



リファレンス・マニュアル：ビルディング・ブロック

Adaptive Server[®] Enterprise

15.7

ドキュメント ID : DC36422-01-1570-01

改訂 : 2011 年 9 月

Copyright © 2012 by Sybase, Inc. All rights reserved.

このマニュアルは Sybase ソフトウェアの付属マニュアルであり、新しいマニュアルまたはテクニカル・ノートで特に示されないかぎり、後続のリリースにも付属します。このマニュアルの内容は予告なしに変更されることがあります。このマニュアルに記載されているソフトウェアはライセンス契約に基づいて提供されるものであり、無断で使用することはできません。

このマニュアルの内容を弊社の書面による事前許可を得ずに、電子的、機械的、手作業、光学的、またはその他のいかなる手段によっても、複製、転載、翻訳することを禁じます。

Sybase の商標は、**Sybase trademarks ページ** (<http://www.sybase.com/detail?id=1011207>) で確認できます。Sybase およびこのリストに掲載されている商標は、米国法人 Sybase, Inc. の商標です。® は、米国における登録商標であることを示します。

このマニュアルに記載されている SAP、その他の SAP 製品、サービス、および関連するロゴは、ドイツおよびその他の国における SAP AG の商標または登録商標です。

Java および Java 関連の商標は、米国およびその他の国における Sun Microsystems, Inc. の商標または登録商標です。

Unicode と Unicode のロゴは、Unicode, Inc. の登録商標です。

IBM および Tivoli は、International Business Machines Corporation の米国およびその他の国における登録商標です。

このマニュアルに記載されている上記以外の社名および製品名は、当該各社の商標または登録商標の場合があります。

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

はじめに	xi	
第 1 章	システム・データ型とユーザ定義データ型	1
データ型の種類	1	
範囲と記憶サイズ	2	
カラム、変数、パラメータのデータ型	4	
テーブル内のカラムのデータ型の宣言	4	
バッチまたはプロシージャ内のローカル変数のデータ型の宣言	5	
ストアド・プロシージャ内のパラメータのデータ型の宣言	5	
リテラルのデータ型の決定	5	
混合モードの式のデータ型	6	
データ型階層の決定	6	
精度と位取りの決定	8	
データ型の変換	8	
固定長 NULL カラムの自動変換	9	
オーバフロー・エラーとトランケーション・エラーの処理	9	
標準規格と準拠レベル	10	
真数値データ型	11	
整数値型	11	
decimal データ型	12	
標準規格と準拠レベル	14	
概数値データ型	14	
概数値データ型について	14	
範囲、精度、記憶サイズ	15	
概数値データの入力	15	
NaN 値と Inf 値	16	
標準規格と準拠レベル	16	
通貨データ型	16	
正確度	16	
範囲と記憶サイズ	16	
通貨値の入力	17	
標準規格と準拠レベル	17	
timestamp データ型	17	
timestamp カラムの作成	17	

日付と時刻のデータ型	18
範囲と記憶サイズ	19
日付および時刻データの入力	19
標準規格と準拠レベル	24
文字データ型	24
unichar、univarchar	24
文字単位の長さ	25
文字データの長さ	26
空文字列の処理	27
文字データの操作	28
標準規格と準拠レベル	28
バイナリ・データ型	29
binary と varbinary の有効な入力値	29
最大カラム・サイズよりも大きいエントリ	29
後続ゼロの扱い	29
プラットフォームの依存	31
標準規格と準拠レベル	31
bit データ型	31
標準規格と準拠レベル	31
sysname データ型および longsysname データ型	32
標準規格と準拠レベル	32
text、image、unitext データ型	32
text、unitext、および image データの保管に使用されるデータ構造	33
text、unitext、および image カラムの初期化	34
NULL によるスペースの節約	35
sysindexes からの情報取得	35
readtext と writetext の使用	36
カラムが使用する領域の決定	36
text、image、および unitext カラムの制限	37
text、unitext、image データの選択	37
text および image データ型の変換	38
unitext へ、または unitext からの変換	38
text データのパターン一致	39
重複ロー	39
ストアド・プロシージャにおけるラージ・オブジェクトの text、 unitext、image データ型の使用	39
標準規格と準拠レベル	41
データ型と暗号化カラム	42
ユーザ定義データ型	42
標準規格と準拠レベル	43

第 2 章	Transact-SQL 関数	45
	abs	46
	acos	47
	ascii	48
	asehostname	49
	asin	50
	atan	51
	atn2	52
	avg	53
	audit_event_name	55
	authmech	57
	biginttohex	58
	bintostr	59
	cache_usage	60
	case	61
	cast	64
	ceiling	67
	char	69
	char_length	71
	charindex	73
	coalesce	75
	col_length	77
	col_name	78
	compare	79
	convert	84
	cos	89
	cot	90
	count	91
	count_big	93
	create_locator	95
	current_bigdatetime	96
	current_bigtime	97
	current_date	98
	current_time	99
	curunreservedpgs	100
	data_pages	102
	datachange	104
	datalength	106
	dateadd	107
	datediff	110
	datename	114
	datepart	116
	day	120
	db_attr	121
	db_id	123
	db_instanceid	124

db_name	125
db_recovery_status	126
degrees	127
derived_stat	128
difference	133
dol_downgrade_check	134
exp	135
floor	136
get_appcontext	138
getdate	139
getutcdate	140
has_role	141
hash	143
hashbytes	145
hextobigint	147
hextoint	148
host_id	149
host_name	150
instance_id	151
identity_burn_max	152
index_col	153
index_colorder	154
index_name	155
instance_name	156
intohex	157
is_quiesced	158
is_sec_service_on	160
is_singleusermode	161
isdate	162
isnull	163
isnumeric	164
lc_id	165
lc_name	166
lct_admin	167
left	170
len	171
license_enabled	172
list_appcontext	173
locator_literal	174
locator_valid	175
lockscheme	176
log	177
log10	178
lower	179
lprofile_id	180
lprofile_name	181

ltrim.....	182
max.....	183
min.....	184
month.....	185
mut_excl_roles.....	186
newid.....	187
next_identity.....	189
nullif.....	190
object_attr.....	192
object_id.....	196
object_name.....	197
object_owner_id.....	198
pagesize.....	199
partition_id.....	201
partition_name.....	202
partition_object_id.....	203
patindex.....	204
pi.....	207
power.....	208
proc_role.....	209
pssinfo.....	211
radians.....	212
rand.....	213
rand2.....	214
replicate.....	215
reserve_identity.....	216
reserved_pages.....	219
return_lob.....	223
reverse.....	224
right.....	225
rm_appcontext.....	227
role_contain.....	228
role_id.....	229
role_name.....	230
round.....	231
row_count.....	233
rtrim.....	234
sdci_intempdbconfig.....	235
set_appcontext.....	236
setdata.....	238
show_cached_plan_in_xml.....	239
show_dynamic_params_in_xml.....	242
show_role.....	243
show_sec_services.....	244
sign.....	245
sin.....	246

sortkey.....	247
soundex.....	252
space.....	253
spid_instance_id.....	254
square.....	255
sqrt.....	256
stddev.....	257
stdev.....	258
stdevp.....	259
stddev_pop.....	260
stddev_samp.....	261
str.....	262
str_replace.....	264
strtobin.....	266
stuff.....	267
substring.....	269
sum.....	270
suser_id.....	272
suser_name.....	273
syb_quit.....	274
syb_sendmsg.....	275
sys_tempdbid.....	276
tan.....	277
tempdb_id.....	278
textptr.....	279
textvalid.....	280
to_unichar.....	281
tran_dumpable_status.....	282
tsequal.....	284
uhighsurr.....	286
ulowsurr.....	287
upper.....	288
uscalar.....	289
used_pages.....	290
user.....	292
user_id.....	293
user_name.....	294
valid_name.....	295
valid_user.....	296
var.....	297
var_pop.....	298
var_samp.....	299
variance.....	300
varp.....	301
workload_metric.....	302
xa_bqual.....	303

	xa_grid.....	305
	xact_connmigrate_check.....	307
	xact_owner_instance.....	308
	xmlextract.....	309
	xmlparse.....	310
	xmlrepresentation.....	311
	xmltable.....	312
	xmltest.....	313
	xmlvalidate.....	314
	year.....	315
第 3 章	グローバル変数.....	317
	Adaptive Server のグローバル変数.....	317
	クラスタ環境でのグローバル変数の使用.....	323
第 4 章	式、識別子、およびワイルドカード文字.....	325
	式.....	325
	式のサイズ.....	326
	算術式と文字式.....	326
	関係式と論理式.....	326
	演算子の優先度.....	327
	算術演算子.....	327
	ビット処理演算子.....	328
	文字列連結演算子.....	329
	比較演算子.....	330
	標準以外の演算子.....	330
	any、all、および in の使用.....	331
	否定と存在確認.....	331
	範囲.....	331
	式での null の使用.....	331
	式の接続.....	333
	式でのカッコの使用.....	334
	文字式の比較.....	334
	空文字列の使用.....	334
	文字式に含む引用符.....	334
	継続文字の使用.....	335
	識別子.....	335
	短い識別子.....	336
	# で始まるテーブル (テンポラリ・テーブル).....	337
	大文字と小文字の区別と識別子.....	337
	オブジェクト名の一意性.....	338
	区切り識別子の使用.....	338
	修飾されたオブジェクト名によるテーブルまたはカラムの識別.....	341
	識別子の有効性の確認.....	343
	データベース・オブジェクト名の変更.....	343
	マルチバイト文字セットの使用.....	343

	like パターン一致	344
	ワイルドカード文字によるパターン一致	344
	like パターン一致	345
	not like の使用	346
	大文字と小文字の区別とアクセントの区別の無視	347
	ワイルドカード文字の使用	347
	マルチバイト・ワイルドカード文字の使用	349
	リテラル文字としてのワイルドカード文字の使用	349
	datetime データでのワイルドカード文字の使用	351
第 5 章	予約語	353
	Transact-SQL 予約語	353
	ANSI SQL 予約語	354
	ANSI SQL の予約語となる可能性のあるワード	356
第 6 章	SQLSTATE コードとメッセージ	357
	警告	357
	例外	358
	基本的な違反	358
	データ例外	359
	整合性の制約違反	359
	無効なカーソル状態	360
	構文エラーおよびアクセス・ルール違反	361
	トランザクション・ロールバック	362
	with check option 違反	362
	索引	363

はじめに

『ASE リファレンス・マニュアル』は、Sybase[®] Adaptive Server[®] Enterprise と Transact-SQL[®] 言語について説明する 4 巻構成のマニュアルです。

- 『ビルディング・ブロック』では、データ型、組み込み関数、グローバル変数、式と識別子、予約語、SQLSTATE エラーなど、Transact-SQL の各要素について説明します。Transact-SQL を使用する前に、ビルディング・ブロックの動作とビルディング・ブロックが Transact-SQL 文の結果にどのように影響するかを理解する必要があります。
- 『コマンド』では、文の作成に使用する Transact-SQL コマンドについてのリファレンス情報が記載されています。
- 『プロシージャ』では、システム・プロシージャ、カタログ・ストアド・プロシージャ、拡張ストアド・プロシージャ、**dbcc** ストアド・プロシージャについてのリファレンス情報が記載されています。すべてのプロシージャは、Transact-SQL 文を使用して作成されます。
- 『テーブル』では、使用しているサーバ、データベース、ユーザ、サーバの他の詳細についての情報を格納するシステム・テーブルに関するリファレンス情報が記載されています。**dbcc** データベースと **dbccalt** データベースのテーブルについても説明します。

規則

次の項では、このマニュアルで使用されている表記について説明します。

SQL は自由な形式の言語で、1 行内のワード数や、改行の仕方に規則はありません。このマニュアルでは、読みやすくするため、例や構文を文の句ごとに改行しています。複数の部分からなり、2 行以上にわたる場合は、字下げしています。複雑なコマンドの書式には、修正された BNF (Backus Naur Form) 記法が使用されています。

表 1 に構文の規則を示します。

表 1: このマニュアルでのフォントと構文規則

要素	例
コマンド名、プロシージャ名、ユーティリティ名、データベース名、データ型、その他のキーワードは sans serif フォントで表記する。	<code>select</code> <code>sp_configure</code> <code>master</code> データベース
ファイル名、変数、パス名は斜体で表記する。	システム管理ガイド <code>sql.ini</code> file <code>column_name</code> \$SYBASE/ASE ディレクトリ
変数 (ユーザが入力する値を表す語) がクエリまたは文の一部である場合は Courier フォントの斜体で表記する。	<code>select column_name</code> <code>from table_name</code> <code>where search_conditions</code>
カッコはコマンドの一部として入力する。	<code>compute row_aggregate (column_name)</code>
2つのコロンと等号は、構文が BNF 表記で記述されていることを示す。この記号は入力しない。「~と定義されている」ことを意味する。	::=
中カッコは、その中のオプションを1つ以上選択しなければならないことを意味する。コマンドには中カッコは入力しない。	{cash, check, credit}
角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。	[cash check credit]
中カッコまたは角カッコの中のカンマで区切られたオプションをいくつでも選択できることを意味する。複数のオプションを選択する場合には、オプションをカンマで区切る。	cash, check, credit
パイプまたは縦線は複数のオプションのうち1つだけを選択できることを意味する。	cash check credit
省略記号 (...) は、直前の要素を必要な回数だけ繰り返し指定できることを意味する。	<code>buy thing = price [cash check credit]</code> <code>[, thing = price [cash check credit]]...</code> この例では、製品 (thing) を少なくとも1つ購入 (buy) し、価格 (price) を指定する必要があります。支払方法を角カッコの中から1つ選択できます。さらに、他の製品を購入することもできます。各 buy に対して、購入した製品 (thing)、価格 (price)、オプションで支払方法 (cash、check、credit のいずれか) を指定します。

- 次は、オプション句のあるコマンドの構文の例です。

`sp_dropdevice [device_name]`

複数のオプションを持つコマンドの例を示します。

```
select column_name
from table_name
where search_conditions
```

構文では、キーワード (コマンド) は通常のフォントで表記し、識別子は小文字で表記します。ユーザが提供するワードは斜体で表記します。

- Transact-SQL コマンドの使用例は次のように表記します。

```
select * from publishers
```

- 次は、コンピュータからの出力例です。

```
pub_id      pub_name                city                state
-----      -
0736        New Age Books           Boston              MA
0877        Binnet & Hardley        Washington          DC
1389        Algodata Infosystems  Berkeley            CA
```

```
(3 rows affected)
```

このマニュアルでは、例に使用する文字はほとんどが小文字ですが、Transact-SQL のキーワードを入力するときは、大文字と小文字は区別されません。たとえば、**SELECT**、**Select**、**select** はすべて同じです。

テーブル名などのデータベース・オブジェクトの大文字と小文字を Adaptive Server が区別するかどうかは、Adaptive Server にインストールされたソート順によって決まります。シングルバイト文字セットを使用している場合は、Adaptive Server のソート順を再設定することによって、大文字と小文字の区別の取り扱い方を変更できます。詳細については、『システム管理ガイド』を参照してください。



この章では、Transact-SQL データ型について説明します。データ型は、タイプ、サイズ、およびカラム、ストアド・プロシージャ・パラメータ、ローカル変数の記憶フォーマットを指定します。

トピック名	ページ
データ型の種類	1
範囲と記憶サイズ	2
カラム、変数、パラメータのデータ型	4
混合モードの式のデータ型	6
データ型の変換	8
標準規格と準拠レベル	10
真数値データ型	11
概数値データ型	14
通貨データ型	16
timestamp データ型	17
日付と時刻のデータ型	18
文字データ型	24
バイナリ・データ型	29
bit データ型	31
sysname データ型および longsysname データ型	32
text、image、unitext データ型	32
データ型と暗号化カラム	42
ユーザ定義データ型	42

