なオプション識別子	無効なAttribute が指定されています。	○
HY117	接続がサスペンド中	－	×
HYC00	オプション機能は実装されていない	指定されたAttribute は，規格にはありますがサポートしていません。	○

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.5.4 SQLGetEnvAttr

(1) 機能

環境ハンドルの属性に設定されている値を返します。

(2) 形式

```
SQLRETURN SQLGetEnvAttr
(
  SQLHENV           EnvironmentHandle,    /* In */
  SQLINTEGER        Attribute,            /* In */
  SQLPOINTER        ValuePtr,             /* Out */
  SQLINTEGER        BufferLength,         /* In */
  SQLINTEGER      * StringLengthPtr       /* Out */
)
```

(3) 引数

EnvironmentHandle：

環境ハンドルを指定します。

Attribute：

取得する属性を指定します。指定できる属性については，「16.15 SQLSetEnvAttr および SQLGetEnvAttr に指定できる属性」を参照してください。

ValuePtr：

Attribute で指定された属性の値を返すバッファのポインタを指定します。ValuePtr にはNULL も指定できます。

BufferLength：

Attribute に指定された属性の値の型が，文字列またはバイナリの場合，*ValuePtr の長さを指定します（単位：バイト長）。それ以外のデータ型の場合，この引数の指定は無視されます。

StringLengthPtr：

返した属性の値の有効な総バイト長を格納するバッファへのポインタを指定します。この総バイト長にNULL 終端文字のバイト長は含みません。ValuePtr がNULL の場合，*StringLengthPtr に値は返されません。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY092	無効な属性識別子，または無効なオプション識別子		○
HYC00	オプション機能は実装されていない		○
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.5.5　SQLSetStmtAttr，SQLSetStmtAttrW

(1)　機能

ステートメントに関連する属性を設定します。

(2)　形式

- SQLSetStmtAttr の場合

```
SQLRETURN SQLSetStmtAttr
(
  SQLHSTMT          StatementHandle,      /* In */
  SQLINTEGER        Attribute,            /* In */
  SQLPOINTER        ValuePtr,             /* In */
  SQLINTEGER        StringLength          /* In */
)
```

- SQLSetStmtAttrW の場合

```
SQLRETURN SQLSetStmtAttrW
(
  SQLHSTMT          StatementHandle,      /* In */
  SQLINTEGER        Attribute,            /* In */
  SQLPOINTER        ValuePtr,             /* In */
  SQLINTEGER        StringLength          /* In */
)
```

(3)　引数

StatementHandle：

ステートメントハンドルを指定します。

Attribute：

設定する属性を指定します。指定できる属性については，「16.16　SQLSetStmtAttr，SQLSetStmtAttrW，SQLGetStmtAttr および SQLGetStmtAttrW に指定できる属性」を参照してください。

ValuePtr：

Attribute で指定した属性に設定する値を指定します。指定できる属性については，「16.16 SQLSetStmtAttr，SQLSetStmtAttrW，SQLGetStmtAttr および SQLGetStmtAttrW に指定できる属性」を参照してください。

StringLength：

Attribute に指定された属性の値の型が，文字列またはバイナリの場合，*ValuePtr の長さを指定します（単位：バイト長）。それ以外のデータ型の場合，この引数の指定は無視されます。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01S02	オプション値の変更		○
08S01	通信リンク失敗		×
24000	無効なカーソル状態		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○
HY011	ここでは属性を設定できない		○
HY013	メモリ管理エラー		×
HY017	自動的に割り当てられるディスクリプタハンドルの不正使用		○
HY024	無効な属性値		○
HY090	無効な文字列長または無効なバッファ長		×
HY092	無効な属性識別子，または無効なオプション識別子		○
HYC00	オプション機能は実装されていない		○
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.5.6 SQLGetStmtAttr，SQLGetStmtAttrW

(1) 機能

ステートメントハンドルに設定されている属性の値を返します。

(2) 形式

- SQLGetStmtAttr の場合

```
SQLRETURN SQLGetStmtAttr
(
  SQLHSTMT           StatementHandle,      /* In */
  SQLINTEGER         Attribute,            /* In */
  SQLPOINTER         ValuePtr,             /* Out */
  SQLINTEGER         BufferLength,         /* In */
  SQLINTEGER       * StringLengthPtr       /* Out */
)
```

- SQLGetStmtAttrW の場合

```
SQLRETURN SQLGetStmtAttrW
(
  SQLHSTMT           StatementHandle,      /* In */
  SQLINTEGER         Attribute,            /* In */
  SQLPOINTER         ValuePtr,             /* Out */
  SQLINTEGER         BufferLength,         /* In */
  SQLINTEGER       * StringLengthPtr       /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

Attribute：

取得する属性を指定します。指定できる属性については，「16.16 SQLSetStmtAttr, SQLSetStmtAttrW，SQLGetStmtAttr および SQLGetStmtAttrW に指定できる属性」を参照してください。

ValuePtr：

Attributeで指定された属性の値を返すバッファのポインタを指定します。ValuePtrにはNULLも指定できます。

BufferLength：

Attributeに指定された属性の値の型が，文字列またはバイナリの場合，*ValuePtrの長さを指定します（単位：バイト長）。それ以外のデータ型の場合，この引数の指定は無視されます。

StringLengthPtr：

返した属性の値の有効なバイト長の合計が格納されるバッファのポインタを指定します。このバイト長の合計にNULL終端文字のバイト長は含みません。ValuePtrがNULLの場合，*StringLengthPtrに値は返されません。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLEが返されます。

(5) SQLSTATE

この関数では次のSQLSTATEを返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
24000	無効なカーソル状態		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY092	無効な属性識別子，または無効なオプション識別子		○
HY109	無効なカーソル位置		×
HYC00	オプション機能は実装されていない		○
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBCドライバが返すことがあるSQLSTATEです。

×：HADB ODBCドライバが返さないSQLSTATEです。

－：なし。

16.6　ディスクリプタ値の設定

ここでは，ディスクリプタ値の設定時に使用する ODBC 関数について説明します。

16.6.1　SQLGetDescField，SQLGetDescFieldW

(1)　機能

引数で指定されたディスクリプタフィールドの値を返します。

(2)　形式

- SQLGetDescField の場合

```
SQLRETURN SQLGetDescField
(
  SQLHDESC        DescriptorHandle,          /* In */
  SQLSMALLINT     RecNumber,                 /* In */
  SQLSMALLINT     FieldIdentifier,           /* In */
  SQLPOINTER      ValuePtr,                  /* Out */
  SQLINTEGER      BufferLength,              /* In */
  SQLINTEGER    * StringLengthPtr            /* Out */
)
```

- SQLGetDescFieldW の場合

```
SQLRETURN SQLGetDescFieldW
(
  SQLHDESC        DescriptorHandle,          /* In */
  SQLSMALLINT     RecNumber,                 /* In */
  SQLSMALLINT     FieldIdentifier,           /* In */
  SQLPOINTER      ValuePtr,                  /* Out */
  SQLINTEGER      BufferLength,              /* In */
  SQLINTEGER    * StringLengthPtr            /* Out */
)
```

(3)　引数

DescriptorHandle：

ディスクリプタハンドルを指定します。

RecNumber：

列番号（ARD またはIRD），またはパラメタ番号（APD またはIPD）を指定します。

FieldIdentifier がヘッダフィールドを示す場合，この引数の指定は無視されます。

FieldIdentifier：

値を返すディスクリプタのフィールド（ヘッダフィールドまたはレコードフィールド）を指定します。指定できる属性については，「16.17　SQLGetDescField，SQLGetDescFieldW，SQLSetDescField および SQLSetDescFieldW に指定できる属性」を参照してください。

ValuePtr：

ディスクリプタ情報を返すバッファへのポインタを指定します。この引数にはNULL も指定できます。

BufferLength：

FieldIdentifier に指定するディスクリプタフィールドのデータ型が，文字列またはバイナリの場合，*ValuePtr の長さを指定します（単位：バイト長）。それ以外のデータ型の場合，この引数の指定は無視されます。SQL_NTS は指定できません。

StringLengthPtr：

*ValuePtr に設定する値の有効な長さ（単位：バイト長）を格納するバッファへのポインタを指定します。

この引数にNULL が指定された場合でもValuePtr に値は設定されます。

(4)　戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，SQL_NO_DATA，またはSQL_INVALID_HANDLE が返されます。

(5)　SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	BufferLength に指定された値は，列名のバイト長より小さいため，列名が切り捨てられました。切り捨てられる前の列名の長さが，*StringLengthPtr に格納されます。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
07009	無効なディスクリプタインデクス	RecNumber に0 以下の値が指定されています。	○
08S01	通信リンク失敗	－	×
HY001	メモリ割り当てエラー		×

SQLSTATE	説明	備考	返却
HY007	関連づけられたステートメントが準備されていない		○
HY010	関数シーケンスエラー	この関数を実行する前に，DescriptorHandle が関連づけられている StatementHandle に対して，SQLExecute，SQLExecDirect，SQLExecDirectW または SQLParamData のどれかが呼び出され，SQL_NEED_DATA を返しました。その後，実行時データパラメタまたは実行時データ列の設定が完了していません。	○
HY013	メモリ管理エラー	－	×
HY021	ディスクリプタ情報の不一致		×
HY090	無効な文字列長または無効なバッファ長	• *ValuePtr が文字列で，BufferLength に 0 より小さい値が指定されています。 • BufferLength に無効な値（SQL_NTS）が指定されました。	○
HY091	無効なディスクリプタフィールド識別子	FieldIdentifier は，ODBC 定義のフィールドではありません。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.6.2 SQLGetDescRec，SQLGetDescRecW

(1) 機能

引数で指定されたディスクリプタレコードの複数フィールドの現在の設定または値を返します。列またはパラメタの名前，データ型，および格納位置を記述するフィールドが返されます。

(2) 形式

- SQLGetDescRec の場合

```
SQLRETURN SQLGetDescRec
(
  SQLHDESC         DescriptorHandle,      /* In */
  SQLSMALLINT      RecNumber,             /* In */
```

```
  SQLCHAR        * Name,                  /* Out */
  SQLSMALLINT      BufferLength,          /* In */
  SQLSMALLINT    * StringLengthPtr,       /* Out */
  SQLSMALLINT    * TypePtr,               /* Out */
  SQLSMALLINT    * SubTypePtr,            /* Out */
  SQLLEN         * LengthPtr,             /* Out */
  SQLSMALLINT    * PrecisionPtr,          /* Out */
  SQLSMALLINT    * ScalePtr,              /* Out */
  SQLSMALLINT    * NullablePtr            /* Out */
)
```

- SQLGetDescRecW の場合

```
SQLRETURN SQLGetDescRecW
(
  SQLHDESC         DescriptorHandle,      /* In */
  SQLSMALLINT      RecNumber,             /* In */
  SQLWCHAR       * Name,                  /* Out */
  SQLSMALLINT      BufferLength,          /* In */
  SQLSMALLINT    * StringLengthPtr,       /* Out */
  SQLSMALLINT    * TypePtr,               /* Out */
  SQLSMALLINT    * SubTypePtr,            /* Out */
  SQLLEN         * LengthPtr,             /* Out */
  SQLSMALLINT    * PrecisionPtr,          /* Out */
  SQLSMALLINT    * ScalePtr,              /* Out */
  SQLSMALLINT    * NullablePtr            /* Out */
)
```

(3) 引数

DescriptorHandle：

ディスクリプタハンドルを指定します。

RecNumber：

列番号（ARD またはIRD），またはパラメタ番号（APD またはIPD）を指定します。

Name：

ディスクリプタレコードのSQL_DESC_NAME フィールドの値を返すバッファへのポインタを指定します。

BufferLength：

*Name バッファの長さ※を指定します。

この長さに NULL 終端文字は含まれます。SQL_NTS は指定できません。

StringLengthPtr：

*Name バッファに返す有効なデータの長さ※を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

重要

ここに格納された*Name の長さ※が，BufferLength から NULL 終端文字分を引いた長さ※より大きい場合，*Name に格納される文字列はBufferLength から NULL 終端文字分を引いた長さ※に切り捨てられ，末尾に NULL 終端文字が付加されます。

注※

長さの単位は，SQLGetDescRec の場合はバイト長，SQLGetDescRecW の場合は文字数となります。

TypePtr：

ディスクリプタレコードのSQL_DESC_TYPE フィールドの値を返すバッファへのポインタを指定します。

SubTypePtr：

ディスクリプタレコードのSQL_DESC_DATETIME_INTERVAL_CODE フィールドの値を返すバッファへのポインタを指定します。

LengthPtr：

ディスクリプタレコードのSQL_DESC_OCTET_LENGTH フィールドの値を返すバッファへのポインタを指定します。

PrecisionPtr：

ディスクリプタレコードのSQL_DESC_PRECISION フィールドの値を返すバッファへのポインタを指定します。

ScalePtr：

ディスクリプタレコードのSQL_DESC_SCALE フィールドの値を返すバッファへのポインタを指定します。

NullablePtr：

ディスクリプタレコードのSQL_DESC_NULLABLE フィールドの値を返すバッファへのポインタを指定します。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，SQL_NO_DATA，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	BufferLength に指定された値が，列名のバイト長より小さいため，列名が切り捨てられました。切り捨てられる前の列名の長さが，*StringLengthPtr に格納されます。こ	○

SQLSTATE	説明	備考	返却
		のとき，SQL_SUCCESS_WITH_INFO を返します。	
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
07009	無効なディスクリプタインデクス	RecNumber に 0 以下の値が指定されています。	○
08S01	通信リンク失敗	－	×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY007	関連づけられたステートメントが準備されていない		○
HY010	関数シーケンスエラー	この関数を実行する前に，DescriptorHandle が関連づけられている StatementHandle に対して，SQLExecute，SQLExecDirect，SQLExecDirectW または SQLParamData のどれかが呼び出され，SQL_NEED_DATA を返しました。その後，実行時データパラメタまたは実行時データ列の設定が完了していません。	○
HY013	メモリ管理エラー	－	×
HY090	無効な文字列長または無効なバッファ長	BufferLength に無効な値（SQL_NTS）が指定されました。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.6.3 SQLSetDescField，SQLSetDescFieldW

(1) 機能

引数で指定されたディスクリプタフィールドの値を設定します。

(2) 形式

- SQLSetDescField の場合

```
SQLRETURN SQLSetDescField
(
  SQLHDESC      DescriptorHandle,       /* In */
  SQLSMALLINT   RecNumber,              /* In */
  SQLSMALLINT   FieldIdentifier,        /* In */
  SQLPOINTER    ValuePtr,               /* In */
  SQLINTEGER    BufferLength            /* In */
)
```

- SQLSetDescFieldW の場合

```
SQLRETURN SQLSetDescFieldW
(
  SQLHDESC      DescriptorHandle,       /* In */
  SQLSMALLINT   RecNumber,              /* In */
  SQLSMALLINT   FieldIdentifier,        /* In */
  SQLPOINTER    ValuePtr,               /* In */
  SQLINTEGER    BufferLength            /* In */
)
```

(3) 引数

DescriptorHandle：

ディスクリプタハンドルを指定します。

RecNumber：

列番号（ARD またはIRD），またはパラメタ番号（APD またはIPD）を指定します。

FieldIdentifier がヘッダフィールドを示す場合，この引数の指定は無視されます。

この引数に該当するレコード番号を持つディスクリプタレコードがない場合，この関数は，そのレコード番号を持つディスクリプタレコードを新しく作成します。

FieldIdentifier：

値を設定するディスクリプタのフィールド（ヘッダフィールドまたはレコードフィールド）を指定します。指定できる属性については，「16.17　SQLGetDescField，SQLGetDescFieldW，SQLSetDescField および SQLSetDescFieldW に指定できる属性」を参照してください。

ValuePtr：

設定するディスクリプタ情報（ポインタまたは整数値）を指定します。指定できる値については，「16.17　SQLGetDescField，SQLGetDescFieldW，SQLSetDescField および SQLSetDescFieldW に指定できる属性」を参照してください。

BufferLength：

FieldIdentifier に指定するディスクリプタフィールドのデータ型が，文字列またはバイナリの場合，この引数に*ValuePtr の長さを指定します（単位：バイト長）。それ以外のデータ型の場合，この引数の指定は無視されます。

(4) 戻り値

SQL_SUCCESS, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	−	×
01S02	オプション値の変更		×
07009	無効なディスクリプタインデクス	• RecNumber に 0 以下の値が指定されています。 • RecNumber がデータソースがサポートしている列の最大個数またはパラメタの最大個数より大きく，DescriptorHandle が ARD または APD と関連づけられています。 • FieldIdentifier がSQL_DESC_COUNT で，ValuePtr に 0 より小さい値が指定されています。	○
08S01	通信リンク失敗	−	×
22001	文字列データの右側が切り捨てられた		×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
HY001	メモリ割り当てエラー	−	○
HY010	関数シーケンスエラー	この関数を実行する前に，DescriptorHandle が関連づけられている StatementHandle に対して，SQLExecute，SQLExecDirect，SQLExecDirectW または SQLParamData のどれかが呼び出され，SQL_NEED_DATA を返しました。その後，実行時データパラメタまたは実行時データ列の設定が完了していません。	○
HY013	メモリ管理エラー	−	×
HY016	インプリメンテーション行ディスクリプタを変更できない	DescriptorHandle が IRD と関連づけられていますが，FieldIdentifier が SQL_DESC_ARRAY_STATUS_PTR または SQL_DESC_ROWS_PROCESSED_PTR 以外です。	○
HY021	ディスクリプタ情報の不一致	−	×
HY090	無効な文字列長または無効なバッファ長		×

SQLSTATE	説明	備考	返却
HY091	無効なディスクリプタフィールド識別子	• FieldIdentifier に指定された値がODBC 定義のフィールドでなく，実装時に定義された値でもありません。 • FieldIdentifier はDescriptorHandle に対して無効です。	○
HY092	無効な属性識別子，または無効なオプション識別子	• FieldIdentifier の指定値に対して，ValuePtr に設定されている値は無効です。 • FieldIdentifier がSQL_DESC_UNNAMED で，ValuePtr がSQL_NAMED です。	○
HY105	無効なパラメタの種類	SQL_DESC_PARAMETER_TYPE フィールドに指定された値が無効（SQL_PARAM_INPUT 以外）です。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.6.4 SQLSetDescRec

(1) 機能

同一ディスクリプタレコード内の複数ディスクリプタレコードフィールドに値を設定します。

(2) 形式

```
SQLRETURN SQLSetDescRec
(
  SQLHDESC        DescriptorHandle,    /* In */
  SQLSMALLINT     RecNumber,           /* In */
  SQLSMALLINT     Type,                /* In */
  SQLSMALLINT     SubType,             /* In */
  SQLLEN          Length,              /* In */
  SQLSMALLINT     Precision,           /* In */
  SQLSMALLINT     Scale,               /* In */
  SQLPOINTER      DataPtr,             /* Deferred  In */
  SQLLEN        * StringLengthPtr,     /* Deferred  In */
  SQLLEN        * IndicatorPtr         /* Deferred  In */
)
```

(3) 引数

DescriptorHandle：

ディスクリプタハンドルを指定します。IRD のハンドルは指定できません。

RecNumber：

列番号（ARD またはIRD），またはパラメタ番号（APD またはIPD）を指定します。

この引数に該当するレコード番号を持つディスクリプタレコードがない場合，この関数は，そのレコード番号を持つディスクリプタレコードを新しく作成します。

Type：

ディスクリプタレコードのSQL_DESC_TYPE フィールドに設定する値を指定します。

SubType：

ディスクリプタレコードのSQL_DESC_DATETIME_INTERVAL_CODE フィールドに設定する値を指定します。

Length：

ディスクリプタレコードのSQL_DESC_OCTET_LENGTH フィールドに設定する値を指定します。

Precision：

ディスクリプタレコードのSQL_DESC_PRECISION フィールドに設定する値を指定します。

Scale：

ディスクリプタレコードのSQL_DESC_SCALE フィールドに設定する値を指定します。

DataPtr：

ディスクリプタレコードのSQL_DESC_DATA_PTR フィールドに設定する値を指定します。

StringLengthPtr：

ディスクリプタレコードのSQL_DESC_OCTET_LENGTH_PTR フィールドに設定する値を指定します。

IndicatorPtr：

ディスクリプタレコードのSQL_DESC_INDICATOR_PTR フィールドに設定する値を指定します。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
07009	無効なディスクリプタインデクス	• RecNumber が，データソースがサポートしている列の最大個数またはパラメタの最大個数より大きく，	○

SQLSTATE	説明	備考	返却
		DescriptorHandle が ARD，APD または IPD と関連づけられています。 • RecNumber に 0 以下の値が指定されています。	
08S01	通信リンク失敗	－	×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY010	関数シーケンスエラー	この関数を実行する前に，DescriptorHandle が関連づけられている StatementHandle に対して，SQLExecute，SQLExecDirect，SQLExecDirectW または SQLParamData のどれかが呼び出され，SQL_NEED_DATA を返しました。その後，実行時データパラメタまたは実行時データ列の設定が完了していません。	○
HY013	メモリ管理エラー	－	×
HY016	インプリメンテーション行ディスクリプタを変更できない	DescriptorHandle が IRD と関連づけられています。	○
HY021	ディスクリプタ情報の不一致	－	○
HY090	無効な文字列長または無効なバッファ長		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.6.5 SQLCopyDesc

(1) 機能

ディスクリプタ情報をディスクリプタハンドル間でコピーします。

(2) 形式

```
SQLRETURN SQLCopyDesc
(
```

```
  SQLHDESC      SourceDescHandle,             /* In */
  SQLHDESC      TargetDescHandle              /* In */
)
```

(3) 引数

SourceDescHandle：

コピー元のディスクリプタハンドルを指定します。

TargetDescHandle：

コピー先のディスクリプタハンドルを指定します。この引数には，アプリケーションディスクリプタまたは IPD のハンドルを指定できます。この引数には，IRD のハンドルを指定できません。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY007	関連づけられたステートメントが準備されていない		○
HY010	関数シーケンスエラー	この関数を実行する前に，SourceDescHandle またはTargetDescHandle のディスクリプタハンドルが関連づけられているStatementHandle に対して，SQLExecute，SQLExecDirect，SQLExecDirectW またはSQLParamData のどれかが呼び出され，SQL_NEED_DATA を返しました。その後，実行時データパラメタまたは実行時データ列の設定が完了していません。	○
HY013	メモリ管理エラー	－	×
HY016	インプリメンテーション行ディスクリプタを変更できない	TargetDescHandle が IRD と関連づけられました。	○
HY021	ディスクリプタ情報の不一致	－	×

SQLSTATE	説明	備考	返却
HY092	無効な属性識別子，または無効なオプション識別子		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すこと性があるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

戻り値にSQL_SUCCESS 以外が返却された場合，CompletionType にSQL_ROLLBACK を指定したSQLEndTran，またはHandleType にSQL_HANDLE_STMT を指定したSQLFreeHandle を実行してください。

16.7　SQL 要求の作成

ここでは，SQL 要求の作成時に使用する ODBC 関数について説明します。

16.7.1　SQLPrepare，SQLPrepareW

(1)　機能

SQL 文をデータソースに送り，SQL 文の前処理を実行します。

(2)　形式

- SQLPrepare の場合

```
SQLRETURN SQLPrepare
(
  SQLHSTMT         StatementHandle,     /* In */
  SQLCHAR        * StatementText,       /* In */
  SQLINTEGER       TextLength           /* In */
)
```

- SQLPrepareW の場合

```
SQLRETURN SQLPrepareW
(
  SQLHSTMT         StatementHandle,     /* In */
  SQLWCHAR       * StatementText,       /* In */
  SQLINTEGER       TextLength           /* In */
)
```

(3)　引数

StatementHandle：

ステートメントハンドルを指定します。

StatementText：

SQL 文字列を指定します。SQL 文字列中に，注釈（/*～*/）は指定できませんが，インデクス指定（/*>>～<<*/）などは指定できます。

TextLength：

*StatementText の長さを指定します（単位：SQLPrepare の場合はバイト長，SQLPrepareW の場合は文字数）。

StatementText に指定した SQL 文字列が NULL 終端文字を保証している場合，SQL_NTS を指定できます。

(4) 戻り値

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	−	×
01S02	オプション値の変更		×
08S01	通信リンク失敗		×
21S01	挿入する値の一覧と列の一覧の不一致		×
21S02	導出したテーブルの次数が列の一覧と不一致		×
22018	キャスト指定に対する無効な文字値		×
22019	無効なエスケープ文字		×
22025	無効なエスケープシーケンス		×
24000	無効なカーソル状態		×
34000	無効なカーソル名		×
3D000	無効なカタログ名		×
3F000	無効なスキーマ名		×
42000	構文エラーまたはアクセス違反		×
42S01	ベーステーブルまたはビューがすでに存在する		×
42S02	ベーステーブルまたはビューが見つからない		×
42S11	インデクスがすでに存在する		×
42S12	インデクスが見つからない		×
42S21	列がすでに存在する		×
42S22	列が見つからない		×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C041	データ型未サポートエラー	ドライバは指定されたデータ型をサポートしていません。	○
5C051	SQL 文のテキスト文字列の長さが，16,000,000 文字を超えた	SQL 文のテキスト文字列が 16,000,000 文字以下の場合でも，ドライバマネージャの文字コード変換によって 16,000,000 文字	○

SQLSTATE	説明	備考	返却
		を超えることがあります。このときも，このSQLSTATE が返されます。	
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー	ドライバは，関数の実行または完了をサポートするメモリを割り当てられません。	○
HY008	動作がキャンセルされた	－	×
HY009	NULL ポインタの不正使用	StatementText に NULL ポインタが指定されています。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	TextLength に 0 以下で，SQL_NTS 以外の値が指定されています。	○
HY117	接続がサスペンド中	－	×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.7.2 SQLBindParameter

(1) 機能

SQL 文の？パラメタにバッファをバインドします。

(2) 形式

```
SQLRETURN SQLBindParameter
(
  SQLHSTMT         StatementHandle,     /* In */
```

```
  SQLUSMALLINT     ParameterNumber,    /* In */
  SQLSMALLINT      InputOutputType,    /* In */
  SQLSMALLINT      ValueType,          /* In */
  SQLSMALLINT      ParameterType,      /* In */
  SQLULEN          ColumnSize,         /* In */
  SQLSMALLINT      DecimalDigits,      /* In */
  SQLPOINTER       ParameterValuePtr,  /* In */
  SQLLEN           BufferLength,       /* In */
  SQLLEN         * StrLen_or_IndPtr    /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

ParameterNumber：

パラメタ番号を指定します。パラメタの昇順に 1 から番号が付けられます。

InputOutputType：

次のパラメタの入出力の種類を指定します。

- SQL_PARAM_INPUT

  入力パラメタです。

- SQL_PARAM_INPUT_OUTPUT

  入出力パラメタです。

  ただし，入出力パラメタは未サポートのため，SQL_PARAM_INPUT が指定されたとして動作します。

ValueType：

パラメタの C データ型またはSQL_C_DEFAULT を指定します。

SQL_C_DEFAULT を指定すると，デフォルトの C データ型を仮定します。

サポートしているデータ型については，「15.5.2 ODBC の SQL データ型と C データ型の対応」を参照してください。サポートされていない C データ型を指定した場合はエラーとなります。

ParameterType：

パラメタの ODBC SQL データ型を指定します。

サポートしているデータ型については，「15.5.1 ODBC の SQL データ型と HADB のデータ型の対応」を参照してください。サポートされていない ODBC SQL データ型を指定した場合はエラーとなります。

ColumnSize：

対応する？パラメタのデータのバイト長を指定します。

ParameterType がSQL_CHAR，SQL_VARCHAR，SQL_DECIMAL，SQL_DOUBLE，SQL_FLOAT またはSQL_NUMERIC の場合にColumnSize の値が使用されます。

そのほかのデータ型の場合，この引数の指定は無視されます。

DecimalDigits：

対応する？パラメタの列または式の小数点以下の桁数を指定します。

ParameterType がSQL_TYPE_TIME，SQL_TYPE_TIMESTAMP，SQL_DECIMAL，SQL_DOUBLE，SQL_FLOAT または SQL_NUMERIC の場合にDecimalDigits の値が使用されます。

そのほかのデータ型の場合，この引数の指定は無視されます。

ParameterValuePtr：

パラメタデータのバッファへのポインタを指定します。データ型は，ValueType で指定される形式である必要があります。*StrLen_or_IndPtr がSQL_NULL_DATA またはSQL_DATA_AT_EXEC の場合，NULL ポインタを指定できます。

SQL_LEN_DATA_AT_EXEC(length)マクロの結果，またはSQL_DATA_AT_EXEC の場合，ParameterValuePtr は，パラメタに関連づけられたアプリケーション定義の 32 ビット値になります。

BufferLength：

文字型の C データの場合，ParameterValuePtr バッファのバイト長を指定します。そのほかの C データの場合，この引数の指定は無視されます。

StrLen_or_IndPtr：

次のうちのどれかの値を格納するバッファへのポインタを指定します。

- *ParameterValuePtr に格納されるパラメタ値の長さ
  文字型の C データ以外の場合，この指定値は使用されません。
- SQL_NTS
  パラメタ値が NULL 終端文字の場合に指定します。
- SQL_NULL_DATA
  パラメタ値がNULL の場合に指定します。
- SQL_LEN_DATA_AT_EXEC(length)マクロの結果
  SQLPutData を使用する場合に指定します。length には0 または整数を指定します。
- SQL_DATA_AT_EXEC
  SQLPutData を使用する場合に指定します。

StrLen_or_IndPtr が NULL ポインタの場合，ドライバはすべての入力パラメタ値がナル値以外であり，文字データの最後にNULL が付いていると見なします。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
07006	データ型属性の制限違反		×
07009	無効なディスクリプタインデクス		○
HY000	一般エラー		×
HY001	メモリ割り当てエラー	ドライバは，関数の実行または完了をサポートするメモリを割り当てられません。	○
HY003	無効なアプリケーションのバッファのデータ型	－	○
HY004	無効な SQL データ型		○
HY009	NULL ポインタの不正使用		○
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY021	ディスクリプタ情報の不一致		×
HY090	無効な文字列長または無効なバッファ長	• BufferLength の値に 0 より小さい値が指定されています。 • SQLBindParameter に NULL ポインタが指定されていますが，パラメタ長が 0，SQL_NULL_DATA または SQL_DATA_AT_EXEC のどれにも一致していないか，SQL_LEN_DATA_AT_EXEC_OFFSET より大きい値になっています。	○
HY104	無効な精度，または無効なスケール値	－	×
HY105	無効なパラメタの種類	InputOutputType に指定された値は無効です。	○
HY117	接続がサスペンド中	－	×
HYC00	オプション機能は実装されていない	ValueType の値とParameterType の値が不整合です。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

- バインドした値は，次のどれかが実行されるまで有効です。
  - 再度同じParameterNumber を指定してSQLBindParameter を呼び出す
  - SQL_RESET_PARAMS オプションを指定してSQLFreeStmt を呼び出す
  - SQLSetDescField を呼び出して APD のSQL_DESC_COUNT ヘッダフィールドに0 を設定する
- オフセットによる再バインドは未サポートです。

16.7.3 SQLGetCursorName，SQLGetCursorNameW

(1) 機能

指定したステートメントハンドルに関連づけられたカーソル名を返します。

(2) 形式

- SQLGetCursorName の場合

```
SQLRETURN SQLGetCursorName
(
  SQLHSTMT         StatementHandle,        /* In */
  SQLCHAR        * CursorName,             /* Out */
  SQLSMALLINT      BufferLength,           /* In */
  SQLSMALLINT    * NameLengthPtr           /* Out */
)
```

- SQLGetCursorNameW の場合

```
SQLRETURN SQLGetCursorNameW
(
  SQLHSTMT         StatementHandle,        /* In */
  SQLWCHAR       * CursorName,             /* Out */
  SQLSMALLINT      BufferLength,           /* In */
  SQLSMALLINT    * NameLengthPtr           /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

CursorName：

カーソル名を返すバッファへのポインタを指定します。

BufferLength：

*CursorName の長さ※を指定します。SQL_NTS は指定できません。

NameLengthPtr：

*CursorName に返す有効な長さ※の合計を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

注※

長さの単位は，SQLGetCursorName の場合はバイト長，SQLGetCursorNameW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	*CursorName バッファの大きさが不足しているため，カーソル名を格納できませんでした（カーソル名が切り捨てられました）。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー	メモリオブジェクトにアクセスできないため，関数の呼び出しを処理できません。	×
HY015	利用できるカーソル名がない	－	×
HY090	無効な文字列長または無効なバッファ長	• BufferLength に 0 より小さい値が指定されています。 • BufferLength に無効な値（SQL_NTS）が指定されました。	○
HYT01	接続タイムアウト終了	データソースが要求に応答する前に接続タイムアウト時間が経過しました。接続タイムアウト時間は，SQLSetConnectAttr またはSQLSetConnectAttrW の SQL_ATTR_CONNECTION_TIMEOUT で設定できます。	×
IM001	ドライバはこの関数をサポートしていない	－	×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

- SQLSetCursorName 関数またはSQLSetCursorNameW 関数を使用しないで，明示的なカーソル名を設定していない場合，この関数は HADB ODBC ドライバが自動で生成するカーソル名を返します。自動で生成するカーソル名は「SQL_CUR*XXXXX*（*XXXXX*：00001 から始まる通番）」です。
- この関数で取得したカーソル名を使用する場合は，マイクロソフト社が提供するカーソルライブラリを使用してください。マイクロソフト社が提供するカーソルライブラリを使用しない場合，HADB ODBC ドライバはステートメントハンドルに関連づけられたカーソル名を無視します。

16.7.4 SQLSetCursorName，SQLSetCursorNameW

(1) 機能

カーソル名をアクティブなステートメントハンドルに関連づけます。アプリケーションがこの関数を呼び出さない場合，HADB ODBC ドライバがカーソル名を生成します。

(2) 形式

- SQLSetCursorName の場合

```
SQLRETURN SQLSetCursorName
(
  SQLHSTMT          StatementHandle,      /* In */
  SQLCHAR         * CursorName,           /* In */
  SQLSMALLINT       NameLength            /* In */
)
```

- SQLSetCursorNameW の場合

```
SQLRETURN SQLSetCursorName
(
  SQLHSTMT          StatementHandle,      /* In */
  SQLWCHAR        * CursorName,           /* In */
  SQLSMALLINT       NameLength            /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

CursorName：

関連づけるカーソルの名称を指定します。

NameLength：

*CursorName の長さを指定します（単位：SQLSetCursorName の場合はバイト長，SQLSetCursorNameW の場合は文字数）。

CursorName に指定した文字列が NULL 終端文字を保証している場合，SQL_NTS を指定できます。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		×
24000	無効なカーソル状態	StatementHandle に関連づけられたステートメントは，すでに実行状態またはカーソルが位置づけられた状態です。	○
34000	無効なカーソル名	• カーソル名のサイズに，HADB ODBC ドライバで扱える上限（30 バイト）よりも大きいサイズが指定されています。 • CursorName に指定されたカーソル名が SQL_CUR で始まっています。	○
3C000	カーソル名の重複	*CursorName に指定されたカーソル名はすでに存在しています。	○
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		×
HY009	NULL ポインタの不正使用	CursorName が NULL ポインタです。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー	メモリオブジェクトにアクセスできないため，関数の呼び出しを処理できません。	×
HY090	無効な文字列長または無効なバッファ長	NameLength に 0 以下かつSQL_NTS 以外の値が指定されています。	○
HYT01	接続タイムアウト終了	データソースが要求に応答する前に接続タイムアウト時間が経過しました。接続タイムアウト時間は，SQLSetConnectAttr ま	×

SQLSTATE	説明	備考	返却
		たはSQLSetConnectAttrW の SQL_ATTR_CONNECTION_TIMEOUT で設定できます。	
IM001	ドライバはこの関数をサポートしていない	－	×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

この関数で設定したカーソル名を使用する場合は，マイクロソフト社が提供するカーソルライブラリを使用してください。マイクロソフト社が提供するカーソルライブラリを使用しない場合，HADB ODBC ドライバは設定されたカーソル名を無視します。

16.7.5 SQLDescribeParam

(1) 機能

SQLPrepare の実行によって得られた，？パラメタの情報を返します。この情報は，IPD のフィールドに設定されている情報です。

(2) 形式

```
SQLRETURN SQLDescribeParam
(
  SQLHSTMT          StatementHandle,    /* In */
  SQLUSMALLINT      ParameterNumber,    /* In */
  SQLSMALLINT     * DataTypePtr,        /* Out */
  SQLULEN         * ParameterSizePtr,   /* Out */
  SQLSMALLINT     * DecimalDigitsPtr,   /* Out */
  SQLSMALLINT     * NullablePtr         /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

ParameterNumber：

パラメタ番号を指定します。パラメタの昇順に 1 から番号が付けられます。

DataTypePtr：

パラメタの SQL データ型を返すバッファへのポインタを指定します。

ParameterSizePtr：

対応するパラメタの列または式のサイズを返すバッファへのポインタを指定します。データソースで定義された内容が返されます。

DecimalDigitsPtr：

対応するパラメタの列または式の小数の桁数を返すバッファへのポインタを指定します。データソースで定義された内容が返されます。

NullablePtr：

パラメタがナル値を受け付けるかどうかを示す値を返すバッファへのポインタを指定します。この値は IPD のSQL_DESC_NULLABLE フィールドから読み出されます。次のどちらかの値が返されます。

- SQL_NO_NULLS
  パラメタはナル値を受け付けません。
- SQL_NULLABLE
  パラメタはナル値を受け付けます。

(4) 戻り値

SQL_SUCCESS, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	−	×
07009	無効なディスクリプタインデクス		○
08S01	通信リンク失敗		×
21S01	挿入する値の一覧と列の一覧の不一致		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.7.6 SQLNumParams

(1) 機能

準備された SQL 文のパラメタの数を返します。

(2) 形式

```
SQLRETURN SQLNumParams
(
  SQLHSTMT        StatementHandle,     /* In */
  SQLSMALLINT   * ParameterCountPtr    /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

ParameterCountPtr：

SQL 文のパラメタ数を返すバッファへのポインタを指定します。

この関数を実行する前にSQLPrepare に渡された SQL 文に含まれるパラメタの数が設定されます。

(4) 戻り値

SQL_SUCCESS, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
HY000	一般エラー		×

SQLSTATE	説明	備考	返却
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用	ParameterCountPtr にNULL が指定されています。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー		×
HY117	接続がサスペンド中		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.8　SQL の実行

ここでは，SQL の実行時に使用する ODBC 関数について説明します。

16.8.1　SQLExecute

(1)　機能

パラメタマーカーがステートメント中にある場合は，パラメタの現在の値を使用し，準備されたステートメントを実行します。

(2)　形式

```
SQLRETURN SQLExecute
(
  SQLHSTMT        StatementHandle     /* In */
)
```

(3)　引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

(4)　戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，SQL_NO_DATA，SQL_NEED_DATA または SQL_INVALID_HANDLE が返されます。

(5)　SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01001	カーソル操作の競合		×
01003	集合関数でナル値が削除された		×
01004	文字列データの右側が切り捨てられた		○
01006	特権が破棄されない		×
01007	特権が与えられない		×

SQLSTATE	説明	備考	返却
01S02	オプション値の変更		×
01S07	小数点以下切り捨て		×
07002	COUNT フィールドが不正	SQLBindParameter に指定されたパラメタの数が？パラメタの数と不一致です。	○
07006	データ型属性の制限違反	－	×
07007	制限パラメタ値違反		×
07S01	デフォルトパラメタの不正使用		×
08S01	通信リンク失敗		×
08003	接続が存在しない		○
21S02	導出したテーブルの次数が列の一覧と不一致		×
22001	文字列データの右側が切り捨てられた		○
22002	必要な標識変数が提供されない		×
22003	数値が範囲外である		○
22007	無効な日付時刻形式		○
22008	日付時刻フィールドのオーバフロー		○
22012	ゼロ除算		×
22015	間隔フィールドのオーバフロー		×
22018	キャスト指定に対する無効な文字値		○
22019	無効なエスケープ文字		×
22025	無効なエスケープシーケンス		×
23000	整合性の制約違反		×
24000	無効なカーソル状態		○
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
42000	構文エラーまたはアクセス違反		×
44000	WITH CHECK OPTION 違反		×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C036	データ変換エラー	設定要求があった入力データの指定内容が誤っています。	○
5C041	データ型未サポートエラー	ドライバは指定されたデータ型をサポートしていません。	○

SQLSTATE	説明	備考	返却
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		○
HY003	無効な C データ型		○
HY004	無効な SQL データ型		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用		○
HY010	関数シーケンスエラー	事前にPrepare が実行されていません。	○
HY013	メモリ管理エラー	－	○
HY014	無効な精度または無効なスケール値		○
HY090	無効な文字列長または無効なバッファ長		○
HY104	無効な精度または無効なスケール値		○
HY105	無効なパラメタの種類		×
HY109	無効なカーソル位置		×
HY117	接続がサスペンド中	Disconnect とr-only 関数は許されます。	×
HYC00	オプション機能は実装されていない	－	○
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

- Microsoft Access Version 2.0 に対応する処理は未サポートです。
- ブックマーク機能は未サポートです。

16.8.2 SQLExecDirect，SQLExecDirectW

(1) 機能

準備されたSQL文を実行します。SQL文にパラメタマーカーがあれば，パラメタマーカー変数の現在の値を使用して実行します。この関数は，SQL文を1回の実行で発行できる最も早い方法です。

(2) 形式

- SQLExecDirectの場合

```
SQLRETURN SQLExecDirect
(
  SQLHSTMT        StatementHandle,    /* In */
  SQLCHAR       * StatementText,      /* In */
  SQLINTEGER      TextLength          /* In */
)
```

- SQLExecDirectWの場合

```
SQLRETURN SQLExecDirectW
(
  SQLHSTMT        StatementHandle,    /* In */
  SQLWCHAR      * StatementText,      /* In */
  SQLINTEGER      TextLength          /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandleの*OutputHandlePtrで出力された値を指定します。

StatementText：

実行するSQL文字列を指定します。NULLポインタは指定できません。必ず1文字以上の文字列を指定してください。SQL文字列中に，注釈（/*～*/）は指定できませんが，インデクス指定（/*>>～<<*/）などは指定できます。

TextLength：

*StatementTextの長さを指定します。指定した値によって次の表のように扱われます。

TextLengthに指定された値	TextLengthの扱い
0より大きい整数値	*StatementTextの先頭から指定されたデータ長（単位：SQLExecDirectの場合はバイト長，SQLExecDirectWの場合は文字数）が有効になります。
SQL_NTS	TextLengthの値は無視され，*StatementTextの先頭からNULLまでが有効になります。

(4) 戻り値

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_ERROR, SQL_NEED_DATA または SQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	−	×
01001	カーソル操作の競合		×
01003	集合関数でナル値が削除された		×
01004	文字列データの右側が切り捨てられた		○
01006	特権が破棄されない		×
01007	特権が与えられない		×
01S02	オプション値の変更		×
01S07	小数点以下切り捨て		×
07002	COUNT フィールドが不正	SQLBindParameter に指定されたパラメタの数が？パラメタの数と不一致です。	○
07006	データ型属性の制限違反	−	×
07S01	デフォルトパラメタの不正使用		×
08003	接続が存在しない		○
08S01	通信リンク失敗		×
21S01	挿入する値の一覧と列の一覧の不一致		×
21S02	導出したテーブルの次数が列の一覧と不一致		×
22001	文字列データの右側が切り捨てられた		○
22002	必要な標識変数が提供されない		×
22003	数値が範囲外である		○
22007	無効な日付時刻形式		○
22008	日付時刻フィールドのオーバフロー		○
22012	ゼロ除算		×
22015	間隔フィールドのオーバフロー		×
22018	キャスト指定に対する無効な文字値		○

SQLSTATE	説明	備考	返却
22019	無効なエスケープ文字		×
22025	無効なエスケープシーケンス		×
23000	整合性の制約違反		×
24000	無効なカーソル状態		○
34000	無効なカーソル名		×
3D000	無効なカタログ名		×
3F000	無効なスキーマ名		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
42000	構文エラーまたはアクセス違反		×
42S01	ベーステーブルまたはビューがすでに存在する		×
42S02	ベーステーブルまたはビューが見つからない		×
42S11	インデクスがすでに存在する		×
42S12	インデクスが見つからない		×
42S21	列がすでに存在する		×
42S22	列が見つからない		×
44000	WITH CHECK OPTION 違反		×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C036	データ変換エラー	設定要求があった入力データの指定内容が誤っています。	○
5C041	データ型未サポートエラー	ドライバは指定されたデータ型をサポートしていません。	○
5C051	SQL 文のテキスト文字列の長さが，16,000,000 文字を超えた	SQL 文のテキスト文字列が 16,000,000 文字以下の場合でも，ドライバマネージャの文字コード変換によって 16,000,000 文字を超えることがあります。このときも，このSQLSTATE が返されます。	○
HY000	一般エラー	−	×
HY001	メモリ割り当てエラー		○
HY003	無効な C データ型		○
HY004	無効な SQL データ型		○
HY008	動作がキャンセルされた		×

SQLSTATE	説明	備考	返却
HY009	NULL ポインタの不正使用		○
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		○
HY014	無効な精度または無効なスケール値		○
HY090	無効な文字列長または無効なバッファ長		○
HY104	無効な精度または無効なスケール値		○
HY105	無効なパラメタの種類		×
HY109	無効なカーソル位置		×
HYC00	オプション機能は実装されていない		○
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

- Microsoft Access Version 2.0 に対応する処理は未サポートです。
- ブックマーク機能は未サポートです。

16.8.3 SQLNativeSql，SQLNativeSqlW

(1) 機能

HADB ODBC ドライバが修正した SQL 文字列を返却します。この関数は SQL 文を実行しません。

(2) 形式

- SQLNativeSql の場合

```
SQLRETURN SQLNativeSql
(
  SQLHDBC         ConnectionHandle,          /* In */
  SQLCHAR       * InStatementText,           /* In */
  SQLINTEGER      TextLength1,               /* In */
```

```
  SQLCHAR       * OutStatementText,         /* Out */
  SQLINTEGER      BufferLength,             /* In */
  SQLINTEGER    * TextLength2Ptr            /* Out */
)
```

- SQLNativeSqlW の場合

```
SQLRETURN SQLNativeSqlW
(
  SQLHDBC         ConnectionHandle,         /* In */
  SQLWCHAR      * InStatementText,          /* In */
  SQLINTEGER      TextLength1,              /* In */
  SQLWCHAR      * OutStatementText,         /* Out */
  SQLINTEGER      BufferLength,             /* In */
  SQLINTEGER    * TextLength2Ptr            /* Out */
)
```

(3) 引数

ConnectionHandle：

接続ハンドルを指定します。

InStatementText：

変換元の SQL 文字列を指定します。SQL 文字列中に，注釈（/*～*/）は指定できませんが，インデクス指定（/*>>～<<*/）などは指定できます。

TextLength1：

*InStatementText の長さ※を指定します。

InStatementText に指定した SQL 文字列が NULL 終端文字で終わる場合，SQL_NTS を指定できます。

OutStatementText：

変換後の SQL 文字列を返すバッファを指すポインタを指定します。

BufferLength：

*OutStatementText の長さ※を指定します。

この長さに NULL 終端文字の長さは含まれます。SQL_NTS は指定できません。

TextLength2Ptr：

*OutStatementText に返す，有効な長さ※の合計を返すバッファへのポインタを指定します。HADB ODBC ドライバは SQL 文字列の有効な長さ※を返します。この長さに NULL 終端文字は含まれません。

> **重要**
>
> ここに格納された SQL 文字列の長さ※が，BufferLength から NULL 終端文字分を引いた長さ※より大きい場合，OutStatementText に格納される文字列はBufferLength から NULL 終端文字分を引いた長さ※に切り捨てられ，末尾に NULL 終端文字が付加されます。

注※

長さの単位は，SQLNativeSql の場合はバイト長，SQLNativeSqlW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	*OutStatementText バッファの大きさが不足しているため，SQL 文字列全体を格納できませんでした（情報が切り捨てられました)。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
08003	接続が存在しない	ConnectionHandle が接続状態ではありません。	○
08S01	通信リンク失敗	－	×
22007	無効な日付時刻形式	*InStatementText のエスケープ句に，無効な日付，時刻またはタイムスタンプの値が指定されています。	×
24000	無効なカーソル状態	ステートメントが指定するカーソルが，結果セットの前または後ろにあります。	×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C051	SQL 文のテキスト文字列の長さが，16,000,000 文字を超えた	SQL 文のテキスト文字列が 16,000,000 文字以下の場合でも，ドライバマネージャの文字コード変換によって 16,000,000 文字を超えることがあります。このときも，このSQLSTATE が返されます。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		×
HY009	NULL ポインタの不正使用	*InStatementText が NULL ポインタです。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー	メモリオブジェクトにアクセスできないため，関数の呼び出しを処理できません。	×
HY090	無効な文字列長または無効なバッファ長	• TextLength1 に 0 より小さく，かつ SQL_NTS 以外の値が指定されています。	○

SQLSTATE	説明	備考	返却
		• BufferLength は 0 より小さい値ですが，OutStatementText は NULL ポインタではありません。 • BufferLength に無効な値（SQL_NTS）が指定されました。	
HY109	無効なカーソル位置	カーソルの現在行はすでに削除されているか，またはフェッチできない行です。	×
HYT01	接続タイムアウト終了	データソースが要求に応答する前に接続タイムアウト時間が経過しました。接続タイムアウト時間は，SQLSetConnectAttr またはSQLSetConnectAttrW の SQL_ATTR_CONNECTION_TIMEOUT で設定できます。	×
IM001	ドライバはこの関数をサポートしていない	－	×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) エスケープ句の構文規則

指定された SQL 文内のエスケープ句を，HADB が実行できる形式に変換して返します。エスケープ句の構文規則を次に示します。

```
エスケープ句 ::= 日付・時刻・時刻印のエスケープシーケンス
               | LIKE述語のエスケープ文字のエスケープシーケンス
               | 外結合のエスケープシーケンス
               | スカラ関数のエスケープシーケンス
日付・時刻・時刻印のエスケープシーケンス ::= 日付のエスケープシーケンス
                                          | 時刻のエスケープシーケンス
                                          | 時刻印のエスケープシーケンス
日付のエスケープシーケンス ::=
        エスケープ開始子 d 日付データの既定の文字列表現 ※1 エスケープ終了子
時刻のエスケープシーケンス ::=
        エスケープ開始子 t 時刻データの既定の文字列表現 ※2 エスケープ終了子
時刻印のエスケープシーケンス ::=
        エスケープ開始子 ts 時刻印データの既定の文字列表現 ※3 エスケープ終了子
LIKE述語のエスケープ文字のエスケープシーケンス ::=
        エスケープ開始子 escape エスケープ文字 エスケープ終了子
外結合のエスケープシーケンス ::= エスケープ開始子 oj 結合表 エスケープ終了子
スカラ関数のエスケープシーケンス ::= エスケープ開始子 fn スカラ関数 エスケープ終了子
スカラ関数 ::= 標準形式のスカラ関数 ※4
エスケープ開始子 ::= '{'
エスケープ終了子 ::= '}'
```

注※1

'*YYYY-MM-DD*' で表される文字列表現のことです。

注※2

'*hh*:*mm*:*ss*[.*nn....n*]' で表される文字列表現のことです。*nn....n* は *p* 桁の小数秒です。*n* は 0～9，*p* は 0，3，6，9，または 12 です。

注※3

'*YYYY-MM-DD hh*:*mm*:*ss*[.*nn....n*]' で表される文字列表現のことです。*nn....n* は *p* 桁の小数秒です。*n* は 0～9，*p* は 0，3，6，9，または 12 です。

注※4

標準形式のスカラ関数については，「表 16-7　標準形式と HADB 形式が異なるスカラ関数の変換内容一覧」を参照してください。

なお，下線部には，エスケープ句を指定できません。また，ODBC ドライバでは構文解析をしないで，変換後もそのままとし，HADB サーバの構文解析に任せます。

エスケープシーケンスのキーワードを次に示します。キーワードでは，大文字と小文字を区別しません。

1. 日付のエスケープシーケンス内の"d"
2. 時刻のエスケープシーケンス内の"t"
3. 時刻印のエスケープシーケンス内の"ts"
4. LIKE 述語のエスケープ文字のエスケープシーケンス内の"escape"
5. 外結合のエスケープシーケンス内の"oj"
6. スカラ関数のエスケープシーケンス内の"fn"

エスケープ句の入力規則を次に示します。

- エスケープ句内の区切り文字には，半角空白が指定できます。
- 区切り文字は，エスケープ開始子の後ろ，キーワードの後ろ，およびエスケープ終了子の前に挿入できます。
- 1 つの SQL 文中に，複数のエスケープ句を指定できます。
- HADB ODBC ドライバは指定された SQL 文内のエスケープ句を，HADB が実行できる形式に変換します。変換するのは，{　} で囲まれたエスケープ句内だけです。エスケープ句外は変換しません。

エスケープ句の変換規則を次の表に示します。

表 16-6　エスケープ句の変換規則

対象エスケープ句	変換前	変換後
日付	*エスケープ開始子* d *日付データの既定の文字列表現 エスケープ終了子*	*DATE 日付データの既定の文字列表現*

対象エスケープ句	変換前	変換後
時刻	*エスケープ開始子* t *時刻データの既定の文字列表現 エスケープ終了子*	*TIME 時刻データの既定の文字列表現*
時刻印	*エスケープ開始子* ts *時刻印データの既定の文字列表現 エスケープ終了子*	*TIMESTAMP 時刻印データの既定の文字列表現*
LIKE	*エスケープ開始子* escape *エスケープ文字 エスケープ終了子*	*escape エスケープ文字*
外結合	*エスケープ開始子* oj *結合表 エスケープ終了子*	*結合表*
スカラ関数	*エスケープ開始子* fn *スカラ関数 エスケープ終了子*	*HADB 形式のスカラ関数※*

注※

標準形式のスカラ関数を，HADB 形式に変換します。

標準形式と HADB 形式が異なるスカラ関数の変換内容を次の表に示します。

基本的に，スカラ関数の引数の個数チェックはしません。

表 16-7　標準形式と HADB 形式が異なるスカラ関数の変換内容一覧

スカラ関数	変換前の形式	変換後の形式（HADB 形式）
数学関数	CEILING(number)	CEIL(number)
	LOG(float)	LN(float)
	RAND([number, number])	RANDOM([number, number])
	TRUNCATE(number[, places])	TRUNC(number[, places])
文字列関数	CHAR(code)	CHR(code)
	LCASE(string)	LOWER(string)
	OCTET_LENGTH(string)	LENGTHB(string)
	SUBSTRING(string, start[, length])	SUBSTR(string, start[, length])
	UCASE(string)	UPPER(string)
時刻と日付の関数	CURRENT_DATE()	CURRENT_DATE
	CURRENT_TIME()	CURRENT_TIME

スカラ関数のエスケープ句内に指定した特定のスカラ関数の場合，HADB がサポートするデータ型名や日時単位以外に ODBC 規約に定められたキーワードおよび HADB が独自に定めたキーワードを指定できます。これらのキーワードを指定した場合，ODBC ドライバが HADB で使用可能な形式に変換します。スカラ関数に指定可能なキーワードを次の表に示します。

表 16-8　スカラ関数に指定可能なキーワード

スカラ関数	指定可能なキーワード
CAST	次の SQL データ型名を指定できます。

スカラ関数	指定可能なキーワード
CONVERT	• SQL_CHAR • SQL_VARCHAR • SQL_LONGVARCHAR • SQL_WCHAR • SQL_WVARCHAR • SQL_WLONGVARCHAR • SQL_DECIMAL • SQL_NUMERIC • SQL_TINYINT • SQL_SMALLINT • SQL_INTEGER • SQL_BIGINT • SQL_REAL • SQL_FLOAT • SQL_DOUBLE • SQL_BIT • SQL_BINARY • SQL_VARBINARY • SQL_LONGVARBINARY • SQL_DATE • SQL_TYPE_DATE • SQL_TIME • SQL_TYPE_TIME • SQL_TIMESTAMP • SQL_TYPE_TIMESTAMP • SQL_INTERVAL_MONTH • SQL_INTERVAL_YEAR • SQL_INTERVAL_YEAR_TO_MONTH • SQL_INTERVAL_DAY • SQL_INTERVAL_HOUR • SQL_INTERVAL_MINUTE • SQL_INTERVAL_SECOND • SQL_INTERVAL_DAY_TO_HOUR • SQL_INTERVAL_DAY_TO_MINUTE • SQL_INTERVAL_DAY_TO_SECOND • SQL_INTERVAL_HOUR_TO_MINUTE • SQL_INTERVAL_HOUR_TO_SECOND • SQL_INTERVAL_MINUTE_TO_SECOND • SQL_GUID
TIMESTAMPADD TIMESTAMPDIFF	次の日時単位キーワードを指定できます。 • SQL_TSI_DAYOFYEAR • SQL_TSI_FRAC_SECOND

スカラ関数	指定可能なキーワード
	• SQL_TSI_SECOND • SQL_TSI_MINUTE • SQL_TSI_HOUR • SQL_TSI_DAY • SQL_TSI_WEEK • SQL_TSI_MONTH • SQL_TSI_QUARTER • SQL_TSI_YEAR

メモ

スカラ関数のエスケープ句内に上記の表のキーワードを指定する場合，大文字および小文字は区別されません。

キーワードの変換規則を次の表に示します。

表 16-9　キーワードの変換規則

変換対象キーワード	変換後の形式	HADB での使用可否
SQL_CHAR	CHAR	○
SQL_VARCHAR	VARCHAR	○
SQL_LONGVARCHAR	SQL_LONGVARCHAR	×
SQL_WCHAR	SQL_WCHAR	×
SQL_WVARCHAR	SQL_WVARCHAR	×
SQL_WLONGVARCHAR	SQL_WLONGVARCHAR	×
SQL_DECIMAL	DECIMAL	○
SQL_NUMERIC	DECIMAL	○
SQL_TINYINT	SQL_TINYINT	×
SQL_SMALLINT	SQL_SMALLINT	×
SQL_INTEGER	SMALLINT	○
SQL_BIGINT	INTEGER	○
SQL_REAL	SQL_REAL	×
SQL_FLOAT	DOUBLE	○
SQL_DOUBLE	DOUBLE	○
SQL_BIT	SQL_BIT	×
SQL_BINARY	BINARY	○

変換対象キーワード	変換後の形式	HADB での使用可否
SQL_VARBINARY	VARBINARY	○
SQL_LONGVARBINARY	SQL_LONGVARBINARY	×
SQL_DATE	DATE	○
SQL_TYPE_DATE	DATE	○
SQL_TIME	TIME	○
SQL_TYPE_TIME	TIME	○
SQL_TIMESTAMP	TIMESTAMP	○
SQL_TYPE_TIMESTAMP	TIMESTAMP	○
SQL_INTERVAL_MONTH	SQL_INTERVAL_MONTH	×
SQL_INTERVAL_YEAR	SQL_INTERVAL_YEAR	×
SQL_INTERVAL_YEAR_TO_MONTH	SQL_INTERVAL_YEAR_TO_MONTH	×
SQL_INTERVAL_DAY	SQL_INTERVAL_DAY	×
SQL_INTERVAL_HOUR	SQL_INTERVAL_HOUR	×
SQL_INTERVAL_MINUTE	SQL_INTERVAL_MINUTE	×
SQL_INTERVAL_SECOND	SQL_INTERVAL_SECOND	×
SQL_INTERVAL_DAY_TO_HOUR	SQL_INTERVAL_DAY_TO_HOUR	×
SQL_INTERVAL_DAY_TO_MINUTE	SQL_INTERVAL_DAY_TO_MINUTE	×
SQL_INTERVAL_DAY_TO_SECOND	SQL_INTERVAL_DAY_TO_SECOND	×
SQL_INTERVAL_HOUR_TO_MINUTE	SQL_INTERVAL_HOUR_TO_MINUTE	×
SQL_INTERVAL_HOUR_TO_SECOND	SQL_INTERVAL_HOUR_TO_SECOND	×
SQL_INTERVAL_MINUTE_TO_SECOND	SQL_INTERVAL_MINUTE_TO_SECOND	×
SQL_GUID	SQL_GUID	×
SQL_TSI_DAYOFYEAR	DAYOFYEAR	○
SQL_TSI_FRAC_SECOND	NANOSECOND	○
SQL_TSI_SECOND	SECOND	○
SQL_TSI_MINUTE	MINUTE	○
SQL_TSI_HOUR	HOUR	○
SQL_TSI_DAY	DAY	○
SQL_TSI_WEEK	WEEK	○
SQL_TSI_MONTH	MONTH	○
SQL_TSI_QUARTER	QUARTER	○

変換対象キーワード	変換後の形式	HADB での使用可否
SQL_TSI_YEAR	YEAR	○

（凡例）

○：HADB で使用できます。

×：HADB で使用できません。

16.8.4 SQLParamData

(1) 機能

SQLPutData とこの関数を一緒に使用すると，SQL 文の実行時，SQLExecute，SQLExecDirect または SQLExecDirectW 実行後にパラメタデータを送信します。

(2) 形式

```
SQLRETURN SQLParamData
(
  SQLHSTMT        StatementHandle,    /* In */
  SQLPOINTER    * ValuePtrPtr         /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

ValuePtrPtr：

SQL 文のパラメタデータの設定先を指すポインタが返されます。

有効な値が返されるのは，戻り値がSQL_NEED_DATA のときだけです。

また，この値はSQLBindParameter のParameterValuePtr の値，またはSQLBindCol のTargetValuePtr の値と同じ値であり，ディスクリプタレコードのSQL_DESC_DATA_PTR フィールドに指定されたものと同じ値です。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_NO_DATA，SQL_NEED_DATA，SQL_ERROR，または SQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		○
07002	COUNT フィールドが不正	SQLBindParameter に指定されたパラメタの数が？パラメタの数と不一致です。	○
07006	データ型属性の制限違反	－	×
08003	接続が存在しない		○
08S01	通信リンク失敗		×
22001	文字列データの右側が切り捨てられた		○
22003	数値が範囲外である		○
22007	無効な日付時刻形式		○
22008	日付時刻フィールドのオーバフロー		○
22018	キャスト指定に対する無効な文字値		○
22026	文字列データの長さが一致しない		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C036	データ変換エラー	設定要求があった入力データの指定内容が誤っています。	○
5C041	データ型未サポートエラー	ドライバは指定されたデータ型をサポートしていません。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		○
HY003	無効な C データ型		○
HY004	無効な SQL データ型		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用	ParameterCountPtr にNULL が指定されています。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー		○
HY014	無効な精度または無効なスケール値		○

SQLSTATE	説明	備考	返却
HY090	無効な文字列長または無効なバッファ長		○
HY104	無効な精度または無効なスケール値		○
HY117	接続がサスペンド中		×
HYC00	オプション機能は実装されていない		○
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.8.5 SQLPutData