データ型の種類

Adaptive Server には、数種類のシステム・データ型とユーザ定義のデータ型 (timestamp、sysname、および longsysname) が用意されています。表 1-1 は、Adaptive Server のデータ型の種類のリストを示します。それぞれの種類については、この章の各項で説明します。

表 1-1: データ型の種類

種類	用途
真数値データ型	正確に表示する必要のある数値 (整数値および小数値)
概数値データ型	算術演算中に丸めることができる数値データ
通貨データ型	通貨データ
timestamp データ型	Client-Library™ アプリケーションでブラウズするテーブル
日付と時刻のデータ型	日付および時刻の情報
文字データ型	文字、数字、記号で構成される文字列
バイナリ・データ型	ピクチャなどの、16 進数に似た表記のバイナリ・データ
bit データ型	true/false (真/偽) および yes/no 型のデータ
sysname データ型および longsysname データ型	システム・テーブル
text、image、unitext データ型	サーバの論理ページ・サイズの最大カラム・サイズよりも大きいカラムを必要とする、印刷可能な文字または 16 進数に似たデータ
抽象データ型	Adaptive Server は、Java クラスを介して抽象データ型をサポートする。詳細については、『Adaptive Server Enterprise における Java』を参照。
ユーザ定義データ型	この表にリストされているデータ型のルール、デフォルト、null 型、IDENTITY プロパティ、および基本データ型を継承するオブジェクトを定義する。クライアントで別の文字セットが使用されている場合に text には文字セットの変換が行われるが、image には行われない。

範囲と記憶サイズ

表 1-2 に、システム定義データ型とその同義語をリストし、各データ型の有効値の範囲と記憶サイズを示します。この表ではわかりやすくするために、データ型はすべて小文字で表記してありますが、Adaptive Server では、システム・データ型は大文字、小文字のどちらでも入力できます。timestamp などのユーザ定義データ型では大文字と小文字が区別されます。Adaptive Server のほとんどのデータ型は予約語ではないため、他のオブジェクトの名前として使用できます。

表 1-2: Adaptive Server のシステム・データ型

データ型 (種類別)	同義語	範囲	記憶サイズ (バイト数)
真数値：整数			
bigint		$2^{63} \sim -2^{63} - 1$ (-9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,807) の間の整数値	8
int	integer	$2^{31} - 1$ (2,147,483,647) ~ -2^{31} (-2,147,483,648)	4
smallint		$2^{15} - 1$ (32,767) ~ -2^{15} (-32,768)	2
tinyint		0 ~ 255 (負の数は使用できない)	1

データ型 (種類別)	同義語	範囲	記憶サイズ (バイト数)
unsigned bigint		0 ~ 18,446,744,073,709,551,615 の間の 整数値	8
unsigned int		0 ~ 4,294,967,295 の間の整数値	4
unsigned smallint		0 ~ 65535 の間の整数値	2
真数値：小数			
numeric (p, s)		$10^{38} - 1 \sim -10^{38}$	17 ~ 2
decimal (p, s)	dec	$10^{38} - 1 \sim -10^{38}$	17 ~ 2
概数値			
float (precision)		マシンに依存する	default precision が 16 未満 の場合は 4、 default precision が 16 以上 の場合は 8
double precision		マシンに依存する	8
real		マシンに依存する	4
通貨			
smallmoney		-214,748.3647 ~ 214,748.3648	4
money		922,337,203,685,477.5807 ~ -922,337,203,685,477.5808	8
日付／時刻			
smalldatetime		1900 年 1 月 1 日 ~ 2079 年 6 月 6 日	4
datetime		1753 年 1 月 1 日 ~ 9999 年 12 月 31 日	8
date		0001 年 1 月 1 日 ~ 9999 年 12 月 31 日	4
time		12:00:00AM ~ 11:59:59:990PM	4
bigdatetime		0001 年 1 月 1 日 ~ 9999 年 12 月 31 日 および 12:00.000000AM ~ 11:59:59.999999 PM	8
bigtime		12:00:00.000000 AM ~ 11:59:59.999999 PM	8
文字			
char(n)	character	ページサイズ	n
varchar(n)	character varying, char varying	ページサイズ	実際のエントリの長さ
unichar	Unicode 文字	ページサイズ	$n * @@unicharsize$ ($@@unicharsize$ は 2)
univarchar	Unicode 文字 varying, char varying	ページサイズ	実際の文字数 * $@@unicharsize$
nchar(n)	national character, national char	ページサイズ	$n * @@ncharsize$
nvarchar(n)	nchar varying, national char varying, national character varying	ページサイズ	$@@ncharsize * 文字数$

データ型 (種類別)	同義語	範囲	記憶サイズ (バイト数)
text		$2^{31} - 1$ (2,147,483,647) バイト以下	初期化前は 0、初期化後は 2K の倍数
unitext		1 ~ 1,073,741,823	初期化前は 0、初期化後は 2K の倍数
バイナリ			
binary(n)		ページサイズ	<i>n</i>
varbinary(n)		ページサイズ	実際のエントリの長さ
image		$2^{31} - 1$ (2,147,483,647) バイト以下	初期化前は 0、初期化後は 2K の倍数
ビット			
bit		0 または 1	1 (1 バイトで 8 つまでの bit カラムを保持する)

カラム、変数、パラメータのデータ型

カラム、ローカル変数、パラメータについては、データ型を宣言する必要があります。システムが提供するデータ型またはデータベース内のユーザ定義データ型の中から、任意のデータ型を宣言することができます。

テーブル内のカラムのデータ型の宣言

`create table` 文または `alter table` 文で新しいカラムのデータ型を宣言するには、次の構文を使用します。

```
create table [[database.]owner.]table_name
    (column_name datatype [identity | not null | null]
    [, column_name datatype [identity | not null |
    null]]...)

alter table [[database.]owner.]table_name
    add column_name datatype [identity | null]...
    [, column_name datatype [identity | null]]...
```

たとえば、次のようにします。

```
create table sales_daily
    (stor_id char(4) not null,
    ord_num numeric(10,0) identity,
    ord_amt money null)
```

また、`select into` 文で `convert` または `cast` を使用して、新しいカラムのデータ型を宣言することもできます。

```
select convert(double precision, x), cast ( int, y) into
    newtable from oldtable
```

バッチまたはプロシージャ内のローカル変数のデータ型の宣言

バッチまたはプロシージャ内でローカル変数のデータ型を宣言するには、次の構文を使用します。

```
declare @variable_name datatype
        [, @variable_name datatype ]...
```

たとえば、次のようにします。

```
declare @hope money
```

ストアド・プロシージャ内のパラメータのデータ型の宣言

ストアド・プロシージャ内のパラメータのデータ型を宣言するには、次の構文を使用します。

```
create procedure [owner.]procedure_name [;number]
        [[(@parameter_name datatype [= default] [output]
        [, @parameter_name datatype [= default]
        [output]]...[])]
[with recompile]
as SQL_statements
```

次に例を示します。

```
create procedure auname_sp @auname varchar(40)
as
    select au_lname, title, au_ord
    from authors, titles, titleauthor
    where @auname = au_lname
    and authors.au_id = titleauthor.au_id
    and titles.title_id = titleauthor.title_id
```

リテラルのデータ型の決定

数字リテラル

E 表記で入力された数字リテラルは `float` として扱われます。そのほかのリテラルはすべて、真数値として扱われます。

- 小数点のない $2^{31} - 1 \sim -2^{31}$ のリテラルは、`integer` として扱われます。
- 小数点のあるリテラルまたは整数値の範囲外にあるリテラルは、`numeric` として扱われます。

注意 下位互換性を保つために、`float` として扱う必要のある数字リテラルには、E 表記を使用してください。

文字リテラル

12.5.1 より前のバージョンの Adaptive Server では、クライアントの文字セットがサーバの文字セットと異なる場合、SQL クエリのテキストを処理する前にサーバの文字セットに変換できるように、通常、変換が有効になっていました。サーバの文字セットで表せないために変換できない文字がある場合は、クエリ全体が拒否されました。Adaptive Server バージョン 12.5.1 からは、この文字セットの「ボトルネック」が解消されました。

文字リテラルのデータ型は宣言できません。Adaptive Server では、文字リテラルは `varchar` として扱われます。ただし、サーバのデフォルト文字セットに変換できない文字を含む文字リテラルは例外です。このようなりテラルは `univarchar` として扱われます。これにより、“sjis” (日本語) クライアントを使用して “iso_1” 用に設定されたサーバ内の `unichar` データを選択するクエリなどを実行できます。次に例を示します。

```
select * from mytable where unichar_column = '五'
```

文字リテラルは `char` データ型を使用して表せないため (“iso_1” の場合)、`unichar` データ型に拡大し、これによってクエリが成功します。

混合モードの式のデータ型

データ型の異なる値の連結または混合モードの算術演算を実行する場合、Adaptive Server によって結果のデータ型、長さ、精度が決定されます。

データ型階層の決定

各システム・データ型には、「データ型の階層」があります。この階層は `systypes` システム・テーブルに保管されています。ユーザ定義データ型は、もともっているシステム・データ型の階層を継承します。

次のクエリは、データベースのデータ型を階層でランク付けします。クエリ結果には、次に示す情報のほかに、データベース内のユーザ定義データ型に関する情報も含まれます。

```
select name, hierarchy
       from systypes
       order by hierarchy

name          hierarchy
-----
floatn        1
float         2
datetimn     3
datetime     4
```

real	5
numericn	6
numeric	7
decimaln	8
decimal	9
moneyn	10
money	11
smallmoney	12
smalldatet	13
intn	14
uintn	15
bigint	16
ubigint	17
int	18
uint	19
smallint	20
usmallint	21
tinyint	22
bit	23
univarchar	24
unichar	25
unitext	26
sysname	27
varchar	27
nvarchar	27
longsysnam	27
char	28
nchar	28
timestamp	29
varbinary	29
binary	30
text	31
image	32
date	33
time	34
daten	35
timen	36
bigdatetime	37
bigtime	38
bigdatetimen	39
bigtimen	40
xml	41
extended time	99

注意 `u<int 型>` は内部で使用される表現です。符号なし型の正しい構文は、`unsigned {int | integer | bigint | smallint}` です。

データ型階層は、異なるデータ型の値を使用して計算の結果を判別します。結果の値には、この階層リストの最上位に最も近い、つまり階層の値が最も小さいデータ型が割り当てられます。

次の例では、`sales` テーブルの `qty` に、`roysched` テーブルの `royalty` を乗算しています。`qty` は `smallint` であり、階層は 20 です。`royalty` は `int` であり、階層は 18 です。したがって、結果のデータ型は `int` になります。

```
smallint(qty) * int(royalty) = int
```

精度と位取りの決定

`numeric` データ型と `decimal` データ型の場合、精度と位取りの各組み合わせがそれぞれ別の Adaptive Server データ型になります。次の 2 つの `numeric` 値または `decimal` 値を対象に算術演算を行うとします。

- 精度が $p1$ 、位取りが $s1$ の値 $n1$
- 精度が $p2$ 、位取りが $s2$ の値 $n2$

この場合、Adaptive Server によって決定される結果の精度と位取りを [表 1-3](#) に示します。

表 1-3: 算術演算後の精度と位取り

演算	精度	位取り
$n1 + n2$	$\max(s1, s2) + \max(p1 - s1, p2 - s2) + 1$	$\max(s1, s2)$
$n1 - n2$	$\max(s1, s2) + \max(p1 - s1, p2 - s2) + 1$	$\max(s1, s2)$
$n1 * n2$	$s1 + s2 + (p1 - s1) + (p2 - s2) + 1$	$s1 + s2$
$n1 / n2$	$\max(s1 + p2 + 1, 6) + p1 - s1 + p2$	$\max(s1 + p2 - s2 + 1, 6)$

データ型の変換

あるデータ型から別のデータ型への変換は、多くの場合、Adaptive Server によって自動的に処理されます。このような自動変換を暗黙的な変換といいます。ほかの変換は、`convert` 関数、`hextoint` 関数、`intohex` 関数、`hextobigint` 関数、および `biginttohex` 関数を使用して明示的に要求する必要があります。Adaptive Server でサポートしているデータ型の変換については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

固定長 NULL カラムの自動変換

null 値を格納できるのは、可変長データ型のカラムだけです。固定長データ型の NULL カラムを作成すると、Adaptive Server によって、そのカラムは対応する可変長データ型に自動的に変換されます。Adaptive Server は、データ型が変更されたことをユーザに通知しません。

表 1-4 は、固定長データ型とその変換後の可変長データ型をリストしたものです。money, などの一部の可変長データ型は予約データ型です。カラム、変数、またはパラメータを作成するときにはこの予約データ型を使用できません。

表 1-4: 固定長データ型の自動変換

変換前の固定長データ型	変換後のデータ型
char	varchar
unichar	univarchar
nchar	nvarchar
binary	varbinary
datetime	datetimn
date	daten
time	timen
float	floatn
bigint, int, smallint, tinyint	intn
unsigned bigint, unsigned int, unsigned smallint	uintn
decimal	decimaln
numeric	numericn
money, smallmoney	moneyn

オーバフロー・エラーとトランケーション・エラーの処理

算術演算エラーが発生した場合の Adaptive Server の動作を指定するには、arithabort オプションを使用します。2 つの arithabort オプションである arithabort arith_overflow と arithabort numeric_truncation は、異なる 2 つのタイプの算術演算エラーを取り扱います。各オプションを別々に設定したり、1 つの set arithabort on 文または set arithabort off 文に両オプションを設定したりできます。

- arithabort arith_overflow には、明示または暗黙のデータ型変換中に 0 による除算エラーや精度ロスが発生したあとの動作を指定します。このタイプのエラーは重大なエラーと見なされます。デフォルト設定の arithabort arith_overflow on では、エラーが発生したトランザクション全体がロールバックされます。トランザクションを含まないバッチでエラーが発生した場合、arithabort arith_overflow on はバッチ内のエラーより前のコマンドはロールバックしませんが、Adaptive Server はバッチ内のエラーを生成した文より後の文を実行しません。

`arith_overflow on` の設定は、Adaptive Server の標準として設定されるのではなく、実行時に適用されます。

`arithabort arith_overflow off` を設定した場合には、Adaptive Server はエラーを発生させた文をアボートしますが、トランザクションまたはバッチ内の残りの文の処理を継続します。

- `arithabort numeric_truncation` は、暗黙のデータ型変換中に真数値型による位取りのロスが発生した後の動作を指定します(明示的変換によって位取りのロスが発生すると、変換結果は警告なしにトランケートされます)。デフォルト設定の `arithabort numeric_truncation on` は、エラーを起こした文をアボートしますが、トランザクションまたはバッチ内のその他の文の処理を継続します。`arithabort numeric_truncation off` を設定すると、Adaptive Server はクエリ結果をトランケートして、処理を継続します。

`arithignore` オプションは、オーバフロー・エラー発生後に Adaptive Server が警告メッセージを出力するかどうかを指定します。デフォルトでは、`arithignore` オプションは `off` になります。この場合、Adaptive Server は数字オーバフローを起こしたクエリの後に、警告メッセージを表示します。オーバフロー・エラーを無視するには、`set arithignore on` を使用してください。

標準規格と準拠レベル

表 1-5 は、Transact-SQL データ型の ANSI SQL 標準規格と準拠レベルを示しています。

表 1-5: Transact-SQL データ型の ANSI SQL 標準規格と準拠レベル

Transact-SQL – ANSI SQL データ型	Transact-SQL 拡張機能 – ユーザ定義データ型
<ul style="list-style-type: none"> • char • varchar • smallint • int • bigint • decimal • numeric • float • real • date • time • double precision 	<ul style="list-style-type: none"> • binary • varbinary • bit • nchar • datetime • smalldatetime • bigdatetime • bigtime • tinyint • unsigned smallint • unsigned int • unsigned bigint • money • smallmoney • text • unitext • image • nvarchar • unichar • univarchar • sysname • longsysname • timestamp

真数値データ型

値を正確に表す必要がある場合に、真数値データ型を使用します。Adaptive Server には、整数 (自然数) 値と小数値のための真数値データ型があります。

整数値型

Adaptive Server には、整数を保管するための真数値データ型 `bigint`、`int` (integer)、`smallint`、`tinyint`、およびこれらの各データ型の符号なしの型があります。保管する数値の予想サイズに基づいて、整数値型を選択してください。表 1-6 に示すように、内部記憶サイズは型によって異なります。

表 1-6: 整数値データ型

データ型	保管するデータ	記憶サイズ (バイト数)
bigint	$-2^{63} \sim 2^{63} - 1$ (-9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,807) の間の整数値	8
int[eger]	$-2^{31} \sim 2^{31} - 1$ (-2,147,483,648 ~ 2,147,483,647) の間の整数値	4
smallint	$-2^{15} \sim 2^{15} - 1$ (-32,768 ~ 32,767) の間の整数値	2
tinyint	0 ~ 255 の間の整数値 (負の数値は使用できない)	1
unsigned bigint	0 ~ 18,446,744,073,709,551,615 の間の整数値	8
unsigned int	0 ~ 4,294,967,295 の間の整数値	4
unsigned smallint	0 ~ 65,535 の間の整数値	2

整数データの入力

整数データは、カンマを含まない数字列として入力します。小数点の右側のすべての桁がゼロであれば、整数データに小数点を含めることができます。smallint、integer、bigint データ型の数値の前に、オプションでプラス符号またはマイナス符号を付けることができます。tinyint データ型の数値の前に、オプションでプラス符号を付けることができます。

表 1-7 に、integer データ型のコラムに有効な入力値と、isql がこれらの値をどのように表示するかを示します。

表 1-7: 有効な整数値

入力値	表示値
2	2
+2	2
-2	-2
2.	2
2.000	2

表 1-8 に、integer コラムの無効な入力値を示します。

表 1-8: 無効な整数値

入力値	エラー・タイプ
2,000	カンマは使用できない。
-2	マイナス符号は数字の前になければならない。
3.45	小数点の右側の数字がゼロ以外の数字である。

decimal データ型

Adaptive Server では、上記のほかに、小数点を含む数値のための 2 つの真数値データ型 numeric と dec[imal] が用意されています。numeric と decimal の 2 つのデータ型は、IDENTITY コラムには位取り 0 の numeric データ型と integer データ型だけが使用できるという点を除いて、同一です。

精度と位取りの指定

`numeric` と `decimal` データ型は、2 つのオプション・パラメータである `precision` と `scale` を受け入れます。これらのパラメータを指定するときには、パラメータをカッコで囲み、カンマで区切ります。

```
datatype [(precision [, scale])]
```

Adaptive Server は、精度と位取りの各組み合わせを別個のデータ型として扱います。たとえば `numeric(10,0)` と `numeric(5,0)` は、2 つの別々のデータ型です。`precision` と `scale` は、`decimal` カラムまたは `numeric` カラムに保管できる値の範囲を決定します。

- 精度 (`precision`) は、カラムに格納できる 10 進数値の最大桁数を指定します。この桁数には、小数点の左右両側にあるすべての桁が含まれます。1 桁から 38 桁までの精度を指定するか、またはデフォルトの精度である 18 桁を使用します。
- 位取りは、小数点の右側に格納できる最大桁数を指定します。位取りの桁数は精度の桁数以下でなければなりません。0 桁から 38 桁までの位取りを指定するか、またはデフォルトの位取りである 0 桁を使用します。

記憶サイズ

`numeric` または `decimal` カラムの記憶サイズは、その精度によって異なります。1 桁または 2 桁のカラムの場合、記憶領域は 2 バイト必要です。記憶サイズは精度が 2 桁増加するたびに約 1 バイトずつ増加します。最大記憶サイズは 17 バイトです。

`numeric` または `decimal` カラムの正確な記憶サイズを計算するには、次の式を使います。

```
ceiling (precision / log10(256)) + 1
```

たとえば、`numeric(18,4)` カラムの記憶サイズは 9 バイトです。

10 進数データの入力

`decimal` データおよび `numeric` データは、数字列として入力します。このとき、オプションでプラス符号またはマイナス符号を数値の前に付けたり、オプションで小数点を含めたりできます。入力された値がカラムに指定されている精度または位取りを超えている場合、Adaptive Server はエラー・メッセージを返します。位取りが 0 の真数型は小数点なしで表示されます。

表 1-9 に、データ型が `numeric(5,3)` のカラムに有効な入力値と、`isql` によってこれらの値がどのように表示されるかを示します。

表 1-9: 有効な 10 進数値

入力値	表示値
12.345	12.345
+12.345	12.345
-12.345	-12.345
12.345000	12.345
12.1	12.100
12	12.000

表 1-10 に、データ型が `numeric(5,3)` のカラムに無効な入力値を示します。

表 1-10: 無効な 10 進値

入力値	エラー・タイプ
1,200	カンマは使用できない。
12-	マイナス符号は数字の前になければならない。
12.345678	小数点の右側にゼロ以外の数字が多すぎる。

標準規格と準拠レベル

Transact-SQL には、ANSI SQL 真数値データ型の `smallint`、`int`、`bigint`、`numeric`、`decimal` があります。unsigned `bigint`、unsigned `int`、unsigned `smallint`、`tinyint` 型は、Transact-SQL 拡張機能です。

概数値データ型

丸めることができる数値データには、概数値型の `float`、`double precision`、`real` を使用できます。概数値型は、値が広範囲にわたるデータに特に適しています。概数値型は、すべての集合関数とすべての算術演算をサポートします。

概数値データ型について

概数値データ型は、浮動小数点の値を保管するために使われるもので、実数表現では多少不正確になります。このために「概数値」と呼ばれます。このデータ型を使う場合には、その制限事項をよく理解してください。

浮動小数点の値の出力や表示のときには、出力された値は保管された値と正確には一致しません。また、保管された値は、ユーザが入力した値とは正確には一致しません。多くの場合、保管された値は十分に正確なものであり、ソフトウェアで出力された値は最初に入力された値と同じように表示されます。しかし浮動小数点を計算に使う場合、特に概数値を使って繰り返し計算する場合には、その結果が不正確なものになることを理解しておいてください。計算結果は、予想外に不正確なものになることがあります。

結果が不正確となるのは、コンピュータでは浮動小数点はバイナリ分数として（つまり、表示された値を 2 の累乗で割った値として）保管されますが、実際に使用される値は、10 進数値（10 の累乗の値）であるためです。このため、ごくわずかな数の数値の集合しか、正確に保管できないこととなります。0.75 (3/4) はバイナリ分数なので、正確に保管することができます（4 は 2 の累乗です）。しかし 0.2 (2/10) は正確に保管することはできません（10 は 2 の累乗ではありません）。

桁数が多すぎて、正確に保管できない数値もあります。double precision は 8 バイナリ・バイトで保管され、かなり正確に約 17 桁の数値を表示できます。real は 4 バイナリ・バイトで保管され、かなり正確に表示できるのは、約 6 桁までです。

ほぼ正確な数値を使って作業を始め、別のほぼ正確な数値を使って計算しても、まったく予想外の結果になることがあります。アプリケーションにおいてこの問題が重要な場合には、真数値データ型を使用してください。

範囲、精度、記憶サイズ

real および double precision 型は、オペレーティング・システムが提供する型を基に構築されます。float データ型には、オプションのバイナリ精度をカッコで囲んで指定できます。精度が 1 ~ 15 の float カラムは、real として格納されます。これより高い精度の場合は、double precision として格納されます。

3 つのデータ型のいずれの場合も、範囲および記憶精度はマシンによって異なります。

表 1-11 に、個々の概数値型の範囲と記憶サイズを示します。isql は、小数点以下の有効桁数を 6 桁しか表示せず、残りを丸めます。

表 1-11: 概数値データ型

データ型	記憶サイズ (バイト数)
float[(default precision)]	default precision < 16 の場合、4 default precision >= 16 の場合、8
double precision	8
real	4

概数値データの入力

概数値データは、仮数の後ろにオプションの指数を付けて入力します。

- 仮数は符号付き、または符号なしの数字です。小数点はあってもなくてもかまいません。カラムのバイナリ精度は、仮数部に使用できるバイナリ桁数の最大数を決定します。
- “e” または “E” で始まる指数は、整数です。

入力値が表す値は次の式の積です。

$$\text{仮数} * 10^{\text{指数}}$$

たとえば、2.4E3 は値 2.4×10^3 、すなわち 2400 です。

NaN 値と Inf 値

“NaN” と “Inf” は、IEEE754/854 浮動小数点標準でそれぞれ “not a number” (数値以外) と “infinity” (無限大) を表すために使用される特殊な値です。ANSI SQL92 標準規格に従い、Adaptive Server バージョン 12.5 以降では、これらの値をデータベースに挿入することも、生成することもできません。Adaptive Server 12.5 より前のバージョンでは、ネイティブ・モードの bcp、JDBC、ODBC などの Open Client クライアントによってこれらの値が強制的にテーブルに格納される場合があります。

データベースで NaN 値または Inf 値が検出された場合は、その問題の再現方法に関する詳細を含めて Sybase カスタマ・サポートまでご連絡ください。

標準規格と準拠レベル

ANSI SQL - 準拠レベル：float、double precision、real データ型は初級レベルの標準に準拠します。

通貨データ型

通貨データを保管するには、money データ型と smallmoney データ型を使用します。これらのデータ型は米国ドルなどの 10 進数値の通貨に使用できますが、Adaptive Server には、ある通貨を別の通貨に変換する機能はありません。money および smallmoney データには、modulo を除く算術演算と、すべての集合関数を使用できます。

正確度

money および smallmoney は、いずれも通貨単位の 10,000 分の 1 まで正確ですが、表示される場合は、値が小数点以下 2 桁まで丸められます。デフォルトの出力フォーマットでは、3 桁ごとにカンマが挿入されます。

範囲と記憶サイズ

表 1-12 に、通貨データ型の範囲と記憶サイズを示します。

表 1-12: 通貨データ型

データ型	範囲	記憶サイズ (バイト数)
money	+922,337,203,685,477.5807 と -922,337,203,685,477.5808 の間の通貨値 -922,337,203,685,477.5808	8
smallmoney	+214,748.3647 と 214,748.3648 の間の通貨値	4

通貨値の入力

E 表記で入力した通貨値は `float` として解釈されます。このため、値が `money` または `smallmoney` 値として格納されると、入力が拒否されたり、精度が若干失われたりすることがあります。

`money` および `smallmoney` 値を入力するときには、ドル記号 (\$)、円記号 (¥)、通貨ポンド記号 (£) などの通貨記号を数値の前に付けても、付けなくてもかまいません。負の値を入力する場合には、通貨記号の後にマイナス符号を入力してください。入力にはカンマは含めないでください。

標準規格と準拠レベル

ANSI SQL – `money` データ型および `smallmoney` データ型は Transact-SQL 拡張機能です。

timestamp データ型

Client-Library™ アプリケーションでブラウズするテーブル (詳細については「ブラウズ・モード」を参照) には、ユーザ定義データ型の `timestamp` を使用します。Adaptive Server は、ローが変更されるたびに、その `timestamp` カラムを更新します。各テーブルには、`timestamp` データ型のカラムを 1 つだけ含めることができます。

timestamp カラムの作成

`timestamp` という名前のカラムを作成し、このカラムにデータ型を指定しないと、自動的にこのカラムが `timestamp` データ型として定義されます。

```
create table testing
(c1 int, timestamp, c2 int)
```

次の例に示すように、`timestamp` という名前のカラムに `timestamp` データ型を明示的に割り当てることもできます。

```
create table testing
(c1 int, timestamp timestamp, c2 int)
```

カラム名は他の名前でもかまいません。

```
create table testing
(c1 int, t_stamp timestamp, c2 int)
```

`timestamp` という名前のカラムを作成し、別のデータ型を割り当ててもかまいません (ただし、これは他のユーザにとってはまぎらわしく、Open Client™ の `browse` 関数や `tsequal` 関数を使用することはできません)。

```
create table testing
(c1 int, timestamp datetime)
```

日付と時刻のデータ型

絶対日付と時刻の情報を保管するには、`datetime`、`smalldatetime`、`bigdatetime`、`bigtime`、`date`、および `time` を使用します。バイナリ型の情報を保管するには、`timestamp` を使用します。

Adaptive Server には、日付と時刻の値の保管に使用するさまざまなデータ型が用意されています。

- `date`
- `time`
- `smalldatetime`
- `datetime`
- `bigdatetime`
- `bigtime`

日付のデフォルトの表示フォーマットは、“Apr 15 1987 10:23PM”です。

`bigdatetime/bigtime` 型のデフォルトの表示フォーマットは、“Apr 15 1987 10:23:00.000000PM”です。他のスタイルの日付の表示には、`convert` 関数を使用できます。組み込み日付関数を使用して、`date` 値および `time` 値についていくつかの算術計算を実行することもできますが、Adaptive Server はミリ秒の値を丸めるか、またはトランケートする場合があります。

- `datetime` カラムは、1753 年の 1 月 1 日から 9999 年 12 月 31 日までの日付を保持します。`datetime` 値は、プラットフォームの能力が対応可能であれば、300 分の 1 秒のレベルまで正確です。小数秒の最後の桁は常に 0、3、または 6 です。その他の桁はこれらの 3 つの数字のいずれかに丸められます。そのため、0 と 1 は 0 に、2 と 3 と 4 は 3 に、5 と 6 と 7 と 8 は 6 に、9 は 10 に丸められます。記憶サイズは 8 バイトです。基本の日付である 1900 年 1 月 1 日以降の日数に 4 バイト、1 日の時刻に 4 バイトを使用します。
- `smalldatetime` カラムは、1900 年 1 月 1 日から 2079 年 6 月 6 日までの日付を、分の単位まで正確に保持します。記憶サイズは 4 バイトです。1900 年 1 月 1 日以降の日数に 2 バイト、夜中の 12 時以降の分数に 2 バイトを使用します。
- `bigdatetime` カラムは、0001 年 1 月 1 日から 9999 年 12 月 31 日までの日付および 12:00:00.000000 AM から 11:59:59.999999 PM までの時刻を保持します。記憶サイズは 8 バイトです。`bigdatetime` の内部で使用される表現は、0000 年 1 月 1 日以降のマイクロ秒数を含む 64 ビットの整数です。
- `bigtime` カラムは、12:00:00.000000 AM から 11:59:59.999999 PM までの時刻を保持します。記憶サイズは 8 バイトです。`bigtime` の内部で使用される表現は、午前 0 時以降のマイクロ秒数を含む 64 ビットの整数です。
- `date` カラムは、0001 年 1 月 1 日から 9999 年 12 月 31 日までの日付を保持します。記憶サイズは 4 バイトです。

- `time` は 00:00:00:000 から 23:59:59:990 の間です。`time` 値は、300 分の 1 秒のレベルまで正確です。小数秒の最後の桁は常に 0、3、または 6 です。その他の桁はこれらの 3 つの数字のいずれかに丸められます。そのため、0 と 1 は 0 に、2 と 3 と 4 は 3 に、5 と 6 と 7 と 8 は 6 に、9 は 10 に丸められます。24 時間制、または正午を 12AM、真夜中を 12PM とする 12 時間制のどちらでも使用できます。時刻値には、コロンか AM/PM 指示子を含めてください。AM または PM は、大文字でも小文字でもかまいません。

日付と時刻の情報を入力する場合は、時刻または日付を常に一重または二重の引用符で囲んでください。

範囲と記憶サイズ

表 1-13 に、`datetime`、`smalldatetime`、`bigdatetime`、`bigintime`、`date`、`time` の各データ型の範囲と記憶サイズを示します。