(1) 機能

SQL 文の実行時にパラメタのデータを HADB ODBC ドライバに送ります。

(2) 形式

```
SQLRETURN SQLPutData
(
  SQLHSTMT          StatementHandle,    /* In */
  SQLPOINTER        DataPtr,            /* In */
  SQLLEN            StrLen_or_Ind       /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

DataPtr：

パラメタに対する実際のデータが格納されているバッファを指すポインタを指定します。データは，SQLBindParameter のValueType で指定された C データ型である必要があります。

StrLen_or_Ind：

- SQLBindParameter でＣデータ型にSQL_C_CHAR またはSQL_C_BINARY を指定した場合

  *DataPtr の長さ，SQL_NTS，またはSQL_NULL_DATA を指定します。

- SQLBindParameter でそのほかのＣデータ型を指定した場合

  SQL_NULL_DATA を指定します。SQL_NULL_DATA 以外が指定された場合，この引数の指定は無視され，HADB ODBC ドライバは*DataPtr バッファのサイズをSQLBindParameter のValueType で指定されたＣデータ型と見なします。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字データの右側が切り捨てられた		×
07006	データ型属性の制限違反		×
07S01	デフォルトパラメタの不正使用		×
08S01	通信リンク失敗		×
22001	文字列データの右側が切り捨てられた		×
22003	数値が範囲外である		×
22007	無効な日付時刻形式		×
22008	日付時刻フィールドのオーバフロー		×
22012	ゼロ除算		×
22015	間隔フィールドのオーバフロー		×
22018	キャスト指定に対する無効な文字値		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用	DataPtr に NULL ポインタが指定されていますが，StrLen_or_Ind には0，またはSQL_NULL_DATA 以外が指定されています。	○
HY010	関数シーケンスエラー	SQLParamData で入力待ち実行時のデータパラメタが取得されていません。	○

SQLSTATE	説明	備考	返却
HY013	メモリ管理エラー	－	×
HY019	文字データとバイナリデータ以外のデータが分割送信された		×
HY020	ナル値の連結を試みた		×
HY090	無効な文字列長または無効なバッファ長	次の条件をすべて満たしています。 • DataPtr に NULL ポインタ以外が指定されています。 • StrLen_or_Ind に0 以下の値が指定されています。 • StrLen_or_Ind にSQL_NTS または SQL_NULL_DATA 以外の値が指定されています。	○
HY117	接続がサスペンド中	－	×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

次の機能はサポートしていません。

- 列のデータ対応
- 分割送信

16.9　実行結果および実行結果情報の取得

ここでは，実行結果および実行結果情報の取得時に使用する ODBC 関数について説明します。

16.9.1　SQLRowCount

(1)　機能

この関数を実行する前に行われた次の処理で変更された行数を返します。

- 各種 SQL 文（UPDATE 文，INSERT 文およびDELETE 文）

(2)　形式

```
SQLRETURN SQLRowCount
(
  SQLHSTMT       StatementHandle,     /* In */
  SQLLEN       * RowCountPtr          /* Out */
)
```

(3)　引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

RowCountPtr：

変更された行カウントを返すバッファを指すポインタを指定します。

(4)　戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5)　SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×

SQLSTATE	説明	備考	返却
HY009	NULL ポインタの不正使用	RowCountPtr に無効な値が設定されました。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー		×
HY117	接続がサスペンド中		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

実行した SQL 文がUPDATE 文，INSERT 文またはDELETE 文のどれかの場合，要求によって変更された行数を返します。ただし，次に示す場合は，-1 を返します。

- 変更した行数にオーバフローが発生した場合
- 実行した SQL 文が，UPDATE 文，INSERT 文，またはDELETE 文以外の場合

16.9.2 SQLNumResultCols

(1) 機能

準備された SQL 文の結果セットの列数を返します。

(2) 形式

```
SQLRETURN SQLNumResultCols
(
  SQLHSTMT        StatementHandle,     /* In */
  SQLSMALLINT   * ColumnCountPtr       /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

ColumnCountPtr：

結果セットの列数を返すバッファを指すポインタを指定します。

(4) 戻り値

SQL_SUCCESS, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用	ColumnCountPtr にNULL が指定されています。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー		×
HY117	接続がサスペンド中		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.9.3 SQLDescribeCol，SQLDescribeColW

(1) 機能

SQLPrepare の実行によって得られた，結果セットの列の情報を返します。この情報は，IRD のフィールドに設定されているものです。

(2) 形式

- SQLDescribeCol の場合

```
SQLRETURN SQLDescribeCol
(
  SQLHSTMT         StatementHandle,     /* In */
  SQLUSMALLINT     ColumnNumber,        /* In */
  SQLCHAR        * ColumnName,          /* Out */
  SQLSMALLINT      BufferLength,        /* In */
  SQLSMALLINT    * NameLengthPtr,       /* Out */
  SQLSMALLINT    * DataTypePtr,         /* Out */
  SQLULEN        * ColumnSizePtr,       /* Out */
  SQLSMALLINT    * DecimalDigitsPtr,    /* Out */
  SQLSMALLINT    * NullablePtr          /* Out */
)
```

- SQLDescribeColW の場合

```
SQLRETURN SQLDescribeColW
(
  SQLHSTMT         StatementHandle,     /* In */
  SQLUSMALLINT     ColumnNumber,        /* In */
  SQLWCHAR       * ColumnName,          /* Out */
  SQLSMALLINT      BufferLength,        /* In */
  SQLSMALLINT    * NameLengthPtr,       /* Out */
  SQLSMALLINT    * DataTypePtr,         /* Out */
  SQLULEN        * ColumnSizePtr,       /* Out */
  SQLSMALLINT    * DecimalDigitsPtr,    /* Out */
  SQLSMALLINT    * NullablePtr          /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

ColumnNumber：

結果データの列番号を指定します。

ColumnName：

列名を返すバッファへのポインタを指定します。検索結果列の名称については，マニュアル『HADB SQL リファレンス』の『SELECT 文の指定形式および規則』の『規則』を参照してください。

BufferLength：

*ColumnName バッファの長さ※を指定します。この長さに NULL 終端文字は含まれます。SQL_NTS は指定できません。

NameLengthPtr：

*ColumnName に設定される値の有効な長さ※の合計を格納するバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

> **重要**
>
> ここに格納された*ColumnName の列名の長さ※が，BufferLength から NULL 終端文字分を引いた長さ※より大きい場合，ColumnName に格納される文字列はBufferLength から NULL 終端文字分を引いた長さ※に切り捨てられ，末尾に NULL 終端文字が付加されます。

DataTypePtr：

列の SQL データ型を返すバッファへのポインタを指定します。データ型が不明な場合，SQL_C_DEFAULT が返されます。

ColumnSizePtr：

データソースの列のサイズを返すバッファへのポインタを指定します。列のサイズが不明な場合，0 が返されます。

DecimalDigitsPtr：

データソースの列の小数の桁数を返すバッファへのポインタを指定します。小数の桁数が不明，または適用されない場合，0 が返されます。

NullablePtr：

列がナル値を受け付けるかどうかを示す値を返すバッファへのポインタを指定します。次の値が返されます。

- SQL_NO_NULLS

  列はナル値を受け付けません。

- SQL_NULLABLE

  列はナル値を受け付けます。

注※

長さの単位は，SQLDescribeCol の場合はバイト長，SQLDescribeColW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		○
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
07005	準備されたステートメントが cursor-specification ではない	－	○
07009	無効なディスクリプタインデクス		○
08S01	通信リンク失敗		×
24000	無効なカーソル状態		○
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	BufferLength に無効な値（SQL_NTS）が指定されました。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.9.4 SQLColAttribute，SQLColAttributeW

(1) 機能

結果セットの列のディスクリプタ情報を返します。

(2) 形式

- SQLColAttribute の場合

```
SQLRETURN SQLColAttribute
(
```

```
  SQLHSTMT          StatementHandle,        /* In */
  SQLUSMALLINT      ColumnNumber,           /* In */
  SQLUSMALLINT      FieldIdentifier,        /* In */
  SQLPOINTER        CharacterAttributePtr,  /* Out */
  SQLSMALLINT       BufferLength,           /* In */
  SQLSMALLINT     * StringLengthPtr,        /* Out */
  SQLLEN          * NumericAttributePtr     /* Out */
)
```

- SQLColAttributeW の場合

```
SQLRETURN SQLColAttributeW
(
  SQLHSTMT          StatementHandle,        /* In */
  SQLUSMALLINT      ColumnNumber,           /* In */
  SQLUSMALLINT      FieldIdentifier,        /* In */
  SQLPOINTER        CharacterAttributePtr,  /* Out */
  SQLSMALLINT       BufferLength,           /* In */
  SQLSMALLINT     * StringLengthPtr,        /* Out */
  SQLLEN          * NumericAttributePtr     /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

ColumnNumber：

フィールド値を取得する IRD のレコード番号を指定します。この番号は 1 から列の昇順で番号が付けられていて，結果セットの列番号に対応しています。

FieldIdentifier：

取得する IRD のディスクリプタフィールドに対応する識別子を指定します。

指定できるフィールド識別子を次の表に示します。

表 16-10 SQLColAttribute および SQLColAttributeW の FieldIdentifier に指定できるフィールド識別子

項番	フィールド識別子
1	SQL_COLUMN_LENGTH
2	SQL_COLUMN_PRECISION
3	SQL_COLUMN_SCALE
4	SQL_DESC_AUTO_UNIQUE_VALUE
5	SQL_DESC_BASE_COLUMN_NAME
6	SQL_DESC_BASE_TABLE_NAME
7	SQL_DESC_CASE_SENSITIVE
8	SQL_DESC_CATALOG_NAME

項番	フィールド識別子
9	SQL_DESC_CONCISE_TYPE
10	SQL_DESC_COUNT
11	SQL_DESC_DISPLAY_SIZE
12	SQL_DESC_FIXED_PREC_SCALE
13	SQL_DESC_LABEL
14	SQL_DESC_LENGTH
15	SQL_DESC_LITERAL_PREFIX
16	SQL_DESC_LITERAL_SUFFIX
17	SQL_DESC_LOCAL_TYPE_NAME
18	SQL_DESC_NAME
19	SQL_DESC_NULLABLE
20	SQL_DESC_NUM_PREC_RADIX
21	SQL_DESC_OCTET_LENGTH
22	SQL_DESC_PRECISION
23	SQL_DESC_SCALE
24	SQL_DESC_SCHEMA_NAME
25	SQL_DESC_SEARCHABLE
26	SQL_DESC_TABLE_NAME
27	SQL_DESC_TYPE
28	SQL_DESC_TYPE_NAME
29	SQL_DESC_UNNAMED
30	SQL_DESC_UNSIGNED
31	SQL_DESC_UPDATABLE

CharacterAttributePtr：

FieldIdentifier に対応する IRD のディスクリプタフィールド値を返すバッファへのポインタを指定します。ディスクリプタフィールド値が文字列以外の場合，この引数は使用されません。

BufferLength：

*CharacterAttributePtr の長さを指定します（単位：バイト長）。SQL_NTS は指定できません。

StringLengthPtr：

*CharacterAttributePtr に返す，有効な総バイト長（NULL 終端文字を除く）を返すバッファへのポインタを指定します。

NumericAttributePtr：

FieldIdentifier に対応する IRD のディスクリプタフィールド値を返す整数バッファへのポインタを指定してください。ディスクリプタフィールド値が数値以外の場合は，この引数は使用されません。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	*CharacterAttributePtr バッファの大きさが不足しているため，すべての文字列値を格納できませんでした（文字列値が切り捨てられました）。切り捨てられる前の文字列値の長さが，*StringLengthPtr に格納されます。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
07005	準備されたステートメントが cursor-specification ではない	StatementHandle に関連づけられたステートメントが結果セットを返しません。	○
07009	無効なディスクリプタインデクス	• ColumnNumber に0 が指定されました。 • ColumnNumber に結果セットの列数よりも大きい値が指定されました。	○
24000	無効なカーソル状態	－	○
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY010	関数シーケンスエラー		×
HY013	メモリ管理エラー	メモリオブジェクトにアクセスできないため，関数の呼び出しを処理できません。	×
HY090	無効な文字列長または無効なバッファ長	• CharacterAttributePtr に文字列を指すポインタが指定されていて，BufferLength に 0 以下の値が指定されています。 • BufferLength に無効な値が指定されました（SQL_NTS）。	○

SQLSTATE	説明	備考	返却
HY091	無効なディスクリプタフィールド識別子	FieldIdentifier に指定された値が，定義された値でもなく，実装時に定義された値でもありません。	○
HYC00	オプション機能は実装されていない	FieldIdentifier に指定された値はサポートしていません。	○
HYT01	接続タイムアウト終了	データソースが要求に応答する前に接続タイムアウト時間が経過しました。接続タイムアウト時間は，SQLSetConnectAttr またはSQLSetConnectAttrW の SQL_ATTR_CONNECTION_TIMEOUT で設定できます。	×
IM001	ドライバはこの関数をサポートしていない	－	×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.9.5 SQLBindCol

(1) 機能

アプリケーションデータ領域を結果セットの列に関連づけます。

(2) 形式

```
SQLRETURN SQLBindCol
(
  SQLHSTMT        StatementHandle,        /* In */
  SQLUSMALLINT    ColumnNumber,           /* In */
  SQLSMALLINT     TargetType,             /* In */
  SQLPOINTER      TargetValuePtr,         /* Out */
  SQLLEN          BufferLength,           /* In */
  SQLLEN        * StrLen_or_IndPtr        /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

ColumnNumber：

関連づける結果セットの列の番号を指定します。

HADB はブックマークをサポートしていないため，列番号は 1 から始まります。

TargetType：

TargetValuePtr が指す領域の C データ型の識別子またはSQL_C_DEFAULT を指定します。SQL_C_DEFAULT を指定すると，デフォルトの C データ型を仮定します。

TargetValuePtr：

結果セットの列に関連づける領域を指すポインタを指定します。

関連づけを解除する場合は，NULL ポインタを指定してください。

BufferLength：

TargetValuePtr が指す領域の長さを指定します（単位：バイト長)。

StrLen_or_IndPtr：

HADB ODBC ドライバが返すデータの長さまたは標識を格納する領域を指すポインタを指定します。

実行後のSQLFetch 関数がSQL_SUCCESS またはSQL_SUCCESS_WITH_INFO を返す場合，HADB ODBC ドライバはデータの長さ，または標識を返します。

また，列のデータがナル値の場合，戻り値にSQL_NULL_DATA を返します。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
07006	データ型属性の制限違反	ColumnNumber に0 が指定されていますが，TargetType にSQL_C_BOOKMARK またはSQL_C_VARBOOKMARK 以外が指定されています。	×
07009	無効なディスクリプタインデクス	ColumnNumber に，結果セットの最大列数よりも大きい値が指定されています。	×
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		○
HY003	無効なアプリケーションのバッファのデータ型	TargetType に有効なデータ型またはSQL_C_DEFAULT 以外が指定されています。	×

SQLSTATE	説明	備考	返却
HY010	関数シーケンスエラー	StatementHandle に対して呼び出された非同期実行関数が，この関数が呼び出されたときも実行中です。	×
HY013	メモリ管理エラー	メモリオブジェクトにアクセスできないため，関数の呼び出しを処理できません。	×
HY090	無効な文字列長または無効なバッファ長	次の条件をすべて満たしています。 • TargetValuePtr に NULL ポインタ以外が指定されています。 • BufferLength に 0 未満の値が指定されています。	○
HYC00	オプション機能は実装されていない	ColumnNumber に0 が指定されましたが，ブックマークはサポートしていません。	○
HYT01	接続タイムアウト終了	データソースが要求に応答する前に接続タイムアウト時間が経過しました。接続タイムアウト時間は，SQLSetConnectAttr またはSQLSetConnectAttrW の SQL_ATTR_CONNECTION_TIMEOUT で設定できます。	×
IM001	ドライバはこの関数をサポートしていない	－	×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

次の機能はサポートしていません。

- バインドオフセット
- 配列のバインド
- 行方向バインド

16.9.6 SQLFetch

(1) 機能

結果セットから次の行セットのデータをフェッチし，SQLBindCol で関連づけられたすべての列にデータを返します。

(2) 形式

```
SQLRETURN SQLFetch
(
  SQLHSTMT          StatementHandle    /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_NO_DATA，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	列に返された文字列またはバイナリのデータで，空白以外の文字またはNULL 以外のバイナリデータが切り捨てられました。文字列の値は，その右側が切り捨てられました。	○
01S01	行のエラー	1 つ以上の行のフェッチでエラーが発生しました。	×
01S07	小数点以下切り捨て	数値の小数部分が切り捨てられました。時刻部分を含む時刻，タイムスタンプまたは間隔データ型の場合，時刻の小数部分が切り捨てられました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
07006	データ型属性の制限違反	結果セットの列のデータ値をSQLBindCol の TargetType で指定されたデータ型に変換できません。	○
07009	無効なディスクリプタインデクス	－	○
08S01	通信リンク失敗		×
22001	文字列データの右側が切り捨てられた		×

SQLSTATE	説明	備考	返却
22002	必要な標識変数が提供されない	次のどちらかが NULL ポインタである列に，NULL データがフェッチされました。 • SQLBindCol によって設定された StrLen_or_IndPtr • SQLSetDescField，SQLSetDescFieldW，SQLSetDescRec またはSQLSetDescRecW のどれかによって設定された SQL_DESC_INDICATOR_PTR	○
22003	数値が範囲外である	数値（数値または文字列）の整数部分が削除されました。	○
22007	無効な日付時刻形式	文字の列が日付，時刻またはタイムスタンプの C の構造体にバインドされましたが，列の値が無効な日付，時刻，またはタイムスタンプです。	○
22012	ゼロ除算	ゼロ除算が行われた算術式の値が返されました。	×
22015	間隔フィールドのオーバフロー	－	×
22018	キャスト指定に対する無効な文字値		○
24000	無効なカーソル状態	StatementHandle は実行状態ですが，関連する結果セットがありません。	○
40001	直列化の失敗	－	×
40003	ステートメントの完了が不明		×
5C002	文字コード変換エラー		○
5C037	データフォーマットエラー		○
5C038	データ変換エラー	取得した結果データまたは受け取り領域の指定内容のどちらかに誤りがあります。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー	HADB ODBC ドライバは，関数の実行または完了をサポートする必要なメモリを割り当てられていません。	×
HY003	無効なアプリケーションのバッファのデータ型	SQLBindCol によって設定された C データ型が不正なデータ型です。	○
HY008	動作がキャンセルされた	－	×
HY009	NULL ポインタの不正使用		○
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		○
HY090	無効な文字列長または無効なバッファ長		○

SQLSTATE	説明	備考	返却
HY104	無効な精度または無効なスケール値		○
HY107	行の値が範囲外である		×
HYC00	オプション機能は実装されていない	HADB ODBC ドライバまたは HADB サーバは，SQLBindCol のTargetType と対応する列の SQL データ型の組み合わせで指定される変換をサポートしていません。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

行ステータス配列に対して行ステータスを返却する機能はサポートしていません。

16.9.7 SQLFetchScroll

(1) 機能

結果セットから指定された行セットのデータをフェッチし，バインドされたすべての列のデータを返します。

ただし，HADB ではSQLSTATE がHYC00 で，SQL_ERROR を返します。

(2) 形式

```
SQLRETURN SQLFetchScroll
(
  SQLHSTMT        StatementHandle,          /* In */
  SQLSMALLINT     FetchOrientation,         /* In */
  SQLLEN          FetchOffset               /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

FetchOrientation：

この引数の指定は無視されます。

FetchOffset：

この引数の指定は無視されます。

(4) 戻り値

SQL_ERROR またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		×
01S01	行のエラー		×
01S06	結果セットが最初の行セットを返す前にフェッチを試みた		×
01S07	小数点以下切り捨て		×
07006	データ型属性の制限違反		×
07009	無効なディスクリプタインデクス		×
08S01	通信リンク失敗		×
22001	文字列データの右側が切り捨てられた		×
22002	必要な標識変数が提供されない		×
22003	数値が範囲外である		×
22007	無効な日付時刻形式		×
22012	ゼロ除算		×
22015	間隔フィールドのオーバフロー		×
22018	キャスト指定に対する無効な文字値		×
24000	無効なカーソル状態		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×

SQLSTATE	説明	備考	返却
HY008	動作がキャンセルされた		×
HY010	関数シーケンスエラー		×
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY106	フェッチの種類が範囲外である		×
HY107	行の値が範囲外である		×
HY111	無効なブックマーク値		×
HYC00	オプション機能は実装されていない	SQL_ERROR を返します。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.9.8 SQLGetData

(1) 機能

結果セット中の単一列のデータを取得します。

(2) 形式

```
SQLRETURN SQLGetData
(
  SQLHSTMT        StatementHandle,        /* In */
  SQLUSMALLINT    ColumnNumber,           /* In */
  SQLSMALLINT     TargetType,             /* In */
  SQLPOINTER      TargetValuePtr,         /* Out */
  SQLLEN          BufferLength,           /* In */
  SQLLEN        * StrLen_or_IndPtr        /* Out */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

ColumnNumber：

データを取得する列の番号を指定します。

HADB はブックマークをサポートしていないため，列番号は 1 から始まります。

TargetType：

TargetValuePtr が指す領域の C データ型の識別子またはSQL_C_DEFAULT を指定します。SQL_C_DEFAULT を指定すると，デフォルトの C データ型を仮定します。

TargetValuePtr：

列のデータを受け取る領域を指すポインタを指定します。

BufferLength：

TargetValuePtr が指す領域の長さを指定します（単位：バイト長)。

StrLen_or_IndPtr：

HADB ODBC ドライバが返すデータの長さまたは標識を格納する領域を指すポインタを指定します。

HADB ODBC ドライバはデータの長さ，または標識を返します。

また，列のデータがナル値の場合，戻り値にSQL_NULL_DATA を返します。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_NO_DATA，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた	TargetValuePtr バッファが指すバッファの大きさが，ColumnNumber で指定された列のデータを格納するのに十分ではありません。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S07	小数点以下切り捨て	数値の小数部分が切り捨てられました。時刻部分を含む時刻，タイムスタンプまたは間隔データ型の場合，時刻の小数部分が切り捨てられました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
01S51	文字コード変換時に置き換えが発生した	変換できない文字コードを検出し，指定の文字に置き換えました。このとき，SQL_SUCCESS_WITH_INFO を返します。	○
07006	データ型属性の制限違反	結果セットの列のデータ値をTargetType で指定される C データ型に変換できません。	○

SQLSTATE	説明	備考	返却
07009	無効なディスクリプタインデクス	ColumnNumber に存在しない列番号が指定されました。	○
08S01	通信リンク失敗	－	×
22002	必要な標識変数が提供されない	StrLen_or_IndPtr が NULL ポインタで，NULL データが取得されました。	○
22003	数値が範囲外である	数値（数値または文字列）の整数部分が削除されました。	×
22007	無効な日付時刻形式	文字の列が日付，時刻またはタイムスタンプの C の構造体にバインドされましたが，列の値が無効な日付，時刻，またはタイムスタンプです。	○
22012	ゼロ除算	ゼロ除算が行われた算術式の値が返されました。	×
22015	間隔フィールドのオーバフロー	－	×
22018	キャスト指定に対する無効な文字値		○
24000	無効なカーソル状態	StatementHandle 上でカーソルがオープンされ，SQLFetch が呼び出されましたが，カーソルが位置づけられているのは結果セットの先頭の前，または結果セットの末尾です。	○
5C002	文字コード変換エラー	－	○
5C037	データフォーマットエラー		○
5C038	データ変換エラー	取得した結果データまたは受け取り領域の指定内容のどちらかに誤りがあります。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー	HADB ODBC ドライバは，関数の実行または完了をサポートするために必要なメモリを割り当てられていません。	○
HY003	無効なアプリケーションのバッファのデータ型	－	○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用	TargetValuePtr とStrLen_or_IndPtr の両方に NULL ポインタが指定されました。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー		○
HY090	無効な文字列長または無効なバッファ長	次の条件をすべて満たしています。 • TargetValuePtr に NULL ポインタ以外が指定されています。	○

SQLSTATE	説明	備考	返却
		• BufferLength に 0 未満の値が指定されています。	
HY104	無効な精度または無効なスケール値	－	○
HY109	無効なカーソル位置		×
HYC00	オプション機能は実装されていない	TargetType と対応する列の SQL データ型の組み合わせで指定される変換をサポートしていません。	○
HYT00	タイムアウト終了	－	×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

- SQLGetData 関数で分割取得機能が有効になるデータ型の組み合わせを次の表に示します。次の表で示すデータ型の組み合わせ以外は，分割取得機能は無効になります。

表 16-11 SQLGetData 関数で分割取得機能が有効になるデータ型の組み合わせ

HADB のデータ型	C データ型
CHAR VARCHAR BINARY VARBINARY	SQL_C_CHAR SQL_C_WCHAR SQL_C_BINARY

なお，上記のデータ型の組み合わせの場合でも分割取得機能を無効にしたいときは，環境変数ADBODBSGDSTにNOUSE を指定してください。環境変数については，「4.3 環境変数の設定」を参照してください。

分割取得機能の詳細については，MSDN のドキュメントの『SQLGetData 関数』の『Retrieving Variable-Length Data in Parts』を参照してください。

- 次の機能はサポートしていません。
  - データの最大返却長制限

16.9.9 SQLSetPos

(1) 機能

行セット内のカーソル位置を設定します。

ただし，HADB ではSQLSTATE がHYC00 で，SQL_ERROR を返します。

(2) 形式

```
SQLRETURN SQLSetPos
(
  SQLHSTMT        StatementHandle,      /* In */
  SQLSETPOSIROW   RowNumber,            /* In */
  SQLUSMALLINT    Operation,            /* In */
  SQLUSMALLINT    LockType              /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

RowNumber：

この引数の指定は無視されます。

Operation：

この引数の指定は無視されます。

LockType：

この引数の指定は無視されます。

(4) 戻り値

SQL_ERROR またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01001	カーソル操作の競合		×
01004	文字列データの右側が切り捨てられた		×
01S01	行のエラー		×
01S07	小数点以下切り捨て		×
07006	データ型属性の制限違反		×
07009	無効なディスクリプタインデクス		×

SQLSTATE	説明	備考	返却
21S02	導出したテーブルの次数が列の一覧と不一致		×
22001	文字列データの右側が切り捨てられた		×
22003	数値が範囲外である		×
22007	無効な日付時刻形式		×
22008	日付時刻フィールドのオーバフロー		×
22015	間隔フィールドのオーバフロー		×
22018	キャスト指定に対する無効な文字値		×
23000	整合性の制約違反		×
24000	無効なカーソル状態		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
42000	構文エラーまたはアクセス違反		×
44000	WITH CHECK OPTION 違反		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY010	関数シーケンスエラー		×
HY011	ここでは属性を設定できない		×
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY092	無効な属性識別子，または無効なオプション識別子		×
HY107	行の値が範囲外である		×
HY109	無効なカーソル位置		×
HYC00	オプション機能は実装されていない	SQL_ERROR を返します。	○
HYT00	タイムアウト終了	－	×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.9.10 SQLBulkOperations

(1) 機能

一括挿入処理と，ブックマークによる一括ブックマーク処理を行います。一括ブックマーク処理には更新，削除，およびフェッチを含みます。

ただし，HADB ではSQLSTATE がHYC00 で，SQL_ERROR を返します。

(2) 形式

```
SQLRETURN SQLBulkOperations
(
  SQLHSTMT        StatementHandle,         /* In */
  SQLSMALLINT     Operation                /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

Operation：

この引数の指定は無視されます。

(4) 戻り値

SQL_ERROR またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01004	文字列データの右側が切り捨てられた		×
01S01	行のエラー		×
01S07	小数点以下切り捨て		×

SQLSTATE	説明	備考	返却
07006	データ型属性の制限違反		×
07009	無効なディスクリプタインデクス		×
21S02	導出したテーブルの次数が列の一覧と不一致		×
22001	文字列データの右側が切り捨てられた		×
22003	数値が範囲外である		×
22007	無効な日付時刻形式		×
22008	日付時刻フィールドのオーバフロー		×
22015	間隔フィールドのオーバフロー		×
22018	キャスト指定に対する無効な文字値		×
23000	整合性の制約違反		×
24000	無効なカーソル状態		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
42000	構文エラーまたはアクセス違反		×
44000	WITH CHECK OPTION 違反		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY010	関数シーケンスエラー		×
HY011	ここでは属性を設定できない		×
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY092	無効な属性識別子，または無効なオプション識別子		×
HYC00	オプション機能は実装されていない	SQL_ERROR を返します。	○
HYT00	タイムアウト終了	－	×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.9.11　SQLMoreResults

(1)　機能

SELECT 文，UPDATE 文，INSERT 文およびDELETE 文実行時の結果を初期化します。

ただし，HADB ではSQL_NO_DATA を返します。

(2)　形式

```
SQLRETURN SQLMoreResults
(
  SQLHSTMT          StatementHandle       /* In */
)
```

(3)　引数

StatementHandle：

ステートメントハンドルを指定します。

(4)　戻り値

SQL_NO_DATA，SQL_ERROR またはSQL_INVALID_HANDLE が返されます。

(5)　SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01S02	オプション値の変更		×
08S01	通信リンク失敗		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×

SQLSTATE	説明	備考	返却
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.9.12 SQLGetDiagField，SQLGetDiagFieldW

(1) 機能

ハンドルに関連づけられているエラー，警告，ステータスなどの情報を含む診断データ構造体のフィールドに現在設定されている値を返します。

(2) 形式

- SQLGetDiagField の場合

```
SQLRETURN SQLGetDiagField
(
  SQLSMALLINT     HandleType,          /* In */
  SQLHANDLE       Handle,              /* In */
  SQLSMALLINT     RecNumber,           /* In */
  SQLSMALLINT     DiagIdentifier,      /* In */
  SQLPOINTER      DiagInfoPtr,         /* Out */
  SQLSMALLINT     BufferLength,        /* In */
  SQLSMALLINT   * StringLengthPtr      /* Out */
)
```

- SQLGetDiagFieldW の場合

```
SQLRETURN SQLGetDiagFieldW
(
  SQLSMALLINT     HandleType,          /* In */
  SQLHANDLE       Handle,              /* In */
  SQLSMALLINT     RecNumber,           /* In */
  SQLSMALLINT     DiagIdentifier,      /* In */
  SQLPOINTER      DiagInfoPtr,         /* Out */
  SQLSMALLINT     BufferLength,        /* In */
  SQLSMALLINT   * StringLengthPtr      /* Out */
)
```

(3) 引数

HandleType：

次のどれかのハンドルの種類を指定します。

- SQL_HANDLE_ENV：環境ハンドル
- SQL_HANDLE_DBC：接続ハンドル
- SQL_HANDLE_STMT：ステートメントハンドル
- SQL_HANDLE_DESC：ディスクリプタハンドル

Handle：

ハンドルの値を指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

RecNumber：

アプリケーションが情報を取得する診断情報（ステータスレコード）番号を指定します。

診断ヘッダフィールドの値を取得する（DiagIdentifier に診断ヘッダフィールドを示す値を設定する）場合，この引数の指定は無視されます。

それ以外の場合，この引数には1 以上の値を指定します。

DiagIdentifier：

要求診断フィールド識別子を指定します。大きく分けてヘッダフィールドとレコードフィールドの 2 種類があります。指定できる属性については，「16.18　SQLGetDiagField および SQLGetDiagFieldW の DiagIdentifier に指定できる属性」を参照してください。

DiagInfoPtr：

診断情報を返すバッファへのポインタを返します。データ型はDiagIdentifier の値によって異なります。

BufferLength：

DiagInfoPtr の長さを指定します。

この長さに NULL 終端文字は含まれます。

DiagInfoPtr の種類によって次の値を指定します。

DiagIdentifier の値の種類	DiagInfoPtr の値の種類	BufferLength に指定する値
「16.18　SQLGetDiagField および SQLGetDiagFieldW の DiagIdentifier に指定できる属性」に定義されている値	文字列またはバイナリ	DiagInfoPtr の長さ（単位：バイト長） SQL_NTS は指定できません。
	整数	なし（無視される）

StringLengthPtr：

DiagInfoPtr が文字データの場合だけ使用します。

DiagInfoPtr に返す有効な総バイト長を格納するバッファへのポインタを指定します。この総バイト長に NULL 終端文字のバイト長は含みません。

重要

ここに格納されたDiagInfoPtr に設定する文字列の総バイト長が，BufferLength から NULL 終端文字分を引いた長さより大きい場合，DiagInfoPtr に格納される文字列はBufferLength から NULL 終端文字分を引いた長さに切り捨てられ，末尾に NULL 終端文字が付加されます。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，SQL_INVALID_HANDLE，またはSQL_NO_DATA が返されます。この関数ではSQLSTATE を設定しない代わりに，戻り値に次の実行結果を示します。

戻り値	意味
SQL_SUCCESS	成功
SQL_SUCCESS_WITH_INFO	次のエラーのうちのどちらかが発生しました。 • 取得値に対してDiagInfoPtr のサイズが小さいため，切り捨てが発生しました。BufferLength とStringLengthPtr を比較することで，切り捨てられたサイズを知ることができます。 • 文字コード変換時に変換できない文字コードを検出し，指定の文字に置き換えました。
SQL_ERROR	次のエラーのうちのどれかが発生しました。 • DiagIdentifier に不正な値が指定されました。 • DiagIdentifier に次のうちのどれかが指定されましたが，Handle がステートメントハンドルではありません。 ・SQL_DIAG_CURSOR_ROW_COUNT ・SQL_DIAG_DYNAMIC_FUNCTION ・SQL_DIAG_DYNAMIC_FUNCTION_CODE ・SQL_DIAG_ROW_COUNT • DiagIdentifier に診断レコードフィールドを示す値が指定されましたが，RecNumber に0 以下の値が指定されました。 • 要求フィールドのデータ型種別は文字列ですが，BufferLength が次のすべての条件を満たしています。 ・0 より小さい ・SQL_NTS ではない ・SQL_LEN_BINARY_ATTR(length)マクロの結果ではない • BufferLength に無効な値（SQL_NTS）が指定されました。
SQL_INVALID_HANDLE	HandleType とHandle によって示されるハンドルが有効なハンドルではありません。
SQL_NO_DATA	RecNumber がHandle で指定されたハンドルに存在する診断レコードの数より大きいか，指定されたHandle に読み出しできる診断レコードがありません。

(5) SQLSTATE

この関数ではSQLSTATE を返しません。

エラーの詳細は，「(4)　戻り値」に定義している値で通知します。

(6)　注意事項

- エラーの詳細情報については，「15.6　エラー発生時に返却される情報」を参照してください。
- 診断情報は返しません。

16.9.13　SQLGetDiagRec，SQLGetDiagRecW

(1)　機能

ハンドルに関連づけられているエラー，警告，ステータスなどの情報を含む診断データ構造体のレコードフィールドに現在設定されている値を返します。

(2)　形式

- SQLGetDiagRec の場合

```
SQLRETURN SQLGetDiagRec
(
  SQLSMALLINT     HandleType,         /* In */
  SQLHANDLE       Handle,             /* In */
  SQLSMALLINT     RecNumber,          /* In */
  SQLCHAR       * SQLState,           /* Out */
  SQLINTEGER    * NativeErrorPtr,     /* Out */
  SQLCHAR       * MessageText,        /* Out */
  SQLSMALLINT     BufferLength,       /* In */
  SQLSMALLINT   * TextLengthPtr       /* Out */
)
```

- SQLGetDiagRecW の場合

```
SQLRETURN SQLGetDiagRecW
(
  SQLSMALLINT     HandleType,         /* In */
  SQLHANDLE       Handle,             /* In */
  SQLSMALLINT     RecNumber,          /* In */
  SQLWCHAR      * SQLState,           /* Out */
  SQLINTEGER    * NativeErrorPtr,     /* Out */
  SQLWCHAR      * MessageText,        /* Out */
  SQLSMALLINT     BufferLength,       /* In */
  SQLSMALLINT   * TextLengthPtr       /* Out */
)
```

(3)　引数

HandleType：

次のどれかのハンドルの種類を指定します。

- SQL_HANDLE_ENV：環境ハンドル
- SQL_HANDLE_DBC：接続ハンドル
- SQL_HANDLE_STMT：ステートメントハンドル
- SQL_HANDLE_DESC：ディスクリプタハンドル

Handle：

ハンドルの値を指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

RecNumber：

アプリケーションが情報を取得する診断情報（ステータスレコード）番号を指定します。

この引数には1 以上の値を指定します。

SQLState：

RecNumber が示す診断レコードに関係するSQLSTATE コードを返すバッファへのポインタです。「クラス(2 文字）＋サブクラス（3 文字)」の 5 文字で構成されます。

この情報はSQL_DIAG_SQLSTATE 診断フィールドに格納されている情報が返されます。

このパラメタにNULL が指定された場合は何も設定しません。

NativeErrorPtr：

データソースに固有のネイティブエラーコードを返すバッファへのポインタです。

この情報はSQL_DIAG_NATIVE 診断フィールドに格納されている情報が返されます。

このパラメタにNULL が指定された場合は何も設定しません。

MessageText：

診断メッセージテキスト文字列を返すバッファへのポインタです。

この情報はSQL_DIAG_MESSAGE_TEXT 診断フィールドに格納されている情報が返されます。

このパラメタにNULL が指定された場合は何も設定しません。

BufferLength：

MessageText バッファの長さ※です。

診断メッセージテキストには最大長の定義はありませんが，必ず 512 バイト以上の値を指定してください。

この長さに NULL 終端文字は含まれます。SQL_NTS は指定できません。

TextLengthPtr：

MessageText に返す有効な長さ※の合計を返すバッファへのポインタを指定します。この長さに NULL 終端文字は含まれません。

重要

ここに格納された長さ※が，BufferLength から NULL 終端文字分を引いた長さ※より大きい場合，MessageText に格納される文字列はBufferLength から NULL 終端文字分を引いた長さ※に切り捨てられ，末尾に NULL 終端文字が付加されます。

注※

長さの単位は，SQLGetDiagRec の場合はバイト長，SQLGetDiagRecW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，SQL_INVALID_HANDLE，またはSQL_NO_DATA が返されます。この関数ではSQLSTATE を設定しない代わりに，戻り値に次の実行結果を示します。

戻り値	意味
SQL_SUCCESS	成功
SQL_SUCCESS_WITH_INFO	次のエラーのうちのどちらかが発生しました。 • 診断メッセージテキストに対して，MessageText のサイズが小さいため，切り捨てが発生しました。BufferLength とTextLengthPtr を比較することで，切り捨てられたサイズを知ることができます。 • 文字コード変換時に変換できない文字コードを検出し，指定の文字に置き換えました。
SQL_ERROR	次のエラーのどれかが発生しました。 • RecNumber に 0 以下の値が指定されました。 • BufferLength に 0 より小さい値が指定されました。 • BufferLength に無効な値（SQL_NTS）が指定されました。
SQL_INVALID_HANDLE	HandleType とHandle によって示されるハンドルが有効なハンドルではありません。
SQL_NO_DATA	RecNumber がHandle で指定されたハンドルに存在する診断レコードの数より大きいか，指定Handle に読み出しできる診断レコードがありません。

(5) SQLSTATE

この関数ではSQLSTATE を返しません。

エラーの詳細はすべて「(4) 戻り値」に定義している値で通知します。

(6) 注意事項

- エラーの詳細情報については，「15.6 エラー発生時に返却される情報」を参照してください。
- NativeErrorPtr にはSQLCODE が設定されます。SQLCODE については，マニュアル『HADB メッセージ』の『SQLCODE の見方』を参照してください。
- 診断情報は返しません。

16.10　データソースのシステム情報の取得

ここでは，データソースのシステム情報の取得時に使用する ODBC 関数について説明します。

16.10.1　SQLColumnPrivileges，SQLColumnPrivilegesW

(1)　機能

指定されたテーブルの列と関連づけられた特権の一覧を返します。SQL の結果セットの形式で出力します。

(2)　形式

- SQLColumnPrivileges の場合

```
SQLRETURN SQLColumnPrivileges
(
  SQLHSTMT       StatementHandle,   /* In */
  SQLCHAR      * CatalogName,       /* In */
  SQLSMALLINT    NameLength1,       /* In */
  SQLCHAR      * SchemaName,        /* In */
  SQLSMALLINT    NameLength2,       /* In */
  SQLCHAR      * TableName,         /* In */
  SQLSMALLINT    NameLength3,       /* In */
  SQLCHAR      * ColumnName,        /* In */
  SQLSMALLINT    NameLength4        /* In */
)
```

- SQLColumnPrivilegesW の場合

```
SQLRETURN SQLColumnPrivilegesW
(
  SQLHSTMT       StatementHandle,   /* In */
  SQLWCHAR     * CatalogName,       /* In */
  SQLSMALLINT    NameLength1,       /* In */
  SQLWCHAR     * SchemaName,        /* In */
  SQLSMALLINT    NameLength2,       /* In */
  SQLWCHAR     * TableName,         /* In */
  SQLSMALLINT    NameLength3,       /* In */
  SQLWCHAR     * ColumnName,        /* In */
  SQLSMALLINT    NameLength4        /* In */
)
```

(3)　引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

CatalogName：

テーブルに対するカタログ名を指定します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，空の文字列（””）またはNULL を指定します。

NameLength1：

*CatalogName の長さ※1 またはSQL_NTS を指定します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，0 を指定します。

SchemaName：

テーブルのスキーマ名を指定します。パターン文字列※2 を使用してスキーマ名を指定できます。

SchemaName にNULL ポインタまたは空の文字列（""）だけを指定した場合，SchemaName にパターン文字列'%' だけを指定したと見なされます。

NameLength2：

*SchemaName の長さ※1 またはSQL_NTS を指定します。

NameLength2 に0 を指定した場合，SchemaName にパターン文字列'%' だけを指定したと見なされます。

TableName：

テーブル名を指定します。パターン文字列※2 を使用してテーブル名を指定できます。

TableName にNULL ポインタまたは空の文字列（""）だけを指定した場合，TableName にパターン文字列'%' だけを指定したと見なされます。

NameLength3：

*TableName の長さ※1 またはSQL_NTS を指定します。

NameLength3 に0 を指定した場合，TableName にパターン文字列'%' だけを指定したと見なされます。

ColumnName：

列名を指定します。パターン文字列※2 を使用して列名を指定できます。

ColumnName にNULL ポインタまたは空の文字列（""）だけを指定した場合，ColumnName にパターン文字列'%' だけを指定したと見なされます。

NameLength4：

*ColumnName の長さ※1 またはSQL_NTS を指定します。

NameLength4 に0 を指定した場合，ColumnName にパターン文字列'%' だけを指定したと見なされます。

注※1

長さの単位は，SQLColumnPrivileges の場合はバイト長，SQLColumnPrivilegesW の場合は文字数となります。

注※2

パターン文字列中に指定できる特殊文字については，「表 16-13　パターン文字列中に指定できる特殊文字」を参照してください。

(4) 戻り値

SQL_SUCCESS, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

この関数を実行したときに返される結果セットの形式を次の表に示します。

表 16-12 返却される結果セットの形式

列番号	型	列名	列の意味
1	Varchar	TABLE_CAT	カタログ名（常にナル値を返します）
2	Varchar	TABLE_SCHEM	スキーマ名
3	Varchar	TABLE_NAME	表名
4	Varchar	COLUMN_NAME	列名
5	Varchar	GRANTOR	アクセス権限の付与者
6	Varchar	GRANTEE	アクセス権限の被付与者
7	Varchar	PRIVILEGE	許可されているアクセス権限を返します。 • SELECT：SELECT 権限 • INSERT：INSERT 権限 • UPDATE：UPDATE 権限 • DELETE：DELETE 権限 • TRUNCATE：TRUNCATE 権限 • REFERENCES：REFERENCES 権限 • IMPORT TABLE：IMPORT TABLE 権限 • REBUILD INDEX：REBUILD INDEX 権限 • GET COSTINFO：GET COSTINFO 権限 • EXPORT TABLE：EXPORT TABLE 権限 • MERGE CHUNK：MERGE CHUNK 権限 • CHANGE CHUNK COMMENT：CHANGE CHUNK COMMENT 権限 • CHANGE CHUNK STATUS：CHANGE CHUNK STATUS 権限 • ARCHIVE CHUNK：ARCHIVE CHUNK 権限 • UNARCHIVE CHUNK：UNARCHIVE CHUNK 権限
8	Varchar	IS_GRANTABLE	アクセス権限の被付与者が，ほかのユーザにアクセス権限を付与できるかどうかを返します。 • YES：アクセス権限を付与できます。 • NO：アクセス権限を付与できません。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態	カーソルがオープンしている状態のときに実行されました。	○
40001	直列化の失敗	－	×
40003	ステートメントの完了が不明		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	名前の長さを格納する引数のどれかの値が，対応する名前の最大長を超えているか，またはSQL_NTS 以外の負の値になっています。	○
HY117	接続がサスペンド中	－	×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.10.2 SQLColumns，SQLColumnsW

(1) 機能

列情報の一覧を結果セットとして返します。

(2) 形式

- SQLColumns の場合

```
SQLRETURN SQLColumns
(
  SQLHSTMT          StatementHandle,   /* In */
  SQLCHAR         * CatalogName,       /* In */
  SQLSMALLINT       NameLength1,       /* In */
  SQLCHAR         * SchemaName,        /* In */
  SQLSMALLINT       NameLength2,       /* In */
  SQLCHAR         * TableName,         /* In */
  SQLSMALLINT       NameLength3,       /* In */
  SQLCHAR         * ColumnName,        /* In */
  SQLSMALLINT       NameLength4        /* In */
)
```

- SQLColumnsW の場合

```
SQLRETURN SQLColumnsW
(
  SQLHSTMT          StatementHandle,   /* In */
  SQLWCHAR        * CatalogName,       /* In */
  SQLSMALLINT       NameLength1,       /* In */
  SQLWCHAR        * SchemaName,        /* In */
  SQLSMALLINT       NameLength2,       /* In */
  SQLWCHAR         * TableName,        /* In */
  SQLSMALLINT       NameLength3,       /* In */
  SQLWCHAR        * ColumnName,        /* In */
  SQLSMALLINT       NameLength4        /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

CatalogName：

指定内容をカタログ名として使用します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，空の文字列（""）またはNULL を指定します。

NameLength1：

*CatalogName の長さ※1，またはSQL_NTS を指定します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，0 を指定します。

SchemaName：

スキーマ名のパターン文字列※2 を指定します。NULL を指定した場合は，パターン文字列として'%'が指定されたときと同じ動作となります。

NameLength2：

*SchemaName の長さ※1，またはSQL_NTS を指定します。

0 を指定した場合は，SchemaName にパターン文字列として'%'が指定されたときと同じ動作となります。

TableName：

テーブル名のパターン文字列※2 を指定します。NULL を指定した場合は，パターン文字列として'%'が指定されたときと同じ動作となります。

NameLength3：

*TableName の長さ※1，またはSQL_NTS を指定します。

0 を指定した場合は，TableName にパターン文字列として'%'が指定されたときと同じ動作となります。

ColumnName：

列名のパターン文字列※2 を指定します。NULL を指定した場合は，パターン文字列として'%'が指定されたときと同じ動作となります。

NameLength4：

*ColumnName の長さ※1，またはSQL_NTS を指定します。

0 を指定した場合は，ColumnName にパターン文字列として'%'が指定されたときと同じ動作となります。

注※1

長さの単位は，SQLColumns の場合はバイト長，SQLColumnsW の場合は文字数となります。

注※2

パターン文字列中に指定できる特殊文字を次の表に示します。

表 16-13　パターン文字列中に指定できる特殊文字

指定できる特殊文字	意味
_（下線）	任意の 1 文字です。
%	0 文字以上の任意の長さの文字列です。
¥	エスケープ文字です。パターン文字列中に指定したエスケープ文字の直後の特殊文字を通常の文字として扱います。 ¥は，Shift-JIS の 0x5c（UTF-16LE の 0x5c00）で示される文字です。UTF-8 の場合は，¥（バックスラッシュ）で表示される文字を指定してください。

(4)　戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

実行後に返却される結果セットの形式を次の表に示します。

表 16-14　返却される結果セットの形式

列番号	型	列名	列の意味
1	Varchar	TABLE_CAT	常にナル値を返します。

列番号	型	列名	列の意味
2	Varchar	TABLE_SCHEM	スキーマ名
3	Varchar	TABLE_NAME	表名
4	Varchar	COLUMN_NAME	列名
5	smallint	DATA_TYPE	ODBC SQL データ型識別子
6	Varchar	TYPE_NAME	型名
7	Integer	COLUMN_SIZE	列サイズ
8	Integer	BUFFER_LENGTH	列のデータ定義長
9	Smallint	DECIMAL_DIGITS	小数点以下の桁数
10	Smallint	NUM_PREC_RADIX	基数 • 真数値：10 • 数値以外：ナル値
11	Smallint	NULLABLE	この型にナル値を使用できるかどうかを返します。 • SQL_NO_NULLS：ナル値を使用できないおそれがあります。 • SQL_NULLABLE：ナル値を使用できます。
12	Varchar	REMARKS	常にナル値を返します。
13	Varchar	COLUMN_DEF	列の既定値を返します。 • 返却された値が，アポストロフィ（'）で囲まれている場合は，列の既定値が文字列であることを意味しています。 • 列の既定値にNULL が指定されている場合は，アポストロフィ（'）で囲まれていない NULL という文字列を返します。 • 列の既定値が指定されていない場合は，ナル値を返します。 • そのほかの返却値については，マニュアル『HADB　システム構築・運用ガイド』の『SQL_COLUMNS の内容』の DEFAULT_VALUE 列を参照してください。
14	Smallint	SQL_DATA_TYPE	ODBC SQL データ型識別子
15	Smallint	SQL_DATETIME_SUB	ODBC SQL データ型識別子のサブコード
16	Integer	CHAR_OCTET_LENGTH	文字データ型についての列の最大バイト長
17	Integer	ORDINAL_POSITION	列番号 1 から始まります。
18	Varchar	IS_NULLABLE	この型にナル値が使用できるかどうかを返します。 • YES：ナル値を使用できる可能性があります。

列番号	型	列名	列の意味
			• NO：ナル値を使用できません。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C041	データ型未サポートエラー	ドライバは指定されたデータ型をサポートしていません。	×
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	名前の長さを格納する引数のどれかの値が，対応する名前の最大長を超えました。	○
HY117	接続がサスペンド中	－	×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

この関数を実行したHADBユーザが持っている権限によって，取得できる列情報が変わります。権限と取得できる列情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADBユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

16.10.3 SQLForeignKeys，SQLForeignKeysW

(1) 機能

次の外部キーの列情報をSQLの結果セットの形式で返します。

- 指定されたテーブルの外部キー一覧（別のテーブルの主キーを参照する指定をされたテーブルの列）
- 指定されたテーブルの主キーを参照する，別のテーブルの外部キー一覧

(2) 形式

- SQLForeignKeysの場合

```
SQLRETURN SQLForeignKeys
(
  SQLHSTMT        StatementHandle,    /* In */
  SQLCHAR       * PKCatalogName,      /* In */
  SQLSMALLINT     NameLength1,        /* In */
  SQLCHAR       * PKSchemaName,       /* In */
  SQLSMALLINT     NameLength2,        /* In */
  SQLCHAR       * PKTableName,        /* In */
  SQLSMALLINT     NameLength3,        /* In */
  SQLCHAR       * FKCatalogName,      /* In */
  SQLSMALLINT     NameLength4,        /* In */
  SQLCHAR       * FKSchemaName,       /* In */
  SQLSMALLINT     NameLength5,        /* In */
  SQLCHAR       * FKTableName,        /* In */
  SQLSMALLINT     NameLength6         /* In */
)
```

- SQLForeignKeysWの場合

```
SQLRETURN SQLForeignKeysW
(
  SQLHSTMT        StatementHandle,    /* In */
  SQLWCHAR      * PKCatalogName,      /* In */
  SQLSMALLINT     NameLength1,        /* In */
  SQLWCHAR      * PKSchemaName,       /* In */
  SQLSMALLINT     NameLength2,        /* In */
  SQLWCHAR      * PKTableName,        /* In */
  SQLSMALLINT     NameLength3,        /* In */
  SQLWCHAR      * FKCatalogName,      /* In */
```

```
  SQLSMALLINT     NameLength4,        /* In */
  SQLWCHAR      * FKSchemaName,       /* In */
  SQLSMALLINT     NameLength5,        /* In */
  SQLWCHAR      * FKTableName,        /* In */
  SQLSMALLINT     NameLength6         /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

PKCatalogName：

主キーのテーブルのカタログ名を指定します。ただし，指定しても無視されます。この関数を使用する際は，空の文字列（""），またはNULL を指定します。

NameLength1：

*PKCatalogName の長さ※，またはSQL_NTS を指定します。ただし，指定しても無視されます。この関数を使用する際は，0 を指定します。

PKSchemaName：

主キーのテーブルのスキーマ名を指定します。NULL を指定した場合は，すべてのスキーマ名を対象とします。

NameLength2：

*PKSchemaName の長さ※，またはSQL_NTS を指定します。0 を指定した場合は，すべてのスキーマ名を対象とします。

PKTableName：

主キーのテーブル名を指定します。指定する値については，「表 16-15　PKTableName および FKTableName の指定値の組み合わせと，その組み合わせによって返却される結果セット」を参照してください。空の文字列を指定した場合，結果セットの行数は 0 になります。

NameLength3：

*PKTableName の長さ※，またはSQL_NTS を指定します。PKTableName にNULL を指定した場合，この引数は無視されます。PKTableName に NULL 以外の値を指定して，この引数に0 を指定した場合，PKTableName に空の文字列を指定したときと同じ動作になります。

FKCatalogName：

外部キーのテーブルのカタログ名を指定します。ただし，指定しても無視されます。この関数を使用する際は，空の文字列（""），またはNULL を指定します。

NameLength4：

*FKCatalogName の長さ※，またはSQL_NTS を指定します。ただし，指定しても無視されます。この関数を使用する際は，0 を指定します。

FKSchemaName：

外部キーのテーブルのスキーマ名を指定します。NULL を指定した場合は，すべてのスキーマ名を対象とします。

NameLength5：

*FKSchemaName の長さ※，またはSQL_NTS を指定します。0 を指定した場合は，すべてのスキーマ名を対象とします。

FKTableName：

外部キーのテーブル名を指定します。指定する値については，「表 16-15　PKTableName および FKTableName の指定値の組み合わせと，その組み合わせによって返却される結果セット」を参照してください。空の文字列を指定した場合，結果セットの行数は 0 になります。

NameLength6：

*FKTableName の長さ※，またはSQL_NTS を指定します。PKTableName にNULL を指定した場合，この引数は無視されます。PKTableName にNULL 以外の値を指定して，この引数に0 を指定した場合，PKTableName に空の文字列を指定したときと同じ動作になります。

注※

長さの単位は，SQLForeignKeys の場合はバイト長，SQLForeignKeysW の場合は文字数となります。

PKTableName およびFKTableName の指定値の組み合わせと，その組み合わせによって返却される結果セットを次の表に示します。

表 16-15　PKTableName および FKTableName の指定値の組み合わせと，その組み合わせによって返却される結果セット

PKTableName	FKTableName	返却される結果セット
NULL	NULL	－（エラー）
Not NULL	NULL	PKTableName に指定されたテーブルの主キーと，その主キーを参照する別テーブルの外部キーの情報を返します。 PKTableName に指定されたテーブルのユニーク制約だけのキーを参照する別テーブルの外部キーの情報は返しません。
NULL	Not NULL	FKTableName に指定されたテーブルの外部キーと，その外部キーが参照する別テーブルの主キーの情報を返します。 別テーブルのユニーク制約だけのキーを参照するFKTableName に指定されたテーブルの外部キーの情報は返しません。
Not NULL	Not NULL	FKTableName に指定されたテーブルの外部キーと，その外部キーが参照するテーブルの主キーの情報を返します。この外部キーは，PKTableName に指定されたテーブルの主キーだけを参照します。

(4)　戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

SQLForeignKeys またはSQLForeignKeysW を実行すると結果セットが作成されます。返却される結果セットの形式を次に示します。

表 16-16　返却される結果セットの形式

列番号	型	列名	列の意味
1	Varchar	PKTABLE_CAT	常にナル値を返します。
2	Varchar	PKTABLE_SCHEM	主キーのスキーマ名を返します。スキーマ名がない場合，空の文字列を返します。
3	Varchar	PKTABLE_NAME	主キーの表名を返します。
4	Varchar	PKCOLUMN_NAME	主キーの列名を返します。列名がない場合，空の文字列を返します。
5	Varchar	FKTABLE_CAT	常にナル値を返します。
6	Varchar	FKTABLE_SCHEM	外部キーのスキーマ名を返します。スキーマ名がない場合，空の文字列を返します。
7	Varchar	FKTABLE_NAME	外部キーの表名を返します。
8	Varchar	FKCOLUMN_NAME	外部キーの列名を返します。列名がない場合，空の文字列を返します。
9	Smallint	KEY_SEQ	1 から始まる外部キーの列順の番号を返します。
10	Smallint	UPDATE_RULE	更新を行う SQL 文が要求されたときに，外部キーに適用される動作を返します。 • SQL_NO_ACTION 主キーの値の更新，または外部キーの値の更新によって，外部キーの値に対応する主キーの値がない場合，更新はできません。ただし，CREATE TABLE 文の参照制約定義で参照制約チェック抑止指定（DISABLE）を指定した場合は，主キーの値または外部キーの値を更新できます。
11	Smallint	DELETE_RULE	削除を行う SQL 文が要求されたときに，外部キーに適用される動作を返します。 • SQL_NO_ACTION 参照先テーブルの行削除によって，外部キーの値に対応する主キーの値がない場合，行削除はできません。ただし，CREATE TABLE 文の参照制約定義で参照制約チェック抑止指定（DISABLE）を指定した場合は，行を削除できます。
12	Varchar	FK_NAME	外部キー名を返します。
13	Varchar	PK_NAME	主キー名を返します。
14	Smallint	DEFERRABILITY	外部キーに対する制約のチェックを遅延するかどうかを返します。

列番号	型	列名	列の意味
			• SQL_NOT_DEFERRABLE SQL 文が実行されるたびに制約がチェックされます。遅延できるように変更することはできません。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	−	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態	カーソルがオープンしている状態のときに実行されました。	○
40001	直列化の失敗	−	×
40003	ステートメントの完了が不明		×
5C002	文字コード変換エラーが発生	変換できない文字コードを検出しました。	○
HY000	一般エラー	−	×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用	PKTableName およびFKTableName の両方がNULL です。	○
HY010	関数シーケンスエラー	−	○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	名前の長さを格納する引数のどれかの値が，対応する名前の最大長を超えました。	○
HY117	接続がサスペンド中	−	×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

この関数を実行した HADB ユーザが持っている権限によって，取得できる外部キーの情報が変わります。権限と取得できる情報の対応については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

16.10.4 SQLPrimaryKeys，SQLPrimaryKeysW

(1) 機能

テーブルに対する主キーを構成する列名を返します。SQL 文の結果セットの形式で出力します。

1 回の呼び出しで複数のテーブルから主キーを返すことはできません。

(2) 形式

- SQLPrimaryKeys の場合

```
SQLRETURN SQLPrimaryKeys
(
  SQLHSTMT       StatementHandle,     /* In */
  SQLCHAR      * CatalogName,         /* In */
  SQLSMALLINT    NameLength1,         /* In */
  SQLCHAR      * SchemaName,          /* In */
  SQLSMALLINT    NameLength2,         /* In */
  SQLCHAR      * TableName,           /* In */
  SQLSMALLINT    NameLength3          /* In */
)
```

- SQLPrimaryKeysW の場合

```
SQLRETURN SQLPrimaryKeysW
(
  SQLHSTMT       StatementHandle,     /* In */
  SQLWCHAR     * CatalogName,         /* In */
  SQLSMALLINT    NameLength1,         /* In */
  SQLWCHAR     * SchemaName,          /* In */
  SQLSMALLINT    NameLength2,         /* In */
  SQLWCHAR     * TableName,           /* In */
  SQLSMALLINT    NameLength3          /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

CatalogName：

指定内容をカタログ名として使用します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，空の文字列（””）またはNULL を指定します。

NameLength1：

*CatalogName の長さ※，またはSQL_NTS を指定します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，0 を指定します。

SchemaName：

スキーマ名を指定します。NULL を指定した場合は，すべてのスキーマ名を対象にします。

NameLength2：

*SchemaName の長さ※，またはSQL_NTS を指定します。

0 を指定した場合は，SchemaName による絞り込みを行いません。

TableName：

テーブル名を指定します。NULL を指定した場合は，すべてのテーブル名を対象にします。

NameLength3：

*TableName の長さ※，またはSQL_NTS を指定します。

0 を指定した場合は，TableName にNULL が指定されたときと同じ動作となります。

注※

長さの単位は，SQLPrimaryKeys の場合はバイト長，SQLPrimaryKeysW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

SQLPrimaryKeys またはSQLPrimaryKeysW を実行すると結果セットが作成されます。返却される結果セットの形式を次に示します。

表 16-17　返却される結果セットの形式

列番号	型	列名	列の意味
1	Varchar	TABLE_CAT	常にナル値を返します。
2	Varchar	TABLE_SCHEM	スキーマ名
3	Varchar	TABLE_NAME	表名

列番号	型	列名	列の意味
4	Varchar	COLUMN_NAME	列名
5	Smallint	KEY_SEQ	主キー内で付けられる列順の番号
6	Varchar	PK_NAME	主キー名

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態	カーソルがオープンしている状態のときに実行されました。	○
40001	直列化の失敗	－	×
40003	ステートメントの完了が不明		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用	TableName が NULL ポインタです。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	名前の長さを格納する引数のどれかの値が，対応する名前の最大長を超えました。	○
HY117	接続がサスペンド中	－	×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

この関数を実行したHADBユーザが持っている権限によって，取得できる主キーの情報が変わります。権限と取得できる主キーの情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADBユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

16.10.5 SQLProcedureColumns，SQLProcedureColumnsW

(1) 機能

入力パラメタと出力パラメタの一覧，および指定されたプロシジャに対する結果セットを構成する列の一覧を返します。SQL文の結果セットの形式で出力します。

ただし，プロシジャをサポートしていないため，検索結果セットの行数は常に0になります。

(2) 形式

- SQLProcedureColumnsの場合

```
SQLRETURN SQLProcedureColumns
(
  SQLHSTMT        StatementHandle,    /* In */
  SQLCHAR       * CatalogName,        /* In */
  SQLSMALLINT     NameLength1,        /* In */
  SQLCHAR       * SchemaName,         /* In */
  SQLSMALLINT     NameLength2,        /* In */
  SQLCHAR       * ProcName,           /* In */
  SQLSMALLINT     NameLength3,        /* In */
  SQLCHAR       * ColumnName,         /* In */
  SQLSMALLINT     NameLength4         /* In */
)
```

- SQLProcedureColumnsWの場合

```
SQLRETURN SQLProcedureColumnsW
(
  SQLHSTMT        StatementHandle,    /* In */
  SQLWCHAR      * CatalogName,        /* In */
  SQLSMALLINT     NameLength1,        /* In */
  SQLWCHAR      * SchemaName,         /* In */
  SQLSMALLINT     NameLength2,        /* In */
  SQLWCHAR      * ProcName,           /* In */
  SQLSMALLINT     NameLength3,        /* In */
  SQLWCHAR      * ColumnName,         /* In */
  SQLSMALLINT     NameLength4         /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