表 1-13: 日付と時刻を保管する Transact-SQL データ型

データ型	範囲	記憶サイズ (バイト数)
<code>datetime</code>	1753 年 1 月 1 日 ~ 9999 年 12 月 31 日	8
<code>smalldatetime</code>	1900 年 1 月 1 日 ~ 2079 年 6 月 6 日	4
<code>bigdatetime</code>	0001 年 1 月 1 日 ~ 9999 年 12 月 31 日	8
<code>bigintime</code>	12:00:00.000000AM ~ 11:59:59.999999PM	8
<code>date</code>	0001 年 1 月 1 日 ~ 9999 年 12 月 31 日	4
<code>time</code>	12:00:00 AM ~ 11:59:59:990 PM	4

日付および時刻データの入力

`datetime`、`smalldatetime`、`bigdatetime`、`bigintime` の各データ型は、日付部分と、その前後の時刻部分からなります (日付または時刻のいずれか、あるいは両方を省略できます)。`date` データ型は日付のみを含み、`time` データ型は時刻のみを含みます。値は、一重または二重引用符で囲む必要があります。

日付の入力

日付は月、日、年で構成され、`date`、`datetime`、`bigdatetime`、`bigintime`、および `smalldatetime` に対してさまざまなフォーマットで入力できます。

- 日付全体を、セパレータで区切らずに 4 桁、6 桁、または 8 桁の数字列で入力したり、日付の各要素をセパレータのスラッシュ (/)、ハイフン (-)、またはピリオド (.) で区切って入力したりすることもできます。
- 日付を区切りのない数字列として入力する場合、その数字列の長さに適したフォーマットを使用してください。1 桁の年、月、日の前には、先行ゼロを入力してください。間違ったフォーマットで日付を入力すると、誤って解釈されたり、エラーを引き起こしたりします。

- 日付をセパレータで区切って入力する場合、**set dateformat** オプションを使って、日付要素の順序を決定してください。区切られた数字列の最初の日付部分が 4 桁の場合、Adaptive Server ではその数字列を *yyyy-mm-dd* のフォーマットとして解釈します。
- 日付フォーマットの中には、2 桁の年 (*yy*) を使用するものもあります。
 - 50 未満の数値は 20*yy* として解釈されます。たとえば、01 は 2001、32 は 2032、49 は 2049 です。
 - 50 以上の数値は 19*yy* として解釈されます。たとえば、50 は 1950、74 は 1974、99 は 1999 です。
- 月の指定には、数字または名前を使用できます。月名とその省略形は言語ごとに異なります。月名は大文字でも小文字でも、また両方を組み合わせても入力できます。
- **datetime** 値または **smalldatetime** 値の日付部分を省略すると、Adaptive Server はデフォルト日付の 1900 年 1 月 1 日を使用します。**bigdatetime** の日付部分を省略すると、デフォルト値の 0001 年 1 月 1 日が追加されます。

表 1-14 に、**datetime** または **smalldatetime** 値の日付部分を入力するのに使用できるフォーマットを示します。

表 1-14: 日付と時刻のデータ型の日付フォーマット

日付フォーマット	解釈	入力例	意味
セパレータのない 4 桁の文字列	<i>yyyy</i> と解釈される。日付のデフォルトは、指定の年の 1 月 1 日。	“1947”	1947 年 1 月 1 日
セパレータのない 6 桁の文字列	<i>yyymmdd</i> と解釈される。 <i>yy</i> < 50 の場合、年は 20 <i>yy</i> となる。 <i>yy</i> ≥ 50 の場合、年は 19 <i>yy</i> となる。	“450128” “520128”	2045 年 1 月 28 日 1952 年 1 月 28 日
セパレータのない 8 桁の文字列	<i>yyyyymmdd</i> と解釈される。	“19940415”	1994 年 4 月 15 日
スラッシュ、ハイフン、ピリオド、またはこれらの組み合わせで区切った 2 桁ずつの月、日、年で構成される文字列	日付要素の順序は dateformat と language set オプションによって決まる。 us_english の場合、デフォルト順序は <i>mdy</i> 。 <i>yy</i> < 50 の場合、年は 20 <i>yy</i> と解釈される。 <i>yy</i> ≥ 50 の場合、年は 19 <i>yy</i> と解釈される。	“4/15/94” “4.15.94” “4-15-94” “04.15/94”	dateformat オプションが <i>mdy</i> に設定されている場合、これらの入力値はすべて 1994 年 4 月 15 日と解釈される。
スラッシュ、ハイフン、ピリオド、またはこれらの組み合わせで区切った 2 桁の月、2 桁の日、4 桁の年で構成される文字列	日付要素の順序は dateformat と language set オプションによって決まる。 us_english の場合、デフォルト順序は <i>mdy</i> 。	“04/15.1994”	dateformat オプションが <i>mdy</i> に設定されている場合、1994 年 4 月 15 日と解釈される。

日付フォーマット	解釈	入力例	意味
月はオプションで後ろにカンマを付けて文字形式で入力 (月のフルネームまたは標準省略名)	4 桁の年を入力する場合、日付要素は任意の順序で入力できる。	“April 15, 1994” “1994 15 apr” “1994 April 15” “15 APR 1994”	これらの入力値はすべて 1994 年 4 月 15 日と解釈される。
	日を省略する場合、年は 4 桁全体を指定する必要がある。デフォルトの日付は月の最初の日。	“apr 1994”	1994 年 4 月 1 日
	年が 2 桁だけ (yy) である場合、年は日の後ろに位置するものと見なされる。 yy < 50 の場合、年は 20yy と解釈される。 yy >= 50 の場合、年は 19yy と解釈される。	“mar 16 17” “apr 15 94”	2017 年 3 月 16 日 1994 年 4 月 15 日
空の文字列 “”	デフォルト日付は 1900 年 1 月 1 日。	“”	1900 年 1 月 1 日

時刻の入力

`datetime` 値、`smalldatetime` 値、または `time` 値の時刻部分は次のように指定してください。

```
hours[:minutes[:seconds[:milliseconds]]] [AM | PM]
```

`bigdatetime` 値または `bigtime` 値の時刻部分は次のように指定してください。

```
hours[:minutes[:seconds[.microseconds]]] [AM | PM]
```

- 午前 0 時には 12AM を、正午には 12PM を使用してください。
- 時刻値には、コロンか AM/PM 指示子を含めてください。AM と PM は大文字でも小文字でも、または両者を組み合わせても入力できます。
- 秒指定には、小数点のあとに小数部分を、またはコロンのあとにミリ秒数を含めることができます。たとえば、“15:30:20.1” は午後 3 時 30 分から 20 秒と 1 ミリ秒が過ぎた時刻を示し、“15:30:20.1” は午後 3 時 30 分から 20.1 秒が過ぎた時刻を示します。ミリ秒は小数点で表す必要があります。
- `datetime` 値または `smalldatetime` 値の時刻部分を省略すると、デフォルト時刻の 12:00:00:000AM が使用されます。

`datetime` 値、
`smalldatetime` 値、`date`
値の表示フォーマット

`datetime` 値と `smalldatetime` 値の表示フォーマットは、“Mon dd yyyy hh:mmAM” (または “PM”) です。たとえば、“Apr 15 1988 10:23PM” のように表示されます。秒とミリ秒を表示したり、他の日付スタイルや日付部分の順序を指定したりするには、`convert` 関数を使用して、日付を文字列に変換してください。Adaptive Server によってミリ秒値が丸められるか、トランケートされます。

表 1-15 に、`datetime` の入力例と表示される値を示します。

表 1-15: `datetime` と `date` の入力例

入力値	表示値
“1947”	Jan 1 1947 12:00AM
“450128 12:30:1PM”	Jan 28 2045 12:30PM
“12:30.1PM 450128”	Jan 28 2045 12:30PM
“14:30.22”	Jan 1 1900 14:30:00
“4am”	Jan 1 1900 04:00:00
日付の例	
“1947”	1947 年 1 月 1 日
“450128”	2045 年 1 月 28 日
“520317”	Mar 17 1952

`bigdatetime` と `bigtime` の表示フォーマット

`bigdatetime` および `bigtime` の場合、表示される値はマイクロ秒の値で示されます。`bigdatetime` と `bigtime` には、この増えた精度に対応するデフォルトの表示フォーマットがあります。

- hh:mm:ss.zzzzzzAM または PM
- hh:mm:ss.zzzzzz
- mon dd yyyy
hh:mm:ss.zzzzzzAM(PM)
- mon dd yyyy
hh:mm:ss.zzzzzz
- yyyy-mm-dd
hh:mm:ss.zzzzzz

時刻のフォーマットは次のように指定する必要があります。

hours[:minutes[:seconds[:microseconds]]] [AM | PM]

hours[:minutes[:seconds[number of milliseconds]]] [AM | PM]

午前 0 時には 12AM を、正午には 12PM を使用してください。`bigtime` 値には、コロンか AM/PM 指示子を含めてください。AM と PM は大文字でも小文字でも、または両者を組み合わせても入力できます。

秒指定には、小数点の後に小数部分を、またはコロンの後にミリ秒数を含めることができます。たとえば、“12:30:20.1” は午後 12 時 30 分から 20 秒と 1 ミリ秒が過ぎた時刻を示し、“12:30:20.1” は午後 12 時 30 分から 20.1 秒が過ぎた時刻を示します。

ミリ秒が含まれる `bigdatetime` または `bigtime` 時刻値を格納するには、小数点を使用したりテラル文字列を指定します。“00:00:00.1” は午前 0 時から 10 分の 1 秒が経過したことを示し、“00:00:00.000001” は午前 0 時から 100 万分の 1 秒が経過したことを示します。コロン後の小数秒を指定する値は、ミリ秒数を示します。たとえば、“00:00:00.5” は 5 ミリ秒を意味します。

time 値の表示フォーマット

time 値の表示フォーマットは “hh:mm:ss:mmmAM” (または “PM”) です。たとえば、“10:23:40:022PM” のように表示されます。

表 1-16: time の入力例

入力値	表示値
“12:12:00”	12:12PM
“01:23PM” または i01:23:1PM”	1:23PM
“02:24:00:001”	2:24AM

パターンに一致する値の検索

特定のパターンに一致する日付を検索するには、like キーワードを使用してください。等号演算子 (=) を使用して日付や時刻の値の特定の月、日、年を検索すると、Adaptive Server は時刻が正確に 12:00:00:000AM である値だけを返します。

たとえば、arrival_time という名前のカラムに値 “9:20” を挿入すると、Adaptive Server によってこのエントリが “Jan 1, 1900 9:20AM” に変換されます。等号演算子を使用してこの入力値を検索しても、この値は見つかりません。

```
where arrival_time = "9:20" /* does not match */
```

like 演算子を使用すれば、この入力値は見つかります。

```
where arrival_time like "%9:20%"
```

like を使用すると、日付はまず datetime または date フォーマットに変換され、次に varchar に変換されます。表示フォーマットは、現在の言語で 3 文字の月名、2 文字の日、4 文字の年、時刻の時と分、“AM” または “PM” で構成されます。

datetime 値、smalldatetime 値、bigdatetime 値、bigtime 値、date 値、および time 値の日付部分の入力には、多種類の入力フォーマットを使用できますが、like を使用して検索するときは、こうした多種類の入力フォーマットは使用できません。style 9 または 109 と convert 関数を併用しない限り、like と一致パターンで秒数やミリ秒数を検索することはできません。

like を使用していて、月の日付が 1～9 の場合、datetime 値の変換後の varchar 値に一致させるために、月と日の間にスペースを 2 つ挿入します。同様に、時間が 10 未満である場合、変換すると年と時間の間に 2 つのスペースが入ります。次の句 (“May” と “2” の間に 1 つのスペースがあります) では、5 月 20 日から 5 月 29 日までのすべての日付が検索されますが、5 月 2 日は検索されません。

```
like "May 2%"
```

その他の日付比較では、余分なスペースを挿入する必要はありません。datetime 値が varchar に変換されるのは like 比較の場合だけなので、like 比較を実行するときだけに、スペースを挿入してください。

日付の操作

組み込み日付関数を使用して、date データ型と time データ型の値についていくつかの算術計算を実行できます。『Transact-SQL ユーザーズ・ガイド』を参照してください。

標準規格と準拠レベル

ANSI SQL – 準拠レベル：`datetime` データ型および `smalldatetime` データ型は Transact-SQL 拡張機能です。`date` データ型および `time` データ型は初級レベルの標準に準拠します。

文字データ型

特定の状況で使用するデータ型は、保管するデータの型によって異なります。

- 文字、数字、記号からなる文字列を保管するには、文字データ型を使用します。
- `varchar(n)` と `char(n)` は、`us_english` などのシングルバイト文字セットと日本語などのマルチバイト文字の両方に使用します。
- `unicar(n)` データ型と `univarchar(n)` データ型は、Unicode 文字の保管に使用します。これらは、1 文字に固定のバイト数が必要な場合、シングルバイトまたはマルチバイトの文字に利用できます。
- シングルバイト文字セットと日本語などのマルチバイト文字セットの両方には、固定長データ型の `nchar(n)` と可変長データ型の `nvarchar(n)` を使用します。`nchar(n)` と `char(n)`、および `nvarchar(n)` と `varchar(n)` の違いは、`nchar(n)` と `nvarchar(n)` では、1 文字あたりのバイト数 (デフォルト文字セットによって異なる) の n 倍に基づいて記憶領域が割り付けられるのに対し、`char(n)` と `varchar(n)` では、 n バイトの記憶領域が割り付けられる点です。
- 文字データ型には、最大でページ・サイズ分のデータを保管できます。
- `char` データ型または `varchar` データ型で許可されているより長い文字列を保管するには、`text` データ型 (詳細については、「[text、image、unitext データ型](#)」(32 ページ) を参照) またはサブテーブル内の複数のローを使用します。

unicar、univarchar

`char` 文字データ型および `varchar` 文字データ型が使用できる場所であれば、構文を変更することなく、`unicar` データ型および `univarchar` データ型をどこでも使用できます。

Adaptive Server バージョン 12.5.1 以降では、サーバの文字セットで表現できない文字リテラルを含むクエリは自動的に `unicar` データ型に拡大されるので、データ操作言語 (DML) 文の構文を変更する必要はありません。文字リテラルに任意の文字を指定するには追加の構文も使用できますが、リテラルを `unicar` に「拡大」する決定は、サーバの文字セットで表現できるかどうかのみに基づいています。

データ定義言語 (DDL) 文では、構文を変更する必要性は最低限です。たとえば、`create table` コマンドでは、Unicode カラムのサイズはバイトではなく 16 ビットの Unicode 値の単位で指定されます。そのため、`char(200)` と `unichar(200)` の類似性が保持されます。カラムの長さをレポートする `sp_help` は、同じ単位を使用します。乗算因子 (2) は新しいグローバル変数 `@@unicharsize` に保管されます。

Unicode の詳細については、『システム管理ガイド』の「第 8 章 文字セット、ソート順、言語の設定」を参照してください。

文字単位の長ささと記憶サイズ

文字変数は、カーソルの `varchar` カラムに移植されるときに文字列から後続スペースを取り除きます。

`char` データ型と `varchar` データ型の記憶領域のバイト数を指定するには、 n を使用します。`unichar` の場合は、 n を使用して Unicode 文字数を指定します (割り付けられる記憶領域は各文字に対して 2 バイトです)。`nchar` と `nvarchar` の場合、 n は文字数です (割り付けられる記憶領域は、サーバの現在のデフォルトの文字セットにおける、各文字のバイト数の n 倍です)。

長さの指定に n を使用しない場合は、次の点に注意してください。

- `create table`、`alter table` で作成されたカラム、および `declare` で作成された変数のデフォルトの長さは 1 バイトである。
- `convert` 関数で作成された値のデフォルトの長さは 30 バイトである。

指定の長さより短い入力値には空白が埋め込まれます。指定の長さより長い入力値は、`set` コマンドの `string_truncation` オプションが `on` に設定されている場合を除き、警告が出ずにトランケートされます。`null` 値を保管できる固定長カラムは、内部で可変長カラムに変換されます。

可変長データ型の `varchar(n)`、`univarchar(n)`、`nvarchar(n)` では、 n に文字単位で最大長を指定します。可変長カラムのデータの後続空白は削除されます。したがって、記憶サイズは入力されたデータの実際の長さです。可変長変数およびパラメータのデータの場合、後続空白はすべて保持されますが、定義されている長さになるまで空白が埋め込まれることはありません。文字リテラルは可変長データ型として扱われます。

固定長カラムには、一般に、可変長カラムより大きな記憶領域が必要ですが、アクセスにかかる時間は多少短くなります。表 1-17 に、各文字データ型に必要な記憶サイズを示します。

表 1-17: 文字データ型

データ型	保管するデータ	記憶サイズ (バイト数)
char(n)	文字	<i>n</i>
unichar(n)	Unicode 文字	<i>n</i> *@@unicharsize (@@unicharsize は 2)
nchar(n)	各国文字	<i>n</i> * @@ncharsize
varchar(n)	可変長文字	実際に入力した文字数
univarchar(n)	可変長の Unicode 文字	実際の文字数 * @@unicharsize
nvarchar(n)	可変長の各国文字	実際の文字数 * @@ncharsize

システム関数によるカラム長の決定

カラムの長さを決定するには、`char_length` 文字列関数と、`datalength` システム関数を使用します。

- `char_length` は、カラムの文字数を返し、可変長データ型の後続ブランクを削除します。
- `datalength` は、可変長カラムに保管されたデータの後続ブランクを削除し、バイト数を返します。

char 値に NULL 値を使用できることが宣言されている場合、Adaptive Server 内部ではこの値は `varchar` として保管されます。

char カラムに対して `min` 集合関数または `max` 集合関数が実行されると、結果として `varchar` の値が返されます。このため、返された値では後続スペースがすべて削除されています。

文字データの入力

文字列は必ず一重引用符または二重引用符で囲みます。set quoted_identifier を使用している場合は、文字列を一重引用符で囲んでください。そうしないと、Adaptive Server は文字列を識別子として扱います。

二重引用符が含まれている文字列は一重引用符で囲んでください。一重引用符が含まれている文字列は二重引用符で囲んでください。たとえば、次のようにします。

```
'George said, "There must be a better way."'
"Isn't there a better way?"
```

もう 1 つの方法として、文字列に含めたい引用符がある場合に、その引用符を 2 つ連続して入力する方法があります。たとえば、次のようにします。

```
"George said, ""There must be a better way.""
'Isn't there a better way?'
```

文字列を画面の次の行に続けるには、次の行に進む前に円記号 (¥) を入力してください。

引用符で囲む識別子の詳細については、『Transact SQL ユーザーズ・ガイド』の「区切り識別子」を参照してください。

Unicode 文字の入力

オプションの新しい構文を使用して、任意の Unicode 文字を指定できます。文字リテラルの直前に U& または u& (間にスペースなし) がある場合、パーサはリテラル内のエスケープ・シーケンスを認識します。¥xxxx (xxxx は 4 桁の 16 進数を表す) の形式のエスケープ・シーケンスは、スカラ値が xxxx の Unicode 文字に置き換えられます。同様に、¥+yyyyyy の形式のエスケープ・シーケンスは、スカラ値が yyyyyy の Unicode 文字に置き換えられます。エスケープ・シーケンス ¥は、単一の ¥ で置き換えられます。たとえば、次の文は同じ意味を持ちます。

```
select * from mytable where unichar_column = '五'

select * from mytable where unichar_column = U'¥4e94'
```

プレフィクスの U& または u& は、単にエスケープの認識を有効にするものです。リテラルのデータ型は、文字セットで表現できるかどうかのみに基づいて選択されます。そのため、たとえば次の 2 つのクエリは同等です。

```
select * from mytable where char_column = 'A'

select * from mytable where char_column = U'¥0041'
```

どちらの場合も、文字リテラルのデータ型は char です。これは、“A” は ASCII 文字で、ASCII は Sybase がサポートしているサーバの文字セットすべてのサブセットであるためです。

プレフィクスの U& および u& は、二重引用符で囲まれた文字リテラルでも、また引用符で囲まれた識別子に対しても有効です。ただし、引用符で囲まれた識別子は、すべてのデータベース・オブジェクトがシステム・テーブル内の名前で識別され、これらの名前が char データ型である場合、サーバの文字セットで表現可能である必要があります。

空文字列の処理

次の例では、固定長文字カラムと可変長文字カラムの両方がある spaces という名前のテーブルが作成されます。

```
create table spaces (cnot char(5) not null,
  cnull char(5) null,
  vnnot varchar(5) not null,
  vnull varchar(5) null,
  explanation varchar(25) not null)

insert spaces values ("a", "b", "c", "d", "pads char-not-null only")
insert spaces values ("1  ", "2  ", "3  ", "4  ", "truncates trailing blanks")
insert spaces values (" e", "  f", "   g", "    h", "leading blanks, no change")
insert spaces values (" w", "  x", "   y", "    z", "truncates trailing blanks")
insert spaces values ("", "", "", "", "empty string equals space")
```

```
select "[" + cnot + "]",
       "[" + cnull + "]",
       "[" + vnot + "]",
       "[" + vnull + "]",
       explanation from spaces
```

				explanation
-----	-----	-----	-----	-----
[a]	[b]	[c]	[d]	pads char-not-null only
[1]	[2]	[3]	[4]	truncates trailing blanks
[e]	[f]	[g]	[h]	leading blanks, no change
[w]	[x]	[y]	[z]	truncates trailing blanks
[]	[]	[]	[]	empty string equals space

(5 rows affected)

この例は、ブランク・スペースの扱い方を決定するのにカラムのデータ型および null 型がどのように作用するかを示しています。

- **char not null** カラムと **nchar not null** カラムだけが、カラム幅いっぱいまでブランクを埋め込まれます。**char null** カラムは **varchar** と同様に扱われます。**nchar null** カラムは **nvarchar** と同様に扱われます。
- **unichar not null** カラムだけが、カラム幅いっぱいまでブランクを埋め込まれます。**unichar null** カラムは **univarchar** と同様に扱われます。
- 先行ブランクは影響を受けません。
- **char**、**unichar**、**nchar not null** のカラム以外では、後続ブランクがトランケートされます。
- 空の文字列 (“ ”) はシングル・スペースとして扱われます。**char**、**nchar**、**unichar not null** のカラムでは、結果はスペースだけからなるカラム長フィールドです。

文字データの操作

文字列内の特定の文字を検索するには **like** キーワードを使用し、文字列の内容を操作するには組み込み関数を使用します。数字からなる文字列は、**convert** 関数で真数値データ型および概数値データ型に変換してから、算術演算に使用できます。

標準規格と準拠レベル

ANSI SQL – 準拠レベル: Transact-SQL には、ANSI SQL データ型の **char** および **varchar** があります。**nchar**、**nvarchar**、**unichar**、**univarchar** データ型は Transact-SQL 拡張機能です。

バイナリ・データ型

ピクチャなどのロー・バイナリ・データを 16 進数に似た表記で、最大でサーバの論理ページ・サイズの最大カラム・サイズまで保管するには、バイナリ・データ型の `binary(n)` と `varbinary(n)` を使用します。

binary と *varbinary* の有効な入力値

バイナリ・データは文字 “0x” で始まり、数字と大文字および小文字の A ~ F との任意の組み合わせで構成されます。

n を使用してカラム長をバイト数単位で指定するか、デフォルト長の 1 バイトを使用してください。各バイトは、2 桁のバイナリ数を保管します。*n* より長い値を入力すると、Adaptive Server は、警告やエラーを発生せずにエントリを指定の長さにはトランケートします。

すべての入力値の長さがほぼ等しいと予測されるデータには、固定長バイナリ型の `binary(n)` を使用します。

長さが大きく異なることが予測されるデータには、可変長バイナリ型の `varbinary(n)` を使用してください。

`binary` カラムの入力値には、カラム長 (*n*) に一致するようにゼロが埋め込まれるため、この入力値は `varbinary` カラムよりも多くの記憶領域を使用する場合がありますが、アクセスは多少速くなります。

長さの指定に *n* を使用しない場合は、次の点に注意してください。

- `create table`、`alter table` で作成されたカラム、および `declare` で作成された変数のデフォルトの長さは 1 バイトである。
- `convert` 関数で作成された値のデフォルトの長さは 30 バイトである。

最大カラム・サイズよりも大きいエントリ

さらに大きいブロックのバイナリ・データ (最大 2,147,483,647 バイトまで) を外部データ・ページに保管するには、`image` データ型を使用してください。`image` データ型は変数やストアド・プロシージャ内のパラメータには使用できません。詳細については、「[text, image, unitext データ型](#)」(32 ページ) を参照してください。

後続ゼロの扱い

`binary not null` カラムはすべて、カラム幅いっぱいまでゼロが埋め込まれます。すべての `varbinary` データおよび `binary null` カラムでは、後続ゼロはトランケートされます。これは、`null` 値を受け入れるカラムは可変長として扱われるためです。

次の例では、NULL と NOT NULL、binary データ型と varbinary データ型の 4 つの組み合わせのカラムがすべて揃っているテーブルを作成します。4 つのカラムすべてに同じデータを挿入しますが、ゼロが埋め込まれるか、トランケートされるかどうかは、カラムのデータ型によって異なります。

```
create table zeros (bnot binary(5) not null,
                  bnull binary(5) null,
                  vnot varbinary(5) not null,
                  vnull varbinary(5) null)

insert zeros values (0x12345000, 0x12345000, 0x12345000, 0x12345000)
insert zeros values (0x123, 0x123, 0x123, 0x123)

select * from zeros
```

bnot	bnull	vnot	vnull
0x1234500000	0x123450	0x123450	0x123450
0x0123000000	0x0123	0x0123	0x0123

記憶領域の各バイトによってそれぞれ 2 桁のバイナリ数字が保持されるため、Adaptive Server は文字列 “0x” のあとに偶数個の数字が続くバイナリ・エントリを予期します。“0x” のあとに奇数個の数字が続く場合、Adaptive Server は先行ゼロが省略されているものとみなし、0 を追加します。

可変長バイナリ・カラム (binary null、image、varbinary カラム) では、入力値 “0x00” と “0x0” はどちらも “0x00” として保管されます。固定長バイナリ (binary not null) カラムでは、値はフィールド幅いっぱいまでゼロを埋め込まれます。

```
insert zeros values (0x0, 0x0,0x0, 0x0)
select * from zeros where bnot = 0x00

bnot          bnull          vnot          vnull
-----
0x0000000000 0x00          0x00          0x00
```

入力値に “0x” がいない場合、Adaptive Server はその値が ASCII 値であると想定して、値を変換します。たとえば、次のようにします。

```
create table sample (col_a binary(8))

insert sample values ('002710000000ae1b')

select * from sample
col_a
-----
0x3030323731303030
```

プラットフォームの依存

特定の値の正確な入力形式は、使用しているプラットフォームに応じて異なります。このため、バイナリ・データを使う計算では、マシンによって異なった結果となることがあります。

集合関数 `sum` と `avg` では、バイナリ・データ型を処理できません。

16 進数の文字列と整数との間の変換で、どのプラットフォームでも同じ結果が得られるようにするには、プラットフォーム固有の変換関数ではなく、`inttohex` 関数と `hextoint` 関数を使います。詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

標準規格と準拠レベル

ANSI SQL – 準拠レベル：binary データ型および varbinary データ型は Transact-SQL 拡張機能です。

bit データ型

true/false (真/偽) と yes/no タイプのデータを格納するカラムには、bit データ型を使用します。`syscolumns` システム・テーブルの `status` カラムは、bit データ型のカラムのユニークなオフセット位置を示します。

bit カラムは、0 または 1 を保持します。0 または 1 以外の整数値も受け入れませんが、すべて 1 と解釈されます。

記憶サイズは 1 バイトです。テーブル内の複数の bit データ型がバイト単位にまとめられます。たとえば 7 つの bit カラムは 1 バイトになり、9 つの bit カラムは 2 バイトを使用します。

bit データ型のカラムには NULL を保管できません。また、インデックスも設定できません。

標準規格と準拠レベル

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

sysname データ型および longsysname データ型

sysname および longsysname は、Adaptive Server インストール・メディアによって配布されるユーザ定義データ型で、システム・テーブルで使用されず。定義は次のとおりです。

- sysname – varchar(30) "not null"
- longsysname – varchar(255) "not null"

カラム、パラメータ、または変数を sysname 型および longsysname 型として宣言できます。また、sysname および longsysname の基本となる型を持つユーザ定義データ型を作成し、このユーザ定義データ型を使用してカラム、パラメータ、変数を定義することもできます。

標準規格と準拠レベル

ANSI SQL – 準拠レベル：sysname 型および longsysname 型を含め、すべてのユーザ定義データ型は Transact-SQL 拡張機能です。

text、image、unitext データ型

text カラムは、最大 2,147,483,647 ($2^{31} - 1$) バイトまでの印刷可能な文字を保持できる可変長カラムです。

可変長の unitext データ型は、Unicode 文字で最大 1,073,741,823 文字 (2,147,483,646 バイト) まで保持できます。

image カラムは、最大 2,147,483,647 ($2^{31} - 1$) バイトまでのロー・バイナリ・データを保持できる可変長カラムです。

text と image の大きな違いは、Adaptive Server のデフォルト文字セットと異なる文字セットでアクセスする場合、text は文字セット変換の対象になることです。image は、文字セット変換の対象になりません。

text、unitext、または image カラムは、他のカラムと同様に、create table 文または alter table 文を使用して定義します。text、unitext、または image データ型の定義には、長さは含まれません。text、unitext、image カラムでは、null 値が許可されます。これらのカラム定義の形式は次のとおりです。

```
column_name {text | image | unitext} [null]
```

たとえば、pubs2 データベースに作者の blurbs テーブルを作成し、null 値を格納できる text カラム blurb をこのテーブルに作成する create table 文は次のとおりです。

```
create table blurbs
  (au_id id not null,
  copy text null)
```

次の例では、null 値が使用できる `unitext` カラムが作成されます。

```
create table tb (ut unitext null)
```

`image` カラムのある `au_pix` テーブルを `pubs2` データベースに作成するには、次の文を使用します。

```
create table au_pix
(au_id          char(11) not null,
 pic           image null,
 format_type   char(11) null,
 bytesize     int null,
 pixwidth_hor  char(14) null,
 pixwidth_vert char(14) null)
```

Adaptive Server では、`text`、`unitext`、および `image` データは、テーブルの他の部分とは別のデータ・ページのリンク・リストに保管されます。各 `text`、`unitext`、または `image` ページには、1 論理ページ・サイズ分のデータ (2、4、8、または 16K) が保管されます。テーブルに含まれている `text`、`unitext`、および `image` カラムの数には関係なく、1 つのテーブルの `text`、`unitext`、および `image` データは、1 つのページ・チェーンにすべて保管されます。

`sp_placeobject` を使用して、後続の `text`、`unitext`、`image` データ・ページの割り当てを別の論理デバイスに置くことができます。

16 進数数値の桁数が奇数の `image` 値には、先行ゼロが埋め込まれます (“0xaaabb” を挿入すると、“0x0aaabb” になります)。

`alter table` コマンドの `partition` オプションを使用して、`text`、`unitext`、`image` カラムが含まれているテーブルを分割できます。テーブルを分割することにより、テーブル内の他のカラムのための追加ページ・チェーンが作成されますが、これで `text`、`unitext`、および `image` カラムの保管方法が影響を受けることはありません。

`unitext` は、`text` データ型を使用する任意の場所で同じセマンティックで使用できます。`unitext` カラムは、Adaptive Server で使用されているデフォルトの文字セットにかかわらず、UTF-16 エンコーディングで保管されます。

text、unitext、および image データの保管に使用されるデータ構造

`text`、`unitext`、または `image` データを割り付ける場合、割り付けるローに 16 バイトのテキスト・ポインタが挿入されます。このテキスト・ポインタの一部は、`text`、`unitext`、または `image` データの先頭にある `text` ページのページ番号を示します。テキスト・ポインタは先頭 `text` ページ (FTP) とも呼ばれます。