CatalogName：

プロシジャのカタログ名を指定します。

NameLength1：

*CatalogName の長さ※を指定します。

SchemaName：

プロシジャのスキーマ名に対するパターン文字列を指定します。

NameLength2：

*SchemaName の長さ※を指定します。

ProcName：

プロシジャ名に対するパターン文字列を指定します。

NameLength3：

*ProcName の長さ※を指定します。

ColumnName：

列名に対するパターン文字列を指定します。

NameLength4：

*ColumnName の長さ※を指定します。

注※

長さの単位は，SQLProcedureColumns の場合はバイト長，SQLProcedureColumnsW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×

SQLSTATE	説明	備考	返却
24000	無効なカーソル状態	カーソルがオープンしている状態のときに実行されました。	○
40001	直列化の失敗	－	×
40003	ステートメントの完了が不明		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○
HY090	無効な文字列長または無効なバッファ長		×
HY117	接続がサスペンド中		×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.10.6 SQLProcedures，SQLProceduresW

(1) 機能

特定のデータソースに格納されたプロシジャの一覧を返します。SQL 文の結果セットの形式で出力します。

ただし，プロシジャをサポートしていないため，検索結果セットの行数は常に 0 になります。

(2) 形式

- SQLProcedures の場合

```
SQLRETURN SQLProcedures
(
  SQLHSTMT       StatementHandle,   /* In */
  SQLCHAR      * CatalogName,       /* In */
  SQLSMALLINT    NameLength1,       /* In */
  SQLCHAR      * SchemaName,        /* In */
  SQLSMALLINT    NameLength2,       /* In */
  SQLCHAR      * ProcName,          /* In */
  SQLSMALLINT    NameLength3        /* In */
)
```

- SQLProceduresW の場合

```
SQLRETURN SQLProceduresW
(
  SQLHSTMT       StatementHandle,   /* In */
  SQLWCHAR     * CatalogName,       /* In */
  SQLSMALLINT    NameLength1,       /* In */
  SQLWCHAR     * SchemaName,        /* In */
  SQLSMALLINT    NameLength2,       /* In */
  SQLWCHAR     * ProcName,          /* In */
  SQLSMALLINT    NameLength3        /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

CatalogName：

プロシジャのカタログ名を指定します。

NameLength1：

*CatalogName の長さ※を指定します。

SchemaName：

プロシジャのスキーマ名に対するパターン文字列を指定します。

NameLength2：

*SchemaName の長さ※を指定します。

ProcName：

プロシジャ名に対するパターン文字列を指定します。

NameLength3：

*ProcName の長さ※を指定します。

注※

長さの単位は，SQLProcedures の場合はバイト長，SQLProceduresW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態	カーソルがオープンしている状態のときに実行されました。	○
40001	直列化の失敗	－	×
40003	ステートメントの完了が不明		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY117	接続がサスペンド中		×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.10.7 SQLSpecialColumns，SQLSpecialColumnsW

(1) 機能

指定されたテーブルの列に関する次のどちらかの情報を取得します。

- テーブルの行を一意に識別する最適な列のセット
- 行の値がトランザクションによって更新されるときに自動的に更新される列

SQL 文の結果セットの形式で出力します。

ただし，自動的に列を更新する機能をサポートしていないため，検索結果セットの行数は常に 0 になります。

(2) 形式

- SQLSpecialColumns の場合

```
SQLRETURN SQLSpecialColumns
(
  SQLHSTMT       StatementHandle,    /* In */
  SQLUSMALLINT   IdentifierType,     /* In */
  SQLCHAR      * CatalogName,        /* In */
  SQLSMALLINT    NameLength1,        /* In */
  SQLCHAR      * SchemaName,         /* In */
  SQLSMALLINT    NameLength2,        /* In */
  SQLCHAR      * TableName,          /* In */
  SQLSMALLINT    NameLength3,        /* In */
  SQLUSMALLINT   Scope,              /* In */
  SQLUSMALLINT   Nullable            /* In */
)
```

- SQLSpecialColumnsW の場合

```
SQLRETURN SQLSpecialColumnsW
(
  SQLHSTMT       StatementHandle,    /* In */
  SQLUSMALLINT   IdentifierType,     /* In */
  SQLWCHAR     * CatalogName,        /* In */
  SQLSMALLINT    NameLength1,        /* In */
  SQLWCHAR     * SchemaName,         /* In */
  SQLSMALLINT    NameLength2,        /* In */
  SQLWCHAR     * TableName,          /* In */
  SQLSMALLINT    NameLength3,        /* In */
  SQLUSMALLINT   Scope,              /* In */
  SQLUSMALLINT   Nullable            /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

IdentifierType：

返す列の種類として，次の値のどちらかを指定します。

IdentifierType	内容
SQL_BEST_ROWID	列から値を取得することで，指定されたテーブルの行が一意に識別できる最適な列または列のセットを返します。 行を一意に識別できるインデクスの列がテーブル内に存在する場合はその列を返します。テーブル内にそういった列が存在しない場合は行を識別できるように一時的に作成した擬似列が返されます。
SQL_ROWVER	トランザクションによって行の値が更新される場合，指定されたテーブル内にデータソースによって自動的に更新される列があるとき，その列が返されます。

CatalogName：

テーブルに対するカタログ名を指定します。

NameLength1：

*CatalogName の長さ※を指定します。

SchemaName：

テーブルに対するスキーマ名を指定します。

NameLength2：

*SchemaName の長さ※を指定します。

TableName：

テーブル名を指定します。

NameLength3：

*TableName の長さ※を指定します。

Scope：

必要最小限の行 ID の有効範囲を次の値から選んで指定します。

Scope	内容
SQL_SCOPE_CURROW	行 ID は，その行に位置づけられている間だけ有効です。
SQL_SCOPE_TRANSACTION	行 ID は，現在のトランザクション中だけ有効です。
SQL_SCOPE_SESSION	行 ID は，セッション（トランザクション境界にわたる）中，有効です。

Nullable：

ナル値を格納する特別な列を返すかどうかを指定します。

次の値のどちらかを指定します。

Nullable	内容
SQL_NO_NULLS	ナル値を格納できる特別な列を除きます。ドライバには，SQL_NO_NULLS をサポートできないものもあり，SQL_NO_NULLS が指定されると，空の結果セットを

Nullable	内容
	返すため，注意が必要です。アプリケーションはこれを踏まえて，必要な場合にだけSQL_NO_NULLS を指定します。
SQL_NULLABLE	ナル値が格納されている場合でも，特別な列を返します。

注※

長さの単位は，SQLSpecialColumns の場合はバイト長，SQLSpecialColumnsW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態	カーソルがオープンしている状態のときに実行されました。	○
40001	直列化の失敗	－	×
40003	ステートメントの完了が不明		×
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長		×
HY097	IdentifierType に無効な値を指定された		×
HY098	Scope に無効な値を指定された		×
HY099	Nullable に無効な値を指定された		×
HY117	接続がサスペンド中		×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×

SQLSTATE	説明	備考	返却
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.10.8 SQLStatistics，SQLStatisticsW

(1) 機能

テーブルに関連づけられたインデクス情報の一覧を SQL 文の結果セットの形式で返します。

(2) 形式

- SQLStatistics の場合

```
SQLRETURN SQLStatistics
(
  SQLHSTMT          StatementHandle,    /* In */
  SQLCHAR         * CatalogName,        /* In */
  SQLSMALLINT       NameLength1,        /* In */
  SQLCHAR         * SchemaName,         /* In */
  SQLSMALLINT       NameLength2,        /* In */
  SQLCHAR         * TableName,          /* In */
  SQLSMALLINT       NameLength3,        /* In */
  SQLUSMALLINT      Unique,             /* In */
  SQLUSMALLINT      Reserved            /* In */
)
```

- SQLStatisticsW の場合

```
SQLRETURN SQLStatisticsW
(
  SQLHSTMT          StatementHandle,    /* In */
  SQLWCHAR        * CatalogName,        /* In */
  SQLSMALLINT       NameLength1,        /* In */
  SQLWCHAR        * SchemaName,         /* In */
  SQLSMALLINT       NameLength2,        /* In */
  SQLWCHAR        * TableName,          /* In */
  SQLSMALLINT       NameLength3,        /* In */
  SQLUSMALLINT      Unique,             /* In */
```

```
  SQLUSMALLINT      Reserved            /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

CatalogName：

指定内容をカタログ名として使用します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，空の文字列（””）またはNULL を指定します。

NameLength1：

*CatalogName の長さ※，またはSQL_NTS を指定します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，0 を指定します。

SchemaName：

スキーマ名を指定します。NULL を指定した場合は，すべてのスキーマ名を対象にします。

NameLength2：

*SchemaName の長さ※，またはSQL_NTS を指定します。

0 を指定した場合は，SchemaName による絞り込みを行いません。

TableName：

テーブル名を指定します。

NULL を指定した場合は，すべてのテーブル名を対象にします。

NameLength3：

*TableName の長さ※，またはSQL_NTS を指定します。

0 を指定した場合は，TableName にNULL が指定されたとして動作します。

Unique：

インデクスの種類を指定します。

- SQL_INDEX_UNIQUE
  ユニークインデクスの情報だけを取得します。
- SQL_INDEX_ALL
  すべてのインデクス情報を取得します。

Reserved：

指定内容を結果セットのCARDINALITY 列とPAGES 列の重要度を示すオプションとして使用します。ただし，指定しても無視されます。

注※

長さの単位は，SQLStatistics の場合はバイト長，SQLStatisticsW の場合は文字数となります。

(4) 戻り値

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

返却される結果セットの形式を次の表に示します。

表 16-18 返却される結果セットの形式

列番号	型	列名	列の意味
1	Varchar	TABLE_CAT	常にナル値を返します。
2	Varchar	TABLE_SCHEM	スキーマ名
3	Varchar	TABLE_NAME	表名
4	Smallint	NON_UNIQUE	インデクスを定義するキー値（インデクスとして定義する 1 つまたは複数の列全体の値）を一意でない値にできる場合は，SQL_TRUE を返します。できない場合は，SQL_FALSE を返します。
5	Varchar	INDEX_QUALIFIER	常にナル値を返します。
6	Varchar	INDEX_NAME	インデクス識別子
7	Smallint	TYPE	インデクス種別を返します。 常にSQL_INDEX_OTHER を返します。
8	Smallint	ORDINAL_POSITION	• 単一列インデクスの場合，1 を返します。 • 複数列インデクスの場合，インデクスを構成する列の順序（インデクスを構成する列名順を識別する番号で，1 から始まる整数）を返します。
9	Varchar	COLUMN_NAME	列名
10	Char(1)	ASC_OR_DESC	• B-tree インデクスを昇順に定義する場合，"A"を返します。 • B-tree インデクスを降順に定義する場合，"D"を返します。 • テキストインデクスまたはレンジインデクスの場合，ナル値を返します。
11	Integer	CARDINALITY	常に0 を返します。
12	Integer	PAGES	
13	Varchar	FILTER_CONDITION	常にナル値を返します。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態	カーソルがオープンしている状態のときに実行されました。	×
40001	直列化の失敗	－	×
40003	ステートメントの完了が不明		×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C041	データ型未サポートエラー	ドライバは指定されたデータ型をサポートしていません。	×
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用	TableName に NULL ポインタが指定されているか，またはNameLength3 に0 が指定されています。	○
HY010	関数シーケンスエラー	－	○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	名前の長さを格納する引数のどれかの値が，対応する名前の最大長を超えました。	○
HY100	一意性のオプションが範囲外である	Unique に指定された値が無効です。	○
HY101	精度のオプションが範囲外である	－	×
HY117	接続がサスペンド中		×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

この関数を実行したHADBユーザが持っている権限によって，取得できるインデクス情報が変わります。権限と取得できるインデクス情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADBユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

16.10.9 SQLTablePrivileges，SQLTablePrivilegesW

(1) 機能

テーブルの一覧と各テーブルに関連づけられたアクセス権限の一覧を返します。SQL文の結果セットの形式で出力します。

(2) 形式

- SQLTablePrivilegesの場合

```
SQLRETURN SQLTablePrivileges
(
  SQLHSTMT       StatementHandle,    /* In */
  SQLCHAR      * CatalogName,        /* In */
  SQLSMALLINT    NameLength1,        /* In */
  SQLCHAR      * SchemaName,         /* In */
  SQLSMALLINT    NameLength2,        /* In */
  SQLCHAR      * TableName,          /* In */
  SQLSMALLINT    NameLength3         /* In */
)
```

- SQLTablePrivilegesWの場合

```
SQLRETURN SQLTablePrivilegesW
(
  SQLHSTMT       StatementHandle,    /* In */
  SQLWCHAR     * CatalogName,        /* In */
  SQLSMALLINT    NameLength1,        /* In */
  SQLWCHAR     * SchemaName,         /* In */
  SQLSMALLINT    NameLength2,        /* In */
  SQLWCHAR     * TableName,          /* In */
  SQLSMALLINT    NameLength3         /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandleの*OutputHandlePtrで出力された値を指定します。

CatalogName：

テーブルのカタログ名を指定します。

ただし，指定しても無視されます。この関数を使用する際は，空の文字列（""）またはNULL を指定してください。

NameLength1：

*CatalogName の長さ※1，またはSQL_NTS を指定します。

ただし，指定しても無視されます。この関数を使用する際は，0 を指定してください。

SchemaName：

テーブルのスキーマ名に対するパターン文字列※2 を指定します。NULL を指定した場合は，すべてのスキーマ名を対象とします。

NameLength2：

*SchemaName の長さ※1，またはSQL_NTS を指定します。0 を指定した場合は，SchemaName の指定による絞り込みを行いません。

TableName：

テーブル名に対するパターン文字列※2 を指定します。NULL を指定した場合は，すべてのテーブル名を対象とします。

NameLength3：

*TableName の長さ※1，またはSQL_NTS を指定します。0 を指定した場合は，TableName の指定による絞り込みを行いません。

注※1

長さの単位は，SQLTablePrivileges の場合はバイト長，SQLTablePrivilegesW の場合は文字数となります。

注※2

パターン文字列中に指定できる特殊文字については，「表 16-13　パターン文字列中に指定できる特殊文字」を参照してください。

(4)　戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

SQLTablePrivileges またはSQLTablePrivilegesW を実行すると結果セットが作成されます。返却される結果セットの形式を次に示します。

表 16-19　返却される結果セットの形式

列番号	型	列名	列の意味
1	Varchar	TABLE_CAT	カタログ名 常にナル値を返します。
2	Varchar	TABLE_SCHEM	スキーマ名

列番号	型	列名	列の意味
3	Varchar	TABLE_NAME	表名
4	Varchar	GRANTOR	アクセス権限の付与者
5	Varchar	GRANTEE	アクセス権限の被付与者
6	Varchar	PRIVILEGE	許可されているアクセス権限 • "SELECT"：SELECT 権限 • "INSERT"：INSERT 権限 • "UPDATE"：UPDATE 権限 • "DELETE"：DELETE 権限 • "TRUNCATE"：TRUNCATE 権限 • "REFERENCES"：REFERENCES 権限 • "IMPORT TABLE"：IMPORT TABLE 権限 • "REBUILD INDEX"：REBUILD INDEX 権限 • "GET COSTINFO"：GET COSTINFO 権限 • "EXPORT TABLE"：EXPORT TABLE 権限 • "MERGE CHUNK"：MERGE CHUNK 権限 • "CHANGE CHUNK COMMENT"：CHANGE CHUNK COMMENT 権限 • "CHANGE CHUNK STATUS"：CHANGE CHUNK STATUS 権限 • "ARCHIVE CHUNK"：ARCHIVE CHUNK 権限 • "UNARCHIVE CHUNK"：UNARCHIVE CHUNK 権限
7	Varchar	IS_GRANTABLE	アクセス権限の被付与者が，別の HADB ユーザにアクセス権限を与えることができるかどうかを返します。 • "YES"：別の HADB ユーザにアクセス権限を与えられます。 • "NO"：別の HADB ユーザにアクセス権限を与えられません。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態	カーソルがオープンしている状態のときに実行されました。	○
40001	直列化の失敗	－	×
40003	ステートメントの完了が不明		×

SQLSTATE	説明	備考	返却
HY000	一般エラー		×
HY001	メモリ割り当てエラー		○
HY008	動作がキャンセルされた		×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	名前の長さを格納する引数のどれかの値が，対応する名前の最大長を超えました。	○
HY117	接続がサスペンド中	－	×
HYC00	オプション機能は実装されていない		×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

この関数を実行した HADB ユーザが持っている権限によって，取得できる情報が変わります。権限と取得できる情報の対応については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

16.10.10 SQLTables，SQLTablesW

(1) 機能

HADB サーバに格納されたテーブル名，スキーマ名およびテーブル種別の一覧を返します。

SQL 文の結果セットの形式で出力します。

(2) 形式

- SQLTables の場合

```
SQLRETURN SQLTables
(
  SQLHSTMT         StatementHandle,        /* In */
  SQLCHAR        * CatalogName,            /* In */
  SQLSMALLINT      NameLength1,            /* In */
  SQLCHAR        * SchemaName,             /* In */
  SQLSMALLINT      NameLength2,            /* In */
  SQLCHAR        * TableName,              /* In */
  SQLSMALLINT      NameLength3,            /* In */
  SQLCHAR        * TableType,              /* In */
  SQLSMALLINT      NameLength4             /* In */
)
```

- SQLTablesW の場合

```
SQLRETURN SQLTablesW
(
  SQLHSTMT         StatementHandle,        /* In */
  SQLWCHAR       * CatalogName,            /* In */
  SQLSMALLINT      NameLength1,            /* In */
  SQLWCHAR       * SchemaName,             /* In */
  SQLSMALLINT      NameLength2,            /* In */
  SQLWCHAR       * TableName,              /* In */
  SQLSMALLINT      NameLength3,            /* In */
  SQLWCHAR       * TableType,              /* In */
  SQLSMALLINT      NameLength4             /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

CatalogName：

カタログ名として使用します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，空の文字列（""）またはNULL を指定します。

NameLength1：

*CatalogName の長さ※1，またはSQL_NTS を指定します。

指定内容は無効になりますが指定自体は必要なため，この関数を使用する際は，0 を指定します。

SchemaName：

認可識別子または認可識別子に対するパターン文字列※2 を指定します。NULL ポインタまたは空の文字列（""）を指定した場合は，パターン文字列として'%'が指定されたと見なされます。

NameLength2：

*SchemaName の長さ※1，またはSQL_NTS を指定します。

0 を指定した場合，SchemaName にパターン文字列として'%'が指定されたときと同じ動作になります。

TableName：

表識別子または表識別子に対するパターン文字列※2 を指定します。NULL ポインタまたは空の文字列（""）を指定した場合は，パターン文字列として'%'が指定されたと見なされます。

NameLength3：

*TableName の長さ※1，またはSQL_NTS を指定します。

0 を指定した場合，TableName にパターン文字列として'%'が指定されたときと同じ動作になります。

TableType：

一致する表種別を示す文字列を指すポインタを指定します。HADB ODBC ドライバが表種別として認識する文字列は次のとおりです。

- SYSTEM TABLE
- TABLE
- VIEW

指定をする際，任意で各文字列の両端をアポストロフィ(')で囲んでください。また，複数の表種別の結果を取得する場合は，指定値をコンマ（,）で区切ります。

(例)

- ディクショナリ表およびシステム表を取得する場合

  「SYSTEM TABLE」または「'SYSTEM TABLE'」と指定します。

- 実表およびビュー表を取得する場合

  「TABLE,VIEW」または「'TABLE','VIEW'」と指定します。

この引数に NULL ポインタ，空の文字列（""），またはSQL_ALL_TABLE_TYPES を指定した場合，表種別として「'SYSTEM TABLE','TABLE','VIEW'」が指定されたと見なします。

また，HADB ODBC ドライバが表種別として認識する文字列以外を指定した場合，HADB ODBC ドライバが認識する文字列だけが有効となり，それ以外の指定値は無視されます。したがって，無効な文字列だけを指定した場合，NULL ポインタ，空の文字列（""）が指定されたときと同様に，「'SYSTEM TABLE','TABLE','VIEW'」が指定されたと見なします。

NameLength4：

*TableType の長さ※1，またはSQL_NTS を指定します。

0 を指定した場合，TableType に「'SYSTEM TABLE','TABLE','VIEW'」が指定されたと見なされます。

注※1

長さの単位は，SQLTables の場合はバイト長，SQLTablesW の場合は文字数となります。

注※2

パターン文字列中に指定できる特殊文字については，「表 16-13　パターン文字列中に指定できる特殊文字」を参照してください。

(4) 戻り値

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

返却される結果セットの形式を次の表に示します。

表 16-20 返却される結果セットの形式

列番号	型	列名	列の意味
1	Varchar	TABLE_CAT	常にナル値を返します。
2	Varchar	TABLE_SCHEM	スキーマ名
3	Varchar	TABLE_NAME	表名
4	Varchar	TABLE_TYPE	表の種類 • TABLE：実表 • VIEW：ビュー表 • SYSTEM TABLE：ディクショナリ表またはシステム表
5	Varchar	REMARKS	常にナル値を返します。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08S01	通信リンク失敗		×
24000	無効なカーソル状態		×
40001	直列化の失敗		×
40003	ステートメントの完了が不明		×
5C002	文字コードの変換エラー	変換できない文字コードを検出しました。	○
5C041	データ型未サポートエラー	ドライバは指定されたデータ型をサポートしていません。	×
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー	HADB ODBC ドライバは，関数の実行または完了をサポートするために必要なメモリを割り当てられません。	○
HY008	動作がキャンセルされた	－	×
HY009	NULL ポインタの不正使用		×
HY010	関数シーケンスエラー		○

SQLSTATE	説明	備考	返却
HY013	メモリ管理エラー		×
HY090	無効な文字列長または無効なバッファ長	名前の長さを格納する引数のどれかの値が，対応する名前の最大長を超えました。	○
HYC00	オプション機能は実装されていない	－	×
HYT00	タイムアウト終了		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

この関数を実行した HADB ユーザが持っている権限によって，取得できるテーブルの情報が変わります。権限と取得できるテーブルの情報については，マニュアル『HADB　システム構築・運用ガイド』の『HADB ユーザが参照できるディクショナリ表とシステム表の範囲』を参照してください。

16.11　SQL 実行の終了

ここでは，SQL 文の実行終了時に使用する ODBC 関数について説明します。

16.11.1　SQLFreeStmt

(1)　機能

指定したステートメントについて，オプションで指定された処理をします。関連する処理の停止，カーソルのクローズ，未処理結果の廃棄，ステートメントハンドルに関連するすべてのリソースの解放処理などをします。

(2)　形式

```
SQLRETURN SQLFreeStmt
(
  SQLHSTMT       StatementHandle,       /* In */
  SQLUSMALLINT   Option                 /* In */
)
```

(3)　引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

Option：

次のうちのどれかのオプションを指定します。

オプションシンボル	内容
SQL_CLOSE	StatementHandle が管理しているカーソルをクローズします。SQLCloseCursor と同じ処理をしますが，この関数では，カーソルがオープンしていない場合，エラーになりません。SQL_SUCCESS で終了します。
SQL_UNBIND	ARD のSQL_DESC_COUNT フィールドに0 を設定し，StatementHandle に対して SQLBindCol で設定された列のバインドをすべて解除します。
SQL_RESET_PARAMS	APD のSQL_DESC_COUNT フィールドに0 を設定し，StatementHandle に対して SQLBindParameter で設定されたパラメタのバインドをすべて解除します。

(4)　戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
24000	無効なカーソル状態		○
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー	この関数を実行する前にSQLExecute，SQLExecDirect，SQLExecDirectW または SQLParamData のどれかがStatementHandle に対して呼び出され，SQL_NEED_DATA を返しました。その後，実行時データパラメタまたは実行時データ列の設定が完了していません。	○
HY013	メモリ管理エラー	－	×
HY092	無効な属性識別子，または無効なオプション識別子	Option にSQL_DROP または無効な識別子が指定されました。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

Option のSQL_DROP はサポートしていないため，ステートメントハンドルの解放はSQLFreeHandle で行ってください。

また，Option のSQL_UNBIND およびSQL_RESET_PARAMS ではアンバインド処理だけをし，メモリの解放はしません。そのため，SQLBindCol およびSQLBindParameter で HADB ODBC ドライバに渡したデータのメモリ領域の解放は AP で実行してください。

16.11.2 SQLCloseCursor

(1) 機能

指定したステートメントハンドルでオープンされたカーソルをクローズし，未処理の結果を破棄します。

(2) 形式

```
SQLRETURN SQLCloseCursor
(
  SQLHSTMT        StatementHandle       /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

(4) 戻り値

SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
24000	無効なカーソル状態	StatementHandle でオープンしているカーソルがありません。	○
HY000	一般エラー	－	×
HY001	メモリ割り当てエラー	HADB ODBC ドライバは，関数の実行または完了をサポートするのに必要なメモリを割り当てられていません。	×
HY010	関数シーケンスエラー	－	×
HY013	メモリ管理エラー		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.11.3 SQLCancel

(1) 機能

実行中の SQL 文の処理をキャンセルします。

(2) 形式

```
SQLRETURN SQLCancel
(
  SQLHSTMT        StatementHandle         /* In */
)
```

(3) 引数

StatementHandle：

ステートメントハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

(4) 戻り値

SQL_SUCCESS, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

SQLSTATE を返しません。

16.11.4 SQLEndTran

(1) 機能

接続に関連づけられたすべてのステートメント上でのアクティブな処理に対して，コミット処理またはロールバック処理を要求します。

(2) 形式

```
SQLRETURN SQLEndTran
(
```

```
  SQLSMALLINT    HandleType,          /* In */
  SQLHANDLE      Handle,              /* In */
  SQLSMALLINT    CompletionType       /* In */
)
```

(3) 引数

HandleType：

次のハンドルの種類を指定します。

- SQL_HANDLE_DBC：接続ハンドル

Handle：

接続ハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

CompletionType ：

次のどちらかの処理を指定します。

- SQL_COMMIT

  コミット処理

- SQL_ROLLBACK

  ロールバック処理

(4) 戻り値

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
08003	接続が存在しない		○
08007	トランザクション時の接続失敗		×
25S01	トランザクション状態が不明		×
25S02	トランザクションがまだアクティブである		×
25S03	トランザクションがロールバックされた		×
40001	直列化の失敗		×
40002	整合性の不一致		×
HY000	一般エラー		×

SQLSTATE	説明	備考	返却
HY001	メモリ割り当てエラー		×
HY008	動作がキャンセルされた		×
HY010	関数シーケンスエラー		○
HY012	無効なトランザクション処理コード		×
HY013	メモリ管理エラー		×
HY092	無効な属性識別子，または無効なオプション識別子	HandleType またはCompletionType に無効な値が設定されました。	○
HY115	非同期実行中の接続を含む環境を指定した	－	×
HYC00	オプション機能は実装されていない		×
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×
IM017	非同期ポーリングが不正		×
IM018	非同期実行が未完了		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

(6) 注意事項

次のハンドルの種類はサポートしていません。

- SQL_HANDLE_ENV：環境ハンドル

また，次のハンドルの種類を指定した場合，SQL_INVALID_HANDLE が返ることがあります。

- SQL_HANDLE_STMT：ステートメントハンドル
- SQL_HANDLE_DESC：ディスクリプタハンドル

16.12　データソースとの切断

ここでは，データソースとの切断時に使用する ODBC 関数について説明します。

16.12.1　SQLDisconnect

(1)　機能

特定の接続ハンドルに関連づけられた接続を切断します。

(2)　形式

```
SQLRETURN SQLDisconnect
(
  SQLHDBC           ConnectionHandle    /* In */
)
```

(3)　引数

ConnectionHandle：

　接続ハンドルを指定します。

(4)　戻り値

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, またはSQL_INVALID_HANDLE が返されます。

(5)　SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
01002	接続解除エラー		×
08003	接続が存在しない		○
25000	無効なトランザクション状態		○
HY000	一般エラー		×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー		○
HY013	メモリ管理エラー		×

SQLSTATE	説明	備考	返却
HYT01	接続タイムアウト終了		×
IM001	ドライバはこの関数をサポートしていない		×

（凡例）

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.12.2 SQLFreeHandle

(1) 機能

特定の環境，接続，ステートメント，ディスクリプタのハンドルと関連づけられたリソースを解放します。

(2) 形式

```
SQLRETURN SQLFreeHandle
(
  SQLSMALLINT     HandleType,         /* In */
  SQLHANDLE       Handle              /* In */
)
```

(3) 引数

HandleType：

次のどれかのハンドルの種類を指定します。

- SQL_HANDLE_ENV：環境ハンドル
- SQL_HANDLE_DBC：接続ハンドル
- SQL_HANDLE_STMT：ステートメントハンドル
- SQL_HANDLE_DESC：ディスクリプタハンドル

Handle：

解放するハンドルを指定します。

この関数を実行する前にSQLAllocHandle の*OutputHandlePtr で出力された値を指定します。

(4) 戻り値

SQL_SUCCESS，SQL_ERROR，またはSQL_INVALID_HANDLE が返されます。

(5) SQLSTATE

この関数では次のSQLSTATE を返します。

SQLSTATE	説明	備考	返却
01000	一般警告	－	×
HY001	メモリ割り当てエラー		×
HY010	関数シーケンスエラー	• すべての従属ハンドルとコネクションを含むほかのリソースが先に解放されていません。 • HandleType がSQL_HANDLE_STMT ですが，そのステートメントハンドルに対して非同期実行関数が終了していません。	○
HY013	メモリ管理エラー	－	×
HY017	自動的に割り当てられるディスクリプタハンドルの不正使用		×
HY092	無効な属性識別子，または無効なオプション識別子	HandleType に指定できない値が指定されました。	○
HYT01	接続タイムアウト終了	－	×
IM001	ドライバはこの関数をサポートしていない		×

(凡例)

○：HADB ODBC ドライバが返すことがあるSQLSTATE です。

×：HADB ODBC ドライバが返さないSQLSTATE です。

－：なし。

16.13 SQLGetInfo および SQLGetInfoW の InfoType に指定できる情報型

InfoType に指定できる情報型を次の表に示します。

表 16-21 InfoType に指定できる情報型

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
1	SQL_ACCESSIBLE_PROCEDURES	次のどちらかの値を返却します。 Y SQLProcedures がプロシジャをすべて実行できます。 N SQLProcedures が実行できないプロシジャがあります。	"N"を返却します。	• SQLCHAR＊ • SQLWCHAR＊
2	SQL_ACCESSIBLE_TABLES	次のどちらかの値を返却します。 Y SQLTables が返すすべてのテーブルに対してSELECT 特権が保証されます。 N SQLTables が返すテーブルに AP がアクセスできないものがあります。	"Y"を返却します。	• SQLCHAR＊ • SQLWCHAR＊
3	SQL_ACTIVE_ENVIRONMENTS	ドライバがサポートするアクティブな環境の最大数を返却します。制限が指定されていない，または不明な場合，この値には0 を返却します。	0 を返却します。	SQLUSMALLINT
4	SQL_AGGREGATE_FUNCTIONS	集計関数に関するサポート状況を示します。	次のビット列を返却します。 • SQL_AF_ALL • SQL_AF_AVG • SQL_AF_COUNT • SQL_AF_DISTINCT • SQL_AF_MAX • SQL_AF_MIN • SQL_AF_SUM	SQLUINTEGER
5	SQL_ALTER_DOMAIN	データソースがサポートしている ALTER DOMAIN ステートメントの句を示します。 戻り値が0 の場合，ALTER DOMAIN ステートメントがサポートされていないことを示します。	0 を返却します。	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
6	SQL_ALTER_TABLE	データソースがサポートしている ALTER TABLE ステートメントの句を示します。	環境ハンドルの SQL_ATTR_ODBC_VERSION の指定値に従い，次のビット列を返却します。 SQL_OV_ODBC2 の場合 • SQL_AT_ADD_COLUMN • SQL_AT_DROP_COLUMN SQL_OV_ODBC3 の場合 • SQL_AT_ADD_COLUMN_SINGLE • SQL_AT_DROP_COLUMN_CASCADE • SQL_AT_DROP_COLUMN_RESTRICT	SQLUINTEGER
7	SQL_ASYNC_MODE	ドライバでの非同期サポートのレベルを示します。	SQL_AM_NONE を返却します。	SQLUINTEGER
8	SQL_BATCH_ROW_COUNT	利用可能な行カウントに対するドライバの動作を示します。	0 を返却します。	SQLUINTEGER
9	SQL_BATCH_SUPPORT	ドライバがサポートするバッチを示します。	0 を返却します。	SQLUINTEGER
10	SQL_BOOKMARK_PERSISTENCE	ブックマークが有効になる処理を示します。	0 を返却します。	SQLUINTEGER
11	SQL_CATALOG_LOCATION	修飾テーブル名のカタログの位置を示す値を返却します。	0 を返却します。	SQLUSMALLINT
12	SQL_CATALOG_NAME	次のどちらかの値を返却します。 Y サーバはカタログ名をサポートしています。 N サーバはカタログ名をサポートしていません。	"N"を返却します。	• SQLCHAR * • SQLWCHAR *
13	SQL_CATALOG_NAME_SEPARATOR	区切り文字として定義する 1 文字以上の文字列を返却します。この区切り文字は，カタログ名とその前後にある要素の間に置かれます。データソースがカタログをサポートしていない場合，空の文字列を返却します。	空の文字列を返却します。	• SQLCHAR * • SQLWCHAR *
14	SQL_CATALOG_TERM	データソースのベンダー名を含む文字列をカタログとして返却します。データソースがカタログをサポート	空の文字列を返却します。	• SQLCHAR * • SQLWCHAR *

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
		していない場合，空の文字列を返却します。		
15	SQL_CATALOG_USAGE	カタログを使用できるステートメントを示します。 データソースがカタログをサポートしていない場合，0 を返却します。	0 を返却します。	SQLUINTEGER
16	SQL_COLLATION_SEQ	照合の順序（サーバのデフォルト文字セットに対するデフォルトの照合）の名前を返却します。名前が不明な場合，空の文字列を返却します。	空の文字列を返却します。	• SQLCHAR * • SQLWCHAR *
17	SQL_COLUMN_ALIAS	次のどちらかの値を返却します。 Y データソースは列の別名をサポートしています。 N データソースは列の別名をサポートしていません。	"Y"を返却します。	• SQLCHAR * • SQLWCHAR *
18	SQL_CONCAT_NULL_BEHAVIOR	データソースがナル値の文字データ型の列をナル値以外のものと連結する方法を示します。	SQL_CB_NULL を返却します。	SQLUSMALLINT
19	SQL_CONVERT_BIGINT	データソースがサポートしている，CONVERT スカラ関数による変換を示します。	次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_CHAR • SQL_CVT_DATE • SQL_CVT_DECIMAL • SQL_CVT_DOUBLE • SQL_CVT_FLOAT • SQL_CVT_INTEGER • SQL_CVT_NUMERIC • SQL_CVT_TIMESTAMP • SQL_CVT_VARCHAR	SQLUINTEGER
20	SQL_CONVERT_BINARY		次のビット列を返却します。 • SQL_CVT_BINARY • SQL_CVT_CHAR • SQL_CVT_VARBINARY • SQL_CVT_VARCHAR	SQLUINTEGER
21	SQL_CONVERT_BIT		0 を返却します。	SQLUINTEGER
22	SQL_CONVERT_CHAR		次のビット列を返却します。	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
			• SQL_CVT_BIGINT • SQL_CVT_BINARY • SQL_CVT_CHAR • SQL_CVT_DATE • SQL_CVT_DECIMAL • SQL_CVT_DOUBLE • SQL_CVT_FLOAT • SQL_CVT_INTEGER • SQL_CVT_NUMERIC • SQL_CVT_TIME • SQL_CVT_TIMESTAMP • SQL_CVT_VARBINARY • SQL_CVT_VARCHAR	
23	SQL_CONVERT_DATE		次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_CHAR • SQL_CVT_DATE • SQL_CVT_INTEGER • SQL_CVT_TIMESTAMP • SQL_CVT_VARCHAR	SQLUINTEGER
24	SQL_CONVERT_DECIMAL		次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_CHAR • SQL_CVT_DECIMAL • SQL_CVT_DOUBLE • SQL_CVT_FLOAT • SQL_CVT_INTEGER • SQL_CVT_NUMERIC • SQL_CVT_VARCHAR	SQLUINTEGER
25	SQL_CONVERT_DOUBLE		次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_CHAR • SQL_CVT_DECIMAL • SQL_CVT_DOUBLE • SQL_CVT_FLOAT • SQL_CVT_INTEGER • SQL_CVT_NUMERIC • SQL_CVT_VARCHAR	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
26	SQL_CONVERT_FLOAT		次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_CHAR • SQL_CVT_DECIMAL • SQL_CVT_DOUBLE • SQL_CVT_FLOAT • SQL_CVT_INTEGER • SQL_CVT_NUMERIC • SQL_CVT_VARCHAR	SQLUINTEGER
27	SQL_CONVERT_INTEGER		次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_CHAR • SQL_CVT_DATE • SQL_CVT_DECIMAL • SQL_CVT_DOUBLE • SQL_CVT_FLOAT • SQL_CVT_INTEGER • SQL_CVT_NUMERIC • SQL_CVT_TIMESTAMP • SQL_CVT_VARCHAR	SQLUINTEGER
28	SQL_CONVERT_INTERVAL_YEAR_MONTH		0 を返却します。	SQLUINTEGER
29	SQL_CONVERT_INTERVAL_DAY_TIME		0 を返却します。	SQLUINTEGER
30	SQL_CONVERT_LONGVARBINARY		0 を返却します。	SQLUINTEGER
31	SQL_CONVERT_LONGVARCHAR		0 を返却します。	SQLUINTEGER
32	SQL_CONVERT_NUMERIC		次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_CHAR • SQL_CVT_DECIMAL • SQL_CVT_DOUBLE • SQL_CVT_FLOAT • SQL_CVT_INTEGER • SQL_CVT_NUMERIC • SQL_CVT_VARCHAR	SQLUINTEGER
33	SQL_CONVERT_REAL		0 を返却します。	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
34	SQL_CONVERT_SMALLINT		0 を返却します。	SQLUINTEGER
35	SQL_CONVERT_TIME		次のビット列を返却します。 • SQL_CVT_CHAR • SQL_CVT_TIME • SQL_CVT_VARCHAR	SQLUINTEGER
36	SQL_CONVERT_TIMESTAMP		次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_CHAR • SQL_CVT_DATE • SQL_CVT_INTEGER • SQL_CVT_TIMESTAMP • SQL_CVT_VARCHAR	SQLUINTEGER
37	SQL_CONVERT_TINYINT		0 を返却します。	SQLUINTEGER
38	SQL_CONVERT_VARBINARY		次のビット列を返却します。 • SQL_CVT_BINARY • SQL_CVT_CHAR • SQL_CVT_VARBINARY • SQL_CVT_VARCHAR	SQLUINTEGER
39	SQL_CONVERT_VARCHAR		次のビット列を返却します。 • SQL_CVT_BIGINT • SQL_CVT_BINARY • SQL_CVT_CHAR • SQL_CVT_DATE • SQL_CVT_DECIMAL • SQL_CVT_DOUBLE • SQL_CVT_FLOAT • SQL_CVT_INTEGER • SQL_CVT_NUMERIC • SQL_CVT_TIME • SQL_CVT_TIMESTAMP • SQL_CVT_VARBINARY • SQL_CVT_VARCHAR	SQLUINTEGER
40	SQL_CONVERT_FUNCTIONS	ドライバおよび関連するデータソースがサポートしているスカラ変換関数を示します。	次のビット列を返却します。 • SQL_FN_CVT_CAST • SQL_FN_CVT_CONVERT	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
41	SQL_CORRELATION_NAME	テーブルの相関名をサポートするかどうかを示します。	SQL_CN_ANY を返却します。	SQLUSMALLINT
42	SQL_CREATE_ASSERTION	データソースがサポートするCREATE ASSERTION ステートメントの句を示します。 戻り値が0 の場合，CREATE ASSERTION ステートメントがサポートされていないことを示します。	0 を返却します。	SQLUINTEGER
43	SQL_CREATE_CHARACTER_SET	データソースがサポートするCREATE CHARACTER SET ステートメントの句を示します。 戻り値が0 の場合，CREATE CHARACTER SET ステートメントがサポートされていないことを示します。	0 を返却します。	SQLUINTEGER
44	SQL_CREATE_COLLATION	データソースがサポートするCREATE COLLATION ステートメントの句を示します。 戻り値が0 の場合，CREATE COLLATION ステートメントがサポートされていないことを示します。	0 を返却します。	SQLUINTEGER
45	SQL_CREATE_DOMAIN	データソースがサポートするCREATE DOMAIN ステートメントの句を示します。 戻り値が0 の場合，CREATE DOMAIN ステートメントがサポートされていないことを示します。	0 を返却します。	SQLUINTEGER
46	SQL_CREATE_SCHEMA	データソースがサポートするCREATE SCHEMA ステートメントの句を示します。 戻り値が0 の場合，CREATE SCHEMA ステートメントがサポートされていないことを示します。	SQL_CS_CREATE_SCHEMA を返却します。	SQLUINTEGER
47	SQL_CREATE_TABLE	データソースがサポートするCREATE TABLE ステートメントの句を示します。 戻り値が0 の場合，CREATE TABLE ステートメントがサポートされていないことを示します。	次のビット列を返却します。 • SQL_CT_CREATE_TABLE • SQL_CT_TABLE_CONSTRAINT • SQL_CT_CONSTRAINT_NAME_DEFINITION • SQL_CT_COLUMN_DEFAULT	SQLUINTEGER
48	SQL_CREATE_TRANSLATION	データソースがサポートするCREATE TRANSLATION ステートメントの句を示します。	0 を返却します。	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
		戻り値が0 の場合，CREATE TRANSLATION ステートメントがサポートされていないことを示します。		
49	SQL_CREATE_VIEW	データソースがサポートするCREATE VIEW ステートメントの句を示します。 戻り値が0 の場合，CREATE VIEW ステートメントがサポートされていないことを示します。	SQL_CV_CREATE_VIEW を返却します。	SQLUINTEGER
50	SQL_CURSOR_COMMIT_BEHAVIOR	COMMIT 処理がカーソルとデータソースの準備されたステートメントに与える影響を示します。	SQL_CB_CLOSE を返却します。	SQLUSMALLINT
51	SQL_CURSOR_ROLLBACK_BEHAVIOR	ROLLBACK 処理がカーソルとデータソースの準備されたステートメントに与える影響を示します。	SQL_CB_CLOSE を返却します。	SQLUSMALLINT
52	SQL_CURSOR_SENSITIVITY	カーソルの動作に対するサポートを示します。	SQL_UNSPECIFIED を返却します。	SQLUINTEGER
53	SQL_DATA_SOURCE_NAME	接続中に使用するデータソース名の文字列を示します。	空の文字列または接続に使用したデータソース名を返却します。	• SQLCHAR * • SQLWCHAR *
54	SQL_DATA_SOURCE_READ_ONLY	次のどちらかの値を返却します。 Y データソースのモードがREAD ONLY N データソースのモードがREAD ONLY 以外	"N"を返却します。	• SQLCHAR * • SQLWCHAR *
55	SQL_DATABASE_NAME	データソースが”database”という名前付きのオブジェクトを定義する場合，現在使用中のデータベース名の文字列になります。	空の文字列を返却します。	• SQLCHAR * • SQLWCHAR *
56	SQL_DATETIME_LITERALS	データソースがサポートするSQL-92 の日付時刻リテラルを示します。	次のビット列を返却します。 • SQL_DL_SQL92_DATE • SQL_DL_SQL92_TIME • SQL_DL_SQL92_TIMESTAMP	SQLUINTEGER
57	SQL_DBMS_NAME	ドライバがアクセスするDBMS 製品の名前を示す文字列です。	”Hitachi Advanced Data Binder”を返却します。	• SQLCHAR * • SQLWCHAR *
58	SQL_DBMS_VER	ドライバがアクセスするDBMS 製品のバージョンを示す文字列です。	DBMS 製品のバージョンを返却します。	• SQLCHAR * • SQLWCHAR *

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
		バージョンは，*##.##.####*の形式で表します。		
59	SQL_DDL_INDEX	インデクスの作成および削除に対するサポートを示します。	次のビット列を返却します。 • SQL_DI_CREATE_INDEX • SQL_DI_DROP_INDEX	SQLUINTEGER
60	SQL_DEFAULT_TXN_ISOLATION	ドライバまたはデータソースがサポートする，デフォルトのトランザクション隔離性水準を示します。	SQL_TXN_READ_COMMITTED を返却します。	SQLUINTEGER
61	SQL_DESCRIBE_PARAMETER	次のどちらかの値を返却します。 Y パラメタを記述できます。 N パラメタを記述できません。	"Y"を返却します。	• SQLCHAR * • SQLWCHAR *
62	SQL_DM_VER	ドライバマネージャのバージョンを示す文字列です。バージョンは*##.##.####.####*の形式で表します。この情報型はドライバマネージャが実装します。	ドライバマネージャが設定する値を返却します。	• SQLCHAR * • SQLWCHAR *
63	SQL_DRIVER_HDBC	ドライバの接続ハンドルで，InfoType で調べることができます。この情報型はドライバマネージャが実装します。	ドライバマネージャが設定する値を返却します。	SQLUINTEGER
64	SQL_DRIVER_HDESC	ドライバマネージャのディスクリプタハンドルによって決まるドライバのディスクリプタハンドルで，アプリケーションが*InfoValuePtr に設定して渡す必要があります。この情報型はドライバマネージャが実装します。	ドライバマネージャが設定する値を返却します。	SQLUINTEGER
65	SQL_DRIVER_HENV	ドライバの環境ハンドルで，InfoType で調べることができます。この情報型はドライバマネージャが実装します。	ドライバマネージャが設定する値を返却します。	SQLUINTEGER
66	SQL_DRIVER_HLIB	ドライバ DLL またはそれに相当するものをロードしたときに，ロードライブラリからドライバマネージャに返されるhinst です。この情報型はドライバマネージャが実装します。	ドライバマネージャが設定する値を返却します。	SQLUINTEGER
67	SQL_DRIVER_HSTMT	ドライバマネージャのステートメントハンドルによって決まるドライバのステートメントハンドルで，アプリケーションが*InfoValuePtr に設	ドライバマネージャが設定する値を返却します。	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
		定して渡す必要があります。この情報型は，ドライバマネージャが実装します。		
68	SQL_DRIVER_NAME	データソースにアクセスするために使用するドライバのファイル名を示す文字列です。	ドライバのファイル名を返却します。	• SQLCHAR * • SQLWCHAR *
69	SQL_DRIVER_ODBC_VER	ドライバがサポートするODBCのバージョンを示す文字列です。バージョンは##.##の形式で表します。	ドライバがサポートするODBCのバージョンを返却します。	• SQLCHAR * • SQLWCHAR *
70	SQL_DRIVER_VER	ドライバのバージョンを示す文字列です。通常，ドライバが値を返却します。最小のバージョンは，##.##.####の形式で表します。	ドライバのバージョンを返却します。	• SQLCHAR * • SQLWCHAR *
71	SQL_DROP_ASSERTION	データソースがサポートするDROP ASSERTIONステートメントの句（SQL-92に定義されている）を示します。	0を返却します。	SQLUINTEGER
72	SQL_DROP_CHARACTER_SET	データソースがサポートするDROP CHARACTER SETステートメントの句（SQL-92に定義されている）を示します。	0を返却します。	SQLUINTEGER
73	SQL_DROP_COLLATION	データソースがサポートするDROP COLLATIONステートメントの句（SQL-92に定義されている）を示します。	0を返却します。	SQLUINTEGER
74	SQL_DROP_DOMAIN	データソースがサポートするDROP DOMAINステートメントの句（SQL-92に定義されている）を示します。	0を返却します。	SQLUINTEGER
75	SQL_DROP_SCHEMA	データソースがサポートするDROP SCHEMAステートメントの句（SQL-92に定義されている）を示します。	次のビット列を返却します。 • SQL_DS_DROP_SCHEMA • SQL_DS_CASCADE • SQL_DS_RESTRICT	SQLUINTEGER
76	SQL_DROP_TABLE	データソースがサポートするDROP TABLEステートメントの句（SQL-92に定義されている）を示します。	次のビット列を返却します。 • SQL_DT_DROP_TABLE • SQL_DT_CASCADE • SQL_DT_RESTRICT	SQLUINTEGER
77	SQL_DROP_TRANSLATION	データソースがサポートするDROP TRANSLATIONステートメントの句を示します。	0を返却します。	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
78	SQL_DROP_VIEW	データソースがサポートするDROP VIEW ステートメントの句を示します。	SQL_DV_DROP_VIEW を返却します。	SQLUINTEGER
79	SQL_DYNAMIC_CURSOR_ATTRIBUTES1	ドライバがサポートしている動的カーソルの属性を示します。このビットマスクには，属性の最初のサブセットが格納されます。	0 を返却します。	SQLUINTEGER
80	SQL_DYNAMIC_CURSOR_ATTRIBUTES2	ドライバがサポートしている動的カーソルの属性を示します。このビットマスクには，属性の 2 番目のサブセットが格納されます。	0 を返却します。	SQLUINTEGER
81	SQL_EXPRESSIONS_IN_ORDERBY	次のどちらかの値を返却します。 Y データソースはORDER BY 一覧の式をサポートしています。 N データソースはORDER BY 一覧の式をサポートしていません。	"Y"を返却します。	• SQLCHAR * • SQLWCHAR *
82	SQL_FILE_USAGE	単一層ドライバがデータソースのファイルを直接扱う方法を示します。	SQL_FILE_NOT_SUPPORTED を返却します。	SQLUSMALLINT
83	SQL_FORWARD_ONLY_CURSOR_ATTRIBUTES1	ドライバがサポートしている，前方にしか進めないカーソルの属性を示します。 このビットマスクには，属性の最初のサブセットが格納されます。	0 を返却します。	SQLUINTEGER
84	SQL_FORWARD_ONLY_CURSOR_ATTRIBUTES2	ドライバがサポートしている，前方にしか進めないカーソルの属性を示します。このビットマスクには，属性の 2 番目のサブセットが格納されます。	0 を返却します。	SQLUINTEGER
85	SQL_GETDATA_EXTENSIONS	SQLGetData の拡張機能を示します。	次の値を返却します。 • SQL_GD_ANY_COLUMN • SQL_GD_ANY_ORDER	SQLUINTEGER
86	SQL_GROUP_BY	GROUP BY 句の列と選択一覧にある集合関数以外の列の関係を示します。	SQL_GB_GROUP_BY_CONTAINS_SELECT を返却します。	SQLUSMALLINT
87	SQL_IDENTIFIER_CASE	SQL の識別子に関する情報を示します。	SQL_IC_UPPER を返却します。	SQLUSMALLINT
88	SQL_IDENTIFIER_QUOTE_CHAR	SQL ステートメントの引用符付き識別子の開始および終端を示す記号として使用する文字列を示します。データソースが引用符付き識別子を	二重引用符（”）を返却します。	• SQLCHAR * • SQLWCHAR *

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
		サポートしていない場合，空白を返却します。		
89	SQL_INDEX_KEYWORDS	ドライバがサポートするCREATE INDEX ステートメント中のキーワードを示します。	次のビット列を返却します。 • SQL_IK_ASC • SQL_IK_DESC	SQLUINTEGER
90	SQL_INFO_SCHEMA_VIEWS	ドライバがサポートするINFORMATION_SCHEMA のビューを示します。	0 を返却します。	SQLUINTEGER
91	SQL_INSERT_STATEMENT	INSERT ステートメントのサポートを示します。	次のビット列を返却します。 • SQL_IS_INSERT_LITERALS • SQL_IS_INSERT_SEARCHED	SQLUINTEGER
92	SQL_INTEGRITY	次のどちらかの値を返却します。 Y データソースは IEF をサポートしています。 N データソースは IEF をサポートしていません。	"N"を返却します。	• SQLCHAR * • SQLWCHAR *
93	SQL_KEYSET_CURSOR_ATTRIBUTES1	ドライバがサポートしているキーセットカーソルの属性を示します。このビットマスクには，属性の最初のサブセットが格納されます。	0 を返却します。	SQLUINTEGER
94	SQL_KEYSET_CURSOR_ATTRIBUTES2	ドライバがサポートしているキーセットカーソルの属性を示します。このビットマスクには，属性の 2 番目のサブセットが格納されます。	0 を返却します。	SQLUINTEGER
95	SQL_KEYWORDS	データソース指定のすべてのキーワードをコンマで区切った一覧を含む文字列を示します。この一覧にはODBC 固有のキーワードまたはデータソースと ODBC の両方が使用するキーワードは含みません。この一覧はすべての予約キーワードを示します。	HADB の予約語からODBC 固有のキーワードを除いたキーワードを返却します。	• SQLCHAR * • SQLWCHAR *
96	SQL_LIKE_ESCAPE_CLAUSE	次のどちらかの値を返却します。 Y 次の条件をすべて満たします。	"Y"を返却します。	• SQLCHAR * • SQLWCHAR *

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
		• データソースがLIKE 述語のパーセント記号（%）とアンダースコア（_）のエスケープ文字をサポートしています。 • ドライバがLIKE 述語のエスケープ文字を定義するためにODBC 構文をサポートしています。 N 上記以外。		
97	SQL_MAX_ASYNC_CONCURRENT_STATEMENTS	指定の接続上でドライバがサポートする，非同期モードのアクティブな同時実行ステートメントの最大数を示します。特定の制限がない，または制限が不明な場合，0 を返却します。	0 を返却します。	SQLUINTEGER
98	SQL_MAX_BINARY_LITERAL_LEN	SQL ステートメントのバイナリリテラルの最大値を示します。最大値がない，または長さが不明な場合，0 を返却します。	64,000 を返却します。	SQLUINTEGER
99	SQL_MAX_CATALOG_NAME_LEN	カタログ名の最大長を示します。最大長がない，または長さが不明な場合，0 を返却します。	0 を返却します。	SQLUSMALLINT
100	SQL_MAX_CHAR_LITERAL_LEN	SQL ステートメントの文字リテラルの最大長を示します。最大長がない，または長さが不明な場合，0 を返却します。	32,000 を返却します。	SQLUINTEGER
101	SQL_MAX_COLUMN_NAME_LEN	データソースの列名の最大長を示します。最大長がない，または長さが不明な場合，0 を返却します。	100 を返却します。	SQLUSMALLINT
102	SQL_MAX_COLUMNS_IN_GROUP_BY	GROUP BY 句に許されるグループ化列の最大数を示します。指定がない，または制限が不明な場合，0 を返却します。	64 を返却します。	SQLUSMALLINT
103	SQL_MAX_COLUMNS_IN_INDEX	インデクスに許される列の最大数を示します。指定がない，または制限が不明な場合，0 を返却します。	16 を返却します。	SQLUSMALLINT
104	SQL_MAX_COLUMNS_IN_ORDER_BY	ORDER BY 句に許される列の最大数を示します。指定がない，または制限が不明な場合，0 を返却します。	16 を返却します。	SQLUSMALLINT

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
105	SQL_MAX_COLUMNS_IN_SELECT	選択一覧に許される列の最大数を示します。指定がない，または制限が不明な場合，0 を返却します。	4,000 を返却します。	SQLUSMALLINT
106	SQL_MAX_COLUMNS_IN_TABLE	テーブルに許される列の最大数を示します。指定がない，または制限が不明な場合，0 を返却します。	4,000 を返却します。	SQLUSMALLINT
107	SQL_MAX_CONCURRENT_ACTIVITIES	ドライバが接続に対してサポートするアクティブなステートメントの最大数を示します。制限が指定されていない，または不明な場合，0 を返却します。	0 を返却します。	SQLUSMALLINT
108	SQL_MAX_CURSOR_NAME_LEN	データソースのカーソル名に許される最大長を示します。最大長がない，または長さが不明な場合，0 を返却します。	0 を返却します。	SQLUSMALLINT
109	SQL_MAX_DRIVER_CONNECTIONS	ドライバが環境に対してサポートするアクティブな接続の最大数を示します。制限が指定されていない，または不明な場合，0 を返却します。	0 を返却します。	SQLUSMALLINT
110	SQL_MAX_IDENTIFIER_LEN	データソースがユーザ定義名に対してサポートする文字列の最大長を示します。	100 を返却します。	SQLUSMALLINT
111	SQL_MAX_INDEX_SIZE	インデクスの結合フィールドに許される最大長（バイト長）を示します。指定がない，または制限が不明な場合，0 を返却します。	4,036 を返却します。	SQLUINTEGER
112	SQL_MAX_PROCEDURE_NAME_LEN	データソースのプロシジャ名の最大長を示します。最大長がない，または長さが不明な場合，0 を返却します。	0 を返却します。	SQLUSMALLINT
113	SQL_MAX_ROW_SIZE	テーブルの単一行に許される最大長を示します。指定がない，または制限が不明な場合，0 を返却します。	0 を返却します。	SQLUINTEGER
114	SQL_MAX_ROW_SIZE_INCLUDES_LONG	次のどちらかの値を返却します。 Y SQL_MAX_ROW_SIZE 情報型に対して返された行サイズの最大長が，行のすべてのSQL_LONGVARCHAR 型の列の長さと SQL_LONGVARBINARY 型の列の長さを含みます。 N 上記以外。	"N"を返却します。	• SQLCHAR * • SQLWCHAR *

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
115	SQL_MAX_SCHEMA_NAME_LEN	データソースのスキーマ名の最大長を示します。最大長がない，または長さが不明な場合，0 を返却します。	100 を返却します。	SQLUSMALLINT
116	SQL_MAX_STATEMENT_LEN	SQL ステートメントの最大長（文字数）を示します。最大長がない，または長さが不明な場合，0 を返却します。	16,000,000 を返却します。	SQLUINTEGER
117	SQL_MAX_TABLE_NAME_LEN	データソーステーブル名の最大長を示します。最大長がない，または長さが不明な場合，0 を返却します。	100 を返却します。	SQLUSMALLINT
118	SQL_MAX_TABLES_IN_SELECT	SELECT ステートメントのFROM 句に許されるテーブル数の最大値を示します。指定がない，または制限が不明な場合，0 を返却します。	64 を返却します。	SQLUSMALLINT
119	SQL_MAX_USER_NAME_LEN	データソースのユーザ名の最大長を示します。最大長がない，または長さが不明な場合，0 を返却します。	100 を返却します。	SQLUSMALLINT
120	SQL_MULT_RESULT_SETS	次のどちらかの値を返却します。 Y データソースは複数の結果セットをサポートしています。 N データソースは複数の結果セットをサポートしていません。	"N"を返却します。	• SQLCHAR * • SQLWCHAR *
121	SQL_MULTIPLE_ACTIVE_TXN	次のどちらかの値を返却します。 Y ドライバは複数のアクティブなトランザクションの同時実行をサポートしています。 N ドライバは単一のアクティブなトランザクションの実行だけをサポートしています。	"Y"を返却します。	• SQLCHAR * • SQLWCHAR *
122	SQL_NEED_LONG_DATA_LEN	次のどちらかの値を返却します。 Y データソースはロングデータ値が送られる前に，そのデータ長が必要です。 N データソースはロングデータ値が送られる前に，そのデータ長を必要としません。	"Y"を返却します。	• SQLCHAR * • SQLWCHAR *

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
123	SQL_NON_NULLABLE_COLUMNS	データソースNOT NULL の列をサポートするかどうかを示します。	SQL_NNC_NON_NULL を返却します。	SQLUSMALLINT
124	SQL_NULL_COLLATION	NULL が結果セット中のどこにソートされるかを示します。	SQL_NC_HIGH を返却します。	SQLUSMALLINT
125	SQL_NUMERIC_FUNCTIONS	ドライバとデータソースがサポートしている数学関数を示します。	次のビット列を返却します。 • SQL_FN_NUM_ABS • SQL_FN_NUM_ACOS • SQL_FN_NUM_ASIN • SQL_FN_NUM_ATAN • SQL_FN_NUM_ATAN2 • SQL_FN_NUM_CEILING • SQL_FN_NUM_COS • SQL_FN_NUM_DEGREES • SQL_FN_NUM_EXP • SQL_FN_NUM_FLOOR • SQL_FN_NUM_LOG • SQL_FN_NUM_MOD • SQL_FN_NUM_PI • SQL_FN_NUM_POWER • SQL_FN_NUM_RADIANS • SQL_FN_NUM_RAND • SQL_FN_NUM_ROUND • SQL_FN_NUM_SIGN • SQL_FN_NUM_SIN • SQL_FN_NUM_SQRT • SQL_FN_NUM_TAN • SQL_FN_NUM_TRUNCATE	SQLUINTEGER
126	SQL_ODBC_INTERFACE_CONFORMANCE	ドライバが準拠する ODBC 3.x のインタフェースのレベルを示します。	SQL_OIC_CORE を返却します。	SQLUINTEGER
127	SQL_ODBC_VER	ドライバマネージャが準拠する ODBC のバージョン情報を含む文字列です。バージョンは *##.##.0000* の形式で表します。この情報型はドライバマネージャが実装します。	ドライバマネージャが設定する値を返却します。	• SQLCHAR * • SQLWCHAR *
128	SQL_OJ_CAPABILITIES	ドライバおよびデータソースがサポートする外結合の種類を示します。	次のビット列を返却します。 • SQL_OJ_ALL_COMPARISON_OPS • SQL_OJ_FULL • SQL_OJ_INNER	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
			• SQL_OJ_LEFT • SQL_OJ_NESTED • SQL_OJ_RIGHT	
129	SQL_ORDER_BY_COLUMNS_IN_SELECT	次のどちらかの値を返却します。 Y ORDER BY 句の列が選択一覧にあります。 N ORDER BY 句の列は選択一覧にありません。	"N"を返却します。	• SQLCHAR * • SQLWCHAR *
130	SQL_PARAM_ARRAY_ROW_COUNTS	パラメタを使った実行で行カウントを利用できるかどうかに関するドライバのプロパティを示します。	0 を返却します。	SQLUINTEGER
131	SQL_PARAM_ARRAY_SELECTS	パラメタ付きの実行結果セットを利用できるかどうかに関するドライバのプロパティを示します。	SQL_PAS_NO_SELECT を返却します。	SQLUINTEGER
132	SQL_PROCEDURE_TERM	プロシジャに対するデータソースのベンダーの名前を含む文字列を示します。	空の文字列を返却します。	• SQLCHAR * • SQLWCHAR *
133	SQL_PROCEDURES	次のどちらかの値を返却します。 Y 次の条件をすべて満たします。 ・データソースはプロシジャをサポートしています。 ・ドライバは ODBC プロシジャ起動構文をサポートしています。 N 上記以外。	"N"を返却します。	• SQLCHAR * • SQLWCHAR *
134	SQL_QUOTED_IDENTIFIER_CASE	SQL ステートメントの引用符付き識別子に関する情報を示します。	SQL_IC_SENSITIVE を返却します。	SQLUSMALLINT
135	SQL_ROW_UPDATES	次のどちらかの値を返却します。 Y キーセットカーソルまたは複合カーソルがフェッチされたすべての行に対して行のバージョンまたは値を保持し，その行が最後にフェッチされてから AP が行った行の更新を検出できます。 N 上記以外。	"N"を返却します。	• SQLCHAR * • SQLWCHAR *