先頭 `text` ページには、次の 2 つの部分があります。

- `text` データと `image` データを含む、`text` ページの双方向にリンクされたリストである `text` データ・ページ・チェーン
- ユーザの `text` データへのアクセスに使用されるオプションのテキスト・ノード構造

いったん先頭 text ページが text、unitext、または image データに割り付けられると、その割り付けは解除されることがありません。既存の text、unitext、または image データ・ローが更新されたことにより、現在割り付けられているよりも text ページが少なくなった場合は、余分な text ページの割り付けが解除されます。text、unitext、または image データを更新して値が NULL に設定されると、先頭 text ページ以外のすべてのページの割り付けは解除されます。

図 1-1 は、データ・ローと text ページの関係を示します。

図 1-1: テキスト・ポインタとデータ・ローの関係

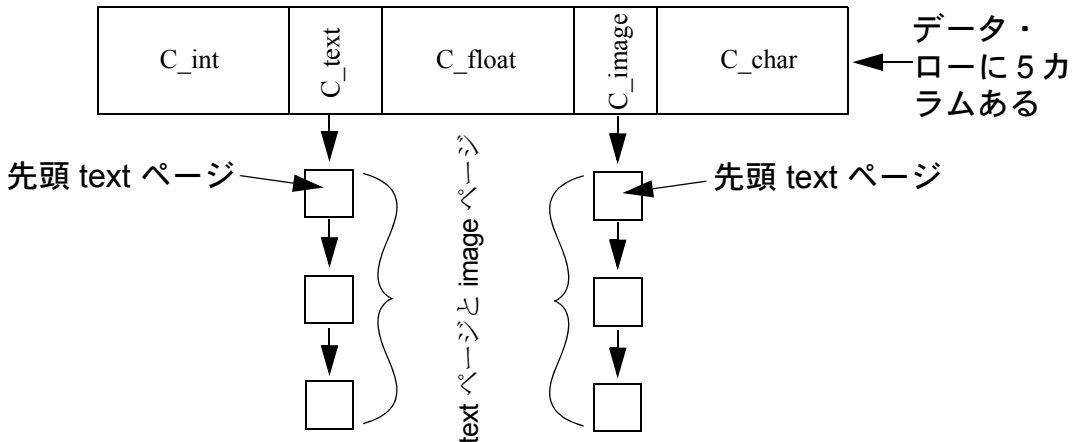


図 1-1 では、カラム c_text と c_image は、図の下部にあるページを含む text カラムおよび image カラムです。

text、unitext、および image カラムの初期化

text、unitext、image カラムは、更新するか、非 null 値を挿入するまで、初期化されません。初期化によって、text、unitext、または image の非 null データ値ごとに、少なくとも 1 つのデータ・ページが割り付けられます。text、unitext、または image データのロケーションを指すポインタもテーブルに作成されます。

たとえば次の文では、テーブル testtext を作成し、非 null 値を挿入して blurb カラムを初期化します。これで、このカラムは有効なテキスト・ポインタを持ち、最初のテキスト・ページが割り付けられます。

```
create table testtext
(title_id varchar(6), blurb text null, pub_id char(4))

insert testtext values
("BU7832", "Straight Talk About Computers is an annotated
analysis of what computers can do for you: a no-hype guide for
the critical user.", "1389")
```


次の文では、`image` 値のためのテーブルを作成し、`image` カラムを初期化します。

```
create table imagetest
(image_id varchar(6), imagecol image null, graphic_id char(4))
insert imagetest values
("94732", 0x00000083000000000000100000000013c, "1389")
```

注意 `text` 値は必ず引用符で囲み、`image` 値の前には文字“0x”を付けてください。

Client-Library プログラムでの `text`、`unitext`、および `image` データの挿入と更新については、『Client-Library/C リファレンス・マニュアル』を参照してください。

unitext カラムの定義

`unitext` カラムは、他のデータ型を定義する場合と同じ方法で、`create table` 文または `alter table` 文を使用して定義できます。`unitext` カラムの長さは定義しません。カラムは `null` でもかまいません。

次の例では、`null` 値が使用できる `unitext` カラムが作成されます。

```
create table tb (ut unitext null)
```

`default unicode sort order` は、`like` 句および `patindex` 関数内でパターン一致に使用する `unitext` カラムのソート順を定義します。このソート順は、Adaptive Server のデフォルト・ソート順とは無関係に適用されます。

NULL によるスペースの節約

空の `text`、`unitext`、または `image` カラムの記憶領域を節約するには、カラムへの `null` 値入力を許可し、カラムを使用するまで `null` を挿入 (`insert`) します。`null` 値を挿入しても、`text`、`unitext`、または `image` カラムは初期化されません。したがって、テキスト・ポインタは作成されず、記憶域も割り付けられません。たとえば次の文では、上で作成した `testtext` テーブルの `title_id` と `pub_id` の両カラムに値が挿入されますが、`blurb` テキスト・カラムは初期化されません。

```
insert texttest
(title_id, pub_id) values ("BU7832", "1389")
```

sysindexes からの情報取得

`text`、`unitext`、または `image` カラムを持つ各テーブルに対して、これらのカラムに関する情報を提供する特別なローが `sysindexes` 内にあります。`sysindexes` の `name` カラムは、“tablename” という形式を使用しています。`indid` は常に 255 です。これらのカラムには、テキストの記憶領域に関する情報が格納されています。

表 1-18: text および image データの記憶領域

コラム	説明
ioampg	テキスト・ページ・チェーンのアロケーション・ページを指すポインタ
first	テキスト・データの最初のページを指すポインタ
root	最後のページを指すポインタ
segment	オブジェクトが置かれているセグメントの数

これらのコラムに関する情報について、**sysindexes** テーブルを問い合わせることができます。たとえば、次のクエリでは、**pubs2** データベースで **blurbs** テーブルが使用しているデータ・ページ数がレポートされます。

```
select name, data_pages(db_id(), object_id("blurbs"), indid)
      from sysindexes
      where name = "tblurbs"
```

注意 システム・テーブルのポスターでは、**sysindexes** と **systabstats** が 1 対 1 (1-1) の関係にあることが示されています。**systabstats** で情報が保持されない **text** コラムと **image** コラムを除く、すべてのコラムにこの関係が該当します。

readtext と writetext の使用

text データを **writetext** を使用して入力したり、**readtext** を使用して読み取ったりする前に、**text** コラムを初期化してください。詳細については、『**ASE** リファレンス・マニュアル: コマンド』の「**readtext**」および「**writetext**」を参照してください。

update を使用して、既存の **text**、**unitext**、**image** データを **NULL** で置き換えると、最初のページを除いて、割り付けられていたすべてのデータ・ページが再び使用できる状態になるため、以降の **writetext** で使用可能になります。ローに割り付けられていたすべての記憶領域を解除するには、**delete** を使用して、ロー全体を削除してください。

unitext で定義されたコラムに対する **readtext** および **writetext** の使用には制限があります。詳細については、『**リファレンス・マニュアル: コマンド**』の「**readtext**」および「**writetext**」の「**使用法**」を参照してください。

コラムが使用する領域の決定

sp_spaceused は、テキスト・データが使用する領域に関する情報を、**index_size** として提供します。

```
sp_spaceused blurbs
name          rowtotal  reserved  data    index_size unused
-----
blurbs        6          32 KB    2 KB   14 KB    16 KB
```

text、image、および unitext カラムの制限

text、image、または unitext カラムには、次の制限があります。

- order by 句、compute 句、group by 句、union 句内では使用できません。
- インデックス内では使用できません。
- サブクエリまたはジョイン内では使用できません。
- where 句内では使用できません。ただし、キーワード like で使用する場合は除きます。

トリガでは、挿入するテキスト値と削除するテキスト値の両方で新しい値が参照され、古い値は参照できません。

text、unitext、image データの選択

次のグローバル変数は text、unitext、および image データに関する情報を返します。

表 1-19: text、unitext、image グローバル変数

変数	説明
<code>@@textptr</code>	プロセスが最後に挿入または更新した、text、unitext、または image カラムのテキスト・ポインタ。このグローバル変数と <code>textptr</code> 関数を混同しないように注意。
<code>@@textcolid</code>	<code>@@textptr</code> が参照するカラムの ID。
<code>@@textdbid</code>	<code>@@textptr</code> が参照するカラムのオブジェクトが入っているデータベースの ID。
<code>@@textobjid</code>	<code>@@textptr</code> が参照するカラムが入っているオブジェクトの ID。
<code>@@textsize</code>	<code>set textsize</code> オプションの現在の値。この値は、select 文が返す text、unitext、または image データの最大長をバイト数単位で指定する。デフォルト値は 32K。@@textsize の最大サイズは $2^{31} - 1$ (2,147,483,647)。
<code>@@textts</code>	<code>@@textptr</code> によって参照されるカラムのタイムスタンプ。

text、unitext、image 値は、非常に大きい場合があります。select リストに text および image 値が含まれる場合、返されるデータの長さの制限値は、@@textsize グローバル変数の設定に依存します。この変数には、select が返す text または image データのバイト数の制限値が含まれています。デフォルトの制限値は、isql の場合 32 キロバイトです。デフォルトは、クライアント・ソフトウェアによって異なります。set textsize を使用して、セッションごとに値を変更できます。

text および image データ型の変換

`convert` 関数を使用すれば、`text` 値から `char`、`unichar`、`varchar`、または `univarchar` への変換、および `image` 値から `binary` または `varbinary` への変換を明示的に実行できます。ただし、文字データ型またはバイナリ・データ型の最大長は制限されています。この制限は、ご使用のサーバの論理ページ・サイズの最大カラム・サイズによって決まります。長さを指定しない場合、変換された値はデフォルトの長さである 30 バイトになります。暗黙的な変換はサポートされていません。

unitext へ、または unitext からの変換

`unitext` を明示的に別のデータ型に変換したり、別のデータ型を `unitext` に変換したりするのと同様に、文字またはバイナリのデータ型を暗黙的に `unitext` に変換することもできます。ただし、変換結果は変換先のデータ型の最大長に制約されます。`unitext` 値が Unicode 文字境界上の変換先バッファに収まらない場合、データはトランケートされます。`enable surrogate processing` を有効にしている場合は、`unitext` 値はサロゲート・ペア値の途中ではトランケートされません。つまり、データ変換後に返されるバイト数が少なくなることがあります。たとえば、テーブル `tb` の `unitext` カラム `ut` に文字列 埋+0041U+0042U+00c2・(U+0041 は Unicode 文字 “A” が保管されている場合、次のクエリはサーバの文字セットが UTF-8 のときに値 “AB” を返します。これは、U+00C2 が 2 バイトの UTF-8 0xc382 に変換されるからです。

```
select convert(char(3), ut) from tb
```

表 1-20: unitext へ、および unitext からの変換

暗黙的に unitext へ変換されるデータ型	暗黙的に unitext から変換されるデータ型	明示的に unitext から変換されるデータ型
char, varchar, unichar, univarchar, binary, varbinary, text, image	text, image	char, varchar, unichar, univarchar, binary, varbinary

`alter table modify` コマンドには、変更対象カラムとして `text`、`image`、または `unitext` カラムを指定できません。`text` カラムを `unitext` カラムにマイグレートするには、次の手順を実行します。

- `bcp out -Jutf8` を使用して `text` カラム・データをコピー・アウトします。
- `unitext` カラムを含むテーブルを作成します。
- `bcp in -Jutf8` を使用して、データを新しいテーブルに挿入します。

text データのパターン一致

`text`、`unitext`、`varchar`、`univarchar`、`unichar`、または `char` カラム内で、指定パターンが最初に出現する位置を検索するには、`patindex` 関数を使用します。`%` ワイルドカード文字は、最初の文字または最後の文字を検索する場合を除いて、パターンの前後に指定する必要があります。

特定のパターンを検索するには、`like` キーワードも使用できます。次の例では、`blurbs` テーブルの `copy` カラムから、パターン “Net Etiquette” が含まれている各 `text` データ値を選択します。

```
select copy from blurbs
where copy like "%Net Etiquette%"
```

重複ロー

`text`、`image`、および `unitext` データを指すポインタによって、各ローが一意に識別されます。したがって、`text`、`image`、および `unitext` データが含まれているテーブルには、すべての `text`、`image`、および `unitext` データが `NULL` の場合を除き、重複するローを格納できません。このような場合、ポインタが初期化されていません。

ストアド・プロシージャにおけるラージ・オブジェクトの `text`、`unitext`、`image` データ型の使用

Adaptive Server は、以下の機能を提供します。

- ローカル変数に対してラージ・オブジェクト (LOB) の `text`、`image`、または `unitext` データ型を宣言し、その変数を入力パラメータとしてストアド・プロシージャに渡すことができます。
- LOB パラメータを含む SQL 文を作成できます。

ステートメント・キャッシュを有効にすると、Adaptive Server は LOB を使用して SQL 文をキャッシュします。『システム管理ガイド 第 2 巻』の「第 3 章 メモリの設定」を参照してください。

ストアド・プロシージャで LOB を使用する場合、次のような制限が適用されます。

- LOB パラメータは複写には対応しない。
- LOB データ型は、`execute immediate` および遅延コンパイルには使用できない。

LOB データ型の宣言

ローカル変数の LOB データ型を宣言するには、次の構文を使用します。

```
declare @variable LOB_datatype
```

ここで、*LOB_datatype* は、text、image、unitext のいずれかです。

次の例では、*text_variable* を text データ型として宣言します。

```
declare @text_variable text
```

LOB パラメータの作成

LOB パラメータを作成するには、次の構文を使用します。

```
create procedure proc_name [@parameter_name LOB_datatype  
as {SQL_statement}
```

次の例では、text LOB データ型を使用する new_proc プロシージャを作成します。

```
create procedure new_proc @v1 text  
as  
select char_length(@v1)
```

LOB データ型の使い方

例 1 LOB をストアド・プロシージャの入力パラメータとして使用します。

1 table_1 を作成します。

```
create table t1 (a1 int, a2 text)  
insert into t1 values(1, "aaaa")  
insert into t1 values(2, "bbbb")  
insert into t1 values(3, "cccc")
```

2 LOB ローカル変数をパラメータとして使用してストアド・プロシージャを作成します。

```
create procedure my_procedure @new_var text  
as select @new_var
```

3 ローカル変数を宣言して、ストアド・プロシージャを実行します。

```
declare @a text  
select @a = a2 from t1 where a1 = 3  
exec my_procedure @a  
-----  
cccc
```

例 2 LOB 変数を text 関数で使用します。

```
declare @a text  
select @a = "abcdefgh"  
select datalength(@a)  
-----
```

例3 LOB テキスト・ローカル変数を宣言します。

```

declare @a text
select @a = '<doc><item><id>1</id><name>Box</name></item>'
          + '<item><id>2</id><name>Jar</name></item></doc>'
select id from xmltable ('/doc/item' passing @a
                        columns id int path 'id', name varchar(20) path 'name')
as items_table
   id
-----
   1
   2

```

次に、同じ LOB パラメータをストアド・プロシージャに渡します。

```

ストアドプロシージャの作成
create proc pr1 @a text
as
select id from xmltable ('/doc/item' passing @a
                        columns id int path 'id', name varchar(20) path 'name') as items_table

```

```

ストアドプロシージャの実行
declare @a text
select @a =
'<doc><item><id>1</id><name>Box</name></item>'
+'<item><id>2</id><name>Jar</name></item></doc>'
exec pr1 @a

   id
-----
   1
   2

```

標準規格と準拠レベル

ANSI SQL – 準拠レベル: text、image、および unitext データ型は Transact-SQL 拡張機能です。

データ型と暗号化カラム

表 1-21 では、暗号化カラムでサポートされるデータ型、および Adaptive Server バージョン 15.0.2 でサポートされるデータ型の暗号化カラムのディスク上の最大長を示します。