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
136	SQL_SCHEMA_TERM	スキーマに対するデータソースベンダーの名前を含む文字列を示します。	”schema”を返却します。	• SQLCHAR＊ • SQLWCHAR＊
137	SQL_SCHEMA_USAGE	スキーマを使用できるステートメントを示します。	次のビット列を返却します。 • SQL_SU_DML_STATEMENTS • SQL_SU_INDEX_DEFINITION • SQL_SU_TABLE_DEFINITION • SQL_SU_PRIVILEGE_DEFINITION	SQLUINTEGER
138	SQL_SCROLL_OPTIONS	スクロール可能なカーソルに対してサポートされているスクロールオプションを示します。	SQL_SO_FORWARD_ONLY を返却します。	SQLUINTEGER
139	SQL_SEARCH_PATTERN_ESCAPE	パターン検索のメタ文字であるアンダースコア（_）とパーセント記号（%）を検索パターンの有効な文字として使用するエスケープ文字を，ドライバがサポートすることを示す文字列を示します。 パターン検索を行う際に，通常はメタ文字であるアンダースコア（_）とパーセント記号（%）を有効な検索文字にするためのエスケープ文字を示します。 このエスケープ文字は，検索文字列をサポートするカタログ関数の引数にだけ適用されます。空の文字列の場合，ドライバはアンダースコア（_）とパーセント記号（%）を検索パターンの文字列として扱えません。	"¥"を返却します。	• SQLCHAR＊ • SQLWCHAR＊
140	SQL_SERVER_NAME	実際のデータソース指定のサーバ名を含む文字列です。	”Hitachi Advanced Data Binder”を返却します。	• SQLCHAR＊ • SQLWCHAR＊
141	SQL_SPECIAL_CHARACTERS	データソースでテーブル，列，インデクス名などのように識別子名として使用できるすべての特殊文字を含む文字列を返却します。	空の文字列を返却します。	• SQLCHAR＊ • SQLWCHAR＊
142	SQL_SQL_CONFORMANCE	ドライバがサポートする SQL-92 のレベルを示します。	SQL_SC_SQL92_ENTRY を返却します。	SQLUINTEGER
143	SQL_SQL92_DATETIME_FUNCTIONS	ドライバおよび関連するデータソースがサポートする日付時刻スカラ関数を示します。	次のビット列を返却します。 • SQL_SDF_CURRENT_DATE • SQL_SDF_CURRENT_TIME	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
			• SQL_SDF_CURRENT_TIMESTAMP	
144	SQL_SQL92_FOREIGN_KEY_DELETE_RULE	DELETE ステートメントの外部キーに対してサポートされている規則を示します。	0 を返却します。	SQLUINTEGER
145	SQL_SQL92_FOREIGN_KEY_UPDATE_RULE	UPDATE ステートメントの外部キーに対してサポートされている規則を示します。	0 を返却します。	SQLUINTEGER
146	SQL_SQL92_GRANT	GRANT ステートメントに対してサポートされている句を示します。	次のビット列を返却します。 • SQL_SG_DELETE_TABLE • SQL_SG_INSERT_TABLE • SQL_SG_REFERENCES_TABLE • SQL_SG_SELECT_TABLE • SQL_SG_UPDATE_TABLE	SQLUINTEGER
147	SQL_SQL92_NUMERIC_VALUE_FUNCTIONS	ドライバおよび関連するデータソースがサポートする数値スカラ関数を示します。	次のビット列を返却します。 • SQL_SNVF_EXTRACT	SQLUINTEGER
148	SQL_SQL92_PREDICATES	SELECT ステートメントに対してサポートされている述語を示します。	次のビット列を返却します。 • SQL_SP_BETWEEN • SQL_SP_COMPARISON • SQL_SP_EXISTS • SQL_SP_IN • SQL_SP_ISNOTNULL • SQL_SP_ISNULL • SQL_SP_LIKE • SQL_SP_QUANTIFIED_COMPARISON	SQLUINTEGER
149	SQL_SQL92_RELATIONAL_JOIN_OPERATORS	SELECT ステートメントに対してサポートされている関係結合演算子を示します。	次のビット列を返却します。 • SQL_SRJO_CROSS_JOIN • SQL_SRJO_FULL_OUTER_JOIN • SQL_SRJO_INNER_JOIN • SQL_SRJO_LEFT_OUTER_JOIN • SQL_SRJO_RIGHT_OUTER_JOIN	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
150	SQL_SQL92_REVOKE	データソースがサポートするREVOKEステートメントに対してサポートされている句を示します。	次のビット列を返却します。 • SQL_SR_CASCADE • SQL_SR_DELETE_TABLE • SQL_SR_INSERT_TABLE • SQL_SR_REFERENCES_TABLE • SQL_SR_RESTRICT • SQL_SR_SELECT_TABLE • SQL_SR_UPDATE_TABLE	SQLUINTEGER
151	SQL_SQL92_ROW_VALUE_CONSTRUCTOR	SELECT ステートメントに対してサポートされている行の値を組み立てる式を示します。	0 を返却します。	SQLUINTEGER
152	SQL_SQL92_STRING_FUNCTIONS	ドライバおよび関連するデータソースがサポートする文字列スカラ関数を示します。	次のビット列を返却します。 • SQL_SSF_CONVERT • SQL_SSF_LOWER • SQL_SSF_SUBSTRING • SQL_SSF_TRIM_BOTH • SQL_SSF_TRIM_LEADING • SQL_SSF_TRIM_TRAILING • SQL_SSF_UPPER	SQLUINTEGER
153	SQL_SQL92_VALUE_EXPRESSIONS	サポートされている値の式を示します。	次のビット列を返却します。 • SQL_SVE_CASE • SQL_SVE_CAST • SQL_SVE_COALESCE • SQL_SVE_NULLIF	SQLUINTEGER
154	SQL_STANDARD_CLI_CONFORMANCE	ドライバが準拠する CLI の標準を示します。	SQL_SCC_ISO92_CLI を返却します。	SQLUINTEGER
155	SQL_STATIC_CURSOR_ATTRIBUTES1	ドライバがサポートする静的カーソルの属性を示します。このビットマスクには，属性の最初のサブセットが格納されます。	0 を返却します。	SQLUINTEGER
156	SQL_STATIC_CURSOR_ATTRIBUTES2	ドライバがサポートする静的カーソルの属性を表します。このビットマスクには，属性の 2 番目のサブセットが格納されます。	0 を返却します。	SQLUINTEGER
157	SQL_STRING_FUNCTIONS	データソースがサポートしている文字列スカラ関数を示します。	次のビット列を返却します。 • SQL_FN_STR_ASCII	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
			• SQL_FN_STR_CONCAT • SQL_FN_STR_LCASE • SQL_FN_STR_LEFT • SQL_FN_STR_LENGTH • SQL_FN_STR_LTRIM • SQL_FN_STR_REPLACE • SQL_FN_STR_RIGHT • SQL_FN_STR_RTRIM • SQL_FN_STR_SUBSTRING • SQL_FN_STR_UCASE	
158	SQL_SUBQUERIES	副問合せをサポートする述語を示します。	次のビット列を返却します。 • SQL_SQ_COMPARISON • SQL_SQ_CORRELATED_SUBQUERIES • SQL_SQ_EXISTS • SQL_SQ_IN • SQL_SQ_QUANTIFIED	SQLUINTEGER
159	SQL_SYSTEM_FUNCTIONS	ドライバと関連するデータソースがサポートするシステムスカラ関数を示します。	0 を返却します。	SQLUINTEGER
160	SQL_TABLE_TERM	テーブルに対するデータソースのベンダー名を含む文字列を返却します。	"table"を返却します。	• SQLCHAR * • SQLWCHAR *
161	SQL_TIMEDATE_ADD_INTERVALS	ドライバと関連するデータソースが TIMESTAMPADD スカラ関数に対してサポートするタイムスタンプの間隔を示します。	次のビット列を返却します。 • SQL_FN_TSI_FRAC_SECOND • SQL_FN_TSI_SECOND • SQL_FN_TSI_MINUTE • SQL_FN_TSI_HOUR • SQL_FN_TSI_DAY • SQL_FN_TSI_WEEK • SQL_FN_TSI_MONTH • SQL_FN_TSI_QUARTER • SQL_FN_TSI_YEAR	SQLUINTEGER
162	SQL_TIMEDATE_DIFF_INTERVALS	ドライバと関連するデータソースが，TIMESTAMPDIFF スカラ関数に対してサポートするタイムスタンプの間隔を示します。	次のビット列を返却します。 • SQL_FN_TSI_FRAC_SECOND • SQL_FN_TSI_SECOND • SQL_FN_TSI_MINUTE • SQL_FN_TSI_HOUR	SQLUINTEGER

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
			• SQL_FN_TSI_DAY • SQL_FN_TSI_WEEK • SQL_FN_TSI_MONTH • SQL_FN_TSI_QUARTER • SQL_FN_TSI_YEAR	
163	SQL_TIMEDATE_FUNCTIONS	ドライバと関連するデータソースがサポートする日付時刻スカラ関数を示します。	次のビット列を返却します。 • SQL_FN_TD_CURRENT_DATE • SQL_FN_TD_CURRENT_TIME • SQL_FN_TD_CURRENT_TIMESTAMP • SQL_FN_TD_DAYOFWEEK • SQL_FN_TD_DAYOFYEAR • SQL_FN_TD_EXTRACT • SQL_FN_TD_TIMESTAMPADD • SQL_FN_TD_TIMESTAMPDIFF	SQLUINTEGER
164	SQL_TXN_CAPABLE	ドライバまたはデータソースがサポートしているトランザクションを示します。	SQL_TC_DDL_COMMIT を返却します。	SQLUSMALLINT
165	SQL_TXN_ISOLATION_OPTION	ドライバまたはデータソースから得られるトランザクション隔離性水準を示します。	次のビット列を返却します。 • SQL_TXN_READ_COMMITTED • SQL_TXN_REPEATABLE_READ	SQLUINTEGER
166	SQL_UNION	UNION 句に対するサポートを示します。	次のビット列を返却します。 • SQL_U_UNION • SQL_U_UNION_ALL	SQLUINTEGER
167	SQL_USER_NAME	特定のデータベースで使用される名前を含む文字列を示します。 ログイン名と異なる場合があります。	接続に使用したユーザ名を返却します。	• SQLCHAR * • SQLWCHAR *
168	SQL_XOPEN_CLI_YEAR	ODBC ドライバマネージャのバージョンが完全に準拠する X/Open 仕様の刊行年を示す文字列を示します。	ドライバマネージャが設定する値を返却します。	• SQLCHAR * • SQLWCHAR *
169	SQL_ODBC_API_CONFORMANCE	ODBC 準拠のレベルを表す SQLSMALLINT 値を示します。	SQL_OAC_LEVEL1 を返却します。	SQLUSMALLINT

項番	情報型（InfoType）	情報型の説明（規約）	返却値	データ型
170	SQL_FETCH_DIRECTION	サポートするフェッチ方向のオプションを示します。	SQL_FD_FETCH_NEXT を返却します。	SQLUINTEGER
171	SQL_LOCK_TYPES	SQLSetPos の引数fLock に指定できるロックタイプを示します。	0 を返却します。	SQLUINTEGER
172	SQL_ODBC_SAG_CLI_CONFORMANCE	SAG 指定の関数の準拠 LV を示します。	SQL_OSCC_COMPLIANT を返却します。	SQLUSMALLINT
173	SQL_ODBC_SQL_CONFORMANCE	ドライバがサポートする SQL 文法を示します。	SQL_OSC_CORE を返却します。	SQLUSMALLINT
174	SQL_OUTER_JOINS	データソースの外結合のサポート状況を示します。	"F"を返却します。	• SQLCHAR * • SQLWCHAR *
175	SQL_POS_OPERATIONS	SQLSetPos がサポートする演算を示します。	0 を返却します。	SQLUINTEGER
176	SQL_POSITIONED_STATEMENTS	データソースがサポートする位置づけ SQL ステートメントを示します。	0 を返却します。	SQLUINTEGER
177	SQL_SCROLL_CONCURRENCY	スクロール可能なカーソルをサポートする同時実行制御オプションを示します。	SQL_SCCO_READ_ONLY を返却します。	SQLUINTEGER
178	SQL_STATIC_SENSITIVITY	次のことによって，アプリケーションが静的またはキーセット駆動カーソルに起こした変化を，そのアプリケーションで検出できるかを示します。 • SQLSetPos 関数 • 位置づけ更新または削除ステートメント	0 を返却します。	SQLUINTEGER

16.14 SQLSetConnectAttr，SQLSetConnectAttrW，SQLGetConnectAttr および SQLGetConnectAttrW に指定できる属性

SQLSetConnectAttr，SQLSetConnectAttrW，SQLGetConnectAttr およびSQLGetConnectAttrW に指定できる属性を次の表に示します。

表 16-22 SQLSetConnectAttr，SQLSetConnectAttrW，SQLGetConnectAttr および SQLGetConnectAttrW に指定できる属性

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
1	SQL_ATTR_ACCESS_MODE	更新を行う SQL ステートメントのサポート要否を指定します。 SQL_MODE_READ_ONLY 更新を行う SQL ステートメントをサポートしていません。 SQL_MODE_READ_WRITE 更新を行う SQL ステートメントをサポートしています。	次の値をサポートします。 • SQL_MODE_READ_ONLY • SQL_MODE_READ_WRITE	SQLUINTEGER
2	SQL_ATTR_ANSI_APP	Unicode 関数およびANSI 関数の切り替えを指定します。 SQL_AA_TRUE ANSI 関数を使用します。 SQL_AA_FALSE Unicode 関数を使用します。	次の値をサポートします。 • SQL_AA_TRUE • SQL_AA_FALSE この属性は，ドライバマネージャで使用されます。アプリケーションでは使用できません。 また，SQLGetConnectAttr および SQLGetConnectAttrW では未サポートエラーを返します。	SQLUINTEGER
3	SQL_ATTR_ASYNC_ENABLE	指定した接続のステートメントに対して呼び出された関数を非同期で実行するかどうかを指定します。 SQL_ASYNC_ENABLE_OFF 非同期実行されません。	次の値をサポートします。 • SQL_ASYNC_ENABLE_OFF	SQLUINTEGER
4	SQL_ATTR_AUTO_IPD	SQLPrepare の呼び出し後に IPD の自動設定をサポートするかどうかを指定します。	未サポート属性です。	－

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
5	SQL_ATTR_AUTOCOMMIT	自動コミットモードを使用するかどうかを指定します。 SQL_AUTOCOMMIT_OFF ドライバは，手動コミットモードを使用します。アプリケーションはSQLEndTran でトランザクションを明示的にコミットまたはロールバックする必要があります。 SQL_AUTOCOMMIT_ON ドライバは，自動コミットモードを使用します。各ステートメントは実行後直ちにコミットされます。	次の値をサポートします。 • SQL_AUTOCOMMIT_OFF • SQL_AUTOCOMMIT_ON	SQLUINTEGER
6	SQL_ATTR_CONNECTION_DEAD	接続状態を示す値を返却します。 この属性は，SQLGetConnectAttr およびSQLGetConnectAttrW で設定できます。SQLSetConnectAttr およびSQLSetConnectAttrW では設定できません。	未サポート属性です。 SQLSetConnectAttr およびSQLSetConnectAttrW では無効な属性です。 SQLGetConnectAttr およびSQLGetConnectAttrW では未サポートエラーを返します。	SQLUINTEGER
7	SQL_ATTR_CONNECTION_TIMEOUT	アプリケーションが接続要求に対する応答を待つ時間（秒数）を指定します。 0 ODBC ドライバでは，タイムアウトを検知しません。クライアントライブラリのタイムアウト制御に依存します。	次の値をサポートします。 • 0	SQLUINTEGER
8	SQL_ATTR_CURRENT_CATALOG	データソースが使用するカタログ名を指定します。	未サポート属性です。	－
9	SQL_ATTR_ENLIST_IN_DTC	Microsoft コンポーネントサービスが分散トランザクションに ODBC ドライバを使用するかどうかを指定します。	未サポート属性です。	－
10	SQL_ATTR_LOGIN_TIMEOUT	アプリケーションがログイン要求に対する応答を待つ時間（秒数）を指定します。	次の値をサポートします。 • 0 0 以外を設定した場合は，HADB ODBC ドライバが0 に書き換えて，	SQLUINTEGER

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
		0 ODBC ドライバでは，タイムアウトを検知しません。クライアントライブラリのタイムアウト制御に依存します。	SQL_SUCCESS_WITH_INFO を返却します。	
11	SQL_ATTR_METADATA_ID	カタログ関数で文字列引数を扱う方法を指定します。 SQL_FALSE カタログ関数の文字列引数は識別子と見なされません。大文字と小文字が区別されます。引数によって，文字列の検索パターンを含む場合と，含まない場合があります。	次の値をサポートします。 • SQL_FALSE ただし，SQLSetConnectAttr のときには，未サポートエラーになります。	－
12	SQL_ATTR_ODBC_CURSORS	ドライバマネージャによる ODBC カーソルライブラリの使用方法を指定します。 SQL_CUR_USE_IF_NEEDED 必要に応じて，ドライバマネージャは ODBC カーソルライブラリを使用します。 SQL_CUR_USE_ODBC ドライバマネージャは ODBC カーソルライブラリを使用します。	次の値をサポートします。 • SQL_CUR_USE_IF_NEEDED • SQL_CUR_USE_ODBC	SQLINTEGER
13	SQL_ATTR_PACKET_SIZE	ネットワークパケットサイズ（バイト長）を指定します。	未サポート属性です。	－
14	SQL_ATTR_QUIET_MODE	32 ビットのウィンドウハンドルを指定します。	未サポート属性です。	－
15	SQL_ATTR_TRACE	トレースの実行をドライバマネージャへ知らせます。 SQL_OPT_TRACE_OFF トレースが無効です。	次の値をサポートします。 • SQL_OPT_TRACE_OFF	SQLUINTEGER
16	SQL_ATTR_TRACEFILE	トレースファイル名を指定します。	未サポート属性です。	－
17	SQL_ATTR_TRANSLATE_LIB	ライブラリ名を指定します。	未サポート属性です。	－
18	SQL_ATTR_TRANSLATE_OPTION	トランスレータ DLL に渡される 32 ビットのフラグ値を指定します。	未サポート属性です。	－

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
19	SQL_ATTR_TXN_ISOLATION	現在の接続に対するトランザクション隔離性水準を指定します。 この関数を呼び出す前にSQLEndTranを呼び出して，接続上にオープンされているすべてのトランザクションをコミットまたはロールバックしておく必要があります。 SQL_TXN_READ_COMMITTED トランザクション隔離性水準をREAD_COMMITTEDに設定します。 SQL_TXN_REPEATABLE_READ トランザクション隔離性水準をREPEATABLE_READに設定します。	次の値をサポートします。 • SQL_TXN_READ_COMMITTED • SQL_TXN_REPEATABLE_READ	SQLUINTEGER
20	SQL_ATTR_HADB_ORDER_MODE	ODBC規約にはないHADB独自の属性です。 現在の接続に対する，ORDER BY句を指定したSELECT文の文字データの並び替え順序を設定します。 SQL_HADB_ORDER_MODE_BYTE 文字データをバイトコード順に並び替えます。 SQL_HADB_ORDER_MODE_ISO 文字データをソートコード順（ISO/IEC14651:2011準拠）に並び替えます。	次の値をサポートします。 • SQL_HADB_ORDER_MODE_BYTE • SQL_HADB_ORDER_MODE_ISO	SQLUINTEGER

（凡例）－：該当しません。

16.15　SQLSetEnvAttr および SQLGetEnvAttr に指定できる属性

SQLSetEnvAttr およびSQLGetEnvAttr に指定できる属性を次の表に示します。

表 16-23　SQLSetEnvAttr および SQLGetEnvAttr に指定できる属性

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
1	SQL_ATTR_CONNECTION_POOLING	環境レベルで接続プールの設定を指定します。	未サポート属性です。	－
2	SQL_ATTR_CP_MATCH	接続プールから接続を選択する方法を指定します。	未サポート属性です。	－
3	SQL_ATTR_ODBC_VERSION	ドライバが準拠する ODBC のバージョンを指定します。 SQL_OV_ODBC3 ドライバマネージャおよびドライバは ODBC3.0 として次のように動作します。 • ドライバは，日付，時刻，およびタイムスタンプに対し ODBC3.0 のコードを返し，ODBC3.0 のコードを前提とします。 • SQLGetDiagField，またはSQLGetDiagRec が呼び出された場合，ドライバは ODBC3.0 の SQLSTATE コードを返します。 SQL_OV_ODBC2 ドライバマネージャおよびドライバは ODBC2.x として次のように動作します。 • ドライバは，日付，時刻，およびタイムスタンプに対して ODBC2.x のコードを返し，ODBC2.x のコードを前提とします。 • SQLGetDiagField，またはSQLGetDiagRec が呼び出された場合，ドライバは ODBC2.x の SQLSTATE コードを返します。	次の値をサポートします。 • SQL_OV_ODBC3 • SQL_OV_ODBC2	SQLUINTEGER

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
		アプリケーションは，SQLHENV 型の引数を指定する関数を呼び出す前に，この環境属性を設定する必要があります。		
4	SQL_ATTR_OUTPUT_NTS	ドライバが文字列データを返却する方法を指定します。 SQL_TRUE ドライバは，NULL 終端文字データを返却します。	次の値をサポートします。 • SQL_TRUE	SQLUINTEGER

（凡例）－：該当しません。

16.16 SQLSetStmtAttr，SQLSetStmtAttrW，SQLGetStmtAttr および SQLGetStmtAttrW に指定できる属性

SQLSetStmtAttr，SQLSetStmtAttrW，SQLGetStmtAttr およびSQLGetStmtAttrW に指定できる属性を次の表に示します。

表 16-24 SQLSetStmtAttr，SQLSetStmtAttrW，SQLGetStmtAttr および SQLGetStmtAttrW に指定できる属性

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
1	SQL_ATTR_APP_PARAM_DESC	次にSQLExecute，SQLExecDirect または SQLExecDirectW を行うときに使用できる APD のハンドルを指定します。 次のどれかを指定します。 • 明示的に割り当てられた APD • SQL_NULL_HDESC 自動的に割り当てられた APD になります。 • 自動的に割り当てられた APD	次の値をサポートします。 • 明示的に割り当てられた APD • SQL_NULL_HDESC • 自動的に割り当てられた APD	SQLHDESC
2	SQL_ATTR_APP_ROW_DESC	次にフェッチを行うときに使用できる ARD のハンドルを指定します。 次のどれかを指定します。 • 明示的に割り当てられた ARD • SQL_NULL_HDESC 自動的に割り当てられた ARD になります。 • 自動的に割り当てられた ARD	次の値をサポートします。 • 明示的に割り当てられた ARD • SQL_NULL_HDESC • 自動的に割り当てられた ARD	SQLHDESC
3	SQL_ATTR_ASYNC_ENABLE	この関数のStatementHandle に指定されたステートメントハンドルを引数として呼び出す関数を非同期に実行するかどうかを指定します。 SQL_ASYNC_ENABLE_OFF 非同期実行しません。	次の値をサポートします。 • SQL_ASYNC_ENABLE_OFF	SQLULEN
4	SQL_ATTR_CONCURRENCY	カーソルの同時実行に関する指定をします。	次の値をサポートします。 • SQL_CONCUR_READ_ONLY	SQLULEN

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
		SQL_CONCUR_READ_ONLY カーソルは読み出しだけが実行できます。更新はできません。	SQL_CONCUR_READ_ONLY 以外を設定した場合，HADB ODBC ドライバが SQL_CONCUR_READ_ONLY に書き換えて，SQL_SUCCESS_WITH_INFO を返却します。	
5	SQL_ATTR_CURSOR_SCROLLABLE	アプリケーションが要求するサポートのレベルを指定します。 SQL_NONSCROLLABLE ステートメントハンドルに対してスクロール可能なカーソルは要求されません。	次の値をサポートします。 • SQL_NONSCROLLABLE SQL_NONSCROLLABLE 以外を設定した場合，HADB ODBC ドライバが SQL_NONSCROLLABLE に書き換えて，SQL_SUCCESS_WITH_INFO を返却します。	SQLULEN
6	SQL_ATTR_CURSOR_SENSITIVITY	ステートメントハンドル上のカーソルが，別のカーソルが結果セットに対して行った変更を検出するかどうかを指定します。 SQL_UNSPECIFIED カーソルの種類とカーソルが，ステートメントハンドル上で別のカーソルが結果セットに対して行った変更を検出するかどうかを指定しません。	次の値をサポートします。 • SQL_UNSPECIFIED SQL_UNSPECIFIED 以外を設定した場合，HADB ODBC ドライバが SQL_UNSPECIFIED に書き換えて，SQL_SUCCESS_WITH_INFO を返却します。	SQLULEN
7	SQL_ATTR_CURSOR_TYPE	カーソルの種類を指定します。 SQL_CURSOR_FORWARD_ONLY カーソルは前方にだけスクロールします。	次の値をサポートします。 • SQL_CURSOR_FORWARD_ONLY SQL_CURSOR_FORWARD_ONLY 以外を設定した場合，HADB ODBC ドライバが SQL_CURSOR_FORWARD_ONLY に書き換えて，SQL_SUCCESS_WITH_INFO を返却します。	SQLULEN
8	SQL_ATTR_ENABLE_AUTO_IPD	IPD の値の自動設定を実行するかどうかを指定します。	未サポート属性です。	－
9	SQL_ATTR_FETCH_BOOKMARK_PTR	バイナリのブックマーク値のポインタを指定します。	未サポート属性です。	－

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
10	SQL_ATTR_IMP_PARAM_DESC	ステートメントハンドルを最初に割り当てたときに割り当てられる IPD のハンドルを指定します。	次の値をサポートします。 • 自動的に割り当てられた IPD	SQLHDESC
11	SQL_ATTR_IMP_ROW_DESC	ステートメントハンドルを最初に割り当てたときに割り当てられる IRD のハンドルを指定します。	次の値をサポートします。 • 自動的に割り当てられた IRD	SQLHDESC
12	SQL_ATTR_KEYSET_SIZE	キーセットカーソルに対するキーセットの行数を指定します。	未サポート属性です。	−
13	SQL_ATTR_MAX_LENGTH	データソース間（サーバ・ホスト間）通信での，文字列またはバイナリ列の最大データ長を指定します。 0 データソースは有効なデータをすべて返します。	次の値をサポートします。 • 0	SQLULEN
14	SQL_ATTR_MAX_ROWS	データソース間（サーバ・ホスト間）通信での，SELECT ステートメントに対して返される行の最大数を指定します。 0 データソースはすべての行を返します。	次の値をサポートします。 • 0	SQLULEN
15	SQL_ATTR_METADATA_ID	カタログ関数の文字列引数を扱う方法を指定します。 SQL_FALSE 大文字と小文字は区別され，カタログ関数の文字列引数は識別子と見なされません。	次の値をサポートします。 • SQL_FALSE	SQLULEN
16	SQL_ATTR_NOSCAN	ドライバがエスケープシーケンスに対して構文解析を行い，DBMS 固有の文法に変換されるかどうかを指定します。 SQL_NOSCAN_OFF ドライバは，エスケープシーケンスに対して，DBMS 固有の文法に変換します。 SQL_NOSCAN_ON ドライバは，エスケープシーケンスに対して，DBMS 固有の文法に変換しません。	次の値をサポートします。 • SQL_NOSCAN_OFF • SQL_NOSCAN_ON	SQLULEN

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
17	SQL_ATTR_PARAM_BIND_OFFSET_PTR	動的パラメタのバインド時に使用されるポインタに加算するオフセットのポインタを指定します。 NULL 加算しません。	次の値をサポートします。 • NULL	SQLULEN *
18	SQL_ATTR_PARAM_BIND_TYPE	動的パラメタに使用するバインドの方向を指定します。 SQL_PARAM_BIND_BY_COLUMN 列方向バインドを指定します。	次の値をサポートします。 • SQL_PARAM_BIND_BY_COLUMN	SQLULEN
19	SQL_ATTR_PARAM_OPERATION_PTR	SQL ステートメントの実行中にパラメタを無視するために使用する配列を指定します。 NULL ドライバはパラメタのステータス値を返しません。	次の値をサポートします。 • NULL	SQLUSMALLINT *
20	SQL_ATTR_PARAM_STATUS_PTR	SQLExecute，SQLExecDirect またはSQLExecDirectW の呼び出し後，パラメタ値の各行に対してステータス情報を格納する配列を指定します。 NULL SQLUSMALLINT のポインタ ドライバはパラメタのステータス値を返しません。	次の値をサポートします。 • NULL • SQLUSMALLINT のポインタ（NULL 以外）	SQLUSMALLINT *
21	SQL_ATTR_PARAMS_PROCESSED_PTR	ドライバが，処理済みのパラメタセット数（エラーセットを含む）を返す変数のアドレスを指定します。 NULL SQLULEN のポインタ パラメタセットの数は返されません。	次の値をサポートします。 • NULL • SQLULEN のポインタ（NULL 以外）	SQLULEN *
22	SQL_ATTR_PARAMSET_SIZE	各パラメタセットの値の個数を指定する属性です。	次の値をサポートします。 • 1 1 以外を設定した場合は，HADB ODBC ドライバが 1 に書き換えて，SQL_SUCCESS_WITH_INFO を返却します。	SQLULEN

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
23	SQL_ATTR_QUERY_TIMEOUT	アプリケーションが，SQL ステートメントが実行されるまで待つ時間（秒数）を指定します。	未サポート属性です。	－
24	SQL_ATTR_RETRIEVE_DATA	SQLFetch が，カーソルを指定の場所に位置づけたあとにデータを取得するかどうかを指定します。 SQL_RD_ON SQLFetch は，カーソルを指定の場所に位置づけたあとにデータを取得します。	次の値をサポートします。 • SQL_RD_ON	SQLULEN
25	SQL_ATTR_ROW_ARRAY_SIZE	SQLFetch またはSQLFetchScroll が返す行数を指定します。	次の値をサポートします。 • 1	SQLULEN
26	SQL_ATTR_ROW_BIND_OFFSET_PTR	列データのバインドを変更するために，ポインタに加算するオフセットのポインタを指定します。 NULL 加算しません。	次の値をサポートします。 • NULL	SQLULEN *
27	SQL_ATTR_ROW_BIND_TYPE	関連づけられたステートメントに対してSQLFetch または SQLFetchScroll が呼び出されたときに使用されるバインドの方向を指定します。 SQL_BIND_BY_COLUMN 列方向バインドを指定します。	次の値をサポートします。 • SQL_BIND_BY_COLUMN	SQLULEN
28	SQL_ATTR_ROW_NUMBER	結果セットの現在の行番号を指定します。	未サポート属性です。	－
29	SQL_ATTR_ROW_OPERATION_PTR	SQLSetPos で一括処理中に行を無視するために使用する配列を指定します。 NULL ドライバは行を無視しません。	次の値をサポートします。 • NULL	SQLUSMALLINT *
30	SQL_ATTR_ROW_STATUS_PTR	SQLFetch またはSQLFetchScroll を呼び出したあと，行のステータス値を格納する配列のポインタを指定します。 NULL ドライバは行のステータス値を返しません。	次の値をサポートします。 • NULL • SQLUSMALLINT のポインタ(NULL 以外)	SQLUSMALLINT *

項番	属性（Attribute）	属性の説明（規約）	サポートの状況	データ型
31	SQL_ATTR_ROWS_FETCHED_PTR	SQLFetch またはSQLFetchScroll を呼び出したあと，フェッチされた行数を格納するバッファのポインタを指定します。 NULL ドライバはフェッチされた行数を返しません。	次の値をサポートします。 • NULL • SQLULEN のポインタ (NULL 以外)	SQLULEN *
32	SQL_ATTR_SIMULATE_CURSOR	位置づけ更新ステートメントと位置づけ削除ステートメントをシミュレートするドライバが，このステートメントで単一の行だけを変更するかどうかを指定します。	未サポート属性です。	－
33	SQL_ATTR_USE_BOOKMARKS	アプリケーションがカーソルでブックマークを使用するかどうかを指定します。	未サポート属性です。	－

（凡例）－：該当しません。

16.17 SQLGetDescField，SQLGetDescFieldW，SQLSetDescField および SQLSetDescFieldW に指定できる属性

SQLGetDescField，SQLGetDescFieldW，SQLSetDescField およびSQLSetDescFieldW に指定できる属性を次の表に示します。

表 16-25　ディスクリプタのヘッダフィールドの値

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
1	SQL_DESC_ALLOC_TYPE	そのディスクリプタが自動的に割り当てられたものか，明示的に割り当てられたものかを示します。 SQL_DESC_ALLOC_AUTO ドライバが自動的に割り当てたディスクリプタです。 SQL_DESC_ALLOC_USER アプリケーションが明示的に割り当てたディスクリプタです。	次の値をサポートします。 • SQL_DESC_ALLOC_AUTO • SQL_DESC_ALLOC_USER	SQLSMALLINT
2	SQL_DESC_ARRAY_SIZE	ARD 行セットの行数（SQLFetch で返される行数）です。 APD 各パラメタの値の個数です。	次の値をサポートします。 • 1	SQLULEN
3	SQL_DESC_ARRAY_STATUS_PTR	IRD SQLFetch 実行後のステータスの値が格納される行ステータス配列へのポインタです。 IPD SQLExecute，SQLExecDirect または SQLExecDirectW 実行後，パラメタ値の各セットに対するステータス情報を格納するパラメタステータス配列へのポインタです。 ARD SQLSetPos の処理に対して行の無視を指定するため，アプリケーションが値を設	次の値をサポートします。 • NULL	SQLUSMALLINT *

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
		定できる操作配列へのポインタです。 APD SQLExecute，SQLExecDirect または SQLExecDirectW 実行時，パラメタセットの無視を指定するため，アプリケーションが値を設定できるパラメタ操作配列へのポインタです。 NULL ドライバはステータスの値やステータス情報の格納，または行やパラメタセットの無視をしません。		
4	SQL_DESC_BIND_OFFSET_PTR	AP が指定するバインドオフセットです。 フェッチ時に，遅延フィールドに加算します。 NULL 加算しません。	次の値をサポートします。 • NULL	SQLLEN *
5	SQL_DESC_BIND_TYPE	列またはパラメタのバインドの方向を指定します。 ARD SQLFetch 実行時のバインドの方向です。 APD 動的パラメタに対するバインドの方向です。 SQL_BIND_BY_COLUMN 列方向バインドを指定します。	次の値をサポートします。 • SQL_BIND_BY_COLUMN	SQLINTEGER
6	SQL_DESC_COUNT	データを格納するレコード番号の最大値です。 指定できる値は 1 以上です。	次の値をサポートします。 • 0 以上の整数値	SQLSMALLINT
7	SQL_DESC_ROWS_PROCESSED_PTR	IRD SQLFetch 実行後，フェッチした行数を格納するバッファへのポインタです。 IPD SQLExecute，SQLExecDirect または	次の値をサポートします。 • NULL	SQLULEN *

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
		SQLExecDirectW 実行時に処理されたパラメタセットの数を格納するバッファへのポインタです。 NULL ドライバはフェッチした行数やパラメタセットの数をバッファに格納しません。		

表 16-26　ディスクリプタのレコードフィールドの値

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
1	SQL_DESC_AUTO_UNIQUE_VALUE	列が自動的にインクリメントされるかどうかを示します。 SQL_FALSE 列が自動的にインクリメントされません。	次の値をサポートします。 • SQL_FALSE	SQLINTEGER
2	SQL_DESC_BASE_COLUMN_NAME	結果セット列に対応するベース列名です。	次の値をサポートします。 • 列名 • "EXP*nnnn*_NO_NAME"（*nnnn* は0001 から 4000 の符号なし整数） 検索項目列が列指定ではありません。	• SQLCHAR＊ • SQLWCHAR＊
3	SQL_DESC_BASE_TABLE_NAME	結果セット列に対応するベーステーブル名です。	次の値をサポートします。 • 空の文字列	• SQLCHAR＊ • SQLWCHAR＊
4	SQL_DESC_CASE_SENSITIVE	照合または比較のときに，列またはパラメタの大文字と小文字を区別するかどうかを示します。 SQL_TRUE 照合・比較時に，列・パラメタの大文字と小文字を区別します。 SQL_FALSE 次のどちらかを示します。 ・照合・比較時に，列・パラメタの大文字と小文字を区別しません。 ・列が文字以外です。	次の値をサポートします。 • SQL_TRUE • SQL_FALSE	SQLINTEGER
5	SQL_DESC_CATALOG_NAME	列を格納するベーステーブルに対するカタログ名です。	次の値をサポートします。 • 空の文字列	• SQLCHAR＊ • SQLWCHAR＊

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
6	SQL_DESC_CONCISE_TYPE	すべてのデータ型に対する簡潔データ型です。	次の値をサポートします。 • 当ドライバでサポートしているデータ型	SQLSMALLINT
7	SQL_DESC_DATA_PTR	パラメタの値（APD）または列の値（ARD）を格納するバッファへのポインタです。	次の値をサポートします。 • NULL • バッファへのポインタ	SQLPOINTER
8	SQL_DESC_DATETIME_INTERVAL_CODE	SQL_DESC_TYPE フィールドの値がSQL_DATETIME または SQL_INTERVAL の場合，特定の日付時刻または間隔のデータ型に対するサブコードです。 上記以外の場合，このフィールドは0 です。	次の値をサポートします。 • 特定の日付時刻または間隔のデータ型に対するサブコード • 0	SQLSMALLINT
9	SQL_DESC_DATETIME_INTERVAL_PRECISION	SQL_DESC_TYPE フィールドが SQL_INTERVAL の場合，間隔先行精度です。 上記以外の場合，このフィールドは0 です。	次の値をサポートします。 • 0	SQLINTEGER
10	SQL_DESC_DISPLAY_SIZE	列からデータを表示するときに必要な最大文字数です。	データ型の種類によって，次の値をサポートします。 • SQL_CHAR, SQL_VARCHAR 定義長（バイト数） • SQL_BIGINT, SQL_INTEGER 桁数 • SQL_DECIMAL 精度（全体の桁数）＋2 • SQL_TYPE_DATE, SQL_DATE 10 • SQL_TYPE_TIME, SQL_TIME 8 ＋ (*p* ＋1) ※ • SQL_TYPE_TIMESTAMP, SQL_TIMESTAMP 19 ＋ (*p* ＋1) ※ • SQL_DOUBLE 24 • SQL_BINARY, SQL_VARBINARY 定義長（バイト数）×2	SQLLEN

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
11	SQL_DESC_FIXED_PREC_SCALE	列が真数値型かどうかを示します。 SQL_FALSE 列が真数値型以外で，固定精度とスケールが設定されています。	次の値をサポートします。 • SQL_FALSE	SQLSMALLINT
12	SQL_DESC_INDICATOR_PTR	列またはパラメタの値がNULLかどうかを示します。	次の値をサポートします。 • NULL • バッファへのポインタ	SQLLEN *
13	SQL_DESC_LABEL	列のラベルまたはタイトルです。	次の値をサポートします。 • 列名 • "EXP*nnnn*_NO_NAME"(*nnnn* は0001 から4000 の符号なし整数) 検索項目列が列指定ではありません。	• SQLCHAR * • SQLWCHAR *
14	SQL_DESC_LENGTH	文字列またはバイナリデータ型の最大長または実際の文字長です。	次の値をサポートします。 • 各データ型の定義長(バイト長)	SQLULEN
15	SQL_DESC_LITERAL_PREFIX	データ型のリテラルに対するプレフィックスとしてドライバが認識する 1 文字以上の文字です。	次の値をサポートします。 • 空の文字列	• SQLCHAR * • SQLWCHAR *
16	SQL_DESC_LITERAL_SUFFIX	データ型のリテラルに対するサフィックスとしてドライバが認識する 1 文字以上の文字です。	次の値をサポートします。 • 空の文字列	• SQLCHAR * • SQLWCHAR *
17	SQL_DESC_LOCAL_TYPE_NAME	普通のデータ型名とは異なるデータ型のローカル名です。	次の値をサポートします。 • 空の文字列	• SQLCHAR * • SQLWCHAR *
18	SQL_DESC_NAME	列のエイリアスです。列のエイリアスが適用されない場合は，列名です。	次の値をサポートします。 • 列名 • "EXP*nnnn*_NO_NAME"(*nnnn* は0001 から4000 の符号なし整数) 検索項目列が列指定ではありません。	• SQLCHAR * • SQLWCHAR *
19	SQL_DESC_NULLABLE	列にNULL を指定できるかどうかを示します。 SQL_NULLABLE 列にナル値を指定できます。	次の値をサポートします。 • SQL_NULLABLE • SQL_NO_NULLS	SQLSMALLINT

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
		SQL_NO_NULLS 列にナル値を指定できません。		
20	SQL_DESC_NUM_PREC_RADIX	SQL_DESC_TYPE フィールドのデータ型が概数値データ型か，真数値データ型かを示します。 10 データ型が真数値データ型です。 0 データ型が数値以外です。	次の値をサポートします。 • 10 • 0	SQLINTEGER
21	SQL_DESC_OCTET_LENGTH	文字列またはバイナリデータ型のバイト長です。	次の値をサポートします。 • 各データ型の定義長(バイト長)	SQLLEN
22	SQL_DESC_OCTET_LENGTH_PTR	動的引数（パラメタディスクリプタの場合）またはバインドされた列の値（行ディスクリプタの場合）の長さの合計(バイト長）を格納する変数を指します。	次の値をサポートします。 • NULL • バッファへのポインタ	SQLLEN *
23	SQL_DESC_PARAMETER_TYPE	パラメタの種類(入力・出力・入出力)を示します。 SQL_PARAM_INPUT 入力パラメタです。 SQL_PARAM_INPUT_OUTPUT 入出力パラメタです。	次の値をサポートします。 • SQL_PARAM_INPUT • SQL_PARAM_INPUT_OUTPUT SQL_PARAM_INPUT に置き換えます。	SQLSMALLINT
24	SQL_DESC_PRECISION	真数値型の場合は桁数，概数値型の場合は仮数（バイナリ精度）のビット数です。	データ型の種類によって，次の値をサポートします。 • SQL_BIGINT, SQL_INTEGER 桁数 • SQL_DECIMAL, SQL_NUMERIC 精度（全体の桁数） • SQL_TYPE_TIME, SQL_TYPE_TIMESTAMP, SQL_TIME, SQL_TIMESTAMP 小数秒の桁数 • SQL_DOUBLE, SQL_FLOAT 15 • そのほかのデータ型	SQLSMALLINT

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
			0	
25	SQL_DESC_SCALE	10 進および数値のデータ型スケールの定義値です。	次の値をサポートします。 • スケールの定義値 • 0 数値以外のデータ型です。	SQLSMALLINT
26	SQL_DESC_SCHEMA_NAME	列を格納するベーステーブルのスキーマ名です。	次の値をサポートします。 • 空の文字列	• SQLCHAR * • SQLWCHAR *
27	SQL_DESC_SEARCHABLE	WHERE 句で列を使用できるかどうかを示します。 SQL_PRED_SEARCHABLE WHERE 句の任意の比較演算子で列を使用できます。	次の値をサポートします。 • SQL_PRED_SEARCHABLE	SQLSMALLINT
28	SQL_DESC_TABLE_NAME	列を格納するベーステーブル名です。	次の値をサポートします。 • 空の文字列	• SQLCHAR * • SQLWCHAR *
29	SQL_DESC_TYPE	• 日付時刻と間隔以外のデータ型 簡潔 SQL データ型または簡潔 C データ型です。 • 日付時刻および間隔のデータ型 冗長データ型（SQL_DATETIME または SQL_INTERVAL）です。	次の値をサポートします。 • 簡潔 SQL データ型 • 簡潔 C データ型 • 冗長データ型	SQLSMALLINT
30	SQL_DESC_TYPE_NAME	データソースに依存するデータ型名です。 データ型が不明な場合，空の文字列が指定されます。	次の値をサポートします。 • データソースに依存するデータ型名 • 空の文字列	• SQLCHAR * • SQLWCHAR *
31	SQL_DESC_UNNAMED	SQL_DESC_NAME フィールドに列名または列のエイリアスが指定されたかどうかを示します。 SQL_NAMED SQL_DESC_NAME フィールドに列名または列のエイリアスが指定されました。 SQL_UNNAMED SQL_DESC_NAME フィールドに列名または列のエイリアスが指定されていません。	次の値をサポートします。 • SQL_NAMED • SQL_UNNAMED	SQLSMALLINT

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
32	SQL_DESC_UNSIGNED	列のデータ型が符号付きか，符号なしかを示します。 SQL_TRUE 列のデータ型が符号なし，または数値以外です。 SQL_FALSE 列のデータ型が符号付きです。	次の値をサポートします。 • SQL_TRUE • SQL_FALSE	SQLSMALLINT
33	SQL_DESC_UPDATABLE	結果セットの列の更新が可能かどうかを示します。 SQL_ATTR_READWRITE_UNKNOWN 結果セット列は更新可能かどうかわかりません。	次の値をサポートします。 • SQL_ATTR_READWRITE_UNKNOWN	SQLSMALLINT
34	SQL_COLUMN_LENGTH	ODBC2.0 で定義されているフィールドです。	データ型の種類によって，次の値をサポートします。 • SQL_CHAR，SQL_VARCHAR，SQL_BINARY，SQL_VARBINARY 定義長（バイト数） • SQL_BIGINT，SQL_INTEGER 桁数 • SQL_DECIMAL，SQL_NUMERIC 精度（全体の桁数）＋2 • SQL_TYPE_DATE 10 • SQL_TYPE_TIME，SQL_TIME 8 ＋ (*p* ＋1) ※ • SQL_TYPE_TIMESTAMP，SQL_TIMESTAMP 19 ＋ (*p* ＋1) ※ • SQL_DOUBLE，SQL_FLOAT 24	SQLLEN
35	SQL_COLUMN_PRECISION	ODBC2.0 で定義されているフィールドです。	次の値をサポートします。 • SQL_DESC_PRECISION と同じ値	SQLSMALLINT
36	SQL_COLUMN_SCALE	ODBC2.0 で定義されているフィールドです。	次の値をサポートします。	SQLSMALLINT

項番	属性(Attribute)	属性の説明(規約)	サポートの状況	データ型
			• SQL_DESC_SCALE と同じ値	

注※

p は最大 12 桁の小数秒精度を意味しています。

（ ）内は，小数秒がある場合に加算します。

16.18 SQLGetDiagField および SQLGetDiagFieldW の DiagIdentifier に指定できる属性

DiagIdentifier に指定できる属性を次の表に示します。

表 16-27 DiagIdentifier に指定できる属性（ヘッダフィールドの場合）

項番	診断フィールド識別子 (DiagIdentifier)	診断フィールド識別子の説明 (規約)	返却値	データ型
1	SQL_DIAG_CURSOR_ROW_COUNT	カーソル内の行カウント値が返されます。 ステートメントハンドルだけが有効です。ステートメントハンドル以外が指定された場合は，SQL_ERROR が返されます。	常に0 が返されます。	SQLLEN
2	SQL_DIAG_DYNAMIC_FUNCTION	関数が実行した SQL ステートメントが返されます。 ステートメントハンドルだけが有効です。ステートメントハンドル以外が指定された場合は，SQL_ERROR が返されます。 SQLExecute，SQLExecDirect またはSQLExecDirectW の直後に取得できます。	次の値が返されます。 • ”ALTER TABLE” • ”ALTER USER” • ”ALTER VIEW” • ”CREATE AUDIT” • ”CREATE INDEX” • ”CREATE SCHEMA” • ”CREATE TABLE” • ”CREATE USER” • ”CREATE VIEW” • ”DELETE” • ”DROP AUDIT” • ”DROP INDEX” • ”DROP SCHEMA” • ”DROP TABLE” • ”DROP USER” • ”DROP VIEW” • ”GRANT” • ”INSERT” • ”PURGE CHUNK” • ”REVOKE” • ”SELECT CURSOR” • ”TRUNCATE TABLE” • ”UPDATE” • ”” SQLExecute，SQLExecDirect，または SQLExecDirectW の直後以	• SQLCHAR * • SQLWCHAR *

項番	診断フィールド識別子(DiagIdentifier)	診断フィールド識別子の説明(規約)	返却値	データ型
			外は，常に""が返されます。	
3	SQL_DIAG_DYNAMIC_FUNCTION_CODE	関数が実行したSQLステートメントを示す値が返されます。 ステートメントハンドルだけが有効です。ステートメントハンドル以外が指定された場合は，SQL_ERRORが返されます。 SQLExecute，SQLExecDirectまたはSQLExecDirectWの直後に取得できます。	次の値が返されます。 • SQL_DIAG_INSERT • SQL_DIAG_CREATE_TABLE • SQL_DIAG_ALTER_TABLE • SQL_DIAG_DROP_TABLE • SQL_DIAG_CREATE_INDEX • SQL_DIAG_DROP_INDEX • SQL_DIAG_CREATE_SCHEMA • SQL_DIAG_DROP_SCHEMA • SQL_DIAG_CREATE_VIEW • SQL_DIAG_DROP_VIEW • SQL_DIAG_UNKNOWN_STATEMENT • SQL_DIAG_SELECT_CURSOR • SQL_DIAG_UPDATE_WHERE • SQL_DIAG_DELETE_WHERE • SQL_DIAG_GRANT • SQL_DIAG_REVOKE SQLExecute，SQLExecDirect，またはSQLExecDirectWの直後以外は，常にSQL_DIAG_UNKNOWN_STATEMENTが返されます。	SQLINTEGER
4	SQL_DIAG_NUMBER	利用可能なステータスレコード数が返されます。	0または1が返されます。	SQLINTEGER
5	SQL_DIAG_RETURNCODE	SQLGetDiagFieldまたはSQLGetDiagFieldW，SQLGetDiagRecまたはSQLGetDiagRecWの実行直前に実行したODBC関数が返したリターンコードが返されます。	常にSQL_SUCCESSが返されます。	SQLRETURN

項番	診断フィールド識別子(DiagIdentifier)	診断フィールド識別子の説明(規約)	返却値	データ型
6	SQL_DIAG_ROW_COUNT	挿入，削除，更新処理によって変更された行数が返されます。 ステートメントハンドルだけが有効です。ステートメントハンドル以外が指定された場合は，SQL_ERROR が返されます。	常に0 が返されます。	SQLLEN

表 16-28　DiagIdentifier に指定できる属性（レコードフィールドの場合）

項番	診断フィールド識別子(DiagIdentifier)	診断フィールド識別子の説明(規約)	返却値	データ型
1	SQL_DIAG_CLASS_ORIGIN	SQLSTATE のクラス部分を定義しているドキュメントを示す文字列が返されます。	次の値が返されます。 • "ODBC 3.0" • "ISO 9075"	• SQLCHAR * • SQLWCHAR *
2	SQL_DIAG_COLUMN_NUMBER	結果セット中の列番号またはパラメタセット中のパラメタ番号を示す値が返されます。 ステートメントハンドルだけ有効です。	常に0 が返されます。	SQLINTEGER
3	SQL_DIAG_CONNECTION_NAME	接続の名前が返されます。	常に""が返されます。	• SQLCHAR * • SQLWCHAR *
4	SQL_DIAG_MESSAGE_TEXT	エラーまたは警告に関するメッセージが返されます。	"[Hitachi Advanced Data Binder ODBC Driver]"で始まる文字列が返されます。 付加情報が付くことがあります。	• SQLCHAR * • SQLWCHAR *
5	SQL_DIAG_NATIVE	ODBC ドライバまたはデータソース固有のネイティブエラーコードが返されます。 ない場合は0 が返されます。	0 またはSQLCODE の値を返します。 SQLCODE については，マニュアル『HADB　メッセージ』の『SQLCODE の見方』を参照してください。	SQLINTEGER
6	SQL_DIAG_ROW_NUMBER	行セット中の列番号またはパラメタセット中のパラメタ番号を示す値が返されます。 ステートメントハンドルだけ有効です。	常に0 が返されます。	SQLLEN
7	SQL_DIAG_SERVER_NAME	サーバ名が返されます。	常に""が返されます。	• SQLCHAR * • SQLWCHAR *

項番	診断フィールド識別子 (DiagIdentifier)	診断フィールド識別子の説明 (規約)	返却値	データ型
8	SQL_DIAG_SQLSTATE	5 文字のSQLSTATE の診断コードが返されます。	ODBC 規約に定められた 5 文字のSQLSTATE の診断コードが返されます。	• SQLCHAR * • SQLWCHAR *
9	SQL_DIAG_SUBCLASS_ORIGIN	SQL_DIAG_CLASS_ORIGIN と同じ形式の有効な値を持った文字列が返されます。SQLSTATE のサブクラス部分の定義を識別するものです。	次の値が返されます。 • "ODBC 3.0" • "ISO 9075"	• SQLCHAR * • SQLWCHAR *

17 トラブルシュート

この章では，ODBC インタフェース使用時のトラブルシュートについて説明します。

17.1　トラブルシュート時に使用する情報

ODBC インタフェース（BI ツールや ODBC モジュールなど）を使用している場合のトラブルシュート時に使用する情報について説明します。

17.1.1　BI ツールや ODBC モジュールの出力するメッセージ

BI ツールや ODBC モジュール（Microsoft Excel など）から，HADB サーバにアクセスした際にエラーが発生した場合，ダイアログボックス，BI ツールのステータス画面やメッセージ欄，またはログにエラーメッセージが出力されます。

メッセージの出力形式は，BI ツールや ODBC モジュールなどの仕様によって異なりますが，その製品独自のメッセージに続いて，エラー発生元のメッセージが出力されるケースが一般的です。出力されたメッセージ中に，「Hitachi」または「KFAA」などの文字列がある場合は，HADB が原因元であるエラーが発生した可能性があります。

重要

ダイアログボックスや，BI ツールのメッセージ欄に出力されたエラーメッセージは，以降表示されないことがあります。そのため，出力されたエラーメッセージのスクリーンショットを取得してください。

17.1.2　ODBC トレース

ドライバマネージャの機能によって，AP とドライバマネージャ間の入出力情報をトレースログに出力します。これを ODBC トレースといいます。

ODBC トレースでは，ドライバマネージャが AP から受け取った要求単位に情報が出力されます。そのため，AP のデバッグや，エラーの原因調査に使用できます。

■ODBC トレースの取得方法（Windows 版の HADB クライアントの場合）

ODBC トレースを取得する場合は，ODBC データソースアドミニストレーターの［トレース］タブで，［トレースの開始］をクリックしてください。

［ログファイルのパス］に指定したパスに，ODBC トレースが出力されます。ODBC トレースの取得方法の詳細については，『MSDN ライブラリ』の『ODBC データソースアドミニストレーターを使う』を参照してください。

■ODBC トレースの取得方法（Linux 版の HADB クライアントの場合）

ODBC トレースを取得する場合は，odbcinst.ini に ODBC トレースを取得するための記述を追加します。ODBC トレースを取得するための記述については，「15.3.1　データソースの設定」を参照してください。

重要

トレースログのファイルへの入出力を，ドライバマネージャが AP から受け取った要求単位で行うため，その分処理性能が低下します。そのため，ODBC トレースを常に出力する運用は推奨しません。

17.1.3 HADB ODBC ドライバトレース情報

Windows 版の HADB クライアントの場合，HADB ODBC ドライバ独自のトレース出力機能（HADB ODBC ドライバトレース）によって，HADB ODBC ドライバトレース情報を出力します。ドライバマネージャと HADB ODBC ドライバ間の入出力情報を出力します。

メモ

Linux 版の HADB クライアントの場合，HADB ODBC ドライバ独自のトレース出力機能（HADB ODBC ドライバトレース）はサポートしていません。

HADB ODBC ドライバトレース情報の取得方法については，「17.3 HADB ODBC ドライバトレース情報を出力するときの設定」を参照してください。出力される情報の詳細については，「17.4 HADB ODBC ドライバトレース情報に出力される情報」を参照してください。

■HADB ODBC ドライバトレース情報の出力先

HADB ODBC ドライバトレース情報の出力先フォルダは，次のどちらかになります。

- ODBC データソースアドミニストレーターの［Trace Setup］ダイアログの［Trace Directory Path］に指定したフォルダ
- 環境変数ADBODBTRCPATH に指定したフォルダ

HADB ODBC ドライバトレースファイルの名称は，次のとおりです。

- adbodbtrace_PID_TID_01.log
- adbodbtrace_PID_TID_02.log

PID は，HADB ODBC ドライバを使用する AP や BI ツールのプロセス ID を 16 進数で表した値です。TID は，HADB ODBC ドライバを使用する AP や BI ツールのプロセスによって起動されたスレッドのスレッド ID を 16 進数で表した値です。

HADB ODBC ドライバトレースファイルは，HADB ODBC ドライバを利用するプロセスのスレッドごとに出力されます。そのため，HADB ODBC ドライバを利用する AP や BI ツールのプロセス数およびスレッド数に比例して，出力されるファイル数が増加します。

なお，出力された情報が，HADB ODBC ドライバトレースファイルの容量の上限に達した場合，出力先をadbodbtrace_PID_TID_01.log とadbodbtrace_PID_TID_02.log で切り替えます。出力先の切り替え時，切り替え先のファイルのサイズが上限に達している場合，切り替え先のファイル内のすべての情報を削除してから，HADB ODBC ドライバトレース情報を出力します。

重要

HADB ODBC ドライバトレース情報を出力するたびに，HADB ODBC ドライバトレースファイルをオープンおよびクローズします。そのため，処理性能が著しく低下することが予想されます。したがって，HADB ODBC ドライバトレース情報を常に出力する運用は推奨しません。

17.1.4　HADB サーバまたは HADB クライアントが出力するメッセージ

HADB サーバまたは HADB クライアントが出力するメッセージには，クライアントからの接続情報や，実行した SQL 文などの情報が出力されます。

17.1.5　SQL トレース情報

SQL トレース情報は，AP から HADB サーバが受け付けた SQL 文のトレース情報を出力する機能です。SQL トレース情報については，マニュアル『HADB　システム構築・運用ガイド』の『SQL トレース機能の運用』を参照してください。

ODBC インタフェースを利用した環境下では，HADB ODBC ドライバから HADB クライアントへの要求，および HADB クライアントから HADB ODBC ドライバに返却された内容が確認できます。

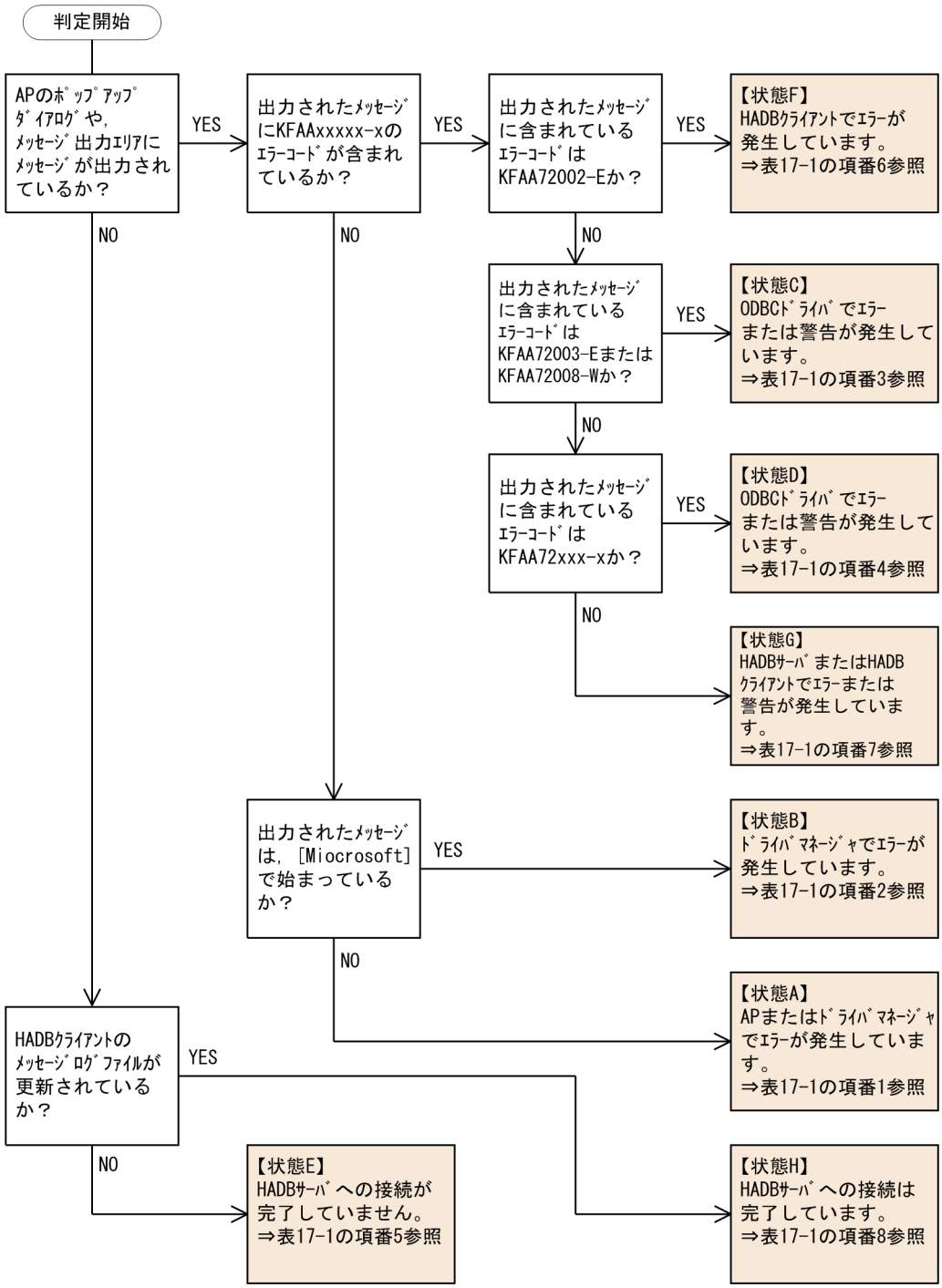
17.2　トラブルシュートの方法

ODBC インタフェースを使用しているときのトラブルシュートの方法について説明します。

17.2.1　エラーの対処方法

トラブルシュートの流れとしては，最初にエラーの発生元を特定します。そのあとに，エラーの対処を行います。エラーの発生元の特定方法を次の図に示します。

図 17-1　エラーの発生元の特定方法



上記のフローからエラーの発生元を特定したあとに，次の表に示す対処方法を参照してください。

表 17-1　エラーの対処方法

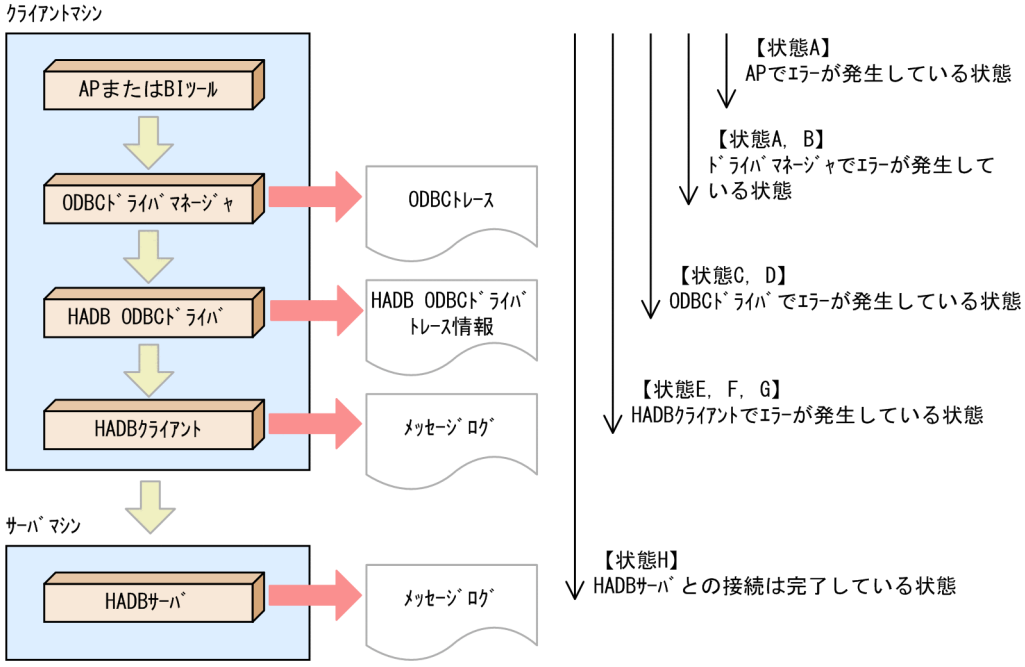
項番	状態	対処方法
1	【状態 A】 AP またはドライバマネージャでエラーが発生しています。	次のことを実施してください。 • AP の設定，またはプログラムの設定を確認してください。 • HADB クライアントのインストールが正しく行われているかを確認してください。

項番	状態	対処方法
		• データソースの登録または接続情報の指定が正しいかを確認してください。 • 出力されたSQLSTATEを確認してください。『MSDN ライブラリ』の『ODBC プログラマーズリファレンス』などで，SQLSTATEやメッセージの内容を確認してください。 • 再現性があるエラーの場合は，ODBC トレースを取得して，エラーの原因となっている ODBC 関数を特定してください。
2	【状態 B】 ドライバマネージャでエラーが発生しています。	項番 1 の対処方法と同じになります。
3	【状態 C】 ODBC ドライバでエラーまたは警告が発生しています。	次のことを実施してください。 • 出力されたSQLSTATEを確認してください。SQLSTATEについては，『MSDN ライブラリ』の『ODBC プログラマーズリファレンス』，または「16. ODBC 関数」の各 ODBC 関数のSQLSTATEの説明を参照してください。このエラーは，ODBC の実装規約に従って出力されるエラーです。 • 再現性があるエラーの場合は，HADB ODBC ドライバトレース情報を取得して，ODBC 関数の実行順序や，どの ODBC 関数でどのようなエラーが発生しているかなどを中心に調査してください。
4	【状態 D】 ODBC ドライバでエラーまたは警告が発生しています。	次のことを実施してください。 • 出力されたメッセージを確認し，マニュアル『HADB　メッセージ』を参照して，エラーまたは警告の原因を調査してください。 • 再現性があるエラーまたは警告の場合は，HADB ODBC ドライバトレース情報を取得して，ODBC 関数の実行順序や，エラーの原因となっている ODBC 関数を特定してください。 • データ変換のエラーや，文字コード変換のエラーが原因となることが多いため，BI ツールや ODBC モジュールの設定，検索データの表示方法などを見直してください。 • サポートしていない属性情報にアクセスしている可能性があります。「16.13 SQLGetInfo および SQLGetInfoW の InfoType に指定できる情報型」～「16.18　SQLGetDiagField および SQLGetDiagFieldW の DiagIdentifier に指定できる属性」を参照し，サポート状況を確認してください。
5	【状態 E】 HADB サーバへの接続が完了していません。	HADB サーバまでアクセスが到達していない可能性があります。アクセス経路のうち，どこまで到達しているかを調査する必要があります。 最初に次のことを実施してください。 • メッセージログファイルの出力先フォルダ，およびメッセージログファイルのアクセス権限の設定が正しいかを確認してください。 • 環境変数の設定が正しいかを確認してください。 • HADB クライアントのインストールが正しく行われているかを確認してください。 上記が正しく設定されているのに，エラーが再現する場合は，ODBC トレース，HADB ODBC ドライバトレース情報，および SQL トレース情報の出力を設定してください。そのあとに，エラーを再現して，各トレースの出力情報を解析してください。 エラーの発生がなく，BI ツールや ODBC モジュールにも何も出力されない場合は，BI ツールや ODBC モジュール側のトラブルの可能性があります。

項番	状態	対処方法
6	【状態 F】 HADB クライアントでエラーが発生しています。	出力されたエラー要因コードを確認し，エラー原因を調査してください。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。 環境変数やクライアント定義のパスの指定が誤っている，または権限がないなどの原因が考えられます。
7	【状態 G】 HADB サーバまたは HADB クライアントでエラーまたは警告が発生しています。	出力された HADB のメッセージを確認し，エラーまたは警告の原因を調査してください。
8	【状態 H】 HADB サーバへの接続は完了しています。	HADB サーバまでアクセスが到達しています。 メッセージログファイルに出力されているメッセージや，SQL 文を解析してトラブルシュートを実施してください。 詳細情報を取得する場合は，ODBC トレース，HADB ODBC ドライバトレース情報，および SQL トレース情報の出力を設定してください。そのあとに，エラーを再現して，各トレースの出力情報を解析してください。 エラーの発生がなく，BI ツールや ODBC モジュールにも何も出力されない場合は，BI ツールや ODBC モジュール側のトラブルの可能性があります。

BI ツールや，ODBC モジュールから，HADB ODBC ドライバを経由して HADB サーバにアクセスする際の処理の流れを次の図に示します。

図 17-2　HADB ODBC ドライバを経由して HADB サーバにアクセスする際の処理の流れ



なお，エラー原因の対処後に，異なるエラーが発生した場合は，再度エラーの発生元の特定からし直してください。

17.2.2　トラブルシュートのポイント

(1)　ODBC トレースを参照する際のポイント

ODBC トレースを参照することによって，ドライバマネージャが，AP や BI ツール，ODBC モジュールなどから受け付けた要求と，その返却内容を確認できます。

AP や BI ツール，ODBC モジュールなどは，ドライバマネージャからエラーが返却された場合，ODBC 関数のSQLGetDiagField およびSQLGetDiagRec を発行し，エラーの詳細情報を取得するのが一般的です。特に，SQLGetDiagRec で得られるSQLSTATE，NativeError，およびMessageText は有用なトラブルシュート情報となります。

> **メモ**
>
> ODBC トレースは，HADB ODBC ドライバからの要求や返却内容を示すものではありません。

(2)　メッセージテキストからエラー発生元を特定する際のポイント

メッセージテキストが出力されている場合，そのメッセージテキストから出力元を判別することができます。ポイントを次に示します。

1. メッセージテキスト中に，KFAA で始まるメッセージ ID がないかを確認してください。さらに，メッセージテキスト中に，[Hitachi Advanced Data Binder] [ODBC Driver]タグがないかを確認してください。

   上記の条件に該当する場合は，HADB（HADB ODBC ドライバも含む）でエラーまたは警告が発生しています。

   - KFAA で始まるメッセージがKFAA72000 番台の場合は，HADB ODBC ドライバでエラーが発生しています。
   - 上記以外のメッセージ ID の場合は，HADB サーバまたは HADB クライアントでエラーが発生しています。
   - メッセージ ID およびタグが途中までしか出力されていない場合は，HADB サーバまたは HADB クライアントで処理が行われたと考えられます。この場合，各種トレースやログを確認して，完全なメッセージを取得してください。

2. メッセージテキスト中に，[Microsoft] [ODBC Driver Manager]タグがないかを確認してください。タグがある場合は，ドライバマネージャでエラーまたは警告が発生しています。

   この場合，HADB（HADB ODBC ドライバも含む）では処理が行われていない可能性が高いと考えられます。シーケンスエラーなど，致命的なエラーのケースが多く見られるため，AP のデバッグ，BI ツールまたは ODBC モジュールの設定の見直しを行ってください。

3. 上記の 1，2 で確認したキーワードがない場合は，ドライバマネージャおよび HADB（HADB ODBC ドライバも含む）では処理が行われていない可能性が高いと考えられます。BI ツールまたは ODBC モ

ジュールの設定で，使用するインタフェースや DBMS の指定に誤りなどがないか，設定を確認してください。

(3) SQLSTATE を参照する際のポイント

エラーの発生元を特定したあとに，SQLSTATE を確認してエラーの原因を特定します。

- ドライバマネージャがエラーの発生元の場合

  ドライバマネージャが返すSQLSTATE は，ODBC の実装規約に基づいています。『MSDN ライブラリ』の『ODBC プログラマーズリファレンス』の『ODBC Error Codes』にSQLSTATE の一覧があります。そこにエラー原因の簡単な説明と，SQLSTATE を返した ODBC 関数が説明されています。詳細な情報については，SQLSTATE と ODBC 関数を基に，『ODBC API Reference』を参照して確認してください。

- HADB ODBC ドライバがエラーの発生元の場合

  ODBC 関数が返すSQLSTATE を確認して，エラーの原因を特定してください。

  > **メモ**
  >
  > HADB ODBC ドライバが返すSQLSTATE は，ドライバマネージャが返すSQLSTATE と同様に，ODBC の実装規約に基づいています。より詳細にエラー原因を特定できる場合は，HADB 独自のSQLSTATE を返しています。

- HADB サーバまたは HADB クライアントがエラーの発生元の場合

  HADB サーバまたは HADB クライアントが返すSQLSTATE は，HADB のメッセージと対応しています。SQLSTATE と HADB のメッセージの対応については，マニュアル『HADB　メッセージ』の『SQLSTATE の一覧』を参照してください。

  対応しているメッセージの内容を確認して，エラーの原因を特定してください。

(4) SQLSTATE およびメッセージテキストの取得方法

SQLSTATE およびメッセージテキストは，次の ODBC 関数を実行して取得します。

- SQLGetDiagField
- SQLGetDiagFieldW
- SQLGetDiagRec
- SQLGetDiagRecW

17.3　HADB ODBC ドライバトレース情報を出力するときの設定

Windows 版の HADB クライアントの場合，HADB ODBC ドライバトレース情報が出力されます。

HADB ODBC ドライバトレース情報を出力するときの設定方法には，次に示す 2 つの方法があります。

- ODBC データソースアドミニストレーターで設定する方法
- 環境変数で設定する方法

どちらの方法を選択するかの目安を次の表に示します。

表 17-2　設定方法を選択する際の目安

使用条件			推奨する設定方法
HADB サーバへの接続時にデータソースを使用するかどうか	HADB サーバへの接続時に使用するデータソース種別	動的にトレース情報を出力，停止する必要があるかどうか	
使用する	ユーザ DSN	必要がある	ODBC データソースアドミニストレーターで設定する方法
		必要がない	どちらの方法でもよい
	システム DSN	必要がある	ODBC データソースアドミニストレーターで設定する方法
		必要がない	どちらの方法でもよい
	ファイル DSN	必要がある	なし
		必要がない	環境変数で設定する方法
使用しない	該当なし	必要がある	なし
		必要がない	環境変数で設定する方法

各方法について，以降で説明します。

17.3.1　ODBC データソースアドミニストレーターで設定する方法

ODBC データソースアドミニストレーターを使用して，HADB ODBC ドライバトレース情報を出力するときの設定方法を説明します。

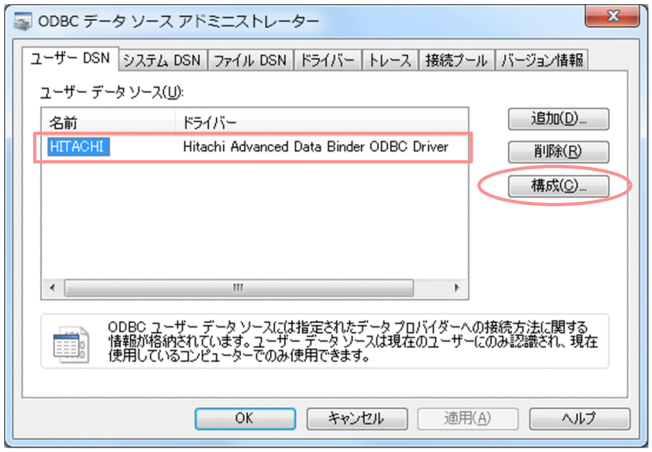
(1)　HADB ODBC ドライバトレース情報を出力する場合

HADB ODBC ドライバトレース情報を出力する際の設定方法を説明します。

手順

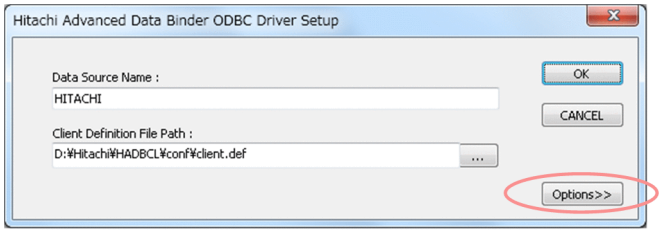
1. [Hitachi Advanced Data Binder ODBC Driver Setup] ダイアログを表示する

   ODBC データソースアドミニストレーターの［ユーザー DSN］タブで，HADB ODBC ドライバを使用するデータソースを選択し，［構成(C)...］ボタンをクリックします。

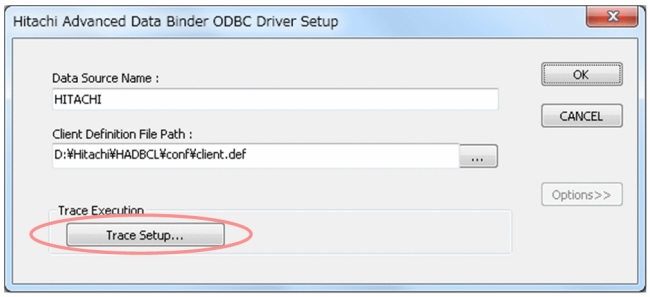


> **メモ**
>
> ODBC データソースアドミニストレーターの［システム DSN］タブでも，同じ操作をして［Hitachi Advanced Data Binder ODBC Driver Setup］ダイアログを表示することができます。

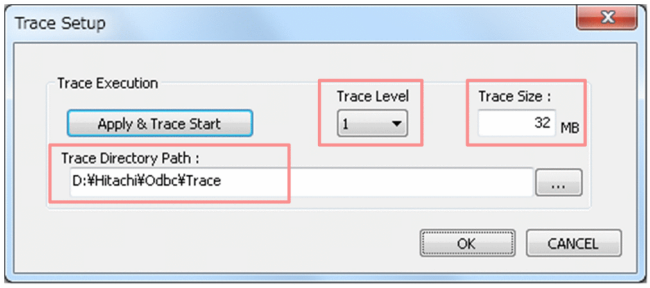
2. [Hitachi Advanced Data Binder ODBC Driver Setup] ダイアログで，［Options>>］ボタンをクリックする



3. [Trace Setup...] ボタンをクリックする



4. [Trace Setup] ダイアログで，HADB ODBC ドライバトレース情報の出力に関する設定を行う



次の指定を行います。

- Trace Level

  トレースレベルを選択します。トレースレベルについては，「17.4.1　トレースレベルとは」を参照してください。

- Trace Size

  HADB ODBC ドライバトレースファイルの 1 ファイル当たりのサイズの上限値（単位：メガバイト）を指定します。32～1,024 の値（半角の数値）を指定できます。

  指定をしない場合，または不正な値を指定した場合は，256 が仮定されます。

- Trace Directory Path

  HADB ODBC ドライバトレースファイルを格納するフォルダを絶対パスで指定します。

  [...] ボタンをクリックして，フォルダを選択してください。

  実行ユーザに対するアクセス権限があるフォルダを指定してください。

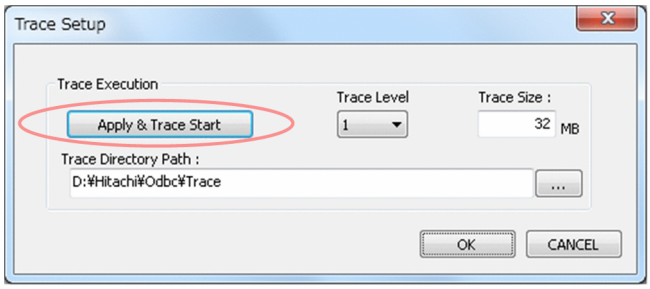
  また，210 バイト以下のパス名を指定してください。

  重要

  次に示す場合は，HADB ODBC ドライバトレース情報は出力されません。

  - [Trace Directory Path] にパス名を指定しない場合
  - [Trace Directory Path] に不正なパスを指定した場合
  - [Trace Directory Path] に 211 バイト以上のパス名を指定した場合

5. [Apply & Trace Start] ボタンをクリックする

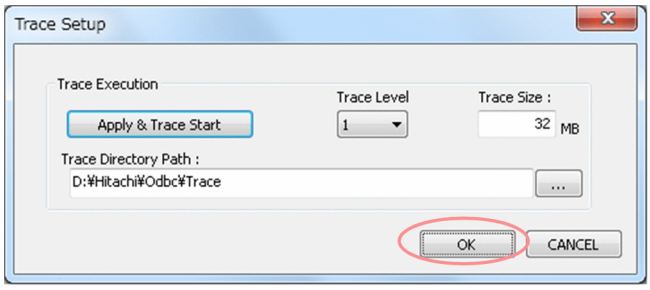


HADB ODBC ドライバトレース情報の出力が開始されます。

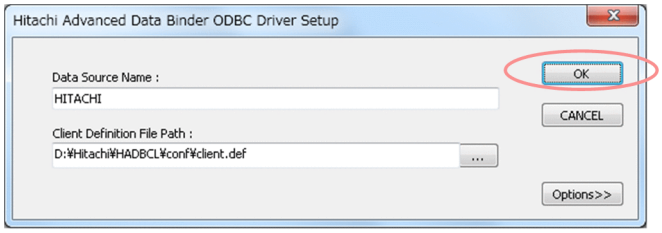
メモ

- 手順の 4.の操作を実行すると，［Apply & Trace Start］ボタンが活性化されます。
- 設定を中止する場合は，［Apply & Trace Start］ボタンをクリックする前に［CANCEL］をクリックしてください。

6. ［OK］をクリックする



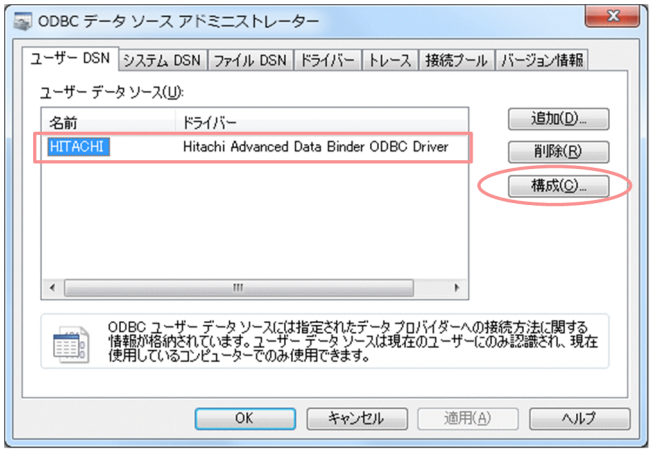
7. ［Hitachi Advanced Data Binder ODBC Driver Setup］ダイアログで，［OK］をクリックする



(2) HADB ODBC ドライバトレース情報の出力をやめる場合

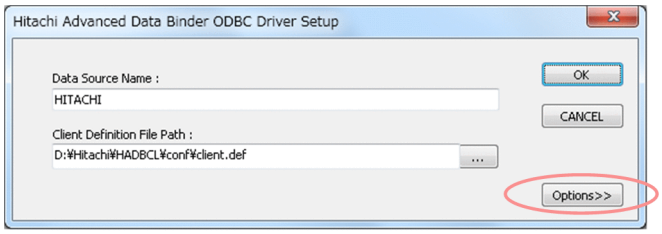
1. ［Hitachi Advanced Data Binder ODBC Driver Setup］ダイアログを表示する

   ODBC データソースアドミニストレーターの［ユーザー DSN］タブで，HADB ODBC ドライバを使用するデータソースを選択し，［構成(C)...］ボタンをクリックします。

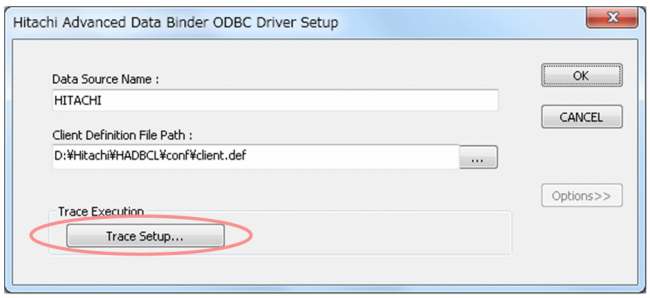


> **メモ**
>
> ODBC データソースアドミニストレーターの［システム DSN］タブでも，同じ操作をして［Hitachi Advanced Data Binder ODBC Driver Setup］ダイアログを表示することができます。

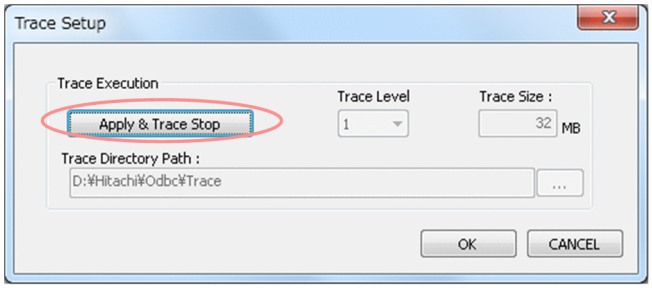
2. ［Hitachi Advanced Data Binder ODBC Driver Setup］ダイアログで，［Options>>］ボタンをクリックする



3. ［Trace Setup...］ボタンをクリックする

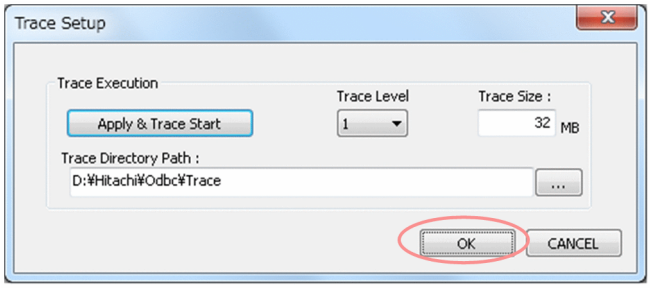


4. ［Apply & Trace Stop］ボタンをクリックする

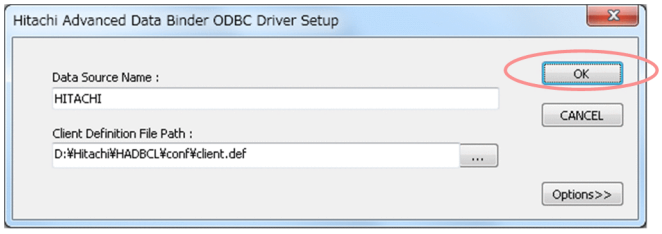


HADB ODBC ドライバトレース情報の出力が停止されます。

5. ［OK］をクリックする



6. ［Hitachi Advanced Data Binder ODBC Driver Setup］ダイアログで，［OK］をクリックする



17.3.2　環境変数で設定する方法

次に示す場合は，環境変数を指定して HADB ODBC ドライバトレース情報を出力してください。

- SQLDriverConnect，SQLDriverConnectW，SQLBrowseConnect，またはSQLBrowseConnectW 関数で，データソースを使用しないで ODBC ドライバ名称で接続（接続属性にDRIVER を指定）する場合
- SQLConnect またはSQLConnectW 関数などの接続関数で，ファイル DSN を使用する場合

指定する環境変数を次に示します。

- ADBODBTRC
  HADB ODBC ドライバトレース情報を出力するかどうかを指定します。
- ADBODBTRCSIZE
  HADB ODBC ドライバトレースファイルの 1 ファイル当たりのサイズの上限値を指定します。
- ADBODBTRCPATH
  HADB ODBC ドライバトレースファイルを格納するフォルダを指定します。
  実行ユーザに対するアクセス権限があるフォルダを指定してください。
- ADBODBTRCLV
  HADB ODBC ドライバトレースのトレースレベルを指定します。

各環境変数の指定方法については，「4.3.1　Windows 版の HADB クライアントの場合」を参照してください。

なお，環境変数を次のように指定した場合，その環境変数の指定は無効になります。この場合，その環境変数にはデフォルト値が仮定されます。

```
export ADBODBTRCSIZE=
```

重要

ODBC ドライバ経由で AP が HADB サーバにアクセスしている最中に，上記の環境変数の値を変更しても，その AP のプロセスに対しては，変更後の値は有効になりません。

17.3.3 ODBC データソースアドミニストレーターと環境変数の設定値の優先順位

ODBC データソースアドミニストレーターの設定と，環境変数の設定の両方を実施した場合，基本的には ODBC データソースアドミニストレーターの設定値が優先されます。ただし，次のケースでは，ODBC データソースアドミニストレーターでの設定が有効になりません。

- 環境変数ADBODBTRC にYES を指定しているときに，ODBC データソースアドミニストレーターで［Apply & Trace Stop］ボタンをクリックして，HADB ODBC ドライバトレース情報の出力を停止した

この場合，環境変数ADBODBTRC の設定値が有効になるため，HADB ODBC ドライバトレース情報の出力は継続されます。

17.4　HADB ODBC ドライバトレース情報に出力される情報

ここでは，HADB ODBC ドライバトレース情報に出力される情報について説明します。

なお，出力する情報量をトレースレベルで選択することができます。

17.4.1　トレースレベルとは

HADB ODBC ドライバトレース情報に出力される情報を，トレースレベルを指定して選択できます。トレースレベルには 2 種類あり，簡易トレースのトレースレベル 1 と，詳細トレースのトレースレベル 2 があります。各トレースレベルと出力される情報の関係を次の表に示します。

表 17-3　各トレースレベルと出力される情報の関係

出力情報	トレースレベル	
	トレースレベル 1	トレースレベル 2
アクセス種別	×	○
関数名	○	○
引数	×	○
ハンドル	○	○
時刻	○	○
実行結果	○	○
SQLSTATE	○	○
エラーメッセージ	○	○
実行 SQL	○	○
付加情報	○	○

（凡例）

○：出力します。

×：出力しません。

なお，トレースレベル 1 の場合は，1 行に 256 バイトまでしか情報を出力しません。一方，トレースレベル 2 の場合は，すべての情報を出力します。そのため，トレースレベル 2 を選択した場合，出力情報が多くなる分，性能に影響を及ぼします。

また，HADB ODBC ドライバトレース情報の出力タイミングが，トレースレベルによって次のように異なります。

- トレースレベル 1

ODBC 関数の終了直前に出力されます。

- トレースレベル 2

  ODBC 関数の開始時および終了直前に出力されます。

> **メモ**
>
> 詳細なトラブルシュート情報を出力する場合は，トレースレベル 2 を選択する必要がありますが，性能に影響があることを考慮してください。

17.4.2 トレースレベル 1 の場合に出力される情報

トレースレベル 1 の場合の出力例を次に示します。

■出力例（トレースレベル 1 の場合）

```
[Trace Start Time] 2023/04/12 19:16:32.108
[Process ID] 7712
[Module Name] odbct32w
[Platform] Win32
[ODBC environment variables]
  ADBCLTLANG(UTF8)
  ADBCOBTRC(YES)
  ADBCOBTRCSIZE(32)
  ADBCOBTRCPATH(C:¥HADBCL¥spool)
  ADBCOBTRCLV(1)
  ADBCOBAPMODE(NORMAL)
  ADBCOBSGDST(USE)

[RETURN Value]
  0 : SQL_SUCCESS
  1 : SQL_SUCCESS_WITH_INFO
  2 : SQL_STILL_EXECUTING
 99 : SQL_NEED_DATA
100 : SQL_NO_DATA
 -1 : SQL_ERROR
 -2 : SQL_INVALID_HANDLE

FUNCTION            HANDLE     START-TIME              END-TIME                RETURN SQL   CON_ID CON_NUM    OPTION
                                                                                   STATE

------------------- ---------- ----------------------- ----------------------- ------ ----- ------ --------- ----------------------------------------------------------------
SQLAllocHandle      0x00000000 2023/04/12 19:16:32.108 2023/04/12 19:16:32.108      0 00000      *         * HandleType=SQL_HANDLE_ENV, Address=0x006ff970
SQLSetEnvAttr       0x006ff970 2023/04/12 19:16:32.111 2023/04/12 19:16:32.112      0 00000      *         * Attribute=SQL_ATTR_ODBC_VERSION
SQLAllocHandle      0x006ff970 2023/04/12 19:16:32.114 2023/04/12 19:16:32.114      0 00000      *         * HandleType=SQL_HANDLE_DBC, Address=0x0074c500
SQLGetInfoW         0x0074c500 2023/04/12 19:16:32.115 2023/04/12 19:16:32.115      0 00000      *         * InfoType=SQL_DRIVER_ODBC_VER
SQLSetConnectAttrW  0x0074c500 2023/04/12 19:16:32.116 2023/04/12 19:16:32.116      0 00000      *         * Attribute=SQL_ATTR_ANSI_APP
SQLDriverConnectW   0x0074c500 2023/04/12 19:16:32.116 2023/04/12 19:16:32.167      0 00000      1        15
InConnectionStringW="DSN=HADB_L_IN30;UID=ADBUSER;PWD=*;"
*** Connect Info ********************************************************************
[DataSourceName] HADB_L_IN30
[Client Definition File Path] C:¥HADBCL¥conf¥client.def
[Hitachi Advanced Data Binder ODBC Driver Version] 05.08
[ProcessID] 7712
******************************************************************************************
SQLGetDiagRecW      0x0074c500 2023/04/12 19:16:32.168 2023/04/12 19:16:32.168    100            1        15 *
SQLGetFunctions     0x0074c500 2023/04/12 19:16:32.169 2023/04/12 19:16:32.169      0 00000      1        15 FunctionID=SQL_API_ODBC3_ALL_FUNCTIONS
SQLGetInfoW         0x0074c500 2023/04/12 19:16:32.170 2023/04/12 19:16:32.170      0 00000      1        15 InfoType=SQL_CURSOR_COMMIT_BEHAVIOR
SQLGetInfoW         0x0074c500 2023/04/12 19:16:32.171 2023/04/12 19:16:32.171      0 00000      1        15 InfoType=SQL_CURSOR_ROLLBACK_BEHAVIOR
SQLGetInfoW         0x0074c500 2023/04/12 19:16:32.172 2023/04/12 19:16:32.172      0 00000      1        15 InfoType=SQL_GETDATA_EXTENSIONS
SQLAllocHandle      0x0074c500 2023/04/12 19:16:32.176 2023/04/12 19:16:32.176      0 00000      1        15 HandleType=SQL_HANDLE_STMT, Address=0x006eaa20
SQLGetStmtAttrW     0x006eaa20 2023/04/12 19:16:32.177 2023/04/12 19:16:32.177      0 00000      1        15 Attribute=SQL_ATTR_APP_ROW_DESC
SQLGetStmtAttrW     0x006eaa20 2023/04/12 19:16:32.178 2023/04/12 19:16:32.178      0 00000      1        15 Attribute=SQL_ATTR_APP_PARAM_DESC
SQLGetStmtAttrW     0x006eaa20 2023/04/12 19:16:32.179 2023/04/12 19:16:32.179      0 00000      1        15 Attribute=SQL_ATTR_IMP_ROW_DESC
SQLGetStmtAttrW     0x006eaa20 2023/04/12 19:16:32.180 2023/04/12 19:16:32.180      0 00000      1        15 Attribute=SQL_ATTR_IMP_PARAM_DESC
SQLGetDescFieldW    0x00702868 2023/04/12 19:16:32.182 2023/04/12 19:16:32.183      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_ALLOC_TYPE
SQLGetDescFieldW    0x00702868 2023/04/12 19:16:32.184 2023/04/12 19:16:32.184      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_ARRAY_SIZE
SQLGetDescFieldW    0x00702868 2023/04/12 19:16:32.185 2023/04/12 19:16:32.185      0 00000      1        15 RecNumber=0,
FieldIdentifier=SQL_DESC_ARRAY_STATUS_PTR
SQLGetDescFieldW    0x00702868 2023/04/12 19:16:32.186 2023/04/12 19:16:32.187      0 00000      1        15 RecNumber=0,
FieldIdentifier=SQL_DESC_BIND_OFFSET_PTR
SQLGetDescFieldW    0x00702868 2023/04/12 19:16:32.188 2023/04/12 19:16:32.188      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_BIND_TYPE
SQLGetDescFieldW    0x00702868 2023/04/12 19:16:32.189 2023/04/12 19:16:32.189      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_COUNT
SQLGetDescFieldW    0x006f99f8 2023/04/12 19:16:32.190 2023/04/12 19:16:32.190      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_ALLOC_TYPE
SQLGetDescFieldW    0x006f99f8 2023/04/12 19:16:32.191 2023/04/12 19:16:32.192      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_ARRAY_SIZE
SQLGetDescFieldW    0x006f99f8 2023/04/12 19:16:32.193 2023/04/12 19:16:32.193      0 00000      1        15 RecNumber=0,
FieldIdentifier=SQL_DESC_ARRAY_STATUS_PTR
SQLGetDescFieldW    0x006f99f8 2023/04/12 19:16:32.194 2023/04/12 19:16:32.194      0 00000      1        15 RecNumber=0,
FieldIdentifier=SQL_DESC_BIND_OFFSET_PTR
SQLGetDescFieldW    0x006f99f8 2023/04/12 19:16:32.195 2023/04/12 19:16:32.195      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_BIND_TYPE
SQLGetDescFieldW    0x006f99f8 2023/04/12 19:16:32.196 2023/04/12 19:16:32.196      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_COUNT
SQLGetDescFieldW    0x00702e68 2023/04/12 19:16:32.198 2023/04/12 19:16:32.198     -1 HY007      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_ALLOC_TYPE
 <Message>[Hitachi Advanced Data Binder][ODBC Driver]KFAA72003-E An error occurred in the ODBC driver. (SQLSTATE = HY007)
SQLGetDescFieldW    0x00702e68 2023/04/12 19:16:32.199 2023/04/12 19:16:32.199     -1 HY007      1        15 RecNumber=0,
FieldIdentifier=SQL_DESC_ARRAY_STATUS_PTR
 <Message>[Hitachi Advanced Data Binder][ODBC Driver]KFAA72003-E An error occurred in the ODBC driver. (SQLSTATE = HY007)
SQLGetDescFieldW    0x00702e68 2023/04/12 19:16:32.200 2023/04/12 19:16:32.200     -1 HY007      1        15 RecNumber=0,
FieldIdentifier=SQL_DESC_ROWS_PROCESSED_PTR
 <Message>[Hitachi Advanced Data Binder][ODBC Driver]KFAA72003-E An error occurred in the ODBC driver. (SQLSTATE = HY007)
SQLGetDescFieldW    0x00702b68 2023/04/12 19:16:32.201 2023/04/12 19:16:32.202      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_ALLOC_TYPE
SQLGetDescFieldW    0x00702b68 2023/04/12 19:16:32.203 2023/04/12 19:16:32.203      0 00000      1        15 RecNumber=0,
FieldIdentifier=SQL_DESC_ARRAY_STATUS_PTR
SQLGetDescFieldW    0x00702b68 2023/04/12 19:16:32.204 2023/04/12 19:16:32.204      0 00000      1        15 RecNumber=0, FieldIdentifier=SQL_DESC_COUNT
SQLGetDescFieldW    0x00702b68 2023/04/12 19:16:32.205 2023/04/12 19:16:32.205      0 00000      1        15 RecNumber=0,
FieldIdentifier=SQL_DESC_ROWS_PROCESSED_PTR
SQLCancel           0x006eaa20 2023/04/12 19:16:37.645 2023/04/12 19:16:37.645      0 00000      1        15 *
SQLFreeHandle       0x006eaa20 2023/04/12 19:16:37.647 2023/04/12 19:16:37.647      0 00000      1        15 *
SQLDisconnect       0x0074c500 2023/04/12 19:16:37.649 2023/04/12 19:16:37.654      0 00000      *         * *
SQLFreeHandle       0x0074c500 2023/04/12 19:16:37.656 2023/04/12 19:16:37.656      0 00000      *         * *
SQLFreeHandle       0x006ff970 2023/04/12 19:16:37.657 2023/04/12 19:16:37.657      0 00000      *         * *
```

トレースレベル 1 の場合，1 行に出力される文字列は最大 256 バイトです。ただし，実行要求した SQL 文，および診断情報メッセージについては，すべての情報が出力されます。

各出力情報について次の表で説明します。

表 17-4　HADB ODBC ドライバトレース情報に出力される情報（トレースレベル 1 の場合）

項番	出力項目	説明
1	[Trace Start Time]	HADB ODBC ドライバトレース情報の出力を開始した日時を出力します。 日時の取得に失敗した場合，日付および時刻のすべての欄に 0 を出力します。
2	[Process ID]	プロセス ID を出力します。
3	[Module Name]	HADB ODBC ドライバを使用するモジュール名称を出力します。モジュール名称が不明な場合は，unknown を出力します。

項番	出力項目	説明
4	[Platform]	HADB ODBC ドライバが動作しているプラットフォームを出力します。
5	[ODBC environment variables]	HADB ODBC ドライバが使用している環境変数の値を出力します。環境変数に指定された値ではなく，HADB ODBC ドライバで有効となっている値（デフォルト値や ODBC データソースアドミニストレーターで変更された値）が出力されます。 • ADBCLTLANG HADB クライアントで使用している文字コード • ADBODBTRC HADB ODBC ドライバトレース情報の出力有無 • ADBODBTRCSIZE HADB ODBC ドライバトレースファイルの 1 ファイル当たりのサイズの上限値 • ADBODBTRCPATH HADB ODBC ドライバトレースファイルを格納するフォルダの絶対パス • ADBODBTRCLV トレースレベルの指定 • ADBODBAPMODE HADB ODBC ドライバのアプリケーションモード • ADBODBSGDST HADB ODBC ドライバのSQLGetData 関数での分割取得機能の使用有無 上記の環境変数については，「4.3.1　Windows 版の HADB クライアントの場合」を参照してください。
6	[RETURN Value]	ODBC 関数の戻り値の説明です。 項番 11 のRETURN のSQLRETURN 型の整数値に対応する定義の説明です。
7	FUNCTION	ODBC 関数名を出力します。
8	HANDLE	ODBC 関数に引数として渡されたハンドルの値を出力します。
9	START-TIME	ODBC 関数の実行開始日時を出力します。 日時の取得に失敗した場合，日付および時刻のすべての欄に 0 を出力します。
10	END-TIME	ODBC 関数の実行終了日時を出力します。 日時の取得に失敗した場合，日付および時刻のすべての欄に 0 を出力します。
11	RETURN	ODBC 関数の実行結果であるSQLRETURN 型の値を出力します。 値の意味については，上部に出力される［RETURN Value］を参照してください。
12	SQLSTATE	SQLSTATE を出力します。 ODBC 関数を実行した結果，SQLSTATE が出力された場合に限り出力されます。
13	CON_ID	コネクション番号を出力します。 SQL トレース情報に出力されるコネクション番号と同じものが出力されます。 HADB サーバと接続する前の場合は，*を出力します。
14	CON_NUM	コネクション通番を出力します。 SQL トレース情報に出力されるコネクション通番と同じものが出力されます。

項番	出力項目	説明
		HADB サーバと接続する前の場合は，*を出力します。
15	OPTION	オプション情報を出力します。詳細については，「表 17-5　OPTION に出力されるオプション情報」を参照してください。 ODBC 関数ごとに出力情報が異なります。 出力する情報がない場合は，*を出力します。 シンボル名で出力する情報のうち，不明な値の場合は，シンボル値の代わりに unknown が出力されます。
16	Connect Info	HADB サーバとの接続が確立したときに出力される情報を出力します。
17	[DataSourceName]	接続しているデータソース名を出力します。 データソースを使用していない場合，またはファイル DSN を使用している場合は，*を出力します。
18	[Client Definition File Path]	クライアント定義ファイルのパスを出力します。
19	[Hitachi Advanced Data Binder ODBC Driver Version]	HADB ODBC ドライバのバージョンを出力します。
20	[ProcessID]	HADB クライアントのプロセス ID を出力します。
21	<SQL>	実行要求した SQL 文を出力します。 次の ODBC 関数が実行された場合に，ODBC 関数の引数として渡された SQL 文を出力します。 • SQLPrepare • SQLExecDirect
22	<Message>	診断情報メッセージを出力します。 ODBC 関数の実行結果が次の場合に出力されます。 • SQL_ERROR • SQL_SUCCESS_WITH_INFO • SQL_NEED_DATA（SQLBrowseConnect の場合）

上記の表のOPTION に出力されるオプション情報を次の表に示します。

表 17-5　OPTION に出力されるオプション情報

項番	分類	ODBC 関数	OPTION に出力されるオプション情報
1	データソースとの接続	SQLAllocHandle	• HandleType（*シンボル値*） • Address：OutputHandlePtr パラメタに返却したアドレス値
2		SQLConnect(W)	• ServerName • UserName
3		SQLDriverConnect(W)	InConnectionString パスワードは，*1 個を出力します。
4		SQLBrowseConnect(W)	
5	ドライバおよびデータソースの情報取得	SQLDataSources(W)	なし

項番	分類	ODBC 関数	OPTION に出力されるオプション情報
6		SQLDrivers(W)	
7		SQLGetInfo(W)	InfoType
8		SQLGetFunctions	FunctionID
9		SQLGetTypeInfo(W)	DataType
10	ドライバオプションの設定および取得	SQLSetConnectAttr(W)	Attribute
11		SQLGetConnectAttr(W)	
12		SQLSetEnvAttr	
13		SQLGetEnvAttr	
14		SQLSetStmtAttr(W)	
15		SQLGetStmtAttr(W)	
16	ディスクリプタ値の設定	SQLGetDescField(W)	• RecNumber • FieldIdentifier（*シンボル値*）
17		SQLGetDescRec(W)	• RecNumber • *Name
18		SQLSetDescField(W)	• RecNumber • FieldIdentifier
19		SQLSetDescRec	• RecNumber • Type • SubType
20		SQLCopyDesc	• Source：SourceDescHandle の値 • Target：TargetDescHandle の値
21	SQL 要求の作成	SQLPrepare(W)	• tran_id：トランザクション ID • stmt_hdl：文ハンドル ID • sql_serial_num：SQL 文通番
22		SQLBindParameter	• ParameterNumber • ValueType • ParameterType
23		SQLGetCursorName(W)	なし
24		SQLSetCursorName(W)	
25		SQLDescribeParam	ParameterNumber
26		SQLNumParams	*ParameterCountPtr
27	SQL の実行	SQLExecute	• tran_id：トランザクション ID • stmt_hdl：文ハンドル ID • sql_serial_num：SQL 文通番
28		SQLExecDirect(W)	

項番	分類	ODBC 関数	OPTION に出力されるオプション情報
29		SQLNativeSql(W)	OutStatementText
30		SQLParamData	なし
31		SQLPutData	StrLen_or_Ind
32	実行結果および実行結果情報の取得	SQLRowCount	*RowCountPtr
33		SQLNumResultCols	*ColumnCountPtr
34		SQLDescribeCol(W)	ColumnNumber
35		SQLColAttribute(W)	• ColumnNumber • FieldIdentifier（*シンボル値*）
36		SQLBindCol	• ColumnNumber • TargetType（*シンボル値*）
37		SQLFetch	なし
38		SQLFetchScroll	
39		SQLGetData	• ColumnNumber • TargetType（*シンボル値*）
40		SQLSetPos	なし
41		SQLBulkOperations	
42		SQLMoreResults	
43		SQLGetDiagField(W)	DiagIdentifier
44		SQLGetDiagRec(W)	なし
45	データソースのシステム情報の取得	SQLColumnPrivileges(W)	なし
46		SQLColumns(W)	
47		SQLForeignKeys(W)	
48		SQLPrimaryKeys(W)	
49		SQLProcedureColumns(W)	
50		SQLProcedures(W)	
51		SQLSpecialColumns(W)	
52		SQLStatistics(W)	
53		SQLTablePrivileges(W)	
54		SQLTables(W)	
55	SQL 実行の終了	SQLFreeStmt	Option Option=SQL_CLOSE の場合は，次の情報も出力します。 • tran_id：トランザクション ID

項番	分類	ODBC 関数	OPTION に出力されるオプション情報
			• stmt_hdl：文ハンドル ID • sql_serial_num：SQL 文通番
56		SQLCloseCursor	• tran_id：トランザクション ID • stmt_hdl：文ハンドル ID • sql_serial_num：SQL 文通番
57		SQLCancel	なし
58		SQLEndTran	• CompletionType • tran_id：トランザクション ID
59	データソースとの切断	SQLDisconnect	なし
60		SQLFreeHandle	

注

- *付きの変数名は出力値を出力します。それ以外の場合は，入力値を出力します。
- (*シンボル値*)は，シンボル値に置き換えた文字列を出力します。それ以外の場合は，数値を出力します。不明な値の場合は，unknown を出力します。
- トランザクションが決着済みのときなど，tran_id，stmt_htdl，およびsql_serial_num が取得できない場合は，各値に*を出力します。
- オプション情報に 2 つ以上の情報を出力する場合，各情報をコンマで区切って出力します。

17.4.3　トレースレベル 2 の場合に出力される情報

トレースレベル 2 の場合の出力例を次に示します。

■出力例（トレースレベル 2 の場合）

```
[Trace Start Time] 2023/04/12 19:15:02.278
[Process ID] 7712
[Module Name] odbct32w
[Platform] Win32
[ODBC environment variables]
  ADBCLTLANG(UTF8)
  ADBODBTRC(YES)
  ADBODBTRCSIZE(32)
  ADBODBTRCPATH(C:¥HADBCL¥spool)
  ADBODBTRCLV(2)
  ADBODBAPMODE(NORMAL)
  ADBODBSGDST(USE)

ACC FUNCTION            HANDLE     START-TIME             END-TIME               RETURN                SQL   CON_ID CON_NUM
ESS                                                                                                    STATE

--- ------------------- ---------- ---------------------- ---------------------- --------------------- ----- ------ ----------
[E] SQLAllocHandle      0x00000000 2023/04/12 19:15:02.278                       *                     *     *      *          *
 [Input]  HandleType(SQLSMALLINT)      = 1(SQL_HANDLE_ENV)
          InputHandle(SQLHANDLE)       = 0x00000000
          OutputHandlePtr(SQLHANDLE)   = 0x0064eef4
[R] SQLAllocHandle      0x00000000 2023/04/12 19:15:02.278 2023/04/12 19:15:02.282 SQL_SUCCESS 00000      *          *
 [Output] OutputHandlePtr(SQLHANDLE)   = 0x0064eef4(0x0074d498)
[E] SQLSetEnvAttr       0x0074d498 2023/04/12 19:15:02.283                       *                     *     *      *          *
 [Input]  EnvironmentHandle(SQLHENV)   = 0x0074d498
          Attribute(SQLINTEGER)        = 200(SQL_ATTR_ODBC_VERSION)
          ValuePtr(SQLPOINTER)         = 3
          StringLength(SQLINTEGER)     = 0
[R] SQLSetEnvAttr       0x0074d498 2023/04/12 19:15:02.283 2023/04/12 19:15:02.284 SQL_SUCCESS 00000      *          *
[E] SQLAllocHandle      0x0074d498 2023/04/12 19:15:02.285                       *                     *     *      *          *
 [Input]  HandleType(SQLSMALLINT)      = 2(SQL_HANDLE_DBC)
          InputHandle(SQLHANDLE)       = 0x0074d498
          OutputHandlePtr(SQLHANDLE)   = 0x0064eee0
[R] SQLAllocHandle      0x0074d498 2023/04/12 19:15:02.285 2023/04/12 19:15:02.286 SQL_SUCCESS 00000      *          *
 [Output] OutputHandlePtr(SQLHANDLE)   = 0x0064eee0(0x006e3df0)
[E] SQLGetInfoW         0x006e3df0 2023/04/12 19:15:02.286                       *                     *     *      *          *
 [Input]  ConnectionHandle(SQLHDBC)    = 0x006e3df0
          InfoType(SQLUSMALLINT)       = 77(SQL_DRIVER_ODBC_VER)
          InfoValuePtr(SQLPOINTER)     = 0x000dc020
          BufferLength(SQLSMALLINT)    = 12
          StringLengthPtr(SQLSMALLINT*) = 0x000db404
[R] SQLGetInfoW         0x006e3df0 2023/04/12 19:15:02.286 2023/04/12 19:15:02.287 SQL_SUCCESS 00000      *          *
 [Output] InfoValuePtr(SQLPOINTER)     = 0x000dc020("03.52")
          StringLengthPtr(SQLSMALLINT*) = 0x000db404(10)
[E] SQLSetConnectAttrW  0x006e3df0 2023/04/12 19:15:02.288                       *                     *     *      *          *
 [Input]  ConnectionHandle(SQLHDBC)    = 0x006e3df0
          Attribute(SQLINTEGER)        = 115(SQL_ATTR_ANSI_APP)
          ValuePtr(SQLPOINTER)         = 0
          StringLength(SQLINTEGER)     = -5
[R] SQLSetConnectAttrW  0x006e3df0 2023/04/12 19:15:02.288 2023/04/12 19:15:02.290 SQL_SUCCESS 00000      *          *
[E] SQLDriverConnectW   0x006e3df0 2023/04/12 19:15:02.291                       *                     *     *      *          *
 [Input]  ConnectionHandle(SQLHDBC)    = 0x006e3df0
          WindowHandle(SQLHWND)        = 0x002309de
          InConnectionStringW(SQLWCHAR*)= 0x0064e728("DSN=HADB_LIN30;UID=ADBUSER;PWD=*;")
          StringLength1(SQLSMALLINT)   = -3
          OutConnectionStringW(SQLWCHAR*)= 0x00000000
          BufferLength(SQLSMALLINT)    = 0
          StringLength2Ptr(SQLSMALLINT*)= 0x00000000
          DriverCompletion(SQLUSMALLINT)= 1(SQL_DRIVER_COMPLETE)
[R] SQLDriverConnectW   0x006e3df0 2023/04/12 19:15:02.291 2023/04/12 19:15:02.362 SQL_SUCCESS 00000      1          14
 [Output] OutConnectionStringW(SQLWCHAR*)= 0x00000000("")
          StringLength2Ptr(SQLSMALLINT*)= 0x00000000()
*** Connect Info ********************************************************************************
[DataSourceName] HADB_LIN30
[Client Definition File Path] C:¥HADBCL¥conf¥client.def
[Hitachi Advanced Data Binder ODBC Driver Version] 05.08
[ProcessID] 7712
*************************************************************************************************
[E] SQLGetDiagRecW      0x006e3df0 2023/04/12 19:15:02.363                       *                     *     *      1          14
 [Input]  HandleType(SQLSMALLINT)      = 2
          Handle(SQLHANDLE)            = 0x006e3df0
          RecNumber(SQLSMALLINT)       = 1
          SQLStateW(SQLWCHAR*)         = 0x000dbae0
          NativeErrorPtr(SQLINTEGER*)  = 0x00000000
          MessageTextW(SQLWCHAR*)      = 0x00000000
          BufferLength(SQLSMALLINT)    = 0
          TextLengthPtr(SQLSMALLINT*)  = 0x000dbacc
[R] SQLGetDiagRecW      0x006e3df0 2023/04/12 19:15:02.363 2023/04/12 19:15:02.364 SQL_NO_DATA           1          14
[E] SQLGetFunctions     0x006e3df0 2023/04/12 19:15:02.365                       *                     *     *      1          14
 [Input]  ConnectionHandle(SQLHDBC)    = 0x006e3df0
          FunctionId(SQLUSMALLINT)     = 999(SQL_API_ODBC3_ALL_FUNCTIONS)
          SupportedPtr(SQLUSMALLINT*)  = 0x000dbe28
[R] SQLGetFunctions     0x006e3df0 2023/04/12 19:15:02.365 2023/04/12 19:15:02.365 SQL_SUCCESS 00000      1          14
 [Output] SupportedPtr(SQLUSMALLINT*)  = 0x000dbe28
[E] SQLGetInfoW         0x006e3df0 2023/04/12 19:15:02.366                       *                     *     *      1          14
 [Input]  ConnectionHandle(SQLHDBC)    = 0x006e3df0
          InfoType(SQLUSMALLINT)       = 23(SQL_CURSOR_COMMIT_BEHAVIOR)
          InfoValuePtr(SQLPOINTER)     = 0x0064ef4a
          BufferLength(SQLSMALLINT)    = 2
          StringLengthPtr(SQLSMALLINT*) = 0x00000000
[R] SQLGetInfoW         0x006e3df0 2023/04/12 19:15:02.366 2023/04/12 19:15:02.367 SQL_SUCCESS 00000      1          14
 [Output] InfoValuePtr(SQLPOINTER)     = 0x0064ef4a(1)
          StringLengthPtr(SQLSMALLINT*) = 0x00000000()
[E] SQLGetInfoW         0x006e3df0 2023/04/12 19:15:02.368                       *                     *     *      1          14
 [Input]  ConnectionHandle(SQLHDBC)    = 0x006e3df0
          InfoType(SQLUSMALLINT)       = 24(SQL_CURSOR_ROLLBACK_BEHAVIOR)
          InfoValuePtr(SQLPOINTER)     = 0x0064ef4c
          BufferLength(SQLSMALLINT)    = 2
          StringLengthPtr(SQLSMALLINT*) = 0x00000000
```

各出力情報について次の表で説明します。

表 17-6　HADB ODBC ドライバトレース情報に出力される情報（トレースレベル 2 の場合）

項番	出力項目	説明
1	[Trace Start Time]	HADB ODBC ドライバトレース情報の出力を開始した日時を出力します。 日時の取得に失敗した場合，日付および時刻のすべての欄に 0 を出力します。
2	[Process ID]	プロセス ID を出力します。

項番	出力項目	説明
3	[Module Name]	HADB ODBC ドライバを使用するモジュール名称を出力します。モジュール名称が不明な場合は，unknown を出力します。
4	[Platform]	HADB ODBC ドライバが動作しているプラットフォームを出力します。
5	[ODBC environment variables]	HADB ODBC ドライバが使用している環境変数の値を出力します。環境変数に指定された値ではなく，HADB ODBC ドライバで有効となっている値（デフォルト値や ODBC データソースアドミニストレーターで変更された値）が出力されます。 • ADBCLTLANG HADB クライアントで使用している文字コード • ADBODBTRC HADB ODBC ドライバトレース情報の出力有無 • ADBODBTRCSIZE HADB ODBC ドライバトレースファイルの 1 ファイル当たりのサイズの上限値 • ADBODBTRCPATH HADB ODBC ドライバトレースファイルを格納するフォルダの絶対パス • ADBODBTRCLV トレースレベルの指定 • ADBODBAPMODE HADB ODBC ドライバのアプリケーションモード • ADBODBSGDST HADB ODBC ドライバのSQLGetData 関数での分割取得機能の使用有無 上記の環境変数については，「4.3.1　Windows 版の HADB クライアントの場合」を参照してください。
6	ACCESS	アクセス種別を出力します。 • [E]：関数の呼び出し • [R]：関数からの戻り
7	FUNCTION	ODBC 関数名を出力します。
8	HANDLE	ODBC 関数に引数として渡されたハンドルの値を出力します。 32 ビット環境の場合は，プレフィックス 0x(2 桁)＋ 8 桁で出力されます。 64 ビット環境の場合は，16 桁の 16 進数で出力されます。
9	START-TIME	ODBC 関数の実行開始日時を出力します。 日時の取得に失敗した場合，日付および時刻のすべての欄に 0 を出力します。
10	END-TIME	ODBC 関数の実行終了日時を出力します。 日時の取得に失敗した場合，日付および時刻のすべての欄に 0 を出力します。 アクセス種別が[E]の場合は，*を出力します。
11	RETURN	ODBC 関数の実行結果であるSQLRETURN 型の値を示すシンボル名を出力します。不明な値の場合は，シンボル名の代わりにunknown を出力します。 アクセス種別が[E]の場合は，*を出力します。
12	SQLSTATE	SQLSTATE を出力します。

項番	出力項目	説明
		ODBC 関数を実行した結果，SQLSTATE が出力された場合に限り出力されます。 アクセス種別が[E]の場合は，*を出力します。
13	CON_ID	コネクション番号を出力します。 SQL トレース情報に出力されるコネクション番号と同じものが出力されます。 HADB サーバと接続する前の場合は，*を出力します。
14	CON_NUM	コネクション通番を出力します。 SQL トレース情報に出力されるコネクション通番と同じものが出力されます。 HADB サーバと接続する前の場合は，*を出力します。
15	[Input]	入力パラメタを出力します。次の形式で出力されます。 • *変数名(データ型名) = 値(シンボル名または参照先データ)* 値の後ろの（）は省略されることがあります。 不明な値の場合は，シンボル名の代わりにunknown を出力します。
16	[Output]	出力パラメタを出力します。次の形式で出力されます。 • *変数名(データ型名) = 値(シンボル名または参照先データ)* 値の後ろの（）は省略されることがあります。 不明な値の場合は，シンボル名の代わりにunknown を出力します。 ODBC 関数が出力パラメタを持たない場合は，出力されません。
17	接続ユーザ名およびパスワード	接続ユーザ名およびパスワードを出力します。 パスワードは，*1 個を出力します。
18	Connect Info	HADB サーバとの接続が確立したときに出力される情報を出力します。
19	[DataSourceName]	接続しているデータソース名を出力します。 データソースを使用していない場合，またはファイル DSN を使用している場合は，*を出力します。
20	[Client Definition File Path]	クライアント定義ファイルのパスを出力します。
21	[Hitachi Advanced Data Binder ODBC Driver Version]	HADB ODBC ドライバのバージョンを出力します。
22	[ProcessID]	HADB クライアントのプロセス ID を出力します。
23	[SYSTEM]	HADB 固有の情報を出力します。 詳細については，「表 17-7 [SYSTEM] に出力される情報」を参照してください。 ODBC 関数ごとに出力情報が異なります。
24	<Message>	診断情報メッセージを出力します。 ODBC 関数の実行結果が次の場合に出力されます。 • SQL_ERROR • SQL_SUCCESS_WITH_INFO • SQL_NEED_DATA（SQLBrowseConnect の場合）

上記の表の［SYSTEM］に出力される情報を次の表に示します。

表 17-7 ［SYSTEM］に出力される情報

項番	分類	ODBC 関数名	［SYSTEM］に出力される情報
1	データソースとの接続	SQLAllocHandle	なし
2		SQLConnect(W)	
3		SQLDriverConnect(W)	
4		SQLBrowseConnect(W)	
5	ドライバおよびデータソースの情報取得	SQLDataSources(W)	なし
6		SQLDrivers(W)	
7		SQLGetInfo(W)	
8		SQLGetFunctions	
9		SQLGetTypeInfo(W)	
10	ドライバオプションの設定および取得	SQLSetConnectAttr(W)	なし
11		SQLGetConnectAttr(W)	
12		SQLSetEnvAttr	
13		SQLGetEnvAttr	
14		SQLSetStmtAttr(W)	
15		SQLGetStmtAttr(W)	
16	ディスクリプタ値の設定	SQLGetDescField(W)	なし
17		SQLGetDescRec(W)	
18		SQLSetDescField(W)	
19		SQLSetDescRec	
20		SQLCopyDesc	
21	SQL 要求の作成	SQLPrepare(W)	• tran_id：トランザクション ID • stmt_hdl：文ハンドル ID • sql_serial_num：SQL 文通番
22		SQLBindParameter	なし
23		SQLGetCursorName(W)	
24		SQLSetCursorName(W)	
25		SQLDescribeParam	
26		SQLNumParams	
27	SQL の実行	SQLExecute	• tran_id：トランザクション ID • stmt_hdl：文ハンドル ID • sql_serial_num：SQL 文通番
28		SQLExecDirect(W)	

項番	分類	ODBC 関数名	［SYSTEM］に出力される情報
29		SQLNativeSql(W)	なし
30		SQLParamData	
31		SQLPutData	
32	実行結果および実行結果情報の取得	SQLRowCount	なし
33		SQLNumResultCols	
34		SQLDescribeCol(W)	
35		SQLColAttribute(W)	
36		SQLBindCol	
37		SQLFetch	
38		SQLFetchScroll	
39		SQLGetData	
40		SQLSetPos	
41		SQLBulkOperations	
42		SQLMoreResults	
43		SQLGetDiagField(W)	
44		SQLGetDiagRec(W)	
45	データソースのシステム情報の取得	SQLColumnPrivileges(W)	なし
46		SQLColumns(W)	
47		SQLForeignKeys(W)	
48		SQLPrimaryKeys(W)	
49		SQLProcedureColumns(W)	
50		SQLProcedures(W)	
51		SQLSpecialColumns(W)	
52		SQLStatistics(W)	
53		SQLTablePrivileges(W)	
54		SQLTables(W)	
55	SQL 実行の終了	SQLFreeStmt	Option=SQL_CLOSE の場合に，次の情報を出力します。 • tran_id：トランザクション ID • stmt_hdl：文ハンドル ID • sql_serial_num：SQL 文通番
56		SQLCloseCursor	• tran_id：トランザクション ID • stmt_hdl：文ハンドル ID

項番	分類	ODBC 関数名	［SYSTEM］に出力される情報
			• sql_serial_num：SQL 文通番
57		SQLCancel	なし
58		SQLEndTran	tran_id：トランザクション ID
59	データソースとの切断	SQLDisconnect	なし
60		SQLFreeHandle	

注

トランザクションが決着済みのときなど，tran_id，stmt_htdl，およびsql_serial_num が取得できない場合は，各値に*を出力します。

17.5 HADB ODBC ドライバトレース情報に関する注意事項

- HADB ODBC ドライバトレース情報が出力されるのは，Windows 版の HADB クライアントの場合だけです。Linux 版の HADB クライアントの場合，HADB ODBC ドライバトレース情報は出力されません。
- HADB ODBC ドライバトレースが原因となるエラーまたは警告が発生した場合，HADB ODBC ドライバトレース情報が出力されなくなることがあります。この場合でも，ODBC 関数の実行に影響はありません。
- HADB ODBC ドライバトレース情報を出力している環境では，ODBC インタフェースを使用するすべての AP に影響があるため，それを考慮するようにしてください。
- HADB ODBC ドライバトレース情報が出力されるたびに HADB ODBC ドライバトレースファイルが単調増加していきます。そのため，出力先のディレクトリの容量不足によって，HADB ODBC ドライバトレース情報が出力されなくなることがあります。

18 APの作成

この章では，C 言語および C++言語で AP を設計，作成するときに考慮する必要がある基本事項について説明します。

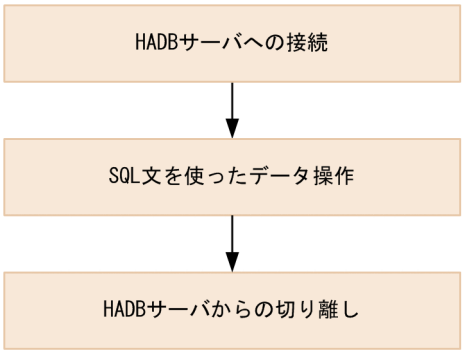
18.1　AP の設計

ここでは，AP を設計するときに考慮する必要がある基本事項について説明します。

18.1.1　AP の処理の流れ

データベースを操作する際の AP の処理の流れを次の図に示します。

図 18-1　データベースを操作する際の AP の処理の流れ



HADB サーバへの接続

AP からデータベースを操作する場合は，AP を HADB サーバに接続する必要があります。最初にコネクションを一意に識別するためのコネクションハンドルを AP に割り当てます。次にコネクションハンドルを利用してコネクションを確立します。これで，AP が HADB サーバに接続されます。