表 1-21: 暗号化カラムのデータ型長

データ型	入力データ長	暗号化カラムのタイプ	暗号化データの最大長 (init_vector なし)	暗号化データの実際の長さ (init_vector なし)	暗号化データの最大長 (init_vector 使用)	暗号化データの実際の長さ (init_vector 使用)
date	4	varbinary	17	17	33	33
time	4	varbinary	17	17	33	33
smalldatetime	4	varbinary	17	17	33	33
bigdatetime	8	varbinary	17	17	33	33
bigtime	8	varbinary	17	17	33	33
datetime	8	varbinary	17	17	33	33
smallmoney	4	varbinary	17	17	33	33
money	8	varbinary	17	17	33	33
bit	8	varbinary	17	17	33	33
bigint	8	varbinary	17	17	33	33
unsigned bigint	8	varbinary	17	17	33	33
unichar(10)	2 (1 unichar 文字)	varbinary	33	17	49	33
unichar(10)	20 (10 unichar 文字)	varbinary	33	33	49	49
univarchar(20)	20 (10 unichar 文字)	varbinary	49	33	65	49

text、image、unitext データ型は、このリリースの Adaptive Server ではサポートされていません。

ユーザ定義データ型

ユーザ定義データ型は、システム・データ型および `sysname` または `longsysname` ユーザ定義データ型から構築されます。ユーザ定義データ型を作成したあと、これを使用して、カラム、パラメータ、変数を定義できます。ユーザ定義データ型から作成されたオブジェクトは、ユーザ定義データ型の基になっているシステム・データ型のデフォルトや null 型を継承するのと同様に、ユーザ定義データ型のルール、デフォルト、null 型、IDENTITY プロパティを継承します。

ユーザ定義データ型は、そのデータ型を使用するデータベースに作成する必要があります。使用頻度の高い型は、**model** データベースに作成します。これらのデータ型は、新しいデータベース (テンポラリ・テーブルに使用する **tempdb** を含む) が作成されると、自動的に追加されます。

Adaptive Server では、**sp_addtype** を使用して、任意のシステム・データ型をもとにユーザ定義データ型を作成できます。**timestamp** や **pubs2** データベースで定義されている **tid** データ型など、別のユーザ定義データ型をもとにユーザ定義データ型を作成することはできません。

sysname データ型および **longsysname** データ型には、この規則は適用されません。**sysname** および **longsysname** はユーザ定義データ型ですが、このデータ型を使用して、ユーザ定義データ型を作成できます。

ユーザ定義データ型はデータベース・オブジェクトです。名前では大文字、小文字が区別され、識別子の規則に従っている必要があります。

sp_bindrule を使用して、ユーザ定義データ型にルールをバインドできます。また、**sp_bindefault** を使用して、ユーザ定義データ型にデフォルトをバインドできます。

ユーザ定義データ型をもとに構築されたオブジェクトは、デフォルトでユーザ定義データ型の **null** 型または **IDENTITY** プロパティを継承します。カラムの定義時に **null** タイプまたは **IDENTITY** プロパティを上書きできます。

ユーザ定義データ型の名前を変更するには、**sp_rename** を使用します。

データベースからユーザ定義データ型を削除するには、**sp_droptype** を使用します。

注意 テーブルで既に使用されているデータ型は、削除できません。

システム・データ型またはユーザ定義データ型のプロパティの情報を表示するには、**sp_help** を使用します。**sp_help** は、テーブル内の各カラムのデータ型、長さ、精度、位取りを表示するためにも使用できます。

標準規格と準拠レベル

ANSI SQL - 準拠レベル：ユーザ定義データ型は Transact-SQL 拡張機能です。

Transact-SQL 関数

この章では、各 Transact-SQL 関数について説明します。関数は、データベースから情報を返すために使用されます。**where** 句内の **select** リスト、および式を使用できる任意の場所で使用可能です。関数は、ストアド・プロシージャまたはプログラムの一部としてよく使用されます。

これらの関数の使用方法の詳細については、『Transact-SQL ユーザーズ・ガイド』の「第 16 章 クエリでの Transact-SQL 関数の使用」を参照してください。

`xmlextract`、`xmlparse`、`xmlrepresentation`、`xmltable`、`xmltest`、および `xmlvalidate` の各 XML 関数の詳細については、『XML サービス』を参照してください。

abs

説明	式の絶対値を返します。
構文	<code>abs(numeric_expression)</code>
パラメータ	numeric_expression カラム、変数、またはデータ型が真数値、概数値、通貨、またはこれらの型の 1 つに暗黙的に変換できる式です。
例	-1 の絶対値である 1 を返します。 <pre>select abs(-1) ----- 1</pre>
使用法	算術関数 abs は、指定された式の絶対値を返します。結果は、数値式と同じ型で、精度や位取りも同じです。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが abs 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 ceiling , floor , round , sign

acos

説明	指定されている余弦の角度 (ラジアン) を返します。
構文	<code>acos(cosine)</code>
パラメータ	cosine カラム名、変数、または (float 、 real 、 double precision 、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の余弦です。
例	余弦が 0.52 である角度 1.023945 ラジアンを返します。 <pre>select acos(0.52) ----- 1.023945</pre>
使用法	算術関数 acos は、指定した値を余弦とする角度 (ラジアン) を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが acos 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 cos 、 degrees 、 radians

ascii

説明	式の先頭文字の ASCII コードを返します。												
構文	<code>ascii(char_expr uchar_expr)</code>												
パラメータ	<p>char_expr 文字型のカラム名、変数、または (<code>char</code>、<code>varchar</code>、<code>nchar</code>、または <code>nvarchar</code> 型の) 定数式です。</p> <p>uchar_expr 文字型のカラム名、変数、または (<code>unichar</code> または <code>univarchar</code> 型の) 定数式です。</p>												
例	<p>ASCII コードが 70 未満の場合に、作者の姓とその先頭文字の ASCII コードを返します。</p> <pre>select au_lname, ascii(au_lname) from authors where ascii(au_lname) < 70</pre> <table> <thead> <tr> <th>au_lname</th> <th></th> </tr> </thead> <tbody> <tr> <td>Bennet</td> <td>66</td> </tr> <tr> <td>Blotchet-Halls</td> <td>66</td> </tr> <tr> <td>Carson</td> <td>67</td> </tr> <tr> <td>DeFrance</td> <td>68</td> </tr> <tr> <td>Dull</td> <td>68</td> </tr> </tbody> </table>	au_lname		Bennet	66	Blotchet-Halls	66	Carson	67	DeFrance	68	Dull	68
au_lname													
Bennet	66												
Blotchet-Halls	66												
Carson	67												
DeFrance	68												
Dull	68												
使用法	<ul style="list-style-type: none"> • <code>ascii</code> は文字列関数として、式の前頭文字の ASCII コードを返します。 • 文字列関数で 2 つの文字式が受け入れられるが、1 つの式が <code>unichar</code> の場合、もう 1 つの式が「拡大」され、内部で <code>unichar</code> に変換されます。これは、混合モードの式に関する既存のルールに従っています。ただし、<code>unichar</code> データが 2 倍のスペースを占有することがあるため、この変換によってトランケーションが発生することがあります。 • <code>char_expr</code> または <code>uchar_expr</code> が <code>NULL</code> の場合は、<code>NULL</code> を返します。 												
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。												
パーミッション	すべてのユーザが <code>ascii</code> 関数を実行できます。												
参照	<p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 char, to_unichar</p>												

asehostname

説明	Adaptive Server が動作する物理ホストまたは仮想ホストを返します。
構文	<code>asehostname</code>
パラメータ	なし。
例	Adaptive Server のホスト名を返します。 <pre>select asehostname() ----- linuxkernel.sybase.com</pre>
標準規格	SQL/92 および SQL/99 対応
パーミッション	<code>asehostname</code> コマンドを実行できるのは、 <code>sa_role</code> が与えられているユーザだけです。

asin

説明	指定された正弦 (サイン) の逆正弦 (アークサイン) を角度 (ラジアン) で返します。
構文	<code>asin(<i>sine</i>)</code>
パラメータ	<i>sine</i> カラム名、変数、または (float、real、double precision、またはこれらの型の1つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の正弦です。
例	<pre>select asin(0.52) ----- 0.546851</pre>
使用法	算術関数 <code>asin</code> は、指定した値を正弦とする角度 (ラジアン) を返します。
標準規格	ANSI SQL – 準拠レベル: Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>asin</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 degrees , radians , sin

atan

説明	指定された正接 (タンジェント) の逆正接 (アークタンジェント) を角度 (ラジアン) で返します。
構文	<code>atan(<i>tangent</i>)</code>
パラメータ	<i>tangent</i> カラム名、変数、または (float、real、double precision、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の正接です。
例	<pre>select atan(0.50) ----- 0.463648</pre>
使用法	算術関数 <code>atan</code> は、指定した値を正接とする角度 (ラジアン) を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>atan</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 atn2 , degrees , radians , tan

atn2

説明	指定された正弦 (サイン) と余弦 (コサイン) の逆正接 (アーктanジェント) を角度 (ラジアン) で返します。
構文	<code>atn2(<i>sine</i>, <i>cosine</i>)</code>
パラメータ	sine カラム名、変数、または (float、real、double precision、またはこれらの型の1つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の正弦です。 cosine カラム名、変数、または (float、real、double precision、またはこれらの型の1つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の余弦です。
例	<pre>select atn2(.50, .48) ----- 0.805803</pre>
使用法	算術関数 atn2 は、正弦と余弦が指定されている角度 (ラジアン) を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが atn2 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 atan , degrees , radians , tan

avg

説明

すべて (重複を除く) の値の平均値を計算します。

構文

```
avg([all | distinct] expression)
```

パラメータ**all**

すべての値に **avg** を適用します。デフォルト設定は **all** です。

distinct

avg を適用する前に重複する値を削除します。**distinct** はオプションです。

expression

カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、**expression** はカラム名です。詳細については、「[式](#)」(325 ページ) を参照してください。

例

例 1 すべてのビジネス雑誌について前払い金の平均と総売上額を計算します。これらの各集合関数は、検索したすべてのローに対して 1 つの合計値を返します。

```
select avg(advance), sum(total_sales)
from titles
where type = "business"
```

```
-----
                        6,281.25      30788
```

例 2 **group by** 句を指定すると、集合関数はテーブル全体ではなくグループごとに 1 つの値を計算します。この文では、本の種類ごとに合計値を計算します。

```
select type, avg(advance), sum(total_sales)
from titles
group by type
```

```
type
-----
UNDECIDED                NULL          NULL
business                  6,281.25     30788
mod_cook                   7,500.00     24278
popular_comp               7,500.00     12875
psychology                 4,255.00      9939
trad_cook                  6,333.33     19566
```

例 3 `titles` テーブルは出版社別のグループに分けられ、前払い総額が 25,000 ドルを超え、本の平均価格が 15 ドルを超える出版社のグループだけが合計されます。

```
select pub_id, sum(advance), avg(price)
from titles
group by pub_id
having sum(advance) > $25000 and avg(price) > $15
```

pub_id	sum(advance)	avg(price)
0877	41,000.00	15.41
1389	30,000.00	18.98

使用法

- **avg** 集合関数は、カラム内の値の平均を計算します。**avg** を使用できるのは、数値 (整数、浮動小数、通貨) のデータ型だけです。平均値の計算では `null` 値は無視されます。
- `int`、`smallint`、`tinyint` データ (符号付きまたは符号なし) を平均すると、Adaptive Server では結果が `int` 値として返されます。`bigint` データ (符号付きまたは符号なし) を平均すると、結果は `bigint` 値として返されます。DB-Library プログラムでのオーバーフロー・エラーを避けるために、結果を格納するのに適した変数を宣言してください。
- バイナリ・データ型では、**avg** を使用できません。
- 平均値は数値データ型としてのみ定義されているため、**avg** に Unicode 式を指定して実行すると、エラーが発生します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが **avg** 関数を実行できます。

参照

マニュアル Transact-SQL ユーザーズ・ガイド

関数 [max](#)、[min](#)

audit_event_name

説明 監査イベントの説明を返します。

構文 `audit_event_name(event_id)`

パラメータ `event_id`
監査イベントの番号です。

例 **例 1** テーブル作成イベントの監査証跡を問い合わせます。

```
select * from audit_data where audit_event_name(event) = "Create Table"
```

例 2 現在の監査イベント値を取得します。監査の値と説明のリストについては、下の「使用法」を参照してください。

```
create table #tmp(event_id int, description varchar(255))
go
declare @a int
select @a=1
while (@a<120)
begin
    insert #tmp values (@a, audit_event_name(@a))
    select @a=@a + 1
end
select * from #tmp
go
```

```
-----
event_id  description
-----
         1  Ad hoc Audit Record
         2  Alter Database
        ...
       104  Create Index
       105  Drop Index
```

使用法 次のリストに、監査イベントの ID と名前を示します。

1	特定の監査レコード	38	ストアド・プロシージャの実行	74	監査の無効化
2	データベースの変更	39	トリガの実行	75	NULL
3	テーブルの変更	40	grant コマンド	76	SSO パスワードの変更
4	BCP 入力	41	テーブルの挿入	79	NULL
5	NULL	42	ビューの挿入	80	役割チェックの実行
6	デフォルトのバインド	43	データベースのロード	81	DBCC コマンド
7	メッセージのバインド	44	トランザクションのロード	82	設定
8	ルールのバインド	45	ログイン	83	データベースのオンライン化
9	データベースの作成	46	ログアウト	84	setuser コマンド
10	テーブルの作成	47	revoke コマンド	85	ユーザ定義関数コマンド
11	プロシージャの作成	48	RPC 入力	86	組み込み関数
12	トリガの作成	49	RPC 出力	87	ディスクの解放
13	ルールの作成	50	サーバのブート	88	set SSA コマンド
14	デフォルトの作成	51	サーバの停止	90	connect コマンド
15	メッセージの作成	52	NULL	91	参照
16	ビューの作成	53	NULL	92	コマンド・テキスト
17	データベースへのアクセス	54	NULL	93	JCS install コマンド
18	テーブルの削除	55	役割のオン/オフ	94	JCS remove コマンド
19	ビューの削除	56	NULL	95	管理者アカウントのロック解除
20	ディスクの初期化	57	NULL	96	quiesce database コマンド
21	ディスクの修復	58	NULL	97	SQLJ 関数の作成
22	ディスクの再初期化	59	NULL	98	SQLJ 関数の削除
23	ディスクのミラーリング	60	NULL	99	SSL の管理
24	ディスクのミラーリング解除	61	監査テーブルへのアクセス	100	ディスクのサイズ変更
25	ディスクの再ミラーリング	62	テーブルの選択	101	データベースのマウント
26	データベースの削除	63	ビューの選択	102	データベースのマウント解除
27	テーブルの削除	64	テーブルのトランケート	103	login コマンド
28	プロシージャの削除	65	NULL	104	インデックスの作成
29	トリガの削除	66	NULL	105	インデックスの削除
30	ルールの削除	67	デフォルトのバインド解除	106	NULL
31	デフォルトの削除	68	ルールのバインド解除	107	NULL
32	メッセージの削除	69	メッセージのバインド解除	108	NULL
33	ビューの削除	70	テーブルの更新	109	NULL
34	データベースのダンプ	71	ビューの更新	110	UDWS の展開
35	トランザクションのダンプ	72	NULL	111	UDWS の展開解除
36	致命的なエラー	73	監査の有効化	115	パスワードの管理
37	致命的でないエラー				

注意 audit_event_name が NULL を返す場合、Adaptive Server はイベントをログ記録しません。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが audit_event_name を実行できます。