同じ AP から複数のコネクションを確立することもできます。ただし，同時に確立できるコネクションの数は決まっています。

なお，HADB サーバへの接続を行う際は，あらかじめ HADB サーバを開始しておく必要があります。

HADB サーバへの接続が完了すると，AP から SQL 文を使ってデータベースを操作できます。

HADB サーバからの切り離し

AP を終了する前に，AP を HADB サーバから切り離してください。最初にコネクションを終了させて，次にコネクションハンドルを解放します。これで，AP が HADB サーバから切り離されて，AP を終了できます。

18.1.2　トランザクション制御

ここでは，トランザクションの開始または終了契機と，トランザクション制御（コミット処理およびロールバック処理）について説明します。

(1)　コネクションとトランザクションの関係

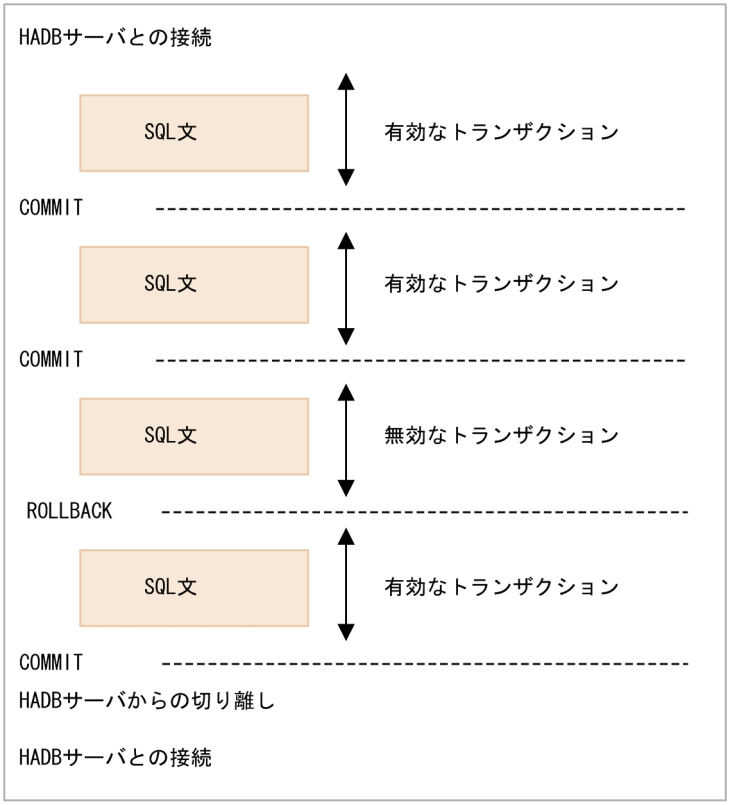
トランザクションを開始する前に，HADB サーバとのコネクションを確立してください。コネクションを確立しないと，トランザクションを開始できません。コネクションを確立するには，CLI 関数の a_rdb_SQLConnect()を使用します。

また，確立したコネクションを終了すると，トランザクションは COMMIT されます（AP 中でCOMMIT を明示的に発行しなくても，コネクションを終了すると，トランザクションが正常終了します)。コネクションを終了するには，CLI 関数のa_rdb_SQLDisconnect()を使用します。

(2)　トランザクションの開始と終了

AP が文ハンドルを確保したときがトランザクションの開始となります。COMMIT またはROLLBACK が実行されたときがトランザクションの終了となります。トランザクションの開始と終了の例を次の図に示します。

図 18-2　トランザクションの開始と終了の例



トランザクションのCOMMIT およびROLLBACK は，CLI 関数のa_rdb_SQLEndTran()で行います。

定義系 SQL を実行した場合は，自動的にCOMMIT 処理が行われます。

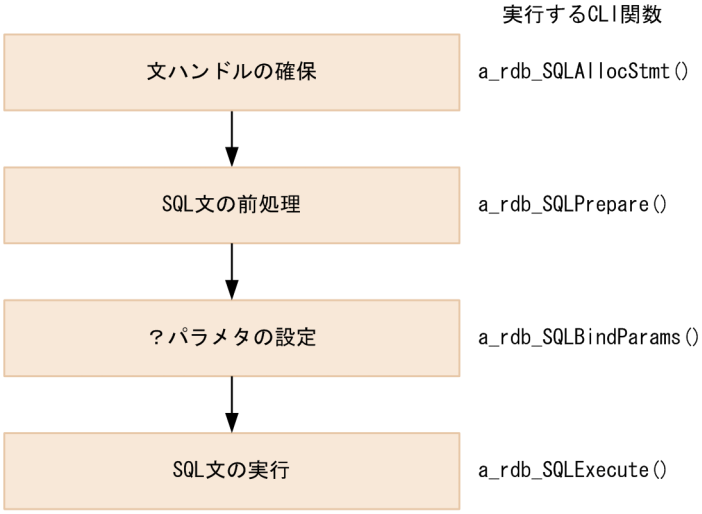
18.1.3　？パラメタを使用する際の処理の流れ

APの実行時に，入力値だけが異なる複数の操作系SQLの処理を行う場合，？パラメタを使用すると，SQL文の前処理を何度も行う必要がなくなり，APでSQL文を組み立てるときに比べて，処理効率を高めることができます。

？パラメタを使用する場合は，SQL文のAPから値を渡される個所に「？」を指定し，CLI関数のa_rdb_SQLPrepare()でSQL文を前処理します。その後，CLI関数のa_rdb_SQLBindParams()で？パラメタを設定することで，？パラメタを指定したSQL文を実行できます。

？パラメタを使用する際の処理の流れを次の図に示します。

図 18-3　？パラメタを使用する際の処理の流れ



？パラメタについては，マニュアル『HADB　SQLリファレンス』の『変数（？パラメタ）』を参照してください。

18.1.4　更新操作によるカーソルを使用した検索への影響

カーソルを使用した検索中に更新操作を行うと，タイミングによっては更新操作の結果が，検索の結果に反映されることがあります。更新操作の結果を検索の結果に反映させないようにするには，次のように運用してください。

- カーソルを閉じたあとに，行の追加または更新を行う
- 追加または更新する行が，検索結果と一致しないようにデータや探索条件などを工夫する

カーソルを使用した検索中の更新操作の例を次に示します。

```
char *selSql = ”SELECT * FROM T1 WHERE C1 BETWEEN 10 AND 20”;
char *updSql = ”UPDATE T1 SET C1=30 WHERE C1=20”;

/* SELECT文の前処理 */
```

```
rtnc = a_rdb_SQLPrepare(cnctContext, hStmt1,selSql);
     :

/* 行の取り出し */
rtnc = a_rdb_SQLFetch(cnctContext, hStmt1);                ...1

/* UPDATEの前処理 */
rtnc = a_rdb_SQLPrepare(cnctContext, hStmt2, updSql);

/* 行の更新 */
rtnc = a_rdb_SQLExecute(cnctContext, hStmt2);              ...2

/* 行の取り出し */
rtnc = a_rdb_SQLFetch(cnctContext, hStmt1);                ...3
     :
```

[説明]

2.の時点でC1 列が20 の行を更新した場合，1.の時点での 1 回目のFETCH 実行によって，すでに HADB サーバが AP とは非同期に検索処理を実行しています。そのため，タイミングによっては 3.の時点でC1 列が20 の行を検索できないことがあります。すでにC1 列が20 の行の検索を完了している場合は，検索できます。

18.1.5　SQL 文のエラー判定と対処方法

AP で実行した SQL 文が正常に実行されたかどうかを判定する必要があります。ここでは，SQL 文が正常に実行されたかどうかを判定する方法と，エラーを検出した場合の対処方法について説明します。

(1)　SQL 文のエラー判定方法

SQL 文が実行されると，CLI 関数の戻り値としてSQLCODE が返却されます。返却されたSQLCODE の値によって SQL 文が正常に実行されたかどうかを判定します。SQLCODE の値とその意味を次の表に示します。

表 18-1　SQLCODE の値とその意味

項番	SQLCODE の値	意味
1	100	検索する行がなくなったことを意味しています。特に，次に示すことを判定するときに有効です。 • FETCH で取り出す行がなくなったかどうか • INSERT，DELETE，またはUPDATE で更新対象の行があるかどうか
2	1	SQL 文の処理は終了しましたが，処理の延長で警告が発生したことを意味しています。発生する警告を次に示します。 • HADB サーバでサーバメッセージログファイルを格納しているディスクが満杯になった • クライアントメッセージログファイルを格納しているディスクが満杯になった • HADB サーバで SQL トレース情報を出力する際にメモリ不足を検知した • コネクション終了の延長でコミット処理に失敗し，ロールバック処理を行ってからコネクションを終了した

項番	SQLCODE の値	意味
3	負の値	SQL エラーが発生したことを意味しています。
4	上記以外	SQL 文が正常に実行されたことを意味しています。

CLI 関数の戻り値については，「19.8　CLI 関数の戻り値」を参照してください。

(2)　エラー検出時の対処方法

SQL 文のエラーを検出した場合，次に示す順序で対処します。なお，ここでは CLI 関数を使用した場合の例について説明します。

(a)　戻り値の出力

CLI 関数の戻り値を出力，または表示します。

(b)　エラーへの対処

SQL 文を使用したデータ操作でエラーが発生した場合，次に示す手順で対処してください。

手順

1. SQL 結果情報のメンバisInConnect の値を判定します。isInConnect の値によって対処方法が次のように異なります。
   - isInConnect の値がa_rdb_SQL_IS_IN_CONNECT の場合

     手順の 2 に進んでください。
   - isInConnect の値がa_rdb_SQL_IS_NOT_IN_CONNECT の場合

     致命的なエラーが発生したため，AP が HADB サーバから切り離されました。クライアントメッセージログファイルに出力されたメッセージを確認し，HADB サーバが異常終了している場合は，HADB 管理者に連絡してください。
2. SQL 結果情報のメンバEndTran の値を判定します。EndTran の値によって対処方法が次の表に示すように異なります。

項番	EndTran の値	対処方法
1	a_rdb_SQL_ROLLBACKED	内部的にロールバックが発生したため，HADB サーバからの切り離しを行って AP を終了してください。そのあと，戻り値を参考にエラーの対策を行ってください。
2	a_rdb_SQL_TRAN_NOT_ENDED	ROLLBACK を実行してトランザクションを取り消してから，HADB サーバからの切り離しを行い AP を終了してください。そのあと，戻り値を参考にエラーの対策を行ってください。
3	上記以外の場合	HADB サーバからの切り離しを行って AP を終了してください。そのあと，戻り値を参考にエラーの対策を行ってください。

18.2　CLI 関数の使い方

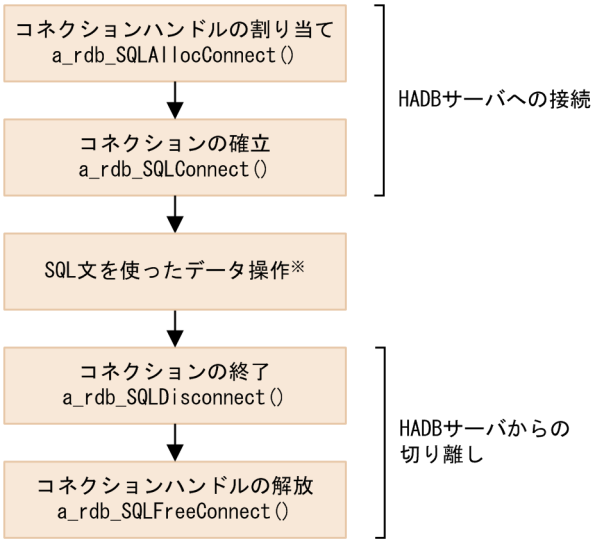
C 言語または C++言語で AP を作成する場合，CLI 関数を使用できます。ここでは，CLI 関数の基本的な使い方について説明します。

18.2.1　HADB サーバへの接続および切り離しをする場合

ここでは，CLI 関数を使用した HADB サーバへの接続，および HADB サーバからの切り離しについて説明します。

HADB サーバへの接続から切り離しまでの処理の流れを次の図に示します。

図 18-4　HADB サーバへの接続から切り離しまでの処理の流れ



注※

SQL 文を使ったデータ操作については，「18.2.2　データを参照する場合」以降で説明します。

(1)　HADB サーバへの接続

HADB サーバに接続する場合は，次に示す CLI 関数を使用します。

- a_rdb_SQLAllocConnect()（コネクションハンドルの割り当て）
- a_rdb_SQLConnect()（コネクションの確立）

(a)　コネクションハンドルの割り当て

コネクションを確立する前に，コネクションハンドルの割り当てが必要になります。コネクションハンドルを割り当てるにはa_rdb_SQLAllocConnect()を使用します。コネクションハンドルの割り当てが成功すると，戻り値としてa_rdb_RC_SQL_SUCCESS が返されます。a_rdb_SQLAllocConnect()の呼び出し例を次に示します。

a_rdb_SQLAllocConnect()の呼び出し例

```
signed short rtnc ;                    /* 戻り値 */
void  *hCnct ;                         /* コネクションハンドルアドレス */

/* コネクションハンドルの割り当て */
rtnc = a_rdb_SQLAllocConnect(&hCnct,
                        "/hadb/client.def",    /* クライアント定義ファイルパス */
                        NULL) ;
```

a_rdb_SQLAllocConnect()の第２引数にクライアント定義ファイルの絶対パスを指定することで，コネクションごとに異なるクライアント定義を使用できます。クライアント定義については，「4.4　クライアント定義の作成」を参照してください。

a_rdb_SQLAllocConnect()については，「19.2.1　a_rdb_SQLAllocConnect()（コネクションハンドルの割り当て）」を参照してください。

(b)　コネクションの確立

コネクションを確立するには，a_rdb_SQLConnect()を使用します。コネクションが確立されると，戻り値としてa_rdb_RC_SQL_SUCCESS が返されます。

a_rdb_SQLConnect()の呼び出し例を次に示します。

a_rdb_SQLConnect()の呼び出し例

```
a_rdb_SQLResultInfo_t  rsltInfo ;          /* SQL結果情報 */

/* コネクションの確立 */
rtnc = a_rdb_SQLConnect(hCnct,             /* コネクションハンドル */
                        "ADBUSER01",       /* 認可識別子 */
                        "password01",      /* パスワード */
                        &rsltInfo,
                        NULL) ;
```

SQL 結果情報のアドレスをa_rdb_SQLConnect()の第４引数に指定することで，CLI 関数の呼び出し単位で各種結果を返却する SQL 結果情報を取得できます。SQL 結果情報については，「19.7.6 a_rdb_SQLResultInfo_t 構造体（SQL 結果情報）」を参照してください。

a_rdb_SQLConnect()については，「19.2.2　a_rdb_SQLConnect()（コネクションの確立）」を参照してください。

(2)　HADB サーバからの切り離し

HADB サーバからの切り離しを行う場合は，次に示す CLI 関数を使用します。

- a_rdb_SQLDisconnect()（コネクションの終了）
- a_rdb_SQLFreeConnect()（コネクションハンドルの解放）

(a)　コネクションの終了

コネクションを終了するには，a_rdb_SQLDisconnect()を使用します。a_rdb_SQLDisconnect()の呼び出し例を次に示します。

a_rdb_SQLDisconnect()の呼び出し例

```
/* コネクションの終了 */
rtnc = a_rdb_SQLDisconnect(hCnct, NULL) ;
```

a_rdb_SQLDisconnect()については，「19.2.4　a_rdb_SQLDisconnect()（コネクションの終了）」を参照してください。

(b)　コネクションハンドルの解放

コネクションを終了したあとに，コネクションハンドルを解放してください。コネクションハンドルを解放するには，a_rdb_SQLFreeConnect()を使用します。a_rdb_SQLFreeConnect()の呼び出し例を次に示します。

a_rdb_SQLFreeConnect()の呼び出し例

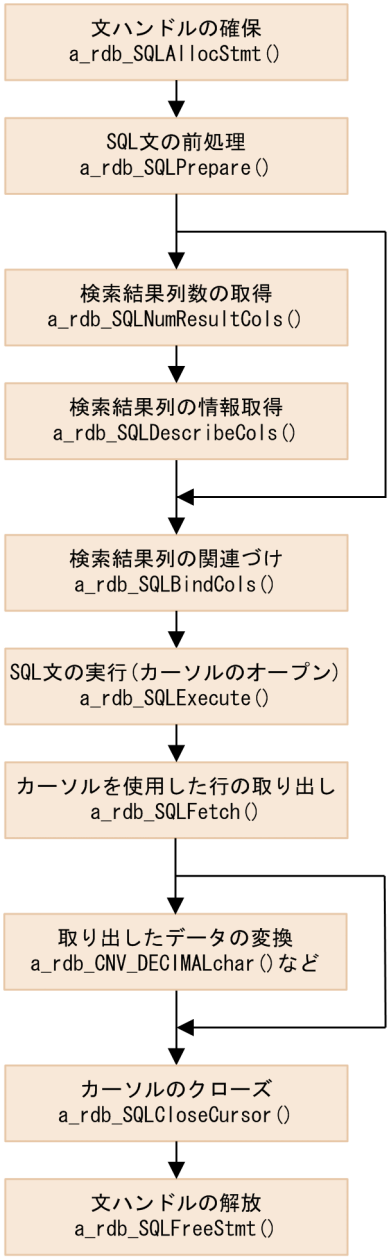
```
/* コネクションハンドルの解放 */
rtnc = a_rdb_SQLFreeConnect(hCnct, NULL) ;
```

a_rdb_SQLFreeConnect()については，「19.2.5　a_rdb_SQLFreeConnect()（コネクションハンドルの解放）」を参照してください。

18.2.2　データを参照する場合

データの参照方法の例を説明します。ここでは，カーソルを使用した行の取り出し方法について説明します。カーソルを使用した行の取り出し方法を次の図に示します。

図 18-5　カーソルを使用した行の取り出し方法



各処理の詳細を次に説明します。

(1)　文ハンドルの確保

SQL 文を実行する前に，a_rdb_SQLAllocStmt()を使用して文ハンドルを確保します。文ハンドルを確保できた場合，戻り値としてa_rdb_RC_SQL_SUCCESS が返されます。文ハンドルの確保例を次に示します。

文ハンドルの確保例

```
void  *hStmt ;                                  /* 文ハンドルアドレス */

/* 文ハンドルの確保 */
rtnc = a_rdb_SQLAllocStmt(hCnct,
```

```
                          &hStmt,
                          NULL) ;
```

a_rdb_SQLAllocStmt()については，「19.4.1　a_rdb_SQLAllocStmt()（文ハンドルの確保）」を参照してください。

(2)　SQL 文の前処理

次に，「(1)　文ハンドルの確保」で取得した文ハンドルに SQL 文を割り当てます。取得した文ハンドルに SQL 文を割り当てるには，a_rdb_SQLPrepare()を使用して SQL 文の前処理を実行します。SELECT 文の前処理の例を次に示します。

SELECT 文の前処理の例

```
/* SELECT文の前処理 */
rtnc = a_rdb_SQLPrepare(hCnct,
                        hStmt,
                        "SELECT C1,C2,C3 FROM T1",
                        NULL) ;
```

a_rdb_SQLPrepare()については，「19.4.14　a_rdb_SQLPrepare()（SQL 文の前処理）」を参照してください。

(3)　検索結果列数の取得

動的に SQL 文を実行するなど，AP の作成時に検索結果列数（検索結果として出力される列の数）が確定していない場合，a_rdb_SQLNumResultCols()を使用して検索結果列数を取得します。検索結果列数の取得例を次に示します。

検索結果列数の取得例

```
/* 検索結果列数の取得 */
rtnc = a_rdb_SQLNumResultCols(hCnct,
                              hStmt,
                              &colCount,     /* 列数 */
                              NULL) ;
```

a_rdb_SQLNumResultCols()については，「19.4.13　a_rdb_SQLNumResultCols()（検索結果列数の取得）」を参照してください。

(4)　検索結果列の情報取得

動的に SQL 文を実行するなど，AP の作成時に検索結果列の列名，データ型，またはデータ長などの列情報が確定していない場合，a_rdb_SQLDescribeCols()を使用して検索結果列の情報を取得します。a_rdb_SQLDescribeCols()で取得できる情報を次に示します。

- 検索結果列の列名
- 検索結果列のデータ型

- 検索結果列の最大要素数
- 検索結果列のデータ長

検索結果列の情報取得例を次に示します。

検索結果列の情報取得例

```
/* 検索結果列の情報の取得 */
rtnc = a_rdb_SQLDescribeCols(hCnct,
                             hStmt,
                             colCount,          /* 検索結果列数 */
                             &(colInf[0]),      /* 全検索結果列の情報返却領域 */
                             NULL) ;
```

a_rdb_SQLDescribeCols()については，「19.4.6　a_rdb_SQLDescribeCols()（検索結果列の情報取得）」を参照してください。

(5)　検索結果列の関連づけ

検索結果列と，検索結果列から取り出した値を格納する領域の関連づけ（検索結果列の関連づけ）を行います。検索結果列の関連づけを行うと，a_rdb_SQLFetch()を使用してカーソルを操作する際に，検索結果列の値が関連づけた領域に自動的に格納されます。

検索結果列の関連づけを行うには，a_rdb_SQLBindCols()を使用します。検索結果列の関連づけの例を次に示します。

検索結果列の関連づけの例

```
/* 検索結果列の関連づけ */
rtnc = a_rdb_SQLBindCols(hCnct,
                         hStmt,
                         colCount,          /* 検索結果列数 */
                         &(colInf[0]),      /* 全列割り当て格納領域 */
                         NULL) ;
```

a_rdb_SQLBindCols()については，「19.4.3　a_rdb_SQLBindCols()（検索結果列の関連づけ）」を参照してください。

(6)　SQL 文の実行（カーソルのオープン）

a_rdb_SQLExecute()を使用して，前処理した SQL 文を実行します。a_rdb_SQLExecute()の引数に，実行する SQL 文の文ハンドルを指定します。SQL 文が正常に実行されると，戻り値としてa_rdb_RC_SQL_SUCCESSが返され，カーソルがオープンします。SQL 文の実行例を次に示します。

SQL 文の実行例

```
/* SQL文の実行 */
rtnc = a_rdb_SQLExecute(hCnct,
```

```
                        hStmt,
                        NULL) ;
```

a_rdb_SQLExecute()については，「19.4.9　a_rdb_SQLExecute()（前処理したSQL文の実行)」を参照してください。

(7)　カーソルを使用した行の取り出し

SQL文の実行によってオープンされたカーソルを使用して行を取り出すには，a_rdb_SQLFetch()を使用します。行の取り出しが成功すると，戻り値としてa_rdb_RC_SQL_SUCCESSが返されます。カーソルを使用した行の取り出し例を次に示します。

カーソルを使用した行の取り出し例

```
/* カーソルを使用した行の取り出し */
rtnc = a_rdb_SQLFetch(hCnct,
                      hStmt,
                      NULL) ;
```

a_rdb_SQLFetch()については，「19.4.10　a_rdb_SQLFetch()（行の取り出し)」を参照してください。

(8)　取り出したデータの変換

取り出したデータが，SQLのDECIMAL型，NUMERIC型，BINARY型，VARBINARY型，DATE型，TIME型，またはTIMESTAMP型のデータの場合，CLI関数を使用してC言語またはC++言語の文字列データに変換できます。DECIMAL型のデータを変換する例を次に示します。

データの変換例

```
#define PRECISION 6
#define SCALE 3

unsigned char data_decimal[4];
char data_char[PRECISION+4];

/* DECIMAL型データを変換 */
rtnc = a_rdb_CNV_DECIMALchar(data_decimal,  /* 出力データの先頭アドレス */
                            PRECISION,      /* 出力データの精度 */
                            SCALE,          /* 出力データの位取り */
                            data_char,      /* 変換後データの格納領域アドレス */
                            (PRECISION+4),  /*変換後データの格納領域長 */
                            NULL);
```

- DECIMAL型またはNUMERIC型のデータを変換する場合は，a_rdb_CNV_DECIMALchar()を使用します。a_rdb_CNV_DECIMALchar()については，「19.5.9　a_rdb_CNV_DECIMALchar()（DECIMAL型またはNUMERIC型データの変換)」を参照してください。
- BINARY型のデータを変換する場合は，a_rdb_CNV_BINARYchar()を使用します。a_rdb_CNV_BINARYchar()については，「19.5.7　a_rdb_CNV_BINARYchar()（BINARY型データの変換)」を参照してください。

- VARBINARY 型のデータを変換する場合は，a_rdb_CNV_VARBINARYchar()を使用します。a_rdb_CNV_VARBINARYchar()については，「19.5.12　a_rdb_CNV_VARBINARYchar()（VARBINARY 型データの変換）」を参照してください。
- DATE 型のデータを変換する場合は，a_rdb_CNV_DATEchar()を使用します。a_rdb_CNV_DATEchar()については，「19.5.8　a_rdb_CNV_DATEchar()（DATE 型データの変換）」を参照してください。
- TIME 型のデータを変換する場合は，a_rdb_CNV_TIMEchar()を使用します。a_rdb_CNV_TIMEchar()については，「19.5.10　a_rdb_CNV_TIMEchar()（TIME 型データの変換）」を参照してください。
- TIMESTAMP 型のデータを変換する場合は，a_rdb_CNV_TIMESTAMPchar()を使用します。a_rdb_CNV_TIMESTAMPchar()については，「19.5.11　a_rdb_CNV_TIMESTAMPchar()（TIMESTAMP 型データの変換）」を参照してください。

(9)　カーソルのクローズ

a_rdb_SQLCloseCursor()を使用して，カーソルをクローズします。カーソルのクローズ例を次に示します。

カーソルのクローズ例

```
/* カーソルのクローズ */
rtnc = a_rdb_SQLCloseCursor(hCnct,
                            hStmt,
                            NULL) ;
```

a_rdb_SQLCloseCursor()については，「19.4.5　a_rdb_SQLCloseCursor()（カーソルのクローズ）」を参照してください。

(10)　文ハンドルの解放

a_rdb_SQLFreeStmt()をして，確保した文ハンドルを解放します。文ハンドルの解放例を次に示します。

文ハンドルの解放例

```
/* 文ハンドルの解放 */
rtnc = a_rdb_SQLFreeStmt(hCnct,
                         hStmt,
                         NULL) ;
```

重要

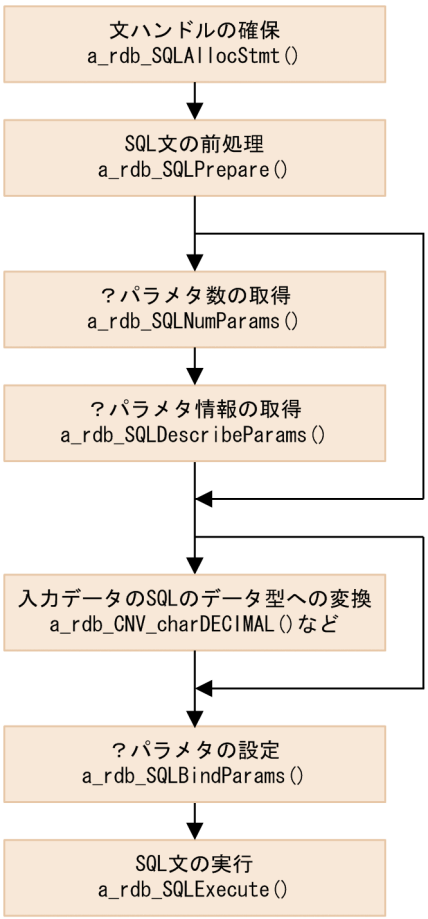
COMMIT またはROLLBACK が実行されると，文ハンドルが解放されます。この場合は，a_rdb_SQLFreeStmt()を実行しないでください。

a_rdb_SQLFreeStmt()については，「19.4.11　a_rdb_SQLFreeStmt()（文ハンドルの解放）」を参照してください。

18.2.3　？パラメタを使用する場合

？パラメタを指定した SQL 文を実行する場合は，SQL 文の割り当て，SQL 文の前処理，？パラメタの設定，および SQL 文の実行という手続きを行います。？パラメタを指定したときの SQL 文実行の流れを次の図に示します。

図 18-6　？パラメタを指定したときの SQL 文実行の流れ



文ハンドルの確保，SQL 文の前処理，および SQL 文の実行方法については，「18.2.2　データを参照する場合」で説明した方法と同じです。ここでは，？パラメタ数の取得，？パラメタ情報の取得，入力データの SQL のデータ型への変換，および？パラメタの設定方法について説明します。

(1)　？パラメタ数の取得

動的に SQL 文を実行するなど，AP の作成時に？パラメタの数が確定していない場合は，a_rdb_SQLNumParams()を使用して？パラメタ数を取得します。？パラメタ数の取得例を次に示します。

？パラメタ数の取得例

```
/* ？パラメタ数の取得 */
rtnc = a_rdb_SQLNumParams(hCnct,
                          hStmt,
                          &paramCount,     /* ？パラメタの数*/
                          NULL) ;
```

a_rdb_SQLNumParams()については，「19.4.12　a_rdb_SQLNumParams()（？パラメタ数の取得)」を参照してください。

(2)　？パラメタ情報の取得

動的に SQL 文を実行するなど，AP の作成時に？パラメタのデータ型やデータ長などの情報が確定していない場合は，a_rdb_SQLDescribeParams()を使用して？パラメタの情報を取得します。
a_rdb_SQLDescribeParams()で取得できる情報は次のとおりです。

- ？パラメタのデータ型
- ？パラメタの最大要素数
- ？パラメタのデータ長

？パラメタ情報の取得例を次に示します。

？パラメタ情報の取得例

```
/* ？パラメタ情報の取得 */
rtnc = a_rdb_SQLDescribeParams(hCnct,
                               hStmt,
                               paramCount,        /* ？パラメタの数 */
                               &(paramInfo[0]),   /* 全？パラメタ情報返却領域 */
                               NULL) ;
```

a_rdb_SQLDescribeParams()については，「19.4.7　a_rdb_SQLDescribeParams()（？パラメタの情報取得)」を参照してください。

(3)　入力データの SQL のデータ型への変換

？パラメタの入力データが，SQL のDECIMAL 型，NUMERIC 型，BINARY 型，VARBINARY 型，DATE 型，TIME 型，またはTIMESTAMP 型のデータの場合，CLI 関数を使用して C 言語または C++言語の文字列データから該当するデータ型に変換できます。C 言語または C++言語の文字列データをDECIMAL 型のデータに変換する例を次に示します。

データの変換例

```
#define PRECISION 6
#define SCALE 3

char data_char[]="-123.567";
unsigned char data_decimal[4];

/* DECIMAL型データに変換 */
rtnc = a_rdb_CNV_charDECIMAL(data_char,           /* 変換するデータの先頭アドレス */
                            (unsigned short)strlen(data_char), /* 変換するデータ長 */
                            PRECISION,           /* 入力データの精度 */
                            SCALE,               /* 入力データの位取り */
                            data_decimal,        /* 入力データの格納領域アドレス */
```

```
                                4,                      /* 入力データの格納領域長 */
                                NULL);
```

- DECIMAL 型またはNUMERIC 型のデータに変換する場合は，a_rdb_CNV_charDECIMAL()を使用します。a_rdb_CNV_charDECIMAL()については，「19.5.3　a_rdb_CNV_charDECIMAL()（DECIMAL 型または NUMERIC 型データへの変換）」を参照してください。
- BINARY 型のデータに変換する場合は，a_rdb_CNV_charBINARY()を使用します。a_rdb_CNV_charBINARY()については，「19.5.1　a_rdb_CNV_charBINARY()（BINARY 型データへの変換）」を参照してください。
- VARBINARY 型のデータに変換する場合は，a_rdb_CNV_charVARBINARY()を使用します。a_rdb_CNV_charVARBINARY()については，「19.5.6　a_rdb_CNV_charVARBINARY()（VARBINARY 型データへの変換）」を参照してください。
- DATE 型のデータに変換する場合は，a_rdb_CNV_charDATE()を使用します。a_rdb_CNV_charDATE()については，「19.5.2　a_rdb_CNV_charDATE()（DATE 型データへの変換）」を参照してください。
- TIME 型のデータに変換する場合は，a_rdb_CNV_charTIME()を使用します。a_rdb_CNV_charTIME()については，「19.5.4　a_rdb_CNV_charTIME()（TIME 型データへの変換）」を参照してください。
- TIMESTAMP 型のデータに変換する場合は，a_rdb_CNV_charTIMESTAMP()を使用します。a_rdb_CNV_charTIMESTAMP()については，「19.5.5　a_rdb_CNV_charTIMESTAMP()（TIMESTAMP 型データへの変換）」を参照してください。

(4)　？パラメタの設定

a_rdb_SQLBindParams()を使用して？パラメタを設定します。設定が正常に完了すると，戻り値として a_rdb_RC_SQL_SUCCESS が返されます。？パラメタの設定例を次に示します。

？パラメタの設定例

```
/* ？パラメタの設定 */
rtnc = a_rdb_SQLBindParams(hCnct,
                           hStmt,
                           paramCount,        /* ？パラメタの数 */
                           &(paramInfo[0]),   /* 全？パラメタ設定用領域 */
                           NULL) ;
```

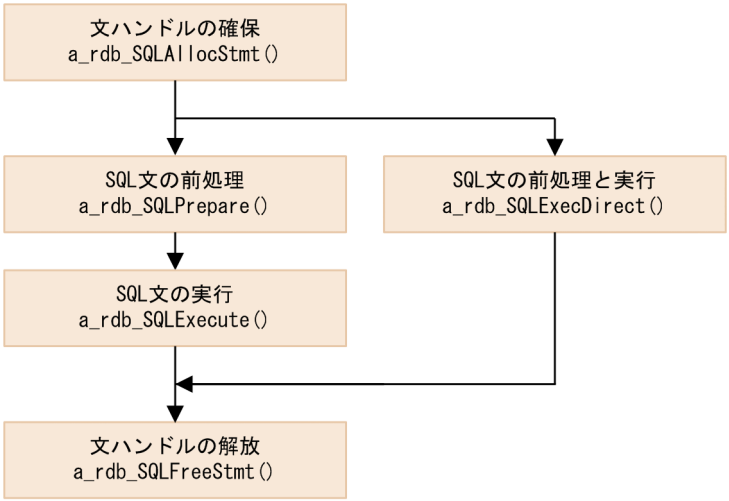
a_rdb_SQLBindParams()については，「19.4.4　a_rdb_SQLBindParams()（？パラメタの関連づけ）」を参照してください。

18.2.4　データを追加，更新，または削除する場合

更新系 SQL によるデータの追加，更新，または削除は，SELECT と同様に文ハンドルを確保してから SQL 文の前処理を行い，そのあとにa_rdb_SQLExecute()を使用して SQL 文を実行します。

データの更新および削除の処理の流れを次の図に示します。

図 18-7　データの更新および削除の処理の流れ



注

a_rdb_SQLPrepare()で事前に SQL 文を前処理していない場合でも，a_rdb_SQLExecDirect()で SQL 文の前処理と実行を同時に行うことができます。

なお，？パラメタを使用する場合は，a_rdb_SQLBindParams()またはa_rdb_SQLBindArrayParams()を使用して？パラメタの関連づけをする必要があります。a_rdb_SQLBindParams()については「19.4.4 a_rdb_SQLBindParams()（？パラメタの関連づけ)」を，a_rdb_SQLBindArrayParams()については「19.4.2　a_rdb_SQLBindArrayParams()（？パラメタの一括関連づけ)」を参照してください。

データの更新および削除の実行例を次に示します。

データの更新および削除の実行例

```
/* 文ハンドルの確保 */
rtnc = a_rdb_SQLAllocStmt(hCnct, &hStmt, NULL) ;
/* INSERT文の準備 */
rtnc = a_rdb_SQLPrepare(hCnct,
                        hStmt,
                        "INSERT INTO TABLE1 VALUES ('A','B','C')",
                        NULL) ;
/* SQL文の実行 */
rtnc = a_rdb_SQLExecute(hCnct, hStmt, NULL) ;
/* DELETEの実行 */
rtnc = a_rdb_SQLExecDirect(hCnct,
                           hStmt,
                           "DELETE FROM TABLE1",
                           NULL) ;
```

a_rdb_SQLExecDirect()については，「19.4.8　a_rdb_SQLExecDirect()（SQL 文の前処理および実行)」を参照してください。

18.2.5　実行中の SQL をキャンセルする場合

実行中の SQL をa_rdb_SQLCancel()でキャンセルできます。キャンセルできる SQL（CLI 関数）を次に示します。

- a_rdb_SQLCloseCursor()
- a_rdb_SQLExecDirect()
- a_rdb_SQLExecute()
- a_rdb_SQLFetch()
- a_rdb_SQLPrepare()

a_rdb_SQLCancel()は，SQL を処理しているスレッドとは別のスレッドで実行します。SQL のキャンセルの実行例を次に示します。

SQL の実行スレッド

```
rtnc = a_rdb_SQLAllocStmt(hCnct, &hStmt, NULL) ;
/* DELETE文の実行 */
rtnc = a_rdb_SQLExecDirect(hCnct,
                           hStmt,
                           ”DELETE FROM TABLE1”,
                           NULL) ;
/* キャンセルされたかどうかをチェック */
if ( rtnc == -955 )
{
  /* キャンセル後の処理 */
}
else ;
```

SQL のキャンセルの実行例（別スレッドで実行）

```
/* SQLキャンセル機能の実行 */
rtnc = a_rdb_SQLCancel(hCnct, NULL) ;
                  /* キャンセル対象のSQLを実行しているコネクションと */
                  /* 同様のコネクションハンドルを指定 */
```

SQL のキャンセルが成功した場合，SQL 処理を中断してトランザクションをロールバックしたあとに SQLCODE が返却されます。

a_rdb_SQLCancel()については，「19.3.1　a_rdb_SQLCancel()（SQL のキャンセル）」を参照してください。

なお，キャンセル処理はa_rdb_SQLCancel()とは非同期に実行されるため，a_rdb_SQLCancel()の正常終了はキャンセル処理の完了（成功）を意味するものではありません。

18.2.6　CLI 関数を使用する場合の留意事項

ここでは，CLI 関数の引数に指定するアドレスの有効期間および境界位置調整について説明します。

(1)　領域の有効期間

CLI 関数の引数に，アドレスが指す領域が解放されている無効なアドレスが指定されている場合，動作は保証されません。

CLI 関数にアドレスを指定する引数のうち，次に示す引数は処理の開始から終了までの間，有効であることを保証する必要があります。

- HADB サーバへの接続から切り離しまでの間，有効であることを保証する必要がある引数

  次に示す引数が該当します。

  - 各 CLI 関数のConnectionHandle
  - a_rdb_SQLConnect()のResultInfo（SQL 結果情報を取得する場合）

- カーソルのオープンからクローズまでの間，有効であることを保証する必要がある引数

  次に示す引数が該当します。

  - a_rdb_SQLBindCols()のColumnInfo 構造体のメンバTargetValue およびStrLen_or_Ind
  - a_rdb_SQLBindParams()およびa_rdb_SQLBindArrayParams()のParameterInfo 構造体のメンバParameterValue およびInd

    a_rdb_SQLBindParams()およびa_rdb_SQLBindArrayParams()の引数については，カーソルを使用しない SQL を前処理する場合でも，a_rdb_SQLExecute()を実行するまで領域が有効であることを保証してください。

なお，トランザクションの開始から終了までの間，有効であることを保証する必要がある引数はありません。

(2)　境界位置調整

検索結果のデータや，？パラメタの入力値などを格納する領域の先頭アドレスは，SQL のデータ型ごとに次の表に従って境界位置調整を行ってください。

表 18-2　境界位置調整

項番	SQL データ型	境界位置調整
1	VARBINARY 型	2 バイト境界
2	SMALLINT 型	4 バイト境界
3	VARCHAR 型	
4	INTEGER 型	8 バイト境界
5	DOUBLE PRECISION 型	
6	FLOAT 型	

18.3　AP のコンパイルとリンケージ

AP の言語に従ったコンパイラで，ソースプログラムをコンパイルおよびリンケージしてください。コンパイルおよびリンケージの方法については，各言語に対応するコンパイラのマニュアルを参照してください。また，OS の『Readme』ファイルも参照してください。

AP をコンパイルおよびリンケージする際は，次の表に従って HADB クライアントが提供しているライブラリから開発環境に合わせたものを指定してください。

表 18-3　AP をコンパイルおよびリンケージする際に指定するライブラリ

項番	開発環境（OS）	指定するライブラリ
1	Windows	• adbclt.lib（64 ビット版の HADB クライアントを使用している場合） • adbclt32.lib（32 ビット版の HADB クライアントを使用している場合）
2	Linux	libadbclt.so

19 CLI 関数

この章では，HADB が提供している CLI 関数の機能と文法について説明します。

19.1　CLI 関数の一覧と共通規則

ここでは，CLI 関数の一覧と共通規則について説明します。

19.1.1　CLI 関数の一覧

HADB では，次の表に示す CLI 関数を提供しています。

表 19-1　CLI 関数の一覧

項番	分類		CLI 関数	機能
1	HADB サーバへの接続および切り離し時に使用する CLI 関数		a_rdb_SQLAllocConnect()	コネクションハンドルを割り当てます。
2			a_rdb_SQLConnect()	HADB サーバとのコネクションを確立します。
3			a_rdb_SQLSetConnectAttr()	コネクション属性を設定します。
4			a_rdb_SQLDisconnect()	コネクションを終了します。
5			a_rdb_SQLFreeConnect()	コネクションハンドルを解放します。
6	トランザクションの制御時に使用する CLI 関数		a_rdb_SQLCancel()	処理中の SQL をキャンセルします。
7			a_rdb_SQLEndTran()	トランザクションを終了します。
8	SQL の実行時に使用する CLI 関数	文ハンドルの確保・解放	a_rdb_SQLAllocStmt()	文ハンドルを確保します。
9			a_rdb_SQLFreeStmt()	文ハンドルを解放します。
10		SQL 文の前処理，実行，カーソル操作，行の取り出し	a_rdb_SQLPrepare()	SQL 文の前処理を実行します。
11			a_rdb_SQLExecute()	前処理した SQL 文を実行します。
12			a_rdb_SQLExecDirect()	SQL 文を前処理して実行します。
13			a_rdb_SQLFetch()	取り出す行を示すカーソルの位置を次の行に進め，その行の列の値を取り出し相手リストで指定した取り出し先に読み込みます。
14			a_rdb_SQLCloseCursor()	カーソルをクローズします。
15		検索結果列	a_rdb_SQLNumResultCols()	検索結果列数を取得します。
16			a_rdb_SQLDescribeCols()	検索結果列の情報を取得します。

項番	分類		CLI 関数	機能
17			a_rdb_SQLBindCols()	検索結果列と，その列から取り出した値を格納する領域を結合します（関連づけます）。
18		？パラメタ	a_rdb_SQLNumParams()	SQL 文中の？パラメタの数を取得します。
19			a_rdb_SQLDescribeParams()	SQL 文中の？パラメタの情報を取得します。
20			a_rdb_SQLBindParams()	SQL 文中の？パラメタの値を設定する領域の結合（関連づけ）を行います。
21			a_rdb_SQLBindArrayParams()	SQL 文中の？パラメタの値を設定する領域の結合（関連づけ）を行います。SQL 文中の複数組の？パラメタの値をまとめて結合します。
22	データ型の変換時に使用する CLI 関数	C 言語または C++言語の文字列データを SQL のデータ型に変換	a_rdb_CNV_charBINARY()	C 言語または C++言語の文字列データ（2 進数または 16 進数）を SQL のBINARY 型のデータに変換します。
23			a_rdb_CNV_charDATE()	C 言語または C++言語の文字列データを SQL のDATE 型のデータに変換します。
24			a_rdb_CNV_charDECIMAL()	C 言語または C++言語の文字列データを，SQL のDECIMAL 型またはNUMERIC 型のデータに変換します。
25			a_rdb_CNV_charTIME()	C 言語または C++言語の文字列データを SQL のTIME 型のデータに変換します。
26			a_rdb_CNV_charTIMESTAMP()	C 言語または C++言語の文字列データを SQL のTIMESTAMP 型のデータに変換します。
27			a_rdb_CNV_charVARBINARY()	C 言語または C++言語の文字列データ（2 進数または 16 進数）を SQL のVARBINARY 型のデータに変換します。
28		SQL のデータ型を C 言語または C++言語の文字列データに変換	a_rdb_CNV_BINARYchar()	SQL のBINARY 型のデータを C 言語または C++言語の文字列データに変換します。
29			a_rdb_CNV_DATEchar()	SQL のDATE 型のデータを C 言語または C++言語の文字列データに変換します。

項番	分類		CLI 関数	機能
30			a_rdb_CNV_DECIMALchar()	SQL のDECIMAL 型または NUMERIC 型のデータを，C 言語または C++言語の文字列データに変換します。
31			a_rdb_CNV_TIMEchar()	SQL のTIME 型のデータを C 言語または C++言語の文字列データに変換します。
32			a_rdb_CNV_TIMESTAMPchar()	SQL のTIMESTAMP 型のデータを C 言語または C++言語の文字列データに変換します。
33			a_rdb_CNV_VARBINARYchar()	SQL のVARBINARY 型のデータを C 言語または C++言語の文字列データに変換します。

これらの CLI 関数は，C 言語または C++言語で記述したソースプログラムに，HADB クライアントが提供するヘッダファイルおよびクライアントライブラリを取り込むことで利用できます。HADB クライアントが提供するヘッダファイルおよびクライアントライブラリを次の表に示します。

表 19-2　HADB クライアントが提供するヘッダファイルおよびクライアントライブラリ

項番	使用する CLI 関数	使用するヘッダファイルおよびクライアントライブラリ	
		ヘッダファイル	クライアントライブラリ
1	HADB サーバへの接続および切り離し時に使用する CLI 関数	• $ADBDIR/include/adbcli.h • $ADBDIR/include/adbtypes.h	• 64 ビット版の HADB クライアントを使用している場合 %ADBCLTDIR%¥client¥lib¥adbclt.lib • 32 ビット版の HADB クライアントを使用している場合 %ADBCLTDIR%¥client¥lib¥adbclt32.lib • Linux の場合 $ADBDIR/client/lib/libadbclt.so
2	トランザクションの制御時に使用する CLI 関数		
3	SQL の実行時に使用する CLI 関数		
4	データ型の変換時に使用する CLI 関数	$ADBDIR/include/adbcnv.h	

注　使用する CLI 関数ごとにヘッダファイルが異なります。

19.1.2　共通規則

CLI 関数の共通規則を次に示します。

(1) コネクションハンドル

HADB サーバとのコネクションを一意に識別するために，コネクションハンドルを使用します。このコネクションハンドルを，HADB サーバへの接続から切り離しまでの間，有効にし続ける必要があります。コネクションハンドルを有効にし続けないと，動作は保証されません。コネクションハンドルを有効にし続けるには，a_rdb_SQLAllocConnect()で割り当てたコネクションハンドルをa_rdb_SQLFreeConnect()で解放するまでの間，各 CLI 関数の引数に同じコネクションハンドルを指定する必要があります。

(2) SQL 結果情報

CLI 関数の呼び出し単位で各種 SQL 結果情報を取得できます。SQL 結果情報を取得する場合，SQL 結果情報の領域は HADB サーバへの接続から切り離しまでの間，解放しないでください。

(3) 文ハンドル

割り当てた SQL 文を一意に識別するために，文ハンドルを使用します。COMMIT またはROLLBACK が行われると，文ハンドルは解放されます。

なお，同一コネクションで割り当てた複数の文ハンドルで SQL 文を同時に実行した場合に，最大 SQL 処理リアルスレッド数分の処理リアルスレッドが使用できないときは，SQL 文の実行要求がエラーとなります（待ち状態にはなりません）。

最大 SQL 処理リアルスレッド数は，次のオペランドで指定しています。

- サーバ定義のadb_sql_exe_max_rthd_num
- クライアント定義のadb_sql_exe_max_rthd_num

(4) 暗黙カーソル

表の検索結果は，通常複数行にわたります。AP で複数行の検索結果を 1 行ずつ取り出すために最新の取り出し位置を保持するものをカーソルといいます。カーソルは，データの検索で使用します。

SELECT 文の前処理時にカーソルが割り当てられて，SELECT 文の実行時にカーソルがオープンされます。COMMIT またはROLLBACK が実行されると，オープンしているカーソルはすべてクローズされます。

19.2　HADB サーバへの接続および切り離し時に使用する CLI 関数

ここでは，HADB サーバへの接続時，および HADB サーバからの切り離し時に使用する CLI 関数について説明します。

19.2.1　a_rdb_SQLAllocConnect()（コネクションハンドルの割り当て）

(1)　機能

HADB サーバとのコネクションを一意に識別するためのコネクションハンドルを割り当てます。

(2)　形式

```
signed short a_rdb_SQLAllocConnect
(
  void                          **ConnectionHandle,    /* Out */
  char                           *ClientDefPath,       /* In  */
  void                           *Option               /* In  */
)
```

(3)　引数の説明

ConnectionHandle：

コネクションハンドルを設定するアドレスを指定します。

ClientDefPath：

クライアント定義ファイルのパスを C 言語または C++言語の文字列表現で指定します。

NULL を指定した場合，%ADBCLTDIR%¥conf¥client.def※が指定されたと仮定します。

注※　Linux 版の HADB クライアントの場合は，$ADBCLTDIR/conf/client.def になります。

Option：

NULL を指定します。

(4)　戻り値

1. a_rdb_SQLAllocConnect()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. コネクションハンドルの割り当ては成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5) 留意事項

次に示す場合，a_rdb_SQLAllocConnect()は実行できません。

- 不正なクライアント定義ファイルのパスを指定した場合
- クライアント定義の指定内容に誤りがある場合
- コネクションハンドルの割り当てに失敗した場合

19.2.2 a_rdb_SQLConnect()（コネクションの確立）

(1) 機能

HADB サーバとのコネクションを確立します。

(2) 形式

```
signed short a_rdb_SQLConnect
(
  void                          *ConnectionHandle,    /* In */
  char                          *UserID,              /* In */
  char                          *Password,            /* In */
  a_rdb_SQLResultInfo_t         *ResultInfo,          /* In */
  void                          *Option               /* In */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

UserID：

HADB サーバに接続する認可識別子を C 言語または C++言語の文字列表現で指定します。

Password：

UserID で指定した認可識別子のパスワードを C 言語または C++言語の文字列表現で指定します。

パスワードの規則については，マニュアル『HADB　SQL リファレンス』の『CREATE USER 文の指定形式および規則』を参照してください。

ResultInfo：

CLI 関数の呼び出し単位で各種結果を返却する SQL 結果情報のアドレスを指定します。
a_rdb_SQLDisconnect()が発行されるまで，指定した領域に SQL 結果情報が返却されます。

アドレスがNULL の場合，SQL 結果情報は返却されません。

a_rdb_SQLResultInfo_t 構造体については，「19.7.6　a_rdb_SQLResultInfo_t 構造体（SQL 結果情報）」を参照してください。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLConnect()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. コネクションハンドルの確立には成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5) 留意事項

次に示す場合，a_rdb_SQLConnect()は実行できません。

- CONNECT 権限を持たない HADB ユーザが実行した場合
- コネクションがすでに確立されている場合
- 不正な認可識別子を指定した場合
- HADB サーバの文字コードと異なる文字コードがADBCLTLANG に指定されている場合
- HADB サーバへの接続数が，最大同時接続数に達している場合

19.2.3 a_rdb_SQLSetConnectAttr()（コネクション属性の設定）

(1) 機能

指定したコネクションハンドルのコネクション属性を設定します。

(2) 形式

```
signed short a_rdb_SQLSetConnectAttr
(
  void                          *ConnectionHandle,     /* In */
  signed short                   Attribute,            /* In */
  void                          *Value,                /* In */
  void                          *Option                /* In */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

Attribute：

次に示すどれかのコネクション属性を指定します。

- トランザクション隔離性水準を設定する場合：

  a_rdb_SQL_ATTR_TXN_ISOLATION

- ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を設定する場合：

  a_rdb_SQL_ATTR_ORDER_MODE

- トランザクションアクセスモードを設定する場合：

  a_rdb_SQL_ATTR_ACCESS_MODE

Value：

属性情報を格納したアドレスを指定します。

・トランザクション隔離性水準を設定する場合

次に示すどれかの値が設定されているunsigned short の領域のアドレスを指定します。

- READ COMMITTED の場合：

  a_rdb_SQL_TXN_READ_COMMITTED

- REPEATABLE READ の場合：

  a_rdb_SQL_TXN_REPEATABLE_READ

- 指定値を未設定（この CLI 関数を使用して一度もトランザクション隔離性水準を設定していない状態）に変更する場合：

  a_rdb_SQL_TXN_UNSPECIFIED

・ORDER BY 句を指定したSELECT 文の文字データの並び替え順序を設定する場合

次に示すどれかの値が設定されているunsigned short の領域のアドレスを指定します。

- 文字データをバイトコード順で並び替える場合：

  a_rdb_SQL_ORDER_MODE_BYTE

- 文字データをソートコード順（ISO/IEC14651:2011 準拠）で並び替える場合：

  a_rdb_SQL_ORDER_MODE_ISO

- 指定値を未設定（この CLI 関数を使用して一度も文字データの並び替え順序を設定していない状態）に変更する場合：

  a_rdb_SQL_ORDER_MODE_UNSPECIFIED

・トランザクションアクセスモードを設定する場合

次に示すどれかの値が設定されているunsigned short の領域のアドレスを指定します。

- 読み書き可能モードの場合：

a_rdb_SQL_ACCESS_MODE_READ_WRITE

- 読み取り専用モードの場合：

  a_rdb_SQL_ACCESS_MODE_READ_ONLY

- 指定値を未設定（この CLI 関数を使用して一度もトランザクションアクセスモードを設定していない状態）に変更する場合：

  a_rdb_SQL_ACCESS_MODE_UNSPECIFIED

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLSetConnectAttr()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. コネクション属性の設定には成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. 次に示す場合，a_rdb_SQLSetConnectAttr()は実行できません。
   - コネクションが確立されていない場合
   - トランザクションが決着していない場合
   - 無効なコネクション属性を指定した場合
   - 無効なトランザクション隔離性水準を指定した場合
   - 無効な文字データの並び替え順序を指定した場合
   - 無効なトランザクションアクセスモードを指定した場合
2. トランザクション隔離性水準は，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 と 2 では 1 の指定が優先されます。
   1. a_rdb_SQLSetConnectAttr()で指定したトランザクション隔離性水準
   2. クライアント定義のadb_clt_trn_iso_lv オペランドで指定したトランザクション隔離性水準
   3. サーバ定義のadb_sys_trn_iso_lv オペランドで指定したトランザクション隔離性水準
3. 文字データの並び替え順序は，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 と 2 では 1 の指定が優先されます。
   1. a_rdb_SQLSetConnectAttr()で指定した文字データの並び替え順序
   2. クライアント定義のadb_clt_sql_order_mode オペランドで指定した文字データの並び替え順序
   3. サーバ定義のadb_sql_order_mode オペランドで指定した文字データの並び替え順序

4. トランザクションアクセスモードは，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 と 2 では 1 の指定が優先されます。

   1. a_rdb_SQLSetConnectAttr()で指定したトランザクションアクセスモード
   2. クライアント定義のadb_clt_trn_access_mode オペランドで指定したトランザクションアクセスモード

19.2.4 a_rdb_SQLDisconnect()（コネクションの終了）

(1) 機能

トランザクションを正常終了させ，同期点を設定し 1 コミットメント単位を生成します。そのあと，確立した HADB サーバとのコネクションを終了します。

(2) 形式

```
signed short a_rdb_SQLDisconnect
(
  void                          *ConnectionHandle,   /* In */
  void                          *Option              /* In */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLDisconnect()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. コネクションの終了には成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. a_rdb_SQLDisconnect()を実行した際にトランザクションが終了していない場合，HADB サーバが自動的にCOMMIT を実行します。そのあとに HADB サーバとのコネクションを終了します。

2. 内部的に実行されたCOMMIT が失敗した場合，HADB サーバがROLLBACK を実行してトランザクションを取り消します。そのあとに HADB サーバとのコネクションを終了します。
3. a_rdb_SQLDisconnect()を実行しないで AP を終了した場合，HADB サーバがROLLBACK を実行します。そのあとに HADB サーバとのコネクションを終了します。
4. コネクションが確立されていない場合，a_rdb_SQLDisconnect()は実行できません。

19.2.5 a_rdb_SQLFreeConnect()（コネクションハンドルの解放）

(1) 機能

終了したコネクションのコネクションハンドルを解放します。

(2) 形式

```
signed short a_rdb_SQLFreeConnect
(
  void                          *ConnectionHandle,    /* In */
  void                          *Option               /* In */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLFreeConnect()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. コネクションハンドルの解放には成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

a_rdb_SQLDisconnect()を実行しないでa_rdb_SQLFreeConnect()を実行した場合，a_rdb_SQLFreeConnect()の延長でa_rdb_SQLDisconnect()相当の処理が実行され，そのあとにコネクションハンドルが解放されます。

19.3　トランザクションの制御時に使用する CLI 関数

ここでは，トランザクションの制御時に使用する CLI 関数について説明します。

19.3.1　a_rdb_SQLCancel()（SQL のキャンセル）

(1)　機能

実行中の SQL をキャンセルします。a_rdb_SQLCancel()でキャンセルできる SQL（CLI 関数）を次に示します。

- a_rdb_SQLCloseCursor()
- a_rdb_SQLExecDirect()
- a_rdb_SQLExecute()
- a_rdb_SQLFetch()
- a_rdb_SQLPrepare()

なお，上記以外の CLI 関数の実行中にa_rdb_SQLCancel()を実行しても，a_rdb_SQLCancel()は正常に終了します。

(2)　形式

```
signed short a_rdb_SQLCancel
(
  void                          *ConnectionHandle,   /* In */
  void                          *Option              /* In */
)
```

(3)　引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

Option：

NULL を指定します。

(4)　戻り値

a_rdb_SQLCancel()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。

(5) 留意事項

1. a_rdb_SQLCancel()は，SQL を処理しているスレッドとは別のスレッドで実行してください。
2. a_rdb_SQLCancel()を実行しても，SQL 結果情報は設定されません。
3. a_rdb_SQLCancel()がエラーとなった場合でも，クライアントメッセージログファイルにメッセージは出力されません。
4. SQL のキャンセルが成功した場合，キャンセル対象の SQL はロールバックされ，SQLCODE が返却されます。
5. キャンセル処理はa_rdb_SQLCancel()とは非同期に実行されるため，a_rdb_SQLCancel()の正常終了はキャンセル処理の完了（成功）を意味するものではありません。
6. コネクションが確立されていない場合，a_rdb_SQLCancel()は実行できません。

19.3.2 a_rdb_SQLEndTran()（トランザクションの終了）

(1) 機能

トランザクションを終了します。トランザクションが終了すると，その時点で割り当てられているすべての文ハンドルが解放されます。

(2) 形式

```
signed short a_rdb_SQLEndTran
(
  void                          *ConnectionHandle,   /* In */
  unsigned short                 CompletionType,     /* In */
  void                          *Option              /* In */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

CompletionType：

次に示すどちらかの値を指定します。

- a_rdb_SQL_COMMIT：COMMIT（トランザクションの正常終了）する場合に指定します。
- a_rdb_SQL_ROLLBACK：ROLLBACK（トランザクションの取り消し）する場合に指定します。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLEndTran()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. トランザクションの終了には成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. 行の取り出し処理で発生したエラーを，コミットの延長で実行されたカーソルクローズで検知した場合，トランザクションは正常終了しないで，自動的にROLLBACK が実行されます。これ以外の理由でCOMMIT に失敗した場合，HADB サーバは異常終了します。
2. ROLLBACK に失敗した場合，HADB サーバは異常終了します。
3. 次に示す場合，a_rdb_SQLEndTran()は実行できません。
   - コネクションが確立されていない場合
   - CompletionType に不正な値を指定した場合
   - 行の取り出し処理で発生したエラーを，コミットの延長で実行されたカーソルクローズで検知した場合
4. a_rdb_SQLEndTran()を実行してトランザクションをコミットまたはロールバックした場合，次に示す状態になります。
   - オープンしているカーソルはすべてクローズされます。
   - 前処理された SQL 文は無効になります。
   - すべての文ハンドルが解放されます。

19.4　SQL の実行時に使用する CLI 関数

ここでは，SQL の実行時に使用する CLI 関数について説明します。

19.4.1　a_rdb_SQLAllocStmt()（文ハンドルの確保）

(1)　機能

文ハンドル（SQL 文のハンドル）を確保します。

(2)　形式

```
signed short a_rdb_SQLAllocStmt
(
  void                          *ConnectionHandle,    /* In  */
  void                         **StatementHandle,    /* Out */
  void                          *Option              /* In  */
)
```

(3)　引数の説明

ConnectionHandle：

　コネクションハンドルを指定します。

StatementHandle：

　文ハンドルを設定するアドレスを指定します。

Option：

　NULL を指定します。

(4)　戻り値

1. a_rdb_SQLAllocStmt()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. 文ハンドルの確保には成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5)　留意事項

1. コネクションが確立されていない場合，a_rdb_SQLAllocStmt()は実行できません。

2. トランザクションの開始に失敗した場合，HADB サーバは異常終了します。

19.4.2　a_rdb_SQLBindArrayParams()（？パラメタの一括関連づけ）

(1)　機能

SQL 文中の？パラメタの値を設定する領域の結合（関連づけ）を行います。

次に示す SQL 文中の，複数組の？パラメタの値をまとめて結合します。

- DELETE 文
- INSERT 文
- UPDATE 文

？パラメタの値の一括転送を行う場合に，この CLI 関数を使用します。？パラメタの値の一括転送については，「5.15　？パラメタの値の一括転送」を参照してください。

(2)　形式

```
signed short a_rdb_SQLBindArrayParams
(
  void                          *ConnectionHandle,  /* In  */
  void                          *StatementHandle,   /* In  */
  int                            ArrayCount,        /* In  */
  unsigned short                 ParameterCount,    /* In  */
  a_rdb_SQLParameterInfo_t      *ParameterInfo[],   /* In  */
  unsigned long long             RowCount[],        /* Out */
  void                          *Option             /* In  */
)
```

(3)　引数の説明

ConnectionHandle：

　コネクションハンドルを指定します。

StatementHandle：

　文ハンドルを指定します。

ArrayCount：

　ParameterInfo およびRowCount の配列数を指定します。

ParameterCount：

　値を与える？パラメタの数を指定します。

　a_rdb_SQLNumParams()で取得した？パラメタの数を指定してください。

ParameterInfo[]：

値の格納領域のアドレスを設定するパラメタ情報領域のアドレス配列を指定します。配列 1 つごとに a_rdb_SQLParameterInfo_t 構造体をParameterCount に指定した数だけ連続させた領域を用意し，その領域の先頭アドレスを各配列値として指定してください。

a_rdb_SQLParameterInfo_t 構造体については，「19.7.5　a_rdb_SQLParameterInfo_t 構造体（パラメタ情報）」を参照してください。

RowCount[]：

SQL 文の処理結果行数を格納する配列の先頭アドレスを指定します。NULL を指定した場合は，処理結果行数は取得されません。

a_rdb_SQLExecute()による SQL 文の実行後，各配列には各組の？パラメタの値を入力して実行した処理結果行数が格納されます。

なお，SQL 文の実行が途中でエラーとなった場合，成功したところまでの処理結果行数が格納されます。ただし，SQL 文のエラーによってロールバックが実行された場合，すべての配列に 0 が設定されます。

Option：

NULL を指定します。

(4)　戻り値

1. a_rdb_SQLBindArrayParams()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. ？パラメタの関連づけには成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5)　留意事項

1. a_rdb_SQLParameterInfo_t 構造体のデータ格納領域の先頭アドレスに対して境界位置調整を行ってください。境界位置調整については，「18.2.6　CLI 関数を使用する場合の留意事項」の「(2)　境界位置調整」を参照してください。
2. a_rdb_SQLBindArrayParams()で？パラメタを関連づけしてa_rdb_SQLExecute()を実行した場合，SQL 文がエラーにならないかぎり処理が続行されます。そのため，処理の途中で処理結果行数が 0 となるような？パラメタの値の組が設定されている場合でも処理が続行されます。
3. a_rdb_SQLResultInfo_t 構造体のRowCount には，RowCount[]配列に格納された処理結果行数の合計が格納されます。
4. 次に示す場合，a_rdb_SQLBindArrayParams()は実行できません。
   - コネクションが確立されていない場合
   - 無効な文ハンドルを指定した場合

- 配列数に 0 以下の値を指定した場合
- ？パラメタ数に，a_rdb_SQLNumParams()で取得した？パラメタの数以外の値を指定した場合
- DELETE 文，INSERT 文，またはUPDATE 文以外の SQL 文に対してa_rdb_SQLBindArrayParams()を実行しようとした場合

19.4.3 a_rdb_SQLBindCols()（検索結果列の関連づけ）

(1) 機能

検索結果列と，検索結果列から取り出した値を格納する領域の結合（検索結果列の関連づけ）を一括で行います。検索結果列の関連づけを行うと，a_rdb_SQLFetch()でのカーソル操作時に，関連づけられた検索結果列の値が格納されます。

なお，検索結果が列以外の場合もa_rdb_SQLBindCols()を実行できます。

(2) 形式

```
signed short a_rdb_SQLBindCols
(
  void                          *ConnectionHandle,    /* In  */
  void                          *StatementHandle,     /* In  */
  unsigned short                 ColumnCount,         /* In  */
  a_rdb_SQLColumnInfo_t         *ColumnInfo,          /* In  */
  void                          *Option               /* In  */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

ColumnCount：

値を取り出す列の数を指定します。

a_rdb_SQLNumResultCols()で取得した検索結果列数を指定してください。

ColumnInfo：

値格納用領域のアドレスを設定する列情報領域の先頭アドレスを指定します。

a_rdb_SQLColumnInfo_t 構造体をColumnCount に指定した数だけ連続させた領域を用意してください。

a_rdb_SQLColumnInfo_t 構造体については，「19.7.1 a_rdb_SQLColumnInfo_t 構造体（列情報）」を参照してください。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLBindCols()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. 検索結果列の関連づけには成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. 取得データがCHAR 型の場合，a_rdb_SQLColumnInfo_t 構造体のデータ格納領域にはデータの末尾にナル文字を 1 文字付加したものを格納します。ここでのナル文字とは，0x00 のことです。
2. a_rdb_SQLColumnInfo_t 構造体のデータ格納領域の先頭アドレスに対して境界位置調整を行ってください。境界位置調整については，「18.2.6 CLI 関数を使用する場合の留意事項」の「(2) 境界位置調整」を参照してください。
3. a_rdb_SQLColumnInfo_t 構造体のデータ格納領域長またはインジケータ値のどちらかを格納する領域には，取得データに応じて，次の表に従って値が設定されます。表中の BL は指定したデータ格納領域長を，DL は取得データを格納するのに必要な領域の大きさ（バイト数）を意味しています。

表 19-3 データ格納領域長またはインジケータ値のどちらかを格納する領域の値

項番	格納データ	データ構造	データ型	BL と DL の関係	データ格納領域長またはインジケータ値
1	ナル値	任意	任意	任意	a_rdb_SQL_NULL_DATA
2	非ナル値	任意	任意	BL ＝ DL	a_rdb_SQL_NOT_NULL_DATA
3				上記以外	－（SQL エラー）※

注※

この場合 SQL エラーとなり，インジケータに値が設定される処理が行われないで，値が不定になります。

4. 次に示す場合，a_rdb_SQLBindCols()は実行できません。
   - コネクションが確立されていない場合
   - 無効な文ハンドルを指定した場合
   - a_rdb_SQLNumResultCols()で取得した検索結果列数以外の値を検索結果列数に指定した場合
   - a_rdb_SQLColumnInfo_t 構造体のデータ格納領域長に，取得データ格納に必要な領域長以外の値を指定した場合
   - 配列型の検索結果列に対して列結合を実行した場合

19.4.4　a_rdb_SQLBindParams()（？パラメタの関連づけ）

(1)　機能

SQL 文中の？パラメタの値を設定する領域の結合（関連づけ）を行います。

(2)　形式

```
signed short a_rdb_SQLBindParams
(
  void                          *ConnectionHandle,    /* In  */
  void                          *StatementHandle,     /* In  */
  unsigned short                 ParameterCount,      /* In  */
  a_rdb_SQLParameterInfo_t      *ParameterInfo,       /* In  */
  void                          *Option               /* In  */
)
```

(3)　引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

ParameterCount：

値を与える？パラメタの数を指定します。

a_rdb_SQLNumParams()で取得した？パラメタの数を指定してください。

ParameterInfo：

値格納用領域のアドレスを設定するパラメタ情報領域の先頭アドレスを指定します。

a_rdb_SQLParameterInfo_t 構造体をParameterCount に指定した数だけ連続させた領域を用意してください。

a_rdb_SQLParameterInfo_t 構造体については，「19.7.5　a_rdb_SQLParameterInfo_t 構造体（パラメタ情報）」を参照してください。

Option：

NULL を指定します。

(4)　戻り値

1. a_rdb_SQLBindParams()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. ？パラメタの関連づけには成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。

3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5)　留意事項

1. a_rdb_SQLParameterInfo_t 構造体のデータ格納領域の先頭アドレスに対して境界位置調整を行ってください。境界位置調整については，「18.2.6　CLI 関数を使用する場合の留意事項」の「(2)　境界位置調整」を参照してください。
2. 次に示す場合，a_rdb_SQLBindParams()は実行できません。
   - コネクションが確立されていない場合
   - 無効な文ハンドルを指定した場合
   - ？パラメタ数に，a_rdb_SQLNumParams()で取得した？パラメタの数以外の値を指定した場合

19.4.5　a_rdb_SQLCloseCursor()（カーソルのクローズ）

(1)　機能

カーソルをクローズして，行の取り出しを終了します。

行を取り出している最中の処理リアルスレッドがある場合，その処理リアルスレッドを終了します。

(2)　形式

```
signed short a_rdb_SQLCloseCursor
(
  void                          *ConnectionHandle,    /* In */
  void                          *StatementHandle,     /* In */
  void                          *Option               /* In */
)
```

(3)　引数の説明

ConnectionHandle：

　コネクションハンドルを指定します。

StatementHandle：

　文ハンドルを指定します。

Option：

　NULL を指定します。

(4) 戻り値

1. a_rdb_SQLCloseCursor()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. カーソルのクローズには成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. 次に示す場合，a_rdb_SQLCloseCursor()は実行できません。
   - コネクションが確立されていない場合
   - 無効な文ハンドルを指定した場合
   - 行の取り出し処理で発生したエラーを，カーソルクローズで検知した場合
2. トランザクションが終了した場合，その時点でオープンしているカーソルはすべてクローズされます。また，暗黙的にデータベースへの更新が無効にされた場合にも，カーソルはすべてクローズされます。なお，暗黙的にデータベースへの更新が無効にされた場合，サーバメッセージログファイルおよびクライアントメッセージログファイルに，KFAA51001-E またはKFAA51002-E メッセージが出力されます。

19.4.6 a_rdb_SQLDescribeCols()（検索結果列の情報取得）

(1) 機能

検索結果列の情報を取得します。次に示す情報を取得します。

- 検索結果列の列名
- 検索結果列のデータ型
- 検索結果列の最大要素数
- 検索結果列のデータ長

(2) 形式

```
signed short a_rdb_SQLDescribeCols
(
  void                          *ConnectionHandle,    /* In  */
  void                          *StatementHandle,     /* In  */
  unsigned short                 ColumnCount,         /* In  */
  a_rdb_SQLColumnInfo_t         *ColumnInfo,          /* Out */
  void                          *Option               /* In  */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

ColumnCount：

情報を取得する検索結果列数を指定します。

a_rdb_SQLNumResultCols()で取得した検索結果列数を指定してください。

ColumnInfo：

検索結果列の情報を取得する領域のアドレスを設定する列情報領域の先頭アドレスを指定します。

a_rdb_SQLColumnInfo_t 構造体をColumnCount に指定した数だけ連続させた領域を用意してください。

a_rdb_SQLColumnInfo_t 構造体については，「19.7.1　a_rdb_SQLColumnInfo_t 構造体（列情報）」を参照してください。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLDescribeCols()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. 検索結果列の情報取得には成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. a_rdb_SQLNameInfo_t 構造体の名称格納領域には，検索結果列名を取得します。検索結果列名については，マニュアル『HADB　SQL リファレンス』の『SELECT 文の指定形式および規則』の『規則』を参照してください。
2. a_rdb_SQLNameInfo_t 構造体の名称格納領域長は，ナル文字 1 文字のバイト数以上の値を指定してください。ここでのナル文字とは，0x00 のことです。
3. a_rdb_SQLNameInfo_t 構造体の名称格納領域長が，検索結果列名の長さ（バイト数）にナル文字 1 文字のバイト数を加算した値以上の場合，名称格納領域には，検索結果列名の末尾にナル文字 1 文字を付加した文字列が取得されます。
4. a_rdb_SQLNameInfo_t 構造体の名称格納領域長が，検索結果列名の長さ（バイト数）にナル文字 1 文字のバイト数を加算した値より小さい場合，検索結果列名の後半部分（名称格納領域に納まらない部分）

が切り捨てられます。名称格納領域には，検索結果列名を先頭から格納していき，最後にナル文字 1 文字を付加した文字列が格納されます。

5. a_rdb_SQLNameInfo_t 構造体の列名長格納領域には，検索結果列名の長さ（バイト数）が取得されます。

6. a_rdb_SQLDataType_t 構造体のデータ型コード格納領域には，検索結果列のデータ型を表すコード（データ型コード）が取得されます。各データ型に対応するデータ型コードについては，マニュアル『HADB SQL リファレンス』の『データ型の種類』を参照してください。

7. a_rdb_SQLDataType_t 構造体の最大要素数格納領域には，検索結果列の最大要素数が取得されます。検索結果列の最大要素数は，検索結果列が配列型でない場合は常に 1 となります。

8. a_rdb_SQLDataType_t 構造体の列長格納領域と列長属性格納領域には，検索結果列のデータ長に関する情報が取得されます。各領域に取得される値は，検索結果列のデータ型に応じて，次の表に示す値になります。

表 19-4　列長と列長属性

項番	データ型	列長	列長属性
1	INTEGER	8	0
2	SMALLINT	4	0
3	DECIMAL(m,n)※1	m	n
4	NUMERIC(m,n)※1		
5	DOUBLE PRECISION	8	0
6	FLOAT		
7	CHAR(n)	n	0
8	VARCHAR(n)	n	0
9	DATE	4	0
10	TIME(p)※2	$3+\uparrow p \div 2 \uparrow$	p
11	TIMESTAMP(p)※2	$7+\uparrow p \div 2 \uparrow$	p
12	BINARY(n)	n	0
13	VARBINARY(n)	n	0
14	ROW	*行長*	0

（凡例）

m，n，p：正の整数

注※1

データ型がDECIMAL 型またはNUMERIC 型の場合，列長として精度を取得します。

また，列長属性として位取りを取得します。

注※2

データ型がTIME 型またはTIMESTAMP 型の場合，列長としてデータ長を取得します。また，列長属性として小数秒の桁数を取得します。

9. 次に示す場合，a_rdb_SQLDescribeCols()は実行できません。

- コネクションが確立されていない場合
- 無効な文ハンドルを指定した場合
- a_rdb_SQLNumResultCols()で取得した検索結果列数以外の値を検索結果列数に指定した場合

19.4.7 a_rdb_SQLDescribeParams()（？パラメタの情報取得）

(1) 機能

SQL 文中の？パラメタの情報を取得します。次に示す情報を取得します。

- ？パラメタのデータ型
- ？パラメタの最大要素数
- ？パラメタのデータ長

(2) 形式

```
signed short a_rdb_SQLDescribeParams
(
  void                          *ConnectionHandle,   /* In  */
  void                          *StatementHandle,    /* In  */
  unsigned short                 ParameterCount,     /* In  */
  a_rdb_SQLParameterInfo_t      *ParameterInfo,      /* Out */
  void                          *Option              /* In  */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

ParameterCount：

情報を取得する？パラメタの数を指定します。

a_rdb_SQLNumParams()で取得した？パラメタの数を指定してください。

ParameterInfo：

各種パラメタ情報取得用領域のアドレスを設定するパラメタ情報領域の先頭アドレスを指定します。

a_rdb_SQLParameterInfo_t 構造体をParameterCount に指定した数だけ連続させた領域を用意してください。

a_rdb_SQLParameterInfo_t 構造体については，「19.7.5　a_rdb_SQLParameterInfo_t 構造体（パラメタ情報）」を参照してください。

Option：

NULL を指定します。

(4)　戻り値

1. a_rdb_SQLDescribeParams()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. ？パラメタの情報取得には成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5)　留意事項

1. a_rdb_SQLDataType_t 構造体のデータ型コード格納領域には，HADB によって仮定された？パラメタのデータ型を表すコード（データ型コード）が取得されます。各データ型に対応するデータ型コードについては，マニュアル『HADB　SQL リファレンス』の『データ型の種類』を参照してください。
2. a_rdb_SQLDataType_t 構造体の最大要素数格納領域には，？パラメタの最大要素数が取得されます。？パラメタの最大要素数は常に 1 となります。
3. a_rdb_SQLDataType_t 構造体のパラメタ長格納領域とパラメタ属性格納領域には，？パラメタのデータ長に関する情報が取得されます。各領域に取得する値は，HADB によって仮定された？パラメタのデータ型に応じて，次の表に示す値になります。

表 19-5　パラメタ長とパラメタ長属性

項番	データ型	パラメタ長	パラメタ長属性
1	INTEGER	8	0
2	SMALLINT	4	0
3	DECIMAL(*m*,*n*)※1	*m*	*n*
4	NUMERIC(*m*,*n*)※1		
5	DOUBLE PRECISION	8	0
6	FLOAT		
7	CHAR(*n*)	*n*	0

項番	データ型	パラメタ長	パラメタ長属性
8	VARCHAR(*n*)	*n*	0
9	DATE	4	0
10	TIME(*p*)※2	3+↑*p*÷2↑	*p*
11	TIMESTAMP(*p*)※2	7+↑*p*÷2↑	*p*
12	BINARY(*n*)	*n*	0
13	VARBINARY(*n*)	*n*	0
14	ROW	*行長*	0

（凡例）

m，*n*，*p*：正の整数

注※1

データ型がDECIMAL 型またはNUMERIC 型の場合，パラメタ長として精度を取得します。

また，パラメタ長属性として位取りを取得します。

注※2

データ型がTIME 型またはTIMESTAMP 型の場合，パラメタ長としてデータ長を取得します。また，パラメタ長属性として小数秒の桁数を取得します。

4. 次に示す場合，a_rdb_SQLDescribeParams()は実行できません。

- コネクションが確立されていない場合
- 無効な文ハンドルを指定した場合
- ？パラメタ数に，a_rdb_SQLNumParams()で取得した？パラメタの数以外の値を指定した場合

19.4.8 a_rdb_SQLExecDirect()（SQL 文の前処理および実行）

(1) 機能

SQL 文を前処理して実行します。実行可能な SQL 文を次に示します。

- DELETE 文※
- INSERT 文※
- PURGE CHUNK 文※
- TRUNCATE TABLE 文
- UPDATE 文※
- 定義系 SQL

注※

？パラメタが指定されている場合は，a_rdb_SQLExecDirect()を実行できません。

(2) 形式

```
signed short a_rdb_SQLExecDirect
(
  void                          *ConnectionHandle,   /* In */
  void                          *StatementHandle,    /* In */
  char                          *StatementText,      /* In */
  void                          *Option              /* In */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

StatementText：

前処理および実行するSQL文のテキストをC言語またはC++言語の文字列表現で指定します。

Option：

NULLを指定します。

(4) 戻り値

1. a_rdb_SQLExecDirect()が正常に終了した場合，a_rdb_RC_SQL_SUCCESSまたはa_rdb_RC_SQL_NO_DATAが返却されます。
2. SQL文を前処理して実行することには成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNINGが返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI関数の戻り値」を参照してください。

(5) 留意事項

次に示す場合，a_rdb_SQLExecDirect()は実行できません。

- コネクションが確立されていない場合
- 無効な文ハンドルを指定した場合

19.4.9　a_rdb_SQLExecute()（前処理したSQL文の実行）

(1)　機能

前処理したSQL文を実行します。実行可能なSQL文を次に示します。

- DELETE文
- INSERT文
- PURGE CHUNK文
- TRUNCATE TABLE文
- SELECT文
- UPDATE文
- 定義系SQL

SELECT文に対してa_rdb_SQLExecute()を実行した場合，カーソルがオープンされます。

(2)　形式

```
signed short a_rdb_SQLExecute
(
  void                          *ConnectionHandle,   /* In */
  void                          *StatementHandle,    /* In */
  void                          *Option              /* In */
)
```

(3)　引数の説明

ConnectionHandle：

　コネクションハンドルを指定します。

StatementHandle：

　文ハンドルを指定します。

Option：

　NULLを指定します。

(4)　戻り値

1. a_rdb_SQLExecute()が正常に終了した場合，a_rdb_RC_SQL_SUCCESSまたはa_rdb_RC_SQL_NO_DATAが返却されます。
2. SQL文の実行には成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNINGが返却されます。

3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5)　留意事項

1. 次に示す場合，a_rdb_SQLExecute()は実行できません。
   - コネクションが確立されていない場合
   - 無効な文ハンドルを指定した場合
2. 指定する文ハンドルには，前処理が完了している SQL 文が準備されている必要があります。
3. 一度オープンしたカーソルを再度オープンする場合は，いったんカーソルをクローズしてから，再度オープンしてください。
4. a_rdb_SQLFetch()による行の取り出しを実行する場合は，カーソルをオープンしてから行の取り出しを実行してください。
5. COMMIT またはROLLBACK を実行した場合（暗黙的ロールバックを含む），その時点でオープンしているカーソルはすべてクローズされます。

19.4.10　a_rdb_SQLFetch()（行の取り出し）

(1)　機能

カーソルの位置を次の行に進めます。列結合している場合は，さらにカーソルの示す 1 行の列の値を，取り出し相手リストで指定した取り出し先に読み込みます。

(2)　形式

```
signed short a_rdb_SQLFetch
(
  void                          *ConnectionHandle,    /* In */
  void                          *StatementHandle,     /* In */
  void                          *Option               /* In */
)
```

(3)　引数の説明

ConnectionHandle：

　コネクションハンドルを指定します。

StatementHandle：

　文ハンドルを指定します。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLFetch()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS またはa_rdb_RC_SQL_NO_DATA が返却されます。
2. 行の取り出しには成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. 次に示す場合，a_rdb_SQLFetch()は実行できません。
   - コネクションが確立されていない場合
   - 無効な文ハンドルを指定した場合
2. 使用するカーソルは，a_rdb_SQLExecute()でオープンしておいてください。

19.4.11 a_rdb_SQLFreeStmt()（文ハンドルの解放）

(1) 機能

a_rdb_SQLAllocStmt()で確保した文ハンドルを解放します。

文ハンドルが解放されると，オープンしているカーソルもクローズされます。

(2) 形式

```
signed short a_rdb_SQLFreeStmt
(
  void                          *ConnectionHandle,   /* In */
  void                          *StatementHandle,    /* In */
  void                          *Option              /* In */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLFreeStmt()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. 文ハンドルの解放には成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

次に示す場合，a_rdb_SQLFreeStmt()は実行できません。

- コネクションが確立されていない場合
- 無効な文ハンドルを指定した場合
- 行の取り出し処理で発生したエラーを，文ハンドルの解放の延長で実行されたカーソルクローズで検知した場合

19.4.12 a_rdb_SQLNumParams()（？パラメタ数の取得）

(1) 機能

SQL 文中の？パラメタの数を取得します。

(2) 形式

```
signed short a_rdb_SQLNumParams
(
  void                          *ConnectionHandle,    /* In  */
  void                          *StatementHandle,     /* In  */
  unsigned short                *ParameterCount,      /* Out */
  void                          *Option               /* In  */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

ParameterCount：

？パラメタの数を取得するアドレスを指定します。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLNumParams()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. ？パラメタ数の取得には成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

次に示す場合，a_rdb_SQLNumParams()は実行できません。

- コネクションが確立されていない場合
- 無効な文ハンドルを指定した場合

19.4.13 a_rdb_SQLNumResultCols()（検索結果列数の取得）

(1) 機能

検索結果列数を取得します。

(2) 形式

```
signed short a_rdb_SQLNumResultCols
(
  void                          *ConnectionHandle,    /* In  */
  void                          *StatementHandle,     /* In  */
  unsigned short                *ColumnCount,         /* Out */
```

```
  void                          *Option             /* In  */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

ColumnCount：

検索結果列数を取得するアドレスを指定します。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLNumResultCols()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. 検索結果列数の取得には成功したが，クライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. 次に示す場合，a_rdb_SQLNumResultCols()は実行できません。
   - コネクションが確立されていない場合
   - 無効な文ハンドルを指定した場合
2. 文ハンドルに割り当てられた SQL 文がSELECT 文でない場合は，検索結果列数を格納する領域に0 が格納されます。

19.4.14　a_rdb_SQLPrepare()（SQL 文の前処理）

(1) 機能

SQL 文の前処理を実行します。前処理できる SQL 文を次に示します。

- DELETE 文
- INSERT 文

- PURGE CHUNK 文
- TRUNCATE TABLE 文
- SELECT 文
- UPDATE 文
- 定義系 SQL

a_rdb_SQLPrepare()を実行すると，指定した SQL 文が実行可能な状態に前処理され，文ハンドルにその SQL 文が割り当てられます。

(2) 形式

```
signed short a_rdb_SQLPrepare
(
  void                          *ConnectionHandle,   /* In */
  void                          *StatementHandle,    /* In */
  char                          *StatementText,      /* In */
  void                          *Option              /* In */
)
```

(3) 引数の説明

ConnectionHandle：

コネクションハンドルを指定します。

StatementHandle：

文ハンドルを指定します。

StatementText：

前処理を実行する SQL 文のテキストを C 言語または C++言語の文字列表現で指定します。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_SQLPrepare()が正常に終了した場合，a_rdb_RC_SQL_SUCCESS が返却されます。
2. SQL 文の前処理には成功したが，サーバメッセージログファイルまたはクライアントメッセージログファイルを格納しているディスクが満杯になった場合，a_rdb_RC_SQL_WARNING が返却されます。
3. クライアントメッセージログファイルにメッセージが出力できない場合にエラーが発生したときは，エラー要因コードが返却されます。エラー要因コードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

次に示す場合，a_rdb_SQLPrepare()は実行できません。

- コネクションが確立されていない場合
- 無効な文ハンドルを指定した場合

19.5　データ型の変換時に使用する CLI 関数

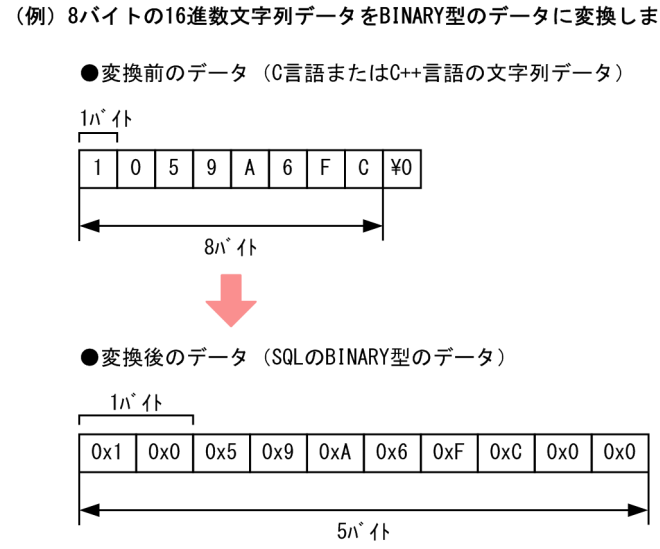
ここでは，データ型の変換時に使用する CLI 関数について説明します。

19.5.1　a_rdb_CNV_charBINARY()（BINARY 型データへの変換）

(1)　機能

C 言語または C++言語の文字列データ（2 進数または 16 進数）を SQL のBINARY 型のデータに変換します。文字列データをBINARY 型のデータに変換する場合の例を次の図に示します。

図 19-1　文字列データを BINARY 型のデータに変換する場合の例



重要

変換前の文字列データは，次に示す値である必要があります。

- 変換前の文字列データの形式が 2 進数の場合

  '0'および'1'

- 変換前の文字列データの形式が 16 進数の場合

  '0'～'9'および'A'～'F'（または'a'～'f'）

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_charBINARY
(
  char                          *char_Data,          /* In  */
  unsigned int                   char_Length,        /* In  */
  unsigned short                 char_Type,          /* In  */
```

```
  unsigned short                BINARY_Length,     /* In  */
  unsigned char                *BINARY_Data,       /* Out */
  unsigned short                BufferLength,      /* In  */
  void                         *Option             /* In  */
)
```

(3) 引数の説明

char_Data：

変換前の文字列データ（C 言語または C++言語の文字列データ）を格納している領域の先頭アドレスを指定します。

char_Length：

変換前の文字列データの長さを指定します（単位：バイト）。次の値を指定してください。

- 変換前の文字列データの形式が 2 進数の場合
  1～*BINARY_Length の値*×8
- 変換前の文字列データの形式が 16 進数の場合
  1～*BINARY_Length の値*×2

char_Type：

変換前の文字列データの形式を指定します。次のどちらかを指定します。

- 変換前の文字列データの形式が 2 進数の場合
  a_rdb_CNV_CHAR_TYPE_BINARY
- 変換前の文字列データの形式が 16 進数の場合
  a_rdb_CNV_CHAR_TYPE_HEX

BINARY_Length：

変換後のBINARY 型データの長さを指定します（単位：バイト）。1～32,000 を指定できます。

BINARY_Data：

変換後のBINARY 型データを格納する領域の先頭アドレスを指定します。

BufferLength：

変換後のBINARY 型データを格納する領域の長さを指定します（単位：バイト）。BINARY_Length と同じ値を指定してください。

Option：

NULL を指定します。値を指定しても無視されます。

(4) 戻り値

1. a_rdb_CNV_charBINARY()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

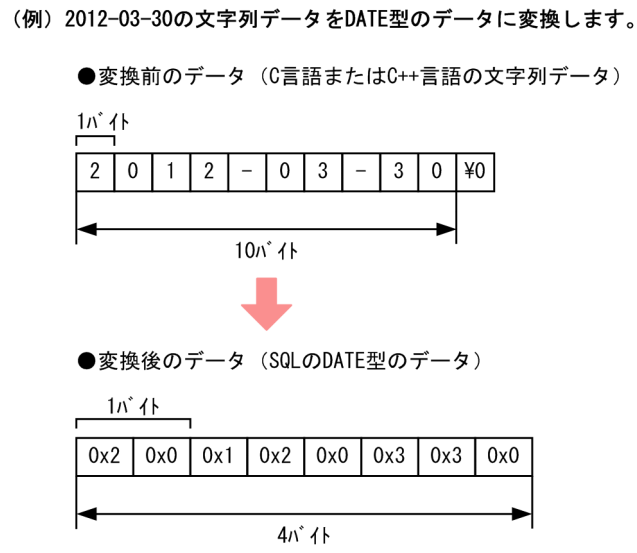
1. 変換前のデータ格納領域（char_Data）と，変換後のデータ格納領域（BINARY_Data）は重複しないようにしてください。
2. 変換前の文字列データの長さは，次の条件を満たしている必要があります。
   - 変換前の文字列データの形式が 2 進数の場合：8 の倍数
   - 変換前の文字列データの形式が 16 進数の場合：2 の倍数
3. 変換前の文字列データをBINARY 型データのデータ形式に変換し，格納領域の先頭から格納します。ただし，変換する文字列データの長さが次に示す値より小さい場合は，右側に0x00 を補います。
   - 変換前の文字列データの形式が 2 進数の場合：*BINARY_Length の値*×8
   - 変換前の文字列データの形式が 16 進数の場合：*BINARY_Length の値*×2

19.5.2 a_rdb_CNV_charDATE()（DATE 型データへの変換）

(1) 機能

C 言語または C++言語の文字列データを SQL のDATE 型のデータに変換します。文字列データをDATE 型のデータに変換する場合の例を次の図に示します。

図 19-2 文字列データを DATE 型のデータに変換する場合の例



重要

変換前の文字列データは，日付を表す既定の入力表現に従った形式にしておく必要があります。日付を表す既定の入力表現については，マニュアル『HADB SQL リファレンス』の『既定の文字列表現』を参照してください。

（例）

2014 年 9 月 30 日を表す場合，文字列データは次のどちらかの形式である必要があります。

2014-09-30

2014/09/30

(2) 形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_charDATE
(
  char              *char_Data,                 /* In  */
  unsigned char     *DATE_Data,                 /* Out */
  unsigned short     BufferLength,              /* In  */
  void              *Option                     /* In  */
)
```

(3) 引数の説明

char_Data：

変換前の文字列データ（C 言語または C++言語の文字列データ）を格納している領域の先頭アドレスを指定します。

DATE_Data：

変換後のDATE 型のデータを格納する領域の先頭アドレスを指定します。

BufferLength：

変換後のDATE 型のデータを格納する領域の長さをバイト数で指定します。4 を指定してください。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_CNV_charDATE()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

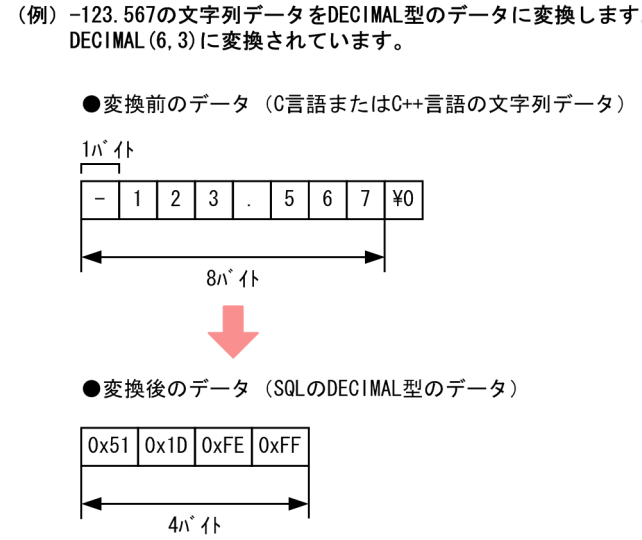
変換前のデータ格納領域（char_Data）と，変換後のデータ格納領域（DATE_Data）は重複しないようにしてください。

19.5.3　a_rdb_CNV_charDECIMAL()（DECIMAL 型または NUMERIC 型データへの変換）

(1)　機能

C 言語または C++言語の文字列データを，SQL のDECIMAL 型またはNUMERIC 型のデータに変換します。文字列データをDECIMAL 型のデータに変換する場合の例を次の図に示します。

図 19-3　文字列データを DECIMAL 型のデータに変換する場合の例



重要

変換前の文字列データは，次に示す形式にしておく必要があります。

```
'[△△…△][+|-][aa....aa][.][bb....bb]'
```

[△△…△]：0 個以上の空白

[+|-]：正符号または負符号（符号を省略した場合は正符号が仮定されます）

[*aa....aa*]：整数部分

[.]：ピリオド（小数部分がない場合はピリオドを省略できます）

[*bb....bb*]：小数部分

注意事項を次に示します。

- 整数部分[*aa....aa*]，または小数点[.]と小数部分[*bb....bb*]のどちらかは必ず指定してください。
- 整数部分[*aa....aa*]と小数部分[*bb....bb*]を合わせて 38 個以下になるようにしてください。
- ピリオド（小数点）を省略した場合は，変換後のデータの末尾に小数点が仮定されます。

(2)　形式

```
#include <adbcnv.h>
```

```
signed short a_rdb_CNV_charDECIMAL
(
  char                *char_Data,                /* In  */
  unsigned short       char_Length,              /* In  */
  unsigned short       DECIMAL_Precision,        /* In  */
  unsigned short       DECIMAL_Scale,            /* In  */
  unsigned char       *DECIMAL_Data,             /* Out */
  unsigned short       BufferLength,             /* In  */
  void                *Option                   /* In  */
)
```

(3) 引数の説明

char_Data：

変換前の文字列データ（C 言語または C++言語の文字列データ）を格納している領域の先頭アドレスを指定します。

char_Length：

変換前の文字列データの長さをバイト数で指定します。1～41 を指定できます。

DECIMAL_Precision：

変換後のDECIMAL 型またはNUMERIC 型のデータの精度（全体の桁数）を指定します。1～38 を指定できます。

？パラメタの関連づけを行う？パラメタと同じ精度を指定してください。

DECIMAL_Scale：

変換後のDECIMAL 型またはNUMERIC 型のデータの位取り（小数点以下の桁数）を指定します。0～38 を指定できます。

？パラメタの関連づけを行う？パラメタと同じ位取りを指定してください。

DECIMAL_Data：

変換後のDECIMAL 型またはNUMERIC 型のデータを格納する領域の先頭アドレスを指定します。

BufferLength：

変換後のDECIMAL 型またはNUMERIC 型のデータを格納する領域の長さをバイト数で指定します。次に示す値を指定してください。

表 19-6　BufferLength の指定値

項番	DECIMAL_Precision の値	BufferLength の指定値
1	1≦DECIMAL_Precision≦4 の場合	2
2	5≦DECIMAL_Precision≦8 の場合	4
3	9≦DECIMAL_Precision≦16 の場合	8
4	17≦DECIMAL_Precision≦38 の場合	16

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_CNV_charDECIMAL()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI関数の戻り値」を参照してください。

(5) 留意事項

1. 変換前のデータ格納領域（char_Data）と，変換後のデータ格納領域（DECIMAL_Data）は重複しないようにしてください。
2. 変換前のデータが”-0.”に相当する場合，同じ文字数の”+0.”と見なします。例えば，変換するデータが”-0.0”の場合は”+0.0”と見なし，変換前のデータが”-0.0000”の場合は”+0.0000”と見なします。
3. 変換前の文字列データをDECIMAL 型またはNUMERIC 型のデータ形式に変換し，格納領域の先頭から格納します。このとき，整数部分が格納できない場合はエラーになります。小数部分が格納できない場合は格納できない部分を切り捨てます。
4. 精度と位取りが一致している場合だけ，整数部に 0 を付加した文字列データを指定できます。その場合，DECIMAL_Precision には，整数部の 0 を除いた整数の数を指定してください。

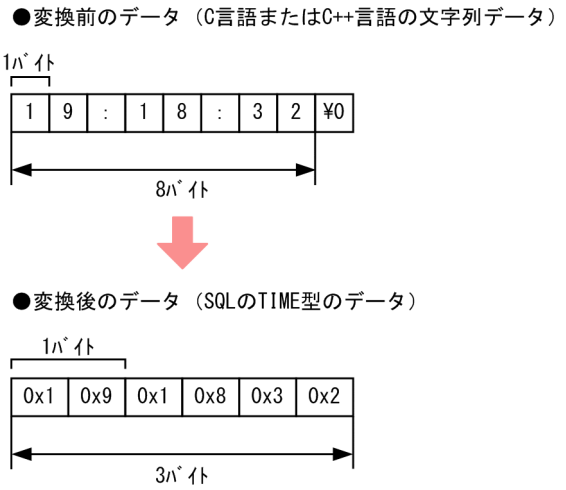
19.5.4 a_rdb_CNV_charTIME()（TIME 型データへの変換）

(1) 機能

C 言語または C++言語の文字列データを SQL のTIME 型のデータに変換します。文字列データをTIME 型のデータに変換する場合の例を次の図に示します。

図 19-4　文字列データを TIME 型のデータに変換する場合の例

（例）19:18:32の文字列データをTIME型のデータに変換します。



重要

変換前の文字列データは，時刻を表す既定の入力表現に従った形式にしておく必要があります。時刻を表す既定の入力表現については，マニュアル『HADB　SQL リファレンス』の『既定の文字列表現』を参照してください。

（例）

11 時 3 分 58.123 秒を表す場合，文字列データは次の形式である必要があります。

11:03:58.123

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_charTIME
(
  char                 *char_Data,                /* In  */
  unsigned short        char_Length,              /* In  */
  unsigned short        TIME_Scale,               /* In  */
  unsigned char        *TIME_Data,                /* Out */
  unsigned short        BufferLength,             /* In  */
  void                 *Option                    /* In  */
)
```

(3)　引数の説明

char_Data：

変換前の文字列データ（C 言語または C++言語の文字列データ）を格納している領域の先頭アドレスを指定します。

char_Length：

変換前の文字列データの長さを指定します（単位：バイト）。8～8 ＋ *TIME_Scale の値* ＋ 1 の範囲の値を指定してください。

TIME_Scale：

変換後のTIME 型データの小数秒の桁数を指定します。0，3，6，9，または12 を指定してください。

TIME_Data：

変換後のTIME 型データを格納する領域の先頭アドレスを指定します。

BufferLength：

変換後のTIME 型データを格納する領域の長さを指定します（単位：バイト）。3 ＋ ↑*TIME_Scale の値* ÷2↑ を指定してください。

Option：

NULL を指定します。値を指定しても無視されます。

(4) 戻り値

1. a_rdb_CNV_charTIME()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. 変換前のデータ格納領域（char_Data）と，変換後のデータ格納領域（TIME_Data）は重複しないようにしてください。
2. 変換する文字列データをTIME 型データのデータ形式に変換し，格納領域の先頭から格納します。ただし，変換する文字列データの小数秒の桁数が3，6，9，または12 桁でない場合は，TIME_Scale に指定した小数秒の桁数になるまで右側に 0 を補います。

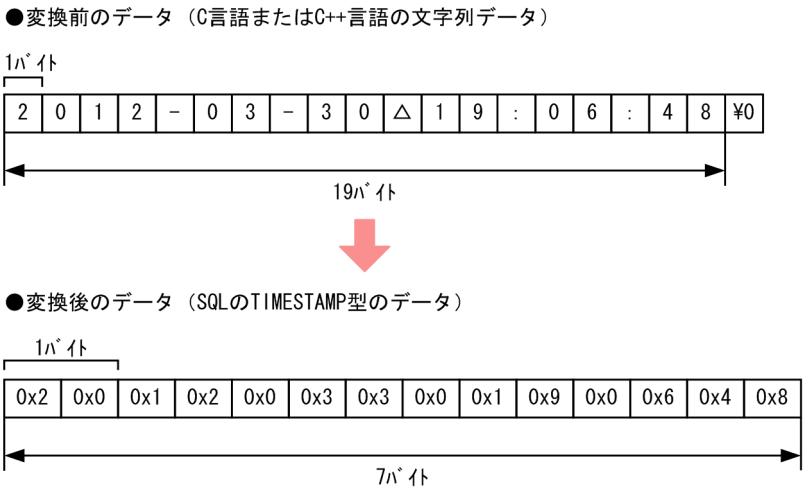
19.5.5 a_rdb_CNV_charTIMESTAMP()（TIMESTAMP 型データへの変換）

(1) 機能

C 言語または C++言語の文字列データを SQL のTIMESTAMP 型のデータに変換します。文字列データをTIMESTAMP 型のデータに変換する場合の例を次の図に示します。

図 19-5　文字列データを TIMESTAMP 型のデータに変換する場合の例

（例）2012-03-30 19:06:48の文字列データをTIMESTAMP型のデータに変換します。



（凡例）△：1バイトの空白文字

重要

変換前の文字列データは，時刻印を表す既定の入力表現に従った形式にしておく必要があります。時刻印を表す既定の入力表現については，マニュアル『HADB　SQL リファレンス』の『既定の文字列表現』を参照してください。

（例）

2014 年 9 月 30 日 19 時 6 分 48 秒を表す場合，文字列データは次のどちらかの形式である必要があります。

```
2014-09-30 19:06:48
```

```
2014/09/30 19:06:48
```

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_charTIMESTAMP
(
  char                  *char_Data,                 /* In  */
  unsigned short         char_Length,               /* In  */
  unsigned short         TIMESTAMP_Scale,           /* In  */
  unsigned char         *TIMESTAMP_Data,            /* Out */
  unsigned short         BufferLength,              /* In  */
  void                  *Option                     /* In  */
)
```

(3) 引数の説明

char_Data：

変換前の文字列データ（C 言語または C++言語の文字列データ）を格納している領域の先頭アドレスを指定します。

char_Length：

変換前の文字列データの長さを指定します（単位：バイト）。19～19 + *TIMESTAMP_Scale の値* + 1 の範囲の値を指定してください。

TIMESTAMP_Scale：

変換後のTIMESTAMP 型データの小数秒の桁数を指定します。0，3，6，9，または12 を指定してください。

TIMESTAMP_Data：

変換後のTIMESTAMP 型のデータを格納する領域の先頭アドレスを指定します。

BufferLength：

変換後のTIMESTAMP 型のデータを格納する領域の長さをバイト数で指定します。7 + ↑*TIMESTAMP_Scale の値*÷2↑を指定してください。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_CNV_charTIMESTAMP()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

1. 変換前のデータ格納領域（char_Data）と，変換後のデータ格納領域（TIMESTAMP_Data）は重複しないようにしてください。
2. 変換する文字列データをTIMESTAMP 型データのデータ形式に変換し，格納領域の先頭から格納します。ただし，変換する文字列データの小数秒の桁数が3，6，9，または12 桁でない場合は，TIMESTAMP_Scale に指定した小数秒の桁数になるまで右側に 0 を補います。

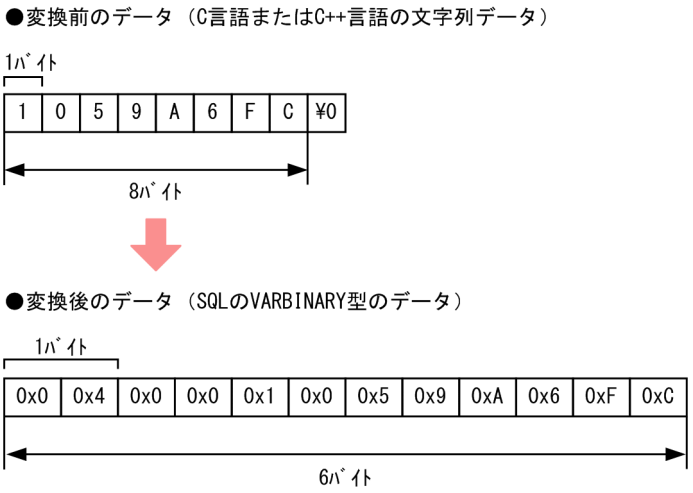
19.5.6 a_rdb_CNV_charVARBINARY()（VARBINARY 型データへの変換）

(1) 機能

C 言語または C++言語の文字列データ（2 進数または 16 進数）を SQL のVARBINARY 型のデータに変換します。文字列データをVARBINARY 型のデータに変換する場合の例を次の図に示します。

図 19-6　文字列データを VARBINARY 型のデータに変換する場合の例

（例）8バイトの16進数文字列データをVARBINARY型のデータに変換します。



重要

変換前の文字列データは，次に示す値である必要があります。

- 変換前の文字列データの形式が 2 進数の場合
  '0'および'1'
- 変換前の文字列データの形式が 16 進数の場合
  '0'～'9'および'A'～'F'（または'a'～'f'）

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_charVARBINARY
(
  char                          *char_Data,           /* In  */
  unsigned int                   char_Length,         /* In  */
  unsigned short                 char_Type,           /* In  */
  unsigned short                 VARBINARY_Length,    /* In  */
  a_rdb_VARBINARY_t             *VARBINARY_Data,      /* Out */
  unsigned short                 BufferLength,        /* In  */
  void                          *Option               /* In  */
)
```

(3)　引数の説明

char_Data：

変換前の文字列データ（C 言語または C++言語の文字列データ）を格納している領域の先頭アドレスを指定します。

char_Length：

変換前の文字列データの長さを指定します（単位：バイト）。次の値を指定してください。

- 変換前の文字列データの形式が 2 進数の場合
  1～*VARBINARY_Length の値*×8
- 変換前の文字列データの形式が 16 進数の場合
  1～*VARBINARY_Length の値*×2

char_Type：

変換前の文字列データの形式を指定します。次のどちらかを指定します。

- 変換前の文字列データの形式が 2 進数の場合
  a_rdb_CNV_CHAR_TYPE_BINARY
- 変換前の文字列データの形式が 16 進数の場合
  a_rdb_CNV_CHAR_TYPE_HEX

VARBINARY_Length：

変換後のVARBINARY 型データの長さを指定します（単位：バイト）。1～32,000 を指定できます。

VARBINARY_Data：

変換後のVARBINARY 型データを格納する領域の先頭アドレスを指定します。

BufferLength：

変換後のVARBINARY 型データを格納する領域の長さを指定します（単位：バイト）。
VARBINARY_Length の値 + 2 を指定してください。

Option：

NULL を指定します。値を指定しても無視されます。

(4) 戻り値

1. a_rdb_CNV_charVARBINARY()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

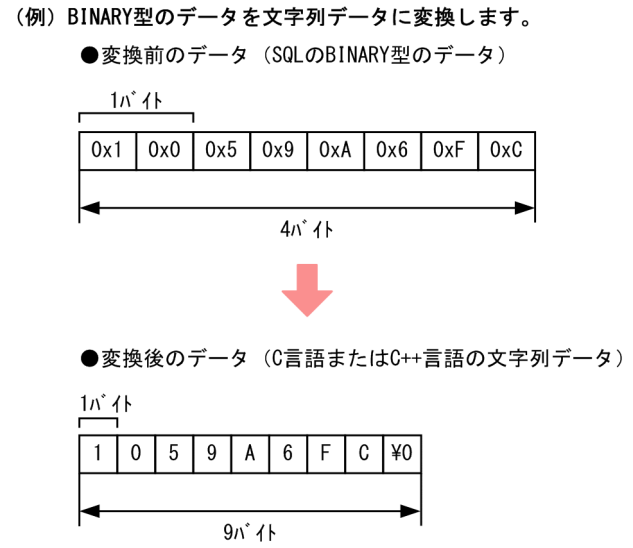
1. 変換前のデータ格納領域（char_Data）と，変換後のデータ格納領域（VARBINARY_Data）は重複しないようにしてください。
2. 変換前の文字列データの長さは，次の条件を満たしている必要があります。
   - 変換前の文字列データの形式が 2 進数の場合：8 の倍数
   - 変換前の文字列データの形式が 16 進数の場合：2 の倍数
3. 変換前の文字列データをVARBINARY 型データのデータ形式に変換し，格納領域の先頭から格納します。

19.5.7　a_rdb_CNV_BINARYchar()（BINARY 型データの変換）

(1)　機能

SQL のBINARY 型のデータを C 言語または C++言語の文字列データに変換します。BINARY 型のデータを文字列データに変換する場合の例を次の図に示します。

図 19-7　BINARY 型のデータを文字列データに変換する場合の例



[説明]

BINARY 型のデータを次の文字列定数の形式に変換し，格納領域の先頭から格納します。末尾には，ナル文字（0x00）が付加されます。

- 変換後の文字列データの形式が 2 進数の場合：2 進形式バイナリ定数
- 変換後の文字列データの形式が 16 進数の場合：16 進形式バイナリ定数

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_BINARYchar
(
  unsigned char                 *BINARY_Data,      /* In  */
  unsigned short                 BINARY_Length,    /* In  */
  unsigned short                 char_Type,        /* In  */
  char                          *char_Data,        /* Out */
  unsigned int                   BufferLength,     /* In  */
  void                          *Option            /* In  */
)
```

(3) 引数の説明

BINARY_Data：

変換前のBINARY 型のデータを格納している領域の先頭アドレスを指定します。

BINARY_Length：

変換前のBINARY 型のデータの長さを指定します（単位：バイト）。1～32,000 を指定できます。

char_Type：

変換後の文字列データの形式を指定します。次のどちらかを指定してください。

- 変換後の文字列データの形式が 2 進数の場合

  a_rdb_CNV_CHAR_TYPE_BINARY
- 変換後の文字列データの形式が 16 進数の場合

  a_rdb_CNV_CHAR_TYPE_HEX

char_Data：

変換後の文字列データ（C 言語または C++言語の文字列データ）を格納する領域の先頭アドレスを指定します。

BufferLength：

変換後の文字列データを格納する領域の長さを指定します（単位：バイト）。次のどちらかの値を指定してください。

- 変換後の文字列データの形式が 2 進数の場合

  BINARY_Length の値×8 ＋ 1
- 変換後の文字列データの形式が 16 進数の場合

  BINARY_Length の値×2 ＋ 1

Option：

NULL を指定します。値を指定しても無視されます。

(4) 戻り値

1. a_rdb_CNV_BINARYchar()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

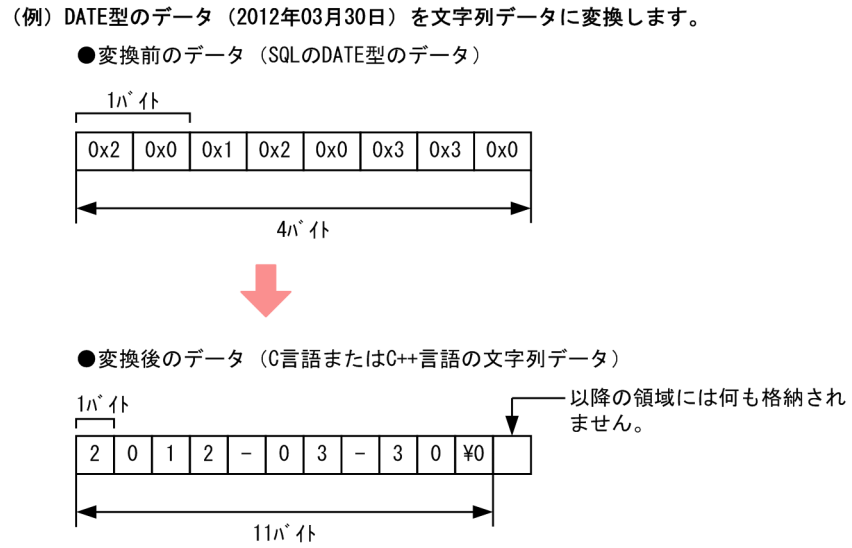
変換前のデータ格納領域（BINARY_Data）と，変換後のデータ格納領域（char_Data）は重複しないようにしてください。

19.5.8　a_rdb_CNV_DATEchar()（DATE 型データの変換）

(1)　機能

SQL のDATE 型のデータを C 言語または C++言語の文字列データに変換します。DATE 型のデータを文字列データに変換する場合の例を次の図に示します。

図 19-8　DATE 型のデータを文字列データに変換する場合の例



[説明]

- DATE 型のデータを，日付を表す既定の出力表現に従って文字列データに変換します。日付を表す既定の出力表現については，マニュアル『HADB　SQL リファレンス』の『既定の文字列表現』を参照してください。
- 末尾にナル文字（0x00）を付加します。

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_DATEchar
(
  unsigned char       *DATE_Data,              /* In  */
  char                *char_Data,              /* Out */
  unsigned short       BufferLength,           /* In  */
  void                *Option                  /* In  */
)
```

(3)　引数の説明

DATE_Data：

変換前のDATE 型のデータを格納している領域の先頭アドレスを指定します。

char_Data：

変換後の文字列データ（C 言語または C++言語の文字列データ）を格納する領域の先頭アドレスを指定します。

BufferLength：

変換後の文字列データを格納する領域の長さをバイト数で指定します。11 を指定してください。

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_CNV_DATEchar()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

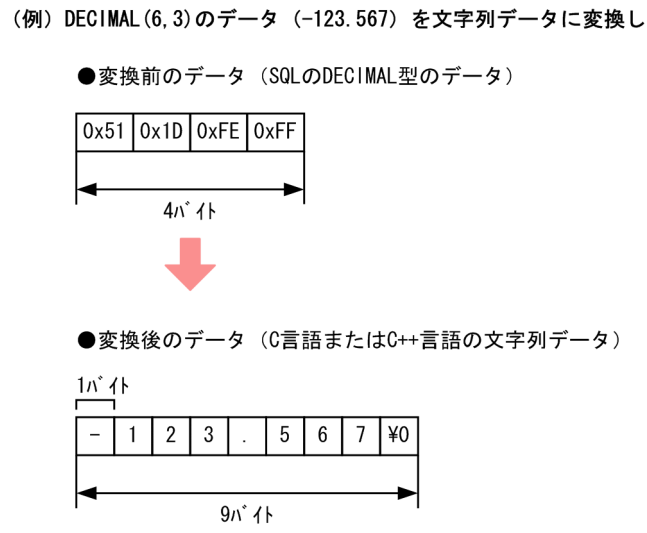
変換前のデータ格納領域（DATE_Data）と，変換後のデータ格納領域（char_Data）は重複しないようにしてください。

19.5.9 a_rdb_CNV_DECIMALchar()（DECIMAL 型または NUMERIC 型データの変換）

(1) 機能

SQL のDECIMAL 型またはNUMERIC 型のデータを，C 言語または C++言語の文字列データに変換します。DECIMAL 型のデータを文字列データに変換する場合の例を次の図に示します。

図 19-9 DECIMAL 型のデータを文字列データに変換する場合の例



[説明]

- DECIMAL 型のデータを符号なしの 10 進数定数の定数表現に文字列変換します。
- 引数のDECIMAL_Precision とDECIMAL_Scale の値が同じ場合は，小数点の前に 0 を付加して文字列変換します。
- DECIMAL_Scale の値が 0 の場合は，小数点は付加されません。
- データが負数の場合は，先頭に負符号'-'（マイナス）を付加します。
- 末尾にナル文字（0x00）を付加して，格納領域に右詰で格納します。
- 格納した文字列の長さよりBufferLength の方が大きい場合，余った領域には空白文字を格納します。

(2) 形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_DECIMALchar
(
  unsigned char          *DECIMAL_Data,             /* In  */
  unsigned short          DECIMAL_Precision,        /* In  */
  unsigned short          DECIMAL_Scale,            /* In  */
  char                   *char_Data,                /* Out */
  unsigned short          BufferLength,             /* In  */
  void                   *Option                    /* In  */
)
```

(3) 引数の説明

DECIMAL_Data：

変換前のDECIMAL 型またはNUMERIC 型のデータを格納している領域の先頭アドレスを指定します。

DECIMAL_Precision：

変換前のDECIMAL 型またはNUMERIC 型のデータの精度（全体の桁数）を指定します。1～38 を指定できます。

取り出した列値のデータ型と同じ精度を指定してください。

DECIMAL_Scale：

変換前のDECIMAL 型またはNUMERIC 型のデータの位取り（小数点以下の桁数）を指定します。0～38 を指定できます。

取り出した列値のデータ型と同じ位取りを指定してください。

char_Data：

変換後の文字列データ（C 言語または C++言語の文字列データ）を格納する領域の先頭アドレスを指定します。

BufferLength：

変換後の文字列データを格納する領域の長さをバイト数で指定します。次に示す値を指定してください。

DECIMAL_Precision ＋ 4

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_CNV_DECIMALchar()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

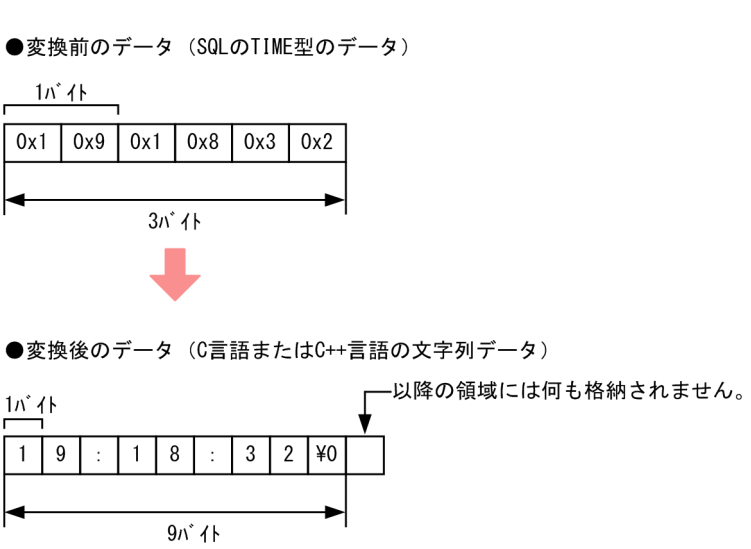
変換前のデータ格納領域（DECIMAL_Data）と，変換後のデータ格納領域（char_Data）は重複しないようにしてください。

19.5.10 a_rdb_CNV_TIMEchar()（TIME 型データの変換）

(1) 機能

SQL のTIME 型のデータを C 言語または C++言語の文字列データに変換します。TIME 型のデータを文字列データに変換する場合の例を次の図に示します。

図 19-10 TIME 型のデータを文字列データに変換する場合の例



［説明］

- TIME 型のデータを，時刻を表す既定の出力表現に従って文字列データに変換します。時刻を表す既定の出力表現については，マニュアル『HADB　SQL リファレンス』の『既定の文字列表現』を参照してください。
- 末尾にナル文字（0x00）を付加します。

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_TIMEchar
(
  unsigned char         *TIME_Data,              /* In  */
  unsigned short         TIME_Scale,             /* In  */
  char                  *char_Data,              /* Out */
  unsigned short         BufferLength,           /* In  */
  void                  *Option                  /* In  */
)
```

(3)　引数の説明

TIME_Data：

変換前のTIME 型データを格納している領域の先頭アドレスを指定します。

TIME_Scale：

変換前のTIME 型データの小数秒の桁数を指定します。0，3，6，9，または12 を指定してください。

char_Data：

変換後の文字列データ（C 言語または C++言語の文字列データ）を格納する領域の先頭アドレスを指定します。

BufferLength：

変換後の文字列データを格納する領域の長さを指定します（単位：バイト）。次に示す値を指定してください。

- 変換前のTIME 型データの小数秒の桁数が0 の場合：9
- 変換前のTIME 型データの小数秒の桁数が3，6，9，または12 の場合：9 ＋ *TIME_Scale の値* ＋ 1

Option：

NULL を指定します。値を指定しても無視されます。

(4)　戻り値

1. a_rdb_CNV_TIMEchar()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8　CLI 関数の戻り値」を参照してください。

(5)　留意事項

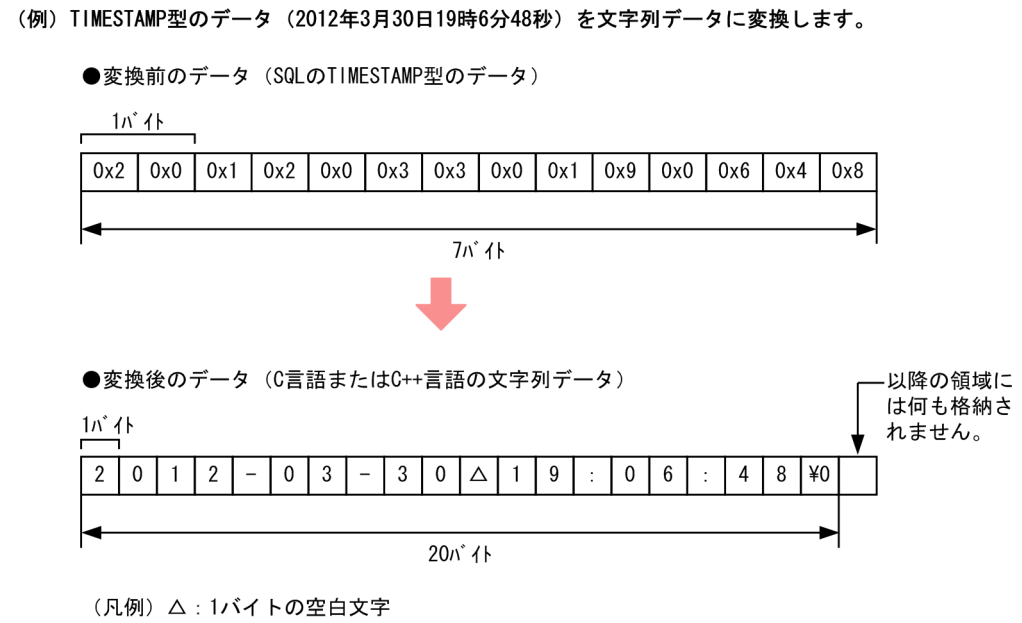
変換前のデータ格納領域（TIME_Data）と，変換後のデータ格納領域（char_Data）は重複しないようにしてください。

19.5.11　a_rdb_CNV_TIMESTAMPchar()（TIMESTAMP 型データの変換）

(1)　機能

SQL のTIMESTAMP 型のデータを C 言語または C++言語の文字列データに変換します。TIMESTAMP 型のデータを文字列データに変換する場合の例を次の図に示します。

図 19-11　TIMESTAMP 型のデータを文字列データに変換する場合の例



[説明]

- TIMESTAMP 型のデータを，時刻印を表す既定の出力表現に従って文字列データに変換します。時刻印を表す既定の出力表現については，マニュアル『HADB　SQL リファレンス』の『既定の文字列表現』を参照してください。
- 末尾にナル文字（0x00）を付加します。

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_TIMESTAMPchar
(
  unsigned char         *TIMESTAMP_Data,          /* In  */
  unsigned short         TIMESTAMP_Scale,         /* In  */
```

```
  char                 *char_Data,              /* Out */
  unsigned short        BufferLength,           /* In  */
  void                 *Option                  /* In  */
)
```

(3) 引数の説明

TIMESTAMP_Data：

変換前のTIMESTAMP 型のデータを格納している領域の先頭アドレスを指定します。

TIMESTAMP_Scale：

変換前のTIMESTAMP 型データの小数秒の桁数を指定します。0，3，6，9，または12 を指定してください。

char_Data：

変換後の文字列データ（C 言語または C++言語の文字列データ）を格納する領域の先頭アドレスを指定します。

BufferLength：

変換後の文字列データを格納する領域の長さを指定します（単位：バイト）。次に示す値を指定してください。

- 変換するTIMESTAMP 型データの小数秒の桁数が0 の場合：20
- 変換するTIMESTAMP 型データの小数秒の桁数が3，6，9，または12 の場合：20 ＋ *TIMESTAMP_Scale の値* ＋ 1

Option：

NULL を指定します。

(4) 戻り値

1. a_rdb_CNV_TIMESTAMPchar()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

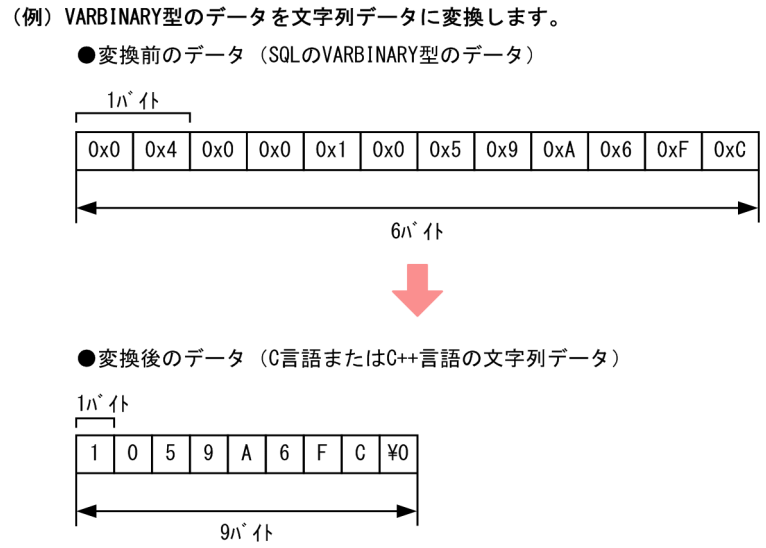
変換前のデータ格納領域（TIMESTAMP_Data）と，変換後のデータ格納領域（char_Data）は重複しないようにしてください。

19.5.12　a_rdb_CNV_VARBINARYchar()（VARBINARY 型データの変換）

(1)　機能

SQL のVARBINARY 型のデータを C 言語または C++言語の文字列データに変換します。VARBINARY 型のデータを文字列データに変換する場合の例を次の図に示します。

図 19-12　VARBINARY 型のデータを文字列データに変換する場合の例



[説明]

VARBINARY 型のデータを次の文字列定数の形式に変換し，格納領域の先頭から格納します。末尾には，ナル文字（0x00）が付加されます。

- 変換後の文字列データの形式が 2 進数の場合：2 進形式バイナリ定数
- 変換後の文字列データの形式が 16 進数の場合：16 進形式バイナリ定数

(2)　形式

```
#include <adbcnv.h>

signed short a_rdb_CNV_VARBINARYchar
(
  a_rdb_VARBINARY_t           *VARBINARY_Data,       /* In  */
  unsigned short               VARBINARY_Length,     /* In  */
  unsigned short               char_Type,            /* In  */
  char                        *char_Data,            /* Out */
  unsigned int                 BufferLength,         /* In  */
  void                        *Option                /* In  */
)
```

(3) 引数の説明

VARBINARY_Data：

変換前のVARBINARY 型のデータを格納している領域の先頭アドレスを指定します。

VARBINARY_Length：

変換前のVARBINARY 型のデータの長さを指定します（単位：バイト）。1～32,000 を指定できます。

char_Type：

変換後の文字列データの形式を指定します。次のどちらかを指定してください。

- 変換後の文字列データの形式が 2 進数の場合
  a_rdb_CNV_CHAR_TYPE_BINARY
- 変換後の文字列データの形式が 16 進数の場合
  a_rdb_CNV_CHAR_TYPE_HEX

char_Data：

変換後の文字列データ（C 言語または C++言語の文字列データ）を格納する領域の先頭アドレスを指定します。

BufferLength：

変換後の文字列データを格納する領域の長さを指定します（単位：バイト）。次のどちらかの値を指定してください。

- 変換後の文字列データの形式が 2 進数の場合
  VARBINARY_Length の値×8 ＋ 1
- 変換後の文字列データの形式が 16 進数の場合
  VARBINARY_Length の値×2 ＋ 1

Option：

NULL を指定します。値を指定しても無視されます。

(4) 戻り値

1. a_rdb_CNV_VARBINARYchar()が正常に終了した場合，a_rdb_RC_CNV_SUCCESS が返却されます。
2. エラーが発生した場合，戻り値にエラーコードが返却されます。エラーコードについては，「19.8 CLI 関数の戻り値」を参照してください。

(5) 留意事項

変換前のデータ格納領域（VARBINARY_Data）と，変換後のデータ格納領域（char_Data）は重複しないようにしてください。

19.6　SQL のデータ型との対応

ここでは，SQL のデータ型と，それに対応する記号定数やデータ型について説明します。

19.6.1　SQL のデータ型と記号定数および値の対応

SQL のデータ型と記号定数および値の対応を次の表に示します。

表 19-7　SQL のデータ型と記号定数および値の対応

項番	SQL のデータ型	記号定数	値
1	CHAR	a_rdb_SQL_DT_CHAR	0xC5
2	VARCHAR	a_rdb_SQL_DT_VARCHAR	0xC1
3	INTEGER	a_rdb_SQL_DT_INT	0xF1
4	SMALLINT	a_rdb_SQL_DT_SMALLINT	0xF5
5	DECIMAL	a_rdb_SQL_DT_DEC	0xE5
6	DOUBLE PRECISION	a_rdb_SQL_DT_DOUBLE	0xE1
7	DATE	a_rdb_SQL_DT_DATE	0x71
8	TIME	a_rdb_SQL_DT_TIME	0x79
9	TIMESTAMP	a_rdb_SQL_DT_TIMESTAMP	0x7D
10	BINARY	a_rdb_SQL_DT_BINARY	0x95
11	VARBINARY	a_rdb_SQL_DT_VARBINARY	0x91
12	ROW	a_rdb_SQL_DT_ROW	0x45

19.6.2　SQL のデータ型とデータ記述の対応

SQL のデータ型と C 言語または C++言語のデータ記述の対応を次の表に示します。

表 19-8　SQL のデータ型と C 言語または C++言語のデータ記述の対応

項番	SQL のデータ型	C 言語または C++言語のデータ記述	領域長（バイト）
1	CHAR(*n*)	char *変数名* [*n*＋1];	*n* ＋ 1
2	VARCHAR(*n*)	a_rdb_M_VARCHAR(*n*) *変数名*;	*n* ＋ 4
3	INTEGER	long long *変数名*;	8
4	SMALLINT	int *変数名*;	4
5	DECIMAL(*m*,*n*)	unsigned char *変数名*[*p*※1];	*p*※1

項番	SQL のデータ型	C 言語または C++言語のデータ記述	領域長（バイト）
6	NUMERIC(*m*,*n*)		
7	DOUBLE PRECISION	double *変数名*;	8
8	FLOAT		
9	DATE	unsigned char *変数名*[4];	4
10	TIME(*p*)※2	unsigned char *変数名*[3＋(*p*＋1)÷2];	3 ＋(*p* ＋ 1)÷2
11	TIMESTAMP(*p*)※2	unsigned char *変数名*[7＋(*p*＋1)÷2];	7 ＋(*p* ＋ 1)÷2
12	BINARY(*n*)	unsigned char *変数名*[*n*];	*n*
13	VARBINARY(*n*)	a_rdb_M_VARBINARY(*n*) *変数名*;	*n* ＋ 2
14	ROW	unsigned char *変数名*[*行長*※3];	*行長*※3

（凡例）

m，*n*：正の整数

注※1

p は*m*（精度）の値によって異なります。

項番	m の値	p の値
1	1≦*m*≦4 の場合	2
2	5≦*m*≦8 の場合	4
3	9≦*m*≦16 の場合	8
4	17≦*m*≦38 の場合	16

注※2

p は小数秒精度で，0，3，6，9，または12 となります。

注※3

各列のデータ長の合計が行長になります。各列のデータ長の計算方法については，マニュアル『HADB SQL リファレンス』の『データ型の種類』の『データ格納長』を参照してください。

データ記述に使用されるマクロは，次のように展開されます。

■a_rdb_M_VARCHAR(*n*) *変数名*;

```
struct
{
  unsigned int Length ;          /* データ長   */
  char Data[n] ;                 /* 文字データ */
}
```

■a_rdb_M_VARBINARY(*n*) *変数名*;

```
struct
{
  unsigned short Length ;         /* データ長       */
  unsigned char Data[n] ;         /* バイナリデータ */
}
```

19.6.3 VARCHAR 型との対応

(1) 機能

a_rdb_VARCHAR_t は，SQL のデータ型のVARCHAR 型に対応します。

(2) 形式

```
typedef struct a_rdb_TG_VARCHAR {
  unsigned int     Length ;
  char             Data[1] ;
} a_rdb_VARCHAR_t ;
```

(3) メンバの説明

Length：

可変長文字列のバイト数を指定します。

Data：

可変長文字列を指定します。

19.6.4 VARBINARY 型との対応

(1) 機能

a_rdb_VARBINARY_t は，SQL のデータ型のVARBINARY 型に対応します。

(2) 形式

```
typedef struct a_rdb_TG_VARBINARY {
  unsigned short   Length ;
  unsigned char   Data[1] ;
} a_rdb_VARBINARY_t ;
```

(3) メンバの説明

Length：

可変長バイナリデータのバイト数を指定します。

Data：

可変長バイナリデータを指定します。

19.7　CLI 関数で使用するデータ型

ここでは，CLI 関数で使用するデータ型について説明します。

19.7.1　a_rdb_SQLColumnInfo_t 構造体（列情報）

(1)　機能

a_rdb_SQLColumnInfo_t 構造体には，a_rdb_SQLDescribeCols()で取得した検索結果列の情報を格納します。

また，a_rdb_SQLBindCols()による検索結果列の関連づけをする際にもこの構造体を使用します。

a_rdb_SQLDescribeCols()については，「19.4.6　a_rdb_SQLDescribeCols()（検索結果列の情報取得）」を，a_rdb_SQLBindCols()については，「19.4.3　a_rdb_SQLBindCols()（検索結果列の関連づけ）」を参照してください。

(2)　形式

```
typedef struct a_rdb_TG_ColumnInfo {
  struct a_rdb_TG_NameInfo    *NameInfo ;              /* Out */
  struct a_rdb_TG_DataType    *TypeInfo ;              /* Out */
  unsigned short               Nullable ;              /* Out */
  char                         _ColumnInfo_rsv01[2] ;
  signed int                   StrLen_or_Ind ;         /* Out */
  void                        *TargetValue ;           /* Out */
  unsigned int                 BufferLength ;          /* In */
  char                         _ColumnInfo_rsv02[28] ;
} a_rdb_SQLColumnInfo_t ;
```

(3)　メンバの説明

NameInfo：

検索結果列の列名に関する情報を取得するアドレスを指定します。

列記述を実行する場合に指定してください。列結合を実行する場合に指定しても無視されます。

0 を指定した場合は，情報は取得しません。

TypeInfo：

検索結果列のデータ型に関する情報を取得するアドレスを指定します。

列記述を実行する場合に指定してください。列結合を実行する場合に指定しても無視されます。

0 を指定した場合は，情報は取得しません。

Nullable：

検索結果列にナル値が返却される可能性があるかどうかを返します。

- ナル値を返却する可能性がない場合：a_rdb_SQL_IS_NOT_NULLABLE

- ナル値を返却する可能性がある場合：a_rdb_SQL_IS_NULLABLE

列記述を実行する場合にこれらの値が設定されます。

StrLen_or_Ind：

値の長さまたはインジケータ値のどちらかを取得します。

行を取り出した際に値が設定されます。

TargetValue：

値を設定するアドレスを指定します。値は，SQL のデータ型に対応する C 言語または C++言語のデータ記述で指定します。

SQL のデータ型に対応する C 言語または C++言語のデータ記述については，「19.6.2　SQL のデータ型とデータ記述の対応」を参照してください。

列結合を実行する場合に指定してください。列記述を実行する場合に指定しても無視されます。

BufferLength：

値を設定する領域の長さを指定します。

データの領域長については，「19.6.2　SQL のデータ型とデータ記述の対応」を参照してください。

列結合を実行する場合に指定してください。列記述を実行する場合に指定しても無視されます。

(4)　留意事項

a_rdb_SQLColumnInfo_t 構造体を使用する場合，最初に領域を 0 で初期化してください。そのあとで，使用する各メンバに値を指定してください。

19.7.2　a_rdb_SQLNameInfo_t 構造体（名称情報）

(1)　機能

a_rdb_SQLNameInfo_t 構造体には，a_rdb_SQLDescribeCols()で取得した検索結果列の列名情報などの名称情報を格納します。a_rdb_SQLDescribeCols()については，「19.4.6　a_rdb_SQLDescribeCols()（検索結果列の情報取得）」を参照してください。

(2)　形式

```
typedef struct a_rdb_TG_NameInfo {
  unsigned short   NameLength ;                /* Out */
  unsigned short   BufferLength ;              /* In * /
  char             _NameInfo_rsv01[4] ;
  char            *Name ;                      /* Out */
} a_rdb_SQLNameInfo_t ;
```

(3) メンバの説明

NameLength：

名称の長さをバイト数で取得します。

BufferLength：

名称を取得する領域の大きさをバイト数で指定します。

Name：

名称を取得する領域のアドレスを指定します。

(4) 留意事項

a_rdb_SQLNameInfo_t 構造体を使用する場合，最初に領域を 0 で初期化してください。そのあとで，使用する各メンバに値を指定してください。

19.7.3 a_rdb_SQLDataType_t 構造体（データ型情報）

(1) 機能

a_rdb_SQLDataType_t 構造体には，次に示す情報を格納します。

- a_rdb_SQLDescribeCols()で取得した検索結果列のデータ型に関する情報
- a_rdb_SQLDescribeParams()で取得した？パラメタのデータ型に関する情報

a_rdb_SQLDescribeCols()については，「19.4.6 a_rdb_SQLDescribeCols()（検索結果列の情報取得）」を，a_rdb_SQLDescribeParams()については，「19.4.7 a_rdb_SQLDescribeParams()（？パラメタの情報取得）」を参照してください。

(2) 形式

```
typedef struct a_rdb_TG_DataType {
  signed int            DataType ;              /* Out */
  unsigned short        ElementCount ;          /* Out */
  char                  _DataType_rsv01[2] ;
  unsigned int          DataLength1 ;           /* Out */
  unsigned int          DataLength2 ;           /* Out */
  char                  _DataType_rsv02[16] ;
} a_rdb_SQLDataType_t ;
```

(3) メンバの説明

DataType：

データ型コードを取得します。

配列型の場合は，要素データ型のデータ型コードを取得します。

ElementCount：

配列型の場合は，最大要素数を取得します。配列型でない場合は，1 を取得します。

DataLength1：

a_rdb_SQLDescribeCols()の場合は列長を取得します。a_rdb_SQLDescribeParams()の場合はパラメタ長を取得します。

取得する値については，「19.4.6　a_rdb_SQLDescribeCols()（検索結果列の情報取得）」または「19.4.7　a_rdb_SQLDescribeParams()（？パラメタの情報取得）」を参照してください。

DataLength2：

a_rdb_SQLDescribeCols()の場合は列長属性を取得します。a_rdb_SQLDescribeParams()の場合はパラメタ長属性を取得します。

詳細については，「19.4.6　a_rdb_SQLDescribeCols()（検索結果列の情報取得）」および「19.4.7　a_rdb_SQLDescribeParams()（？パラメタの情報取得）」を参照してください。

(4)　留意事項

a_rdb_SQLDataType_t 構造体を使用する場合，最初に領域を 0 で初期化してください。そのあとで，使用する各メンバに値を指定してください。

19.7.4　a_rdb_SQLInd_t（インジケータ）

(1)　機能

a_rdb_SQLInd_t（インジケータ）は，値がナル値であるかを識別します。

(2)　形式

```
typedef signed char a_rdb_SQLInd_t ;
```

(3)　インジケータ値

インジケータの意味に対応する記号定数および値を次の表に示します。

表 19-9　インジケータの意味に対応する記号定数および値

項番	意味	記号定数	値
1	ナル値	a_rdb_SQL_NULL_DATA	-1
2	非ナル値データ	a_rdb_SQL_NOT_NULL_DATA	0

19.7.5　a_rdb_SQLParameterInfo_t 構造体（パラメタ情報）

(1)　機能

a_rdb_SQLParameterInfo_t 構造体には，a_rdb_SQLDescribeParams()で取得した？パラメタの情報を格納します。

また，a_rdb_SQLBindParams()またはa_rdb_SQLBindArrayParams()による？パラメタの値を設定する領域の関連づけをする際にもこの構造体を使用します。

a_rdb_SQLDescribeParams()については，「19.4.7　a_rdb_SQLDescribeParams()（？パラメタの情報取得)」を，a_rdb_SQLBindParams()については，「19.4.4　a_rdb_SQLBindParams()（？パラメタの関連づけ)」を，a_rdb_SQLBindArrayParams()については，「19.4.2　a_rdb_SQLBindArrayParams()（？パラメタの一括関連づけ)」を参照してください。

(2)　形式

```
typedef struct a_rdb_TG_ParameterInfo {
  struct a_rdb_TG_DataType     *TypeInfo ;                    /* Out */
  unsigned short                Nullable ;                    /* Out */
  char                          _ParameterInfo_rsv01[2] ;
  a_rdb_SQLInd_t                Ind ;                         /* In  */
  char                          _ParameterInfo_rsv02[3] ;
  void                         *ParameterValue;               /* In  */
  char                          _ParameterInfo_rsv03[24] ;
} a_rdb_SQLParameterInfo_t ;
```

(3)　メンバの説明

TypeInfo：

？パラメタのデータ型に関する情報を取得するアドレスを指定します。

パラメタ記述を実行する場合に指定してください。パラメタ結合を実行する場合は指定する必要はありません。指定した場合は無視されます。

Nullable：

？パラメタにナル値を指定できるかどうかを返します。

- ナル値を指定できない場合：a_rdb_SQL_IS_NOT_NULLABLE
- ナル値を指定できる場合：a_rdb_SQL_IS_NULLABLE

パラメタ記述の場合に値が設定されます。

Ind：

インジケータ値を指定します。

パラメタ結合を実行する場合に指定してください。パラメタ記述を実行する場合は指定する必要はありません。指定した場合は無視されます。

ParameterValue：

値を設定するアドレスを指定します。

値は，SQL データ型に対応する C 言語または C++言語のデータ記述で指定します。SQL のデータ型に対応する C 言語または C++言語のデータ記述については，「19.6.2　SQL のデータ型とデータ記述の対応」を参照してください。

パラメタ結合を実行する場合に指定してください。パラメタ記述を実行する場合は指定する必要はありません。指定した場合は無視されます。

(4)　留意事項

a_rdb_SQLParameterInfo_t 構造体を使用する場合，最初に領域を 0 で初期化してください。そのあとで，使用する各メンバに値を指定してください。

19.7.6　a_rdb_SQLResultInfo_t 構造体（SQL 結果情報）

(1)　機能

a_rdb_SQLResultInfo_t 構造体には，SQL の結果情報が CLI 関数の呼び出し単位で格納されます。

(2)　形式

```
typedef struct a_rdb_TG_ResultInfo {
  unsigned long long  RowCount ;                  /* Out */
  unsigned char       isInConnect ;               /* Out */
  unsigned char       EndTran ;                   /* Out */
  unsigned char       RowCountOverFlowed ;        /* Out */
  unsigned char       isMessageLogFull ;          /* Out */
  unsigned char       isClientLogFull ;           /* Out */
  char                _ResultInfo_rsv01[3] ;
  char                SQLState[5] ;               /* Out */
  char                _ResultInfo_rsv02[11] ;
} a_rdb_SQLResultInfo_t ;
```

(3)　メンバの説明

RowCount：

CLI 関数の戻り値がa_rdb_RC_SQL_SUCCESS またはa_rdb_RC_SQL_NO_DATA の場合，処理結果行数を返します。それ以外の場合は0 を返します。

なお，処理結果行数は 0～18,446,744,073,709,551,615 までの範囲で返却します。

18,446,744,073,709,551,615 を超えてオーバフローした場合は，18,446,744,073,709,551,615 を返却し，RowCountOverFlowed にa_rdb_SQL_OVERFLOWED を設定します。

isInConnect：

コネクションが確立されている（接続中）かどうかを返します。

- 確立されていない場合：a_rdb_SQL_IS_NOT_IN_CONNECT
- 確立されている場合：a_rdb_SQL_IS_IN_CONNECT

EndTran：

トランザクションの実行結果を返します。

- トランザクションが正常終了した場合：a_rdb_SQL_COMMITTED
- トランザクションが取り消された場合：a_rdb_SQL_ROLLBACKED
- トランザクションが実行中の場合：a_rdb_SQL_TRAN_NOT_ENDED
- トランザクションが開始前の場合：a_rdb_SQL_TRAN_NOT_STARTED
- トランザクションが正常終了したときにSQLエラーが発生し，トランザクションを取り消した場合：a_rdb_SQL_DETECTED_ERROR_IN_COMMIT

トランザクションの実行結果とトランザクションの状態の対応を次の表に示します。

表 19-10　トランザクションの実行結果とトランザクションの状態の対応

項番	トランザクションの実行結果	トランザクションの状態
1	a_rdb_SQL_COMMITTED	トランザクション開始可能※
2	a_rdb_SQL_ROLLBACKED	トランザクション開始可能※
3	a_rdb_SQL_TRAN_NOT_ENDED	トランザクション実行中
4	a_rdb_SQL_TRAN_NOT_STARTED	トランザクション開始可能
5	a_rdb_SQL_DETECTED_ERROR_IN_COMMIT	トランザクション開始可能※

注※

COMMITおよびROLLBACKの成功によってトランザクションが決着し，次のトランザクションが開始できる状態です。

RowCountOverFlowed：

RowCountがオーバフローしたかどうかを返します。

- オーバフローしていない場合：a_rdb_SQL_NOT_OVERFLOWED
- オーバフローした場合：a_rdb_SQL_OVERFLOWED

isMessageLogFull：

サーバメッセージログファイルの状態を返します。

- 正常状態の場合：a_rdb_SQL_IS_NOT_LOG_FULL
- 縮退状態の場合：a_rdb_SQL_IS_LOG_FULL

サーバメッセージログファイルが縮退状態の場合は，HADB管理者に連絡してください。

isClientLogFull：

クライアントメッセージログファイルの状態を返します。

- 正常状態の場合：a_rdb_SQL_IS_NOT_LOG_FULL
- 縮退状態の場合：a_rdb_SQL_IS_LOG_FULL

クライアントメッセージログファイルが縮退状態の場合は，クライアントメッセージログファイルを格納しているディスクの空き容量を確保してください。環境変数ADBMSGLOGSIZE で指定したサイズ×2 以上の空き容量が必要になります。

SQLState：

CLI 関数の実行結果のSQLSTATE を返します。

SQLSTATE については，マニュアル『HADB　メッセージ』の『SQLSTATE の一覧』を参照してください。

19.8 CLI 関数の戻り値

次に示す CLI 関数の戻り値を次の表に示します。

- HADB サーバへの接続および切り離し時に使用する CLI 関数
- トランザクションの制御時に使用する CLI 関数
- SQL の実行時に使用する CLI 関数

表 19-11 CLI 関数の戻り値

項番	事象	記号定数	値
1	CLI 関数が正常に終了した	a_rdb_RC_SQL_SUCCESS	0
2	警告あり終了	a_rdb_RC_SQL_WARNING	+1
3	データがない	a_rdb_RC_SQL_NO_DATA	+100
4	SQL エラーが発生した	－	SQLCODE
5	クライアントメッセージログファイルにメッセージが出力できないエラーが発生した	－	*エラー要因コード*※
6	そのほかのエラーが発生した	a_rdb_RC_SQL_ERROR	-1

(凡例)

－：該当しません。

注※

エラー要因コードの一覧を次の表に示します。

表 19-12 エラー要因コードの一覧

項番	エラー要因コード	エラー要因
1	-10000	引数ConnectionHandle の指定に誤りがあります（NULL が指定されました）。
2	-11000	クライアントメッセージログファイルのメッセージカタログファイルの絶対パスの取得に失敗しました。
3	-12*XXX*	クライアントメッセージログファイルのメッセージカタログファイルのopen 処理に失敗しました。*XXX* にはerrno（エラー番号）が返却されます。
4	-13000	環境変数ADBMSGLOGSIZE（クライアントメッセージログファイルのサイズ）の取得に失敗しました。
5	-14000	クライアントメッセージログファイルの絶対パスの取得に失敗しました。
6	-15*XXX*	クライアントメッセージログファイルのopen 処理に失敗しました。*XXX* にはerrno（エラー番号）が返却されます。
7	-16*XXX*	クライアントメッセージログファイルのfstat 処理に失敗しました。*XXX* にはerrno（エラー番号）が返却されます。

項番	エラー要因コード	エラー要因
8	-17*XXX*	クライアントのメッセージログファイルのread 処理に失敗しました。*XXX* にはerrno（エラー番号）が返却されます。
9	-18*XXX*	クライアントのメッセージログファイルのlseek 処理に失敗しました。*XXX* にはerrno（エラー番号）が返却されます。
10	-19*XXX*	クライアントのメッセージログファイルのwrite 処理に失敗しました。*XXX* にはerrno（エラー番号）が返却されます。
11	-21000	HADB クライアントのバージョンと ODBC ドライバのバージョンが不一致です。

データ型の変換時に使用する CLI 関数の戻り値を次の表に示します。

表 19-13　データ型の変換時に使用する CLI 関数の戻り値

項番	事象	記号定数	値	対処方法
1	CLI 関数が正常に終了した	a_rdb_RC_CNV_SUCCESS	0	なし。
2	変換前のデータを格納している領域のアドレスが不正	a_rdb_RC_CNV_INVALID_SRC_ADDRESS	1	変換前のデータを格納している領域のアドレスを確認してください。
3	変換前のデータ長が不正	a_rdb_RC_CNV_INVALID_SRC_LENGTH	2	変換前のデータ長が正しいか確認してください。
4	変換前のデータに変換できないデータがある	a_rdb_RC_CNV_INVALID_FORMAT	3	変換前のデータを確認してください。
5	変換後のデータ型が不正	a_rdb_RC_CNV_INVALID_DEST_LENGTH	4	SQL のデータ型については，マニュアル『HADB　SQL リファレンス』の『データ型』を参照してください。 そのあと，次に示す指定が正しいか確認してください。 • a_rdb_CNV_charDECIMAL()の引数 DECIMAL_Precision または DECIMAL_Scale • a_rdb_CNV_charTIME()の引数 TIME_Scale • a_rdb_CNV_charTIMESTAMP()の引数 TIMESTAMP_Scale • a_rdb_CNV_charBINARY()の引数 BINARY_Length • a_rdb_CNV_charVARBINARY()の引数 VARBINARY_Length
6	変換後のデータを格納する領域のアドレスが不正	a_rdb_RC_CNV_INVALID_DEST_ADDRESS	5	変換後のデータを格納する領域のアドレスを確認してください。

項番	事象	記号定数	値	対処方法
7	変換後のデータを格納する領域の長さが不正	a_rdb_RC_CNV_INVALID_BUF_LENGTH	6	変換されたデータを格納する領域の長さを指定する引数（BufferLength）の値を確認してください。
8	変換前のデータを格納する領域と，変換後のデータを格納する領域が重複している	a_rdb_RC_CNV_OVERRAPPING_AREA	7	領域のアドレスが重複していないか，引数の値を確認してください。
9	指定された文字列種別が不正	a_rdb_RC_CNV_INVALID_CHAR_TYPE	8	正しい文字列種別を指定してください。

付録

付録 A　サンプル AP

HADB サーバへの接続・切り離し，および行の検索，追加，削除を実行する AP をサンプル AP として提供しています。

付録 A.1　サンプル AP の概要

HADB が提供しているサンプル AP を次の表に示します。

表 A-1　HADB が提供しているサンプル AP

項番	サンプル AP の名称	サンプル AP の記述言語	サンプル AP のファイル名	サンプル AP で実行する操作
1	sample1	Java 言語	Sample1.java	• 表の検索（SELECT） • 行の追加（INSERT） • 行の削除（DELETE）
2		ODBC 関数	odbc_sample1.c	
3		C 言語，C++言語	cli_sample1.c	

メモ

サンプル AP は，HADB サーバの次に示すディレクトリ下にあります。

- $ADBDIR/sample

付録 A.2　サンプル AP を実行するための準備

サンプル AP を実行する前に，次に示す準備作業をしてください。

- サンプル AP のmake 環境を作成してください。HADB が提供するインクルードファイルおよびライブラリを設定し，make を実行してください。
- HADB クライアントの環境設定が完了している必要があります。HADB クライアントの環境設定については，「4.　HADB クライアントの環境設定（ODBC ドライバおよび CLI 関数を使用する場合）」を参照してください。
- サンプル AP がアクセスするSAMPLE 表を定義して，SAMPLE 表にデータをインポートしてください。詳細については，「付録 A.3　SAMPLE 表の作成手順」を参照してください。

付録 A.3　SAMPLE 表の作成手順

SAMPLE 表を作成するには，SAMPLE 表を格納するデータ用 DB エリア（ADBUTBL01 およびADBUIDX01）が作成されている必要があります。ADBUTBL01 およびADBUIDX01 を作成していない場合は，ここで説明する方法でSAMPLE 表を作成できません。

SAMPLE 表を作成する手順を次に示します。

手順

1. ADBUSER02 という名前のユーザを作成します。ADBUSER02 ユーザのパスワードは#HelloHADB_02 にします。
   実行する SQL 文

```
CREATE USER ”ADBUSER02” IDENTIFIED BY ’#HelloHADB_02’
```

2. ADBUSER02 ユーザにはCONNECT 権限およびスキーマ定義権限を付与します。
   実行する SQL 文

```
GRANT CONNECT, SCHEMA TO ”ADBUSER02”
```

3. SAMPLE 表を作成するには，SAMPLE 表を作成するシェルスクリプト（$ADBDIR/sample/create_sampledb.sh）を実行します。
   このシェルスクリプトを実行すると，SAMPLE 表が定義されて，SAMPLE 表にデータがインポートされます。

作成されるSAMPLE 表のスキーマ定義，表定義，および B-tree インデクス定義を次に示します。

```
CREATE SCHEMA ”ADBUSER02”

CREATE TABLE ”SAMPLE”(
    ”STATECODE”       SMALLINT,
    ”STATENAME”       VARCHAR(15),
    ”ZIPCODE”         CHAR(15),
    ”ADDRESS”         VARCHAR(100),
    ”AREA”            DECIMAL(19))
  IN ADBUTBL01

CREATE INDEX ”CODE_IDX” ON ”SAMPLE”(”STATECODE” ASC)
  IN ADBUIDX01 EMPTY
```

■SAMPLE 表のイメージ

STATECODE	STATENAME	ZIPCODE	ADDRESS	AREA
1	Alabama	36130-2751	State Capitol N-104 600 Dexter Avenue Montgomery	135,765,000,000
2	Alaska	99811	State Capitol Juneau	1,717,854,000,000
3	Arizona	85007	State Capitol West Wing 1700 W. Washington, 9th Fl. Phoenix	295,254,000,000

STATECODE	STATENAME	ZIPCODE	ADDRESS	AREA
:	:	:	:	:
49	Wisconsin	53707-7863	State Capitol P.O. Box 7863 Madison	169,639,000,000
50	Wyoming	82002-0010	State Capitol Cheyenne	253,336,000,000

[説明]

STATECODE：州番号

STATENAME：州名

ZIPCODE：郵便番号

ADDRESS：州議事堂の住所

AREA：面積

サンプル AP に関連するファイルを次に示します。

- $ADBDIR/sample/Sample1.java：JDBC ドライバを使用したサンプル AP
- $ADBDIR/sample/odbc_sample1.c：ODBC ドライバを使用したサンプル AP
- $ADBDIR/sample/cli_sample1.c：CLI 関数を使用したサンプル AP
- $ADBDIR/sample/create_sampledb.sh：SAMPLE 表を作成するシェルスクリプト
- $ADBDIR/sample/SAMPLE.txt：SAMPLE 表に格納するデータ
- $ADBDIR/sample/SAMPLE_table.sql：SAMPLE 表を定義する定義系 SQL を記述したファイル

付録 A.4　サンプル AP の実行手順

サンプル AP（sample1）の実行手順を次に示します。

手順

1. sample1 の実行ファイルを起動します。
2. 次に示すメッセージが表示されます。実行する SQL 文（1～3）を選択してください。選択した SQL 文ごとの操作方法を(1)以降で説明します。

```
***** HADB CLI Function Sample Program *****
1. Search (specify search range)
2. Add
3. Delete
4. Exit
Please specify a menu item (1 - 4):
```

(1)　SELECT 文を選択した場合（1 を選択した場合）

アメリカの州番号（STATECODE）を指定した検索が実行されます。次に示す SQL 文が実行されます。

```
SELECT ”STATECODE”,”STATENAME”,”ZIPCODE”,”ADDRESS”,”AREA”
    FROM ”SAMPLE”
    WHERE ”STATECODE” BETWEEN ? AND ?
```

SELECT 文を選択すると，次に示すメッセージが標準出力に出力されます。

```
・Please specify the minimum number of search conditions (1 - 50):（入力待ち）
      ← 1つ目の?パラメタの値（下限値）を指定します。
・Please specify the maximum number of search conditions (1 - 50):（入力待ち）
      ← 2つ目の?パラメタの値（上限値）を指定します。
```

指定した値に従ってSELECT 文が実行されます。

> **メモ**
>
> 上記の 2 つの入力が空入力（[改行] だけを入力）の場合，1 つ目に1，2 つ目に50 が指定されたと仮定し，SELECT 文が実行されます。

(2)　INSERT 文を選択した場合（2 を選択した場合）

SAMPLE 表に行が追加されます。次に示す SQL 文が実行されます。

```
INSERT INTO ”SAMPLE” VALUES (?,?,?,?,?)
```

INSERT 文を選択すると，次に示すメッセージが標準出力に出力されます。

```
・Please specify a state code :(入力待ち)
      ← 1つ目の?パラメタの値（州番号）を指定します。
・Please specify a state name :(入力待ち)
      ← 2つ目の?パラメタの値（州名）を指定します。
・Please specify a zip code :(入力待ち)
      ← 3つ目の?パラメタの値（郵便番号）を指定します。
・Please specify an address :(入力待ち)
      ← 4つ目の?パラメタの値（州議事堂の住所）を指定します。
・Please specify an area :(入力待ち)
      ← 5つ目の?パラメタの値（面積）を指定します。
```

指定した値に従ってINSERT 文が実行されます。

なお，不正な値を指定すると SQL エラーになります。

(3)　DELETE 文を選択した場合（3 を選択した場合）

SAMPLE 表から行が削除されます。次に示す SQL 文が実行されます。

```
DELETE FROM ”SAMPLE” WHERE ”STATECODE” = ?
```

DELETE 文を選択すると，次に示すメッセージが標準出力に出力されます。

```
・Please specify the state code of the row to be deleted :(入力待ち)
      ← ?パラメタの値（削除する行の州番号）を指定します。
```

指定した値に従ってDELETE 文が実行されます。

なお，不正な値を指定すると SQL エラーになります。

付録 B　HADB クライアントのディレクトリの構成

ここでは，HADB クライアントのクライアントディレクトリ（インストール時），およびクライアントディレクトリ（運用時）の構成を説明します。

付録 B.1　Windows 版の HADB クライアントの場合

(1)　クライアントディレクトリ（インストール時）の構成

Windows 版の HADB クライアントをインストールしたときのクライアントディレクトリ（インストール時）の構成を次の表に示します。

なお，表中の%INSTCLTDIR%は，クライアントディレクトリ（インストール時）を示します。

表 B-1　クライアントディレクトリ（インストール時）の構成（64 ビット版の HADB クライアントの場合）

項番	フォルダ名およびファイル名	説明
1	%INSTCLTDIR%¥client	HADB クライアントで使用するコマンド，ライブラリ，ODBC ドライバ，および ODBC データソースセットアッププログラムを格納するフォルダ
2	%INSTCLTDIR%¥client¥bin	HADB クライアントで使用するコマンド，ODBC ドライバ，および ODBC データソースセットアッププログラムを格納するフォルダ
3	%INSTCLTDIR%¥client¥bin¥adbclt.dll	HADB クライアント
4	%INSTCLTDIR%¥client¥bin¥adbodbc.dll	ODBC ドライバ
5	%INSTCLTDIR%¥client¥bin¥adbodbcstp.dll	ODBC データソースセットアッププログラム
6	%INSTCLTDIR%¥client¥bin¥adbclt32.dll	HADB クライアント（32bit）
7	%INSTCLTDIR%¥client¥bin¥adbodbc32.dll	ODBC ドライバ（32bit）
8	%INSTCLTDIR%¥client¥bin¥adbodbstp32.dll	ODBC データソースセットアッププログラム（32bit）
9	%INSTCLTDIR%¥client¥lib	HADB クライアントで使用する各種ライブラリを格納するフォルダ
10	%INSTCLTDIR%¥client¥lib¥adbclt.lib	クライアントライブラリ
11	%INSTCLTDIR%¥client¥lib¥adbclt32.lib	クライアントライブラリ（32bit）
12	%INSTCLTDIR%¥client¥lib¥adbjdbc8.jar	JDBC ドライバ（JRE8 版）
13	%INSTCLTDIR%¥conf	クライアント定義を格納するフォルダ
14	%INSTCLTDIR%¥include	ユーザ提供ヘッダファイルを格納するフォルダ
15	%INSTCLTDIR%¥include¥adbcnv.h	ユーザ提供ヘッダファイル

項番	フォルダ名およびファイル名	説明
16	%INSTCLTDIR%¥include¥adbcli.h	
17	%INSTCLTDIR%¥include¥adbtypes.h	
18	%INSTCLTDIR%¥include¥adbodb.h	
19	%INSTCLTDIR%¥lib	各種ライブラリを格納するフォルダ
20	%INSTCLTDIR%¥lib¥sysdef	定義解析情報ファイルを格納するフォルダ
21	%INSTCLTDIR%¥lib¥sysdef¥adbclt.def	定義解析情報ファイル
22	%INSTCLTDIR%¥sample	サンプル AP および定義ファイルのひな形を格納するフォルダ
23	%INSTCLTDIR%¥sample¥cli_sample1.c	サンプル AP
24	%INSTCLTDIR%¥sample¥Sample1.java	サンプル AP（JDBC 用）
25	%INSTCLTDIR%¥sample¥odbc_sample1.c	サンプル AP（ODBC 用）
26	%INSTCLTDIR%¥sample¥conf	定義ファイルのひな形を格納するフォルダ
27	%INSTCLTDIR%¥sample¥conf¥client.def	クライアント定義のひな形ファイル
28	%INSTCLTDIR%¥spool	HADB クライアントの実行結果ログを格納するフォルダ
29	%INSTCLTDIR%¥vclib	Microsoft Visual Studio 2019 の Visual C++再頒布可能パッケージを格納するフォルダ
30	%INSTCLTDIR%¥vclib¥VC_redist.x64.exe	Microsoft Visual Studio 2019 の Visual C++再頒布可能パッケージ
31	%INSTCLTDIR%¥adbreg.reg	レジストリ登録コマンド
32	%INSTCLTDIR%¥adbunreg.reg	レジストリ削除コマンド
33	%INSTCLTDIR%¥readme.txt	README ファイル（リリースノート）

メモ

上記の表に示すフォルダおよびファイルが作成されるタイミングは，HADB クライアントのインストール時です。また，上記の表に示すフォルダおよびファイルが削除されるタイミングは，HADB クライアントのアンインストール時です。

表 B-2　クライアントディレクトリ（インストール時）の構成（32 ビット版の HADB クライアントの場合）

項番	フォルダ名およびファイル名	説明
1	%INSTCLTDIR%¥client	HADB クライアントで使用するコマンド，ライブラリ，ODBC ドライバ，および ODBC データソースセットアッププログラムを格納するフォルダ

項番	フォルダ名およびファイル名	説明
2	%INSTCLTDIR%¥client¥bin	HADB クライアントで使用するコマンド，ODBC ドライバ，および ODBC データソースセットアッププログラムを格納するフォルダ
3	%INSTCLTDIR%¥client¥bin¥adbclt32.dll	HADB クライアント
4	%INSTCLTDIR%¥client¥bin¥adbodbc32.dll	ODBC ドライバ
5	%INSTCLTDIR%¥client¥bin¥adbodbstp32.dll	ODBC データソースセットアッププログラム
6	%INSTCLTDIR%¥client¥lib	HADB クライアントで使用する各種ライブラリを格納するフォルダ
7	%INSTCLTDIR%¥client¥lib¥adbclt32.lib	クライアントライブラリ
8	%INSTCLTDIR%¥client¥lib¥adbjdbc8.jar	JDBC ドライバ（JRE8 版）
9	%INSTCLTDIR%¥conf	クライアント定義を格納するフォルダ
10	%INSTCLTDIR%¥include	ユーザ提供ヘッダファイルを格納するフォルダ
11	%INSTCLTDIR%¥include¥adbcnv.h	ユーザ提供ヘッダファイル
12	%INSTCLTDIR%¥include¥adbcli.h	
13	%INSTCLTDIR%¥include¥adbtypes.h	
14	%INSTCLTDIR%¥include¥adbodb.h	
15	%INSTCLTDIR%¥lib	各種ライブラリを格納するフォルダ
16	%INSTCLTDIR%¥lib¥sysdef	定義解析情報ファイルを格納するフォルダ
17	%INSTCLTDIR%¥lib¥sysdef¥adbclt.def	定義解析情報ファイル
18	%INSTCLTDIR%¥sample	サンプル AP および定義ファイルのひな形を格納するフォルダ
19	%INSTCLTDIR%¥sample¥cli_sample1.c	サンプル AP
20	%INSTCLTDIR%¥sample¥Sample1.java	サンプル AP（JDBC 用）
21	%INSTCLTDIR%¥sample¥odbc_sample1.c	サンプル AP（ODBC 用）
22	%INSTCLTDIR%¥sample¥conf	定義ファイルのひな形を格納するフォルダ
23	%INSTCLTDIR%¥sample¥conf¥client.def	クライアント定義のひな形ファイル
24	%INSTCLTDIR%¥spool	HADB クライアントの実行結果ログを格納するフォルダ
25	%INSTCLTDIR%¥vclib32	Microsoft Visual Studio 2019 の Visual C++再頒布可能パッケージを格納するフォルダ
26	%INSTCLTDIR%¥vclib32¥VC_redist.x86.exe	Microsoft Visual Studio 2019 の Visual C++再頒布可能パッケージ
27	%INSTCLTDIR%¥adbreg32.reg	レジストリ登録コマンド
28	%INSTCLTDIR%¥adbunreg32.reg	レジストリ削除コマンド

項番	フォルダ名およびファイル名	説明
29	%INSTCLTDIR%¥readme.txt	README ファイル（リリースノート）

メモ

上記の表に示すフォルダおよびファイルが作成されるタイミングは，HADB クライアントのインストール時です。また，上記の表に示すフォルダおよびファイルが削除されるタイミングは，HADB クライアントのアンインストール時です。

(2) クライアントディレクトリ（運用時）の構成

Windows 版の HADB クライアントの場合は，クライアントディレクトリ（インストール時）をそのままクライアントディレクトリ（運用時）として使用します。

表 B-3 クライアントディレクトリ（運用時）の構成（64 ビット版の HADB クライアントの場合）

項番	フォルダ名およびファイル名	説明	作成されるタイミング	削除されるタイミング
1	%ADBCLTDIR%¥client	HADB クライアントで使用するコマンド，ライブラリ，ODBC ドライバ，および ODBC データソースセットアッププログラムを格納するフォルダ	HADB クライアントのインストール時	HADB クライアントのアンインストール時
2	%ADBCLTDIR%¥client¥bin	HADB クライアントで使用するコマンド，ODBC ドライバ，および ODBC データソースセットアッププログラムを格納するフォルダ		
3	%ADBCLTDIR%¥client¥bin¥adbclt.dll	HADB クライアント		
4	%ADBCLTDIR%¥client¥bin¥adbodbc.dll	ODBC ドライバ		
5	%ADBCLTDIR%¥client¥bin¥adbodbcstp.dll	ODBC データソースセットアッププログラム		
6	%ADBCLTDIR%¥client¥bin¥adbclt32.dll	HADB クライアント（32bit）		
7	%ADBCLTDIR%¥client¥bin¥adbodbc32.dll	ODBC ドライバ（32bit）		
8	%ADBCLTDIR%¥client¥bin¥adbodbstp32.dll	ODBC データソースセットアッププログラム（32bit）		
9	%ADBCLTDIR%¥client¥lib	HADB クライアントで使用する各種ライブラリを格納するフォルダ		

項番	フォルダ名およびファイル名	説明	作成されるタイミング	削除されるタイミング
10	%ADBCLTDIR%¥client¥lib¥adbclt.lib	クライアントライブラリ		
11	%ADBCLTDIR%¥client¥lib¥adbclt32.lib	クライアントライブラリ（32bit）		
12	%ADBCLTDIR%¥client¥lib¥adbjdbc8.jar	JDBC ドライバ（JRE8 版）		
13	%ADBCLTDIR%¥conf	クライアント定義を格納するフォルダ		
14	%ADBCLTDIR%¥include	ユーザ提供ヘッダファイルを格納するフォルダ		
15	%ADBCLTDIR%¥include¥adbcnv.h	ユーザ提供ヘッダファイル		
16	%ADBCLTDIR%¥include¥adbcli.h			
17	%ADBCLTDIR%¥include¥adbtypes.h			
18	%ADBCLTDIR%¥include¥adbodb.h			
19	%ADBCLTDIR%¥lib	各種ライブラリを格納するフォルダ		
20	%ADBCLTDIR%¥lib¥sysdef	定義解析情報ファイルを格納するフォルダ		
21	%ADBCLTDIR%¥lib¥sysdef¥adbclt.def	定義解析情報ファイル		
22	%ADBCLTDIR%¥sample	サンプル AP および定義ファイルのひな形を格納するフォルダ		
23	%ADBCLTDIR%¥sample¥cli_sample1.c	サンプル AP		
24	%ADBCLTDIR%¥sample¥Sample1.java	サンプル AP（JDBC 用）		
25	%ADBCLTDIR%¥sample¥odbc_sample1.c	サンプル AP（ODBC 用）		
26	%ADBCLTDIR%¥sample¥conf	定義ファイルのひな形を格納するフォルダ		
27	%ADBCLTDIR%¥sample¥conf¥client.def	クライアント定義のひな形ファイル		
28	%ADBCLTDIR%¥spool※1	HADB クライアントの実行結果ログを格納するフォルダ		
29	%ADBCLTDIR%¥spool¥adbmessagecltXX.log	クライアントメッセージログファイル※2	HADB クライアントのインストール後の，HADB クライアントから HADB サーバへの初回接続時	
30	%ADBCLTDIR%¥spool¥.adbmessageclt	クライアントメッセージログファイル番号管理ファイル		

項番	フォルダ名およびファイル名	説明	作成されるタイミング	削除されるタイミング
31	%ADBCLTDIR%¥vclib	Microsoft Visual Studio 2019 の Visual C++再頒布可能パッケージを格納するフォルダ	HADB クライアントのインストール時	
32	%ADBCLTDIR%¥vclib¥VC_redist.x64.exe	Microsoft Visual Studio 2019 の Visual C++再頒布可能パッケージ		
33	%ADBCLTDIR%¥adbreg.reg	レジストリ登録コマンド		
34	%ADBCLTDIR%¥adbunreg.reg	レジストリ削除コマンド		
35	%ADBCLTDIR%¥readme.txt	README ファイル（リリースノート）		
36	%ADBODBTRCPATH%¥adbodbtrace_PID_TID_*xx*.log※1	HADB ODBC ドライバトレースファイル※3	HADB ODBC ドライバを使用した初回のハンドル取得時	HADB クライアントはファイルを削除しません。

注※1

HADB クライアント（ODBC ドライバを含む）を使用する可能性がある OS ユーザに対して，次のフォルダの書き込み権限を与えてください。

- %ADBCLTDIR%¥spool
- %ADBODBTRCPATH%

注※2

クライアントメッセージログファイルは，最大 4 個作成されます。1 ファイル当たりの最大容量は，環境変数ADBMSGLOGSIZE で指定します。

注※3

HADB ODBC ドライバトレースファイルは，1 プロセスまたは 1 スレッド当たり最大 2 個作成されます。1 ファイル当たりの最大容量は，環境変数ADBODBTRCSIZE で指定します。

HADB ODBC ドライバトレースファイルを格納するフォルダは，ユーザが作成します。フォルダが不要になった場合は，ユーザがフォルダを削除します，HADB クライアントは，フォルダを作成または削除しません。

表 B-4　クライアントディレクトリ（運用時）の構成（32 ビット版の HADB クライアントの場合）

項番	フォルダ名およびファイル名	説明	作成されるタイミング	削除されるタイミング
1	%ADBCLTDIR%¥client	HADB クライアントで使用するコマンド，ライブラリ，ODBC ドライバ，および ODBC データソースセットアッププログラムを格納するフォルダ	HADB クライアントのインストール時	HADB クライアントのアンインストール時
2	%ADBCLTDIR%¥client¥bin	HADB クライアントで使用するコマンド，ODBC ドライバ，および		

項番	フォルダ名およびファイル名	説明	作成されるタイミング	削除されるタイミング
		ODBC データソースセットアッププログラムを格納するフォルダ		
3	%ADBCLTDIR%¥client¥bin¥adbclt32.dll	HADB クライアント		
4	%ADBCLTDIR%¥client¥bin¥adbodbc32.dll	ODBC ドライバ		
5	%ADBCLTDIR%¥client¥bin¥adbodbstp32.dll	ODBC データソースセットアッププログラム		
6	%ADBCLTDIR%¥client¥lib	HADB クライアントで使用する各種ライブラリを格納するフォルダ		
7	%ADBCLTDIR%¥client¥lib¥adbclt32.lib	クライアントライブラリ		
8	%ADBCLTDIR%¥client¥lib¥adbjdbc8.jar	JDBC ドライバ（JRE8 版）		
9	%ADBCLTDIR%¥conf	クライアント定義を格納するフォルダ		
10	%ADBCLTDIR%¥include	ユーザ提供ヘッダファイルを格納するフォルダ		
11	%ADBCLTDIR%¥include¥adbcnv.h	ユーザ提供ヘッダファイル		
12	%ADBCLTDIR%¥include¥adbcli.h			
13	%ADBCLTDIR%¥include¥adbtypes.h			
14	%ADBCLTDIR%¥include¥adbodb.h			
15	%ADBCLTDIR%¥lib	各種ライブラリを格納するフォルダ		
16	%ADBCLTDIR%¥lib¥sysdef	定義解析情報ファイルを格納するフォルダ		
17	%ADBCLTDIR%¥lib¥sysdef¥adbclt.def	定義解析情報ファイル		
18	%ADBCLTDIR%¥sample	サンプル AP および定義ファイルのひな形を格納するフォルダ		
19	%ADBCLTDIR%¥sample¥cli_sample1.c	サンプル AP		
20	%ADBCLTDIR%¥sample¥Sample1.java	サンプル AP（JDBC 用）		
21	%ADBCLTDIR%¥sample¥odbc_sample1.c	サンプル AP（ODBC 用）		
22	%ADBCLTDIR%¥sample¥conf	定義ファイルのひな形を格納するフォルダ		

項番	フォルダ名およびファイル名	説明	作成されるタイミング	削除されるタイミング
23	%ADBCLTDIR%¥sample¥conf¥client.def	クライアント定義のひな形ファイル		
24	%ADBCLTDIR%¥spool※1	HADB クライアントの実行結果ログを格納するフォルダ		
25	%ADBCLTDIR%¥spool¥adbmessagecltXX.log	クライアントメッセージログファイル※2	HADB クライアントのインストール後の，HADB クライアントから HADB サーバへの初回接続時	
26	%ADBCLTDIR%¥spool¥.adbmessageclt	クライアントメッセージログファイル番号管理ファイル		
27	%ADBCLTDIR%¥vclib32	Microsoft Visual Studio 2019 の Visual C++再頒布可能パッケージを格納するフォルダ	HADB クライアントのインストール時	
28	%ADBCLTDIR%¥vclib32¥VC_redist.x86.exe	Microsoft Visual Studio 2019 の Visual C++再頒布可能パッケージ		
29	%ADBCLTDIR%¥adbreg32.reg	レジストリ登録コマンド		
30	%ADBCLTDIR%¥adbunreg32.reg	レジストリ削除コマンド		
31	%ADBCLTDIR%¥readme.txt	README ファイル（リリースノート）		
32	%ADBODBTRCPATH%¥adbodbtrace_PID_TID_XX.log※1	HADB ODBC ドライバトレースファイル※3	HADB ODBC ドライバを使用した初回のハンドル取得時	HADB クライアントはファイルを削除しません。

注※1

HADB クライアント（ODBC ドライバを含む）を使用する可能性がある OS ユーザに対して，次のフォルダの書き込み権限を与えてください。

- %ADBCLTDIR%¥spool
- %ADBODBTRCPATH%

注※2

クライアントメッセージログファイルは，最大 4 個作成されます。1 ファイル当たりの最大容量は，環境変数ADBMSGLOGSIZE で指定します。

注※3

HADB ODBC ドライバトレースファイルは，1 プロセスまたは 1 スレッド当たり最大 2 個作成されます。1 ファイル当たりの最大容量は，環境変数ADBODBTRCSIZE で指定します。

HADB ODBC ドライバトレースファイルを格納するフォルダは，ユーザが作成します。フォルダが不要になった場合は，ユーザがフォルダを削除します，HADB クライアントは，フォルダを作成または削除しません。

付録 B.2　Linux 版の HADB クライアントの場合

(1)　クライアントディレクトリ（インストール時）の構成

Linux 版の HADB クライアントディレクトリ（インストール時）の構成を次の表に示します。

なお，表中の$INSTCLTDIR は HADB クライアントディレクトリ（インストール時）を示します。HADB クライアントディレクトリ（インストール時）は，adbinstall コマンドで HADB クライアントをインストールしたディレクトリです。

表 B-5　クライアントディレクトリ（インストール時）の構成

項番	ディレクトリ名およびファイル名	説明
1	$INSTCLTDIR/client	HADB クライアントで使用するコマンドおよびライブラリを格納するディレクトリ
2	$INSTCLTDIR/client/bin	HADB クライアントで使用するコマンドを格納するディレクトリ
3	$INSTCLTDIR/client/bin/adbsql	SQL 実行コマンド
4	$INSTCLTDIR/client/lib	HADB クライアントで使用する各種ライブラリを格納するディレクトリ
5	$INSTCLTDIR/client/lib/libadbclt.so	クライアントライブラリ
6	$INSTCLTDIR/client/lib/libadbodbc.so	ODBC ドライバ
7	$INSTCLTDIR/client/lib/adbjdbc8.jar	JDBC ドライバ（JRE8 版）
8	$INSTCLTDIR/conf	クライアント定義を格納するディレクトリ
9	$INSTCLTDIR/include	ユーザ提供ヘッダファイルを格納するディレクトリ
10	$INSTCLTDIR/include/adbcnv.h	ユーザ提供ヘッダファイル
11	$INSTCLTDIR/include/adbcli.h	
12	$INSTCLTDIR/include/adbtypes.h	
13	$INSTCLTDIR/include/adbodb.h	
14	$INSTCLTDIR/lib	各種ライブラリを格納するディレクトリ
15	$INSTCLTDIR/lib/adbmsg.cat	メッセージカタログファイル
16	$INSTCLTDIR/lib/sysdef	定義解析情報ファイルを格納するディレクトリ
17	$INSTCLTDIR/lib/sysdef/adbclt.def	定義解析情報ファイル
18	$INSTCLTDIR/sample	サンプル AP および定義ファイルのひな形を格納するディレクトリ
19	$INSTCLTDIR/sample/cli_sample1.c	サンプル AP
20	$INSTCLTDIR/sample/Sample1.java	サンプル AP（JDBC 用）

項番	ディレクトリ名およびファイル名	説明
21	$INSTCLTDIR/sample/odbc_sample1.c	サンプル AP（ODBC 用）
22	$INSTCLTDIR/sample/conf	定義ファイルのひな形を格納するディレクトリ
23	$INSTCLTDIR/sample/conf/client.def	クライアント定義のひな形ファイル
24	$INSTCLTDIR/spool	HADB クライアントの実行結果ログを格納するディレクトリ
25	$INSTCLTDIR/adbinstcl.log	HADB クライアントをインストールしたときの実行結果ログファイル

メモ

上記の表に示すディレクトリおよびファイルが作成されるタイミングは，HADB クライアントのインストール時です。また，上記の表に示すディレクトリおよびファイルが削除されるタイミングは，HADB クライアントのアンインストール時です。

(2) クライアントディレクトリ（運用時）の構成

クライアントディレクトリ（運用時）の構成を次の表に示します。

表 B-6 クライアントディレクトリ（運用時）の構成

項番	ディレクトリ名およびファイル名	説明	作成されるタイミング	削除されるタイミング
1	$ADBCLTDIR/client	HADB クライアントで使用するコマンドおよびライブラリを格納するディレクトリ	HADB クライアントのインストール時	HADB クライアントのアンインストール時
2	$ADBCLTDIR/client/bin	HADB クライアントで使用するコマンドを格納するディレクトリ		
3	$ADBCLTDIR/client/bin/adbsql	SQL 実行コマンド		
4	$ADBCLTDIR/client/lib	HADB クライアントで使用する各種ライブラリを格納するディレクトリ		
5	$ADBCLTDIR/client/lib/libadbclt.so	クライアントライブラリ		
6	$ADBCLTDIR/client/lib/libadbodbc.so	ODBC ドライバ		
7	$ADBCLTDIR/client/lib/adbjdbc8.jar	JDBC ドライバ（JRE8 版）		
8	$ADBCLTDIR/conf	クライアント定義を格納するディレクトリ		

項番	ディレクトリ名およびファイル名	説明	作成されるタイミング	削除されるタイミング
9	$ADBCLTDIR/include	ユーザ提供ヘッダファイルを格納するディレクトリ		
10	$ADBCLTDIR/include/adbcnv.h	ユーザ提供ヘッダファイル		
11	$ADBCLTDIR/include/adbcli.h			
12	$ADBCLTDIR/include/adbtypes.h			
13	$ADBCLTDIR/include/adbodb.h			
14	$ADBCLTDIR/lib	各種ライブラリを格納するディレクトリ		
15	$ADBCLTDIR/lib/adbmsg.cat	メッセージカタログファイル		
16	$ADBCLTDIR/lib/sysdef	定義解析情報ファイルを格納するディレクトリ		
17	$ADBCLTDIR/lib/sysdef/adbclt.def	定義解析情報ファイル		
18	$ADBCLTDIR/sample	サンプル AP および定義ファイルのひな形を格納するディレクトリ		
19	$ADBCLTDIR/sample/cli_sample1.c	サンプル AP		
20	$ADBCLTDIR/sample/Sample1.java	サンプル AP（JDBC 用）		
21	$ADBCLTDIR/sample/odbc_sample1.c	サンプル AP（ODBC 用）		
22	$ADBCLTDIR/sample/conf	定義ファイルのひな形を格納するディレクトリ		
23	$ADBCLTDIR/sample/conf/client.def	クライアント定義のひな形ファイル		
24	$ADBCLTDIR/spool	HADB クライアントの実行結果ログを格納するディレクトリ		
25	$ADBCLTDIR/spool/adbmessagecltXX.log	クライアントメッセージログファイル※	HADB クライアントのインストール後の，HADB クライアントから HADB サーバへの初回接続時	
26	$ADBCLTDIR/spool/.adbmessageclt	クライアントメッセージログファイル番号管理ファイル		
27	$ADBCLTDIR/adbinstcl.log	HADB クライアントをインストールしたときの実行結果ログファイル	HADB クライアントのインストール時	

注※

クライアントメッセージログファイルは，最大 4 個作成されます。1 ファイル当たりの最大容量は，環境変数ADBMSGLOGSIZE で指定します。

付録 C　HADB クライアントのメモリ所要量の見積もり

ここでは，HADB クライアントが使用するメモリ所要量の見積もりについて説明します。

付録 C.1　HADB サーバへの接続時に使用するメモリ所要量

HADB サーバへの接続時に使用するメモリ所要量は，18 キロバイトになります。

付録 C.2　HADB クライアントと HADB サーバの通信時に使用するメモリ所要量

HADB クライアントと HADB サーバの通信時に使用するメモリ所要量は，次に示す計算式から求めてください。

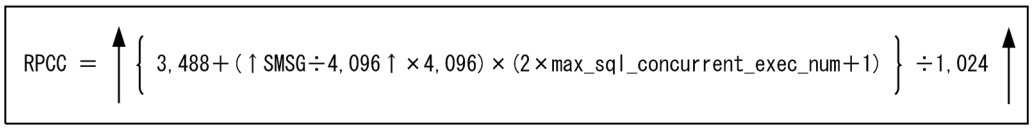
計算式（単位：キロバイト）

通信時に使用するメモリ所要量 ＝ *RPCC* ＋*SBF* ＋256

(1)　RPCC の求め方

RPCC（通信管理情報）は，次に示す計算式から求めてください。

計算式（単位：キロバイト）



SMSG：HADB サーバの送信データサイズ

SMSG 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『通常運用時のメモリ所要量の求め方』の『リアルスレッド固有メモリの所要量の求め方（通常運用時)』にある『変数 RTHD_COMMUSZ の求め方』の，変数 *SMSG* の説明を参照してください。

max_sql_concurrent_exec_num：

1 トランザクション内で同時に実行する SQL 文の最大数

(2)　SBF の求め方

SBF（送信バッファ）の初期確保サイズは 4,096 バイトになります。

初期確保サイズを超える送信データが作成された場合，次に示す計算式で求められた値を再確保します。

計算式（再確保時）（単位：キロバイト）

$$SBF = \uparrow CMSG \div 4{,}096 \uparrow \times 4$$

CMSG：HADB クライアントからの送信データサイズ

送信データサイズは処理の内容によって異なります。各処理で確保される送信データサイズを次の表に示します。

表 C-1　各処理で確保される送信データサイズの一覧

項番	処理内容	送信データサイズの計算式（単位：バイト）
1	前処理および実行をした場合	*CMSG = clt_base_info + clt_exec_direct + clt_sql_text*
2	カーソルをオープンした場合	*CMSG = clt_base_info + clt_open + PARAM_DATA*
3	SQL 文を実行した場合	*CMSG = clt_base_info + clt_exec + PARAM_DATA*
4	前処理をした場合	*CMSG = clt_base_info + clt_prepare + clt_sql_text*

（凡例）

clt_base_info：送信データ基本情報

32 バイトを代入してください。

clt_exec_direct：前処理および実行固有情報

44 バイトを代入してください。

clt_sql_text：SQL テキストのサイズ

実行する SQL 文の長さ（単位：バイト）を代入してください。

clt_open：カーソルオープン固有情報

48 バイトを代入してください。

clt_exec：実行固有情報

48 バイトを代入してください。

clt_prepare：前処理固有情報

44 バイトを代入してください。

PARAM_DATA：？パラメタ情報

次に示す計算式から求めてください。

計算式（単位：バイト）

$$PARAM\_DATA = \left\{ \sum_{i=1}^{param\_num} param\_size(i) \right\} \times array\_num$$

param_num：SQL 文に指定した？パラメタ数

param_size（i）：各？パラメタ数に指定したデータサイズ

array_num：一括更新した？パラメタの組の数※

注※

APの実装方法によって異なります。

- CLI関数を使用したAPの場合

  a_rdb_SQLBindArrayParams()の引数ArrayCountに指定した数になります。
  a_rdb_SQLBindParams()を使用してパラメタ結合した場合は，1で固定されます。

- JDBCドライバを使用したAPの場合

  addBatchメソッドで登録したパラメタリストの数になります。executeBatchメソッドまたはexecuteLargeBatchメソッドを使用しない場合は，1で固定されます。

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

く

け

こ

ひ

ふ

へ

ほ

む

め

も

ゆ

よ

ら

り

れ

ろ

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