参照

コマンド select、sp_audit

authmech

説明	指定したログイン済みサーバのプロセス ID が使用している認証メカニズムを判断します。
構文	<code>authmech ([spid])</code>
例	<p>例 1 サーバ・プロセス ID 42 の認証メカニズムとして、KERBEROS、LDAP、またはその他のメカニズムを返します。</p> <pre>select authmech(42)</pre> <p>例 2 現在のログインのサーバ・プロセス ID の認証メカニズムを返します。</p> <pre>select authmech()</pre> <p>または</p> <pre>select authmech(0)</pre> <p>例 3 各ログイン・セッションに使用される認証メカニズムを出力します。</p> <pre>select suid, authmech(spid) from sysprocesses where suid!=0</pre>
使用法	<ul style="list-style-type: none">この関数は 1 つのオプション引数から型 <code>varchar</code> の出力を返します。サーバ・プロセス ID の値が 0 の場合は、現在のクライアント・セッションのサーバ・プロセス ID で使用されている認証方式を返します。引数を指定しない場合は、サーバ・プロセス ID の値が 0 の場合と同じ出力になります。可能な戻り値は、<code>ldap</code>、<code>ase</code>、<code>pam</code>、<code>NULL</code> などです。
パーミッション	すべてのユーザが <code>authmech</code> を使用して現在の個人セッションにクエリできます。別のユーザのセッションの詳細をクエリするには、 <code>sso_role</code> 権限が必要です。

biginttohex

説明	指定した整数値に相当する、プラットフォームに依存しない 8 バイトの 16 進数を返します。
構文	<code>biginttohex (integer_expression)</code>
パラメータ	<i>integer_expression</i> 16 進文字列に変換される整数値です。
例	この例は、-9223372036854775808 という <code>bigint</code> 値を 16 進文字列に変換します。 <pre>1> select biginttohex(-9223372036854775808) 2> go ----- 8000000000000000</pre>
使用法	<ul style="list-style-type: none">データ型変換関数 <code>biginttohex</code> は、指定された整数値に相当する、プラットフォームに依存しない 16 進文字列を、プレフィクス「0x」を付けずに返します。<code>biginttohex</code> 関数は、プラットフォームに依存しないで整数を 16 進文字列に変換するために使用します。<code>biginttohex</code> 関数では、<code>bigint</code> に評価される任意の式を使用できます。実行するプラットフォームに関係なく、指定の式と同じ 16 進数の値を常に返します。
参照	関数 convert , hextobigint , hextoint , inttohex

bintostr

説明	一連の 16 進数を同等の英数字または varbinary データの文字列に変換します。																																												
構文	<code>select bintostr(sequence of hexadecimal digits)</code>																																												
パラメータ	<p>sequence of hexadecimal digits</p> <p>有効な 16 進数の数列で、[0 - 9]、[a - f]、および [A - F] で構成され、プレフィクス「0x」が付きます。</p>																																												
例	<p>例 1 16 進数列「0x723ad82fe」を同じ値の英数字文字列に変換します。</p> <pre>1> select bintostr(0x723ad82fe) 2> go</pre> <p>----- 0723ad82fe</p> <p>この例では、16 進の数列とそれに相当する英数字文字列のメモリ内表現は次のようになります。</p> <table border="1" style="margin: 10px auto;"> <tr> <td colspan="11" style="text-align: center;">16 進数 (5 バイト)</td> </tr> <tr> <td style="text-align: center;">0</td><td style="text-align: center;">7</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">a</td><td style="text-align: center;">d</td><td style="text-align: center;">8</td><td style="text-align: center;">2</td><td style="text-align: center;">f</td><td style="text-align: center;">e</td><td></td> </tr> <tr> <td colspan="11" style="text-align: center;">英数字文字列 (10 バイト)</td> </tr> <tr> <td style="text-align: center;">0</td><td style="text-align: center;">7</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">a</td><td style="text-align: center;">d</td><td style="text-align: center;">8</td><td style="text-align: center;">2</td><td style="text-align: center;">f</td><td style="text-align: center;">e</td><td></td> </tr> </table> <p>この関数は 16 進数を右から左へと処理します。この例では、入力の桁数は奇数です。そのため、英数字文字列に「0」のプレフィクスが付いて出力に反映されます。</p> <p>例 2 <code>@bin_data</code> というローカル変数の 16 進数を「723ad82fe」の値に相当する英数字文字列に変換します。</p> <pre>declare @bin_data varchar(30) select @bin_data = 0x723ad82fe select bintostr(@bin_data) go</pre> <p>----- 0723ad82fe</p>	16 進数 (5 バイト)											0	7	2	3	a	d	8	2	f	e		英数字文字列 (10 バイト)											0	7	2	3	a	d	8	2	f	e	
16 進数 (5 バイト)																																													
0	7	2	3	a	d	8	2	f	e																																				
英数字文字列 (10 バイト)																																													
0	7	2	3	a	d	8	2	f	e																																				
使用法	<ul style="list-style-type: none"> • 入力の無効な文字は、出力として null になります。 • 入力是有効な varbinary データである必要があります。 • NULL の入力結果は NULL の出力になります。 																																												
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。																																												
パーミッション	bintostr は、すべてのユーザが実行できます。																																												
参照	関数 strtobin																																												

cache_usage

説明	キャッシュの使用率を、テーブルが属するキャッシュのすべてのオブジェクトの割合として返します。
構文	<code>cache_usage(table_name)</code>
パラメータ	<p>table_name</p> <p>テーブルの名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。</p>
例	<p>例 1 titles テーブルが使用しているキャッシュの割合を返します。</p> <pre>select cache_usage("titles") ----- 98.876953</pre> <p>例 2 authors テーブルが使用しているキャッシュの割合を master データベースから返します。</p> <pre>select cache_usage ("pubs2..authors") ----- 98.876953</pre>
使用法	<ul style="list-style-type: none"> • <code>cache_usage</code> は、キャッシュの使用状況をキャッシュのすべてのプールのパーセンテージとして提供します。 • <code>cache_usage</code> は、現在のオブジェクトが使用しているキャッシュの容量についての情報は提供しません。また、別のキャッシュにバウンドされている場合は、インデックスのキャッシュ使用状況についての情報も提供しません。 • (クラスタ環境) <code>cache_usage</code> は、オブジェクトが現在のノードでバウンドされているキャッシュのキャッシュ使用状況を提供します。
パーミッション	<code>cache_usage</code> は、すべてのユーザが実行できます。

case

説明	条件付き SQL 文をサポートします。value 式を使用できる場所であればどこでも使用できます。
構文	<pre> case when <i>search_condition</i> then <i>expression</i> [when <i>search_condition</i> then <i>expression</i>]... [else <i>expression</i>] end </pre> <p>case と値の構文</p> <pre> case <i>expression</i> when <i>expression</i> then <i>expression</i> [when <i>expression</i> then <i>expression</i>]... [else <i>expression</i>] end </pre>
パラメータ	<p>case case 式を開始します。</p> <p>when 探索条件または比較する式の前に置きます。</p> <p>search_condition 選択される結果に対する条件を設定するために使用します。case 式の探索条件は、where 句の中の探索条件と似ています。探索条件の詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。</p> <p>then case の結果値を指定する式の前に置きます。</p> <p>expression カラム名、定数、関数、サブクエリ、またはこれらを算術演算子やビット処理演算子でつないだ任意の組み合わせです。式の詳細については、「式」(325 ページ)を参照してください。</p>
例	<p>例 1 authors テーブルからすべての著者を選択し、特定の著者に関しては住んでいる都市を指定します。</p> <pre> select au_lname, postalcode, case when postalcode = "94705" then "Berkeley Author" when postalcode = "94609" then "Oakland Author" when postalcode = "94612" then "Oakland Author" when postalcode = "97330" then "Corvallis Author" end from authors </pre>

例 2 `discounts` テーブルの `lowqty` カラムまたは `highqty` カラムのどちらかにある最初の NULL 以外の値を返します。

```
select stor_id, discount,
       coalesce (lowqty, highqty)
from discounts
```

また、次のフォーマットを使用して同じ結果を得ることができます。これは、`coalesce` が `case` 式の省略形だからです。

```
select stor_id, discount,
       case
         when lowqty is not NULL then lowqty
         else highqty
       end
from discounts
```

例 3 `titles` テーブルから `titles` と `type` を選択します。本のタイプが UNDECIDED の場合、`nullif` は NULL 値を返します。

```
select title,
       nullif(type, "UNDECIDED")
from titles
```

また、次のフォーマットを使用して同じ結果を得ることができます。これは、`nullif` が `case` 式の省略形だからです。

```
select title,
       case
         when type = "UNDECIDED" then NULL
         else type
       end
from titles
```

例 4 エラー・メッセージが表示されます。少なくとも 1 つの式が、`null` キーワード以外である必要があるからです。

```
select price, coalesce (NULL, NULL, NULL)
from titles
```

All result expressions in a CASE expression must not be NULL.

例 5 エラー・メッセージが表示されます。`coalesce` の後ろには、少なくとも 2 つの式を配置する必要があるからです。

```
select stor_id, discount, coalesce (highqty) from discounts
```

A single coalesce element is illegal in a COALESCE expression.

使用法	<ul style="list-style-type: none">• <code>case</code> 式は、<code>if</code> 文の代わりに <code>when...then</code> 構造を使用して探索条件を表すことによって、標準 SQL 式を簡略化したものです。• <code>case</code> 式は、SQL で式が使用できる場所であればどこでも使用できます。• クエリがさまざまなデータ型を作成する場合は、<code>case</code> 式の結果のデータ型は、データ型の階層によって決定されます。詳細については、「混合モードの式のデータ型」(6 ページ)を参照してください。Adaptive Server で、暗黙的に変換されないデータ型 (<code>char</code> や <code>int</code> など) を 2 つ指定した場合、クエリは正常に動作しません。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	<code>case</code> パーミッションは、すべてのユーザに対してデフォルトで設定されています。これを使用するためのパーミッションは必要ありません。
参照	コマンド coalesce , nullif , if...else , select , where 句

cast

説明

指定された値を別のデータ型に変換します。

構文

```
cast (expression as datatype [(length | precision[, scale]]))
```

パラメータ

expression

あるデータ型や日付フォーマットから別のデータ型や日付フォーマットに変換される値です。カラム、定数、関数、定数の任意の組み合わせ、および算術演算子やビット処理演算子またはサブクエリで連結された関数が含まれます。

データベースで Java が実行可能な場合、*expression* には、Java-SQL クラスに変換される値を指定できます。

変換先データ型として *unichar* が使用されており、長さが指定されていない場合には、デフォルトの長さである 30 Unicode 値が使用されます。

length

char, *nchar*, *unichar*, *univarchar*, *varchar*, *nvarchar*, *binary*, *varbinary* データ型とともに使用されるオプションのパラメータです。この引数を指定しないと、Adaptive Server は文字型のデータを 30 文字にトランケートし、バイナリ型のデータを 30 バイトにトランケートします。文字型データおよびバイナリ型データの最大長は 64K です。

precision

データ型が *numeric* または *decimal* のときの有効桁数です。float データ型のときは仮数内の有効バイナリ桁数です。*numeric* データ型および *decimal* のデータ型の場合、この引数を指定しないと、Adaptive Server はデフォルトの精度 18 を使います。

scale

データ型が *numeric* や *decimal* の場合の、小数点の右側の桁数です。この引数を指定しないと、Adaptive Server はデフォルトの位取り 0 を使います。

例

例 1 日付を読みやすい *datetime* フォーマットに変換します。

```
select cast("01/03/63" as datetime)
go
```

```
-----
                Jan  3 1963 12:00AM
```

```
(1 row affected)
```

例 2 *title* データベース内の *total_sales* カラムを 12 文字のカラムに変換します。

```
select title, cast(total_sales as char(12))
```

使用法

- `cast` は `date` と `time` のデータ型にデフォルトのフォーマットを使用します。
- `cast` 関数の引数が、この関数に定義されている範囲の外にあるときは、ドメイン・エラーとなります。このエラーが起こることはほとんどありません。
- ターゲット・カラムが `null` 入力可能かどうかを指定するには、`null` または `not null` を使用してください。`select into` で `null` または `not null` を使用して新しいテーブルを作成し、ソース・テーブル内の既存カラムのデータ型と `null` 入力可能性を変更することができます。
- `cast` 関数を使用して、`image` カラムを `binary` または `varbinary` に変換できます。変換は、サーバの論理ページ・サイズの最大カラム・サイズによって決定される `binary` データ型の最大長に制限されます。長さを指定しない場合、変換された値はデフォルトの長さである 30 文字になります。
- `unicar` 式を変換先データ型として使用できます。また、この式を別のデータ型へ変換できます。`unicar` 式と、サーバで使用できる任意のデータ型との間で、明示的または暗黙的にデータ型を変換できます。
- `unicar` が変換先の型として使用されているときに長さを指定しないと、デフォルトの長さである 30 Unicode 値が使用されます。変換先の型の長さが、指定された式を格納するのに十分でない場合は、エラー・メッセージが表示されます。

暗黙的変換

プライマリ・フィールドが一致しない場合、データ型の暗黙的な変換が行われると、データのトランケーション、デフォルト値の挿入、またはエラー・メッセージの発生のおいずれかが生じる場合があります。たとえば、`datetime` 値を `date` 値に変換すると、時刻部分がトランケートされ、日付部分のみが残ります。`time` 値を `datetime` 値に変換すると、新しい `datetime` 値の日付部分に、デフォルト日付の Jan 1, 1900 が追加されます。`date` 値を `datetime` 値に変換すると、`datetime` 値の時刻部分に、デフォルト時刻の 00:00:00:000 が追加されます。

```
DATE -> VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME
TIME -> VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME
VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME -> DATE
VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME -> TIME
```

明示的変換

`date` を `datetime` に明示的に変換しようとしたとき、その値が `datetime` の範囲外 (たとえば「Jan 1, 1000」) の場合は、変換が許可されず、そのことを知らせるエラー・メッセージが表示されます。

```
DATE -> UNICHAR, UNIVARCHAR
TIME -> UNICHAR, UNIVARCHAR
UNICHAR, UNIVARCHAR -> DATE
UNICHAR, UNIVARCHAR -> TIME
```

Java クラスに関連する変換

- データベースで Java が実行可能な場合は、**cast** を使用して、次の方法でデータ型を変更できます。
 - Java オブジェクト・タイプを SQL データ型に変換する。
 - SQL データ型を Java タイプに変換する。
 - 式 (ソース・クラス) のコンパイル時のデータ型がターゲット・クラスのサブクラスまたはスーパークラスである場合、Adaptive Server にインストールされた任意の Java-SQL クラスを、Adaptive Server にインストールされた任意の他の Java-SQL クラスに変換する。

変換の結果は、現在のデータベースと関連付けられます。

標準規格

ANSI SQL – 準拠レベル：ANSI レベル。

パーミッション

すべてのユーザが **cast** 関数を実行できます。

ceiling

説明

指定された値以上の最小の整数を返す。

構文

`ceiling(value)`

パラメータ

value

カラム、変数、またはデータ型が真数値、概数値、通貨、またはこれらの型の1つに暗黙的に変換できる式です。

例

例 1 124 の値を返します。

```
select ceiling(123.45)
124
```

例 2 -123 の値を返します。

```
select ceiling(-123.45)
-123
```

例 3 24.000000 の値を返します。

```
select ceiling(1.2345E2)
24.000000
```

例 4 -123.000000 の値を返します。

```
select ceiling(-1.2345E2)
-123.000000
```

例 5 124.00 の値を返します。

```
select ceiling($123.45)
124.00
```

例 6 `title_id` が値「PS3333」の `salesdetail` テーブルから「discount」の値を返します。

```
select discount, ceiling(discount) from salesdetail where
title_id = "PS3333"

discount
-----
45.000000          45.000000
46.700000          47.000000
46.700000          47.000000
50.000000          50.000000
```

使用法	<ul style="list-style-type: none">• ceiling は算術関数として、指定した値以上の最小整数値を返します。戻り値のデータ型は、指定した値のデータ型と同じです。 numeric 値や decimal 値の場合は、結果は指定した値と同じ精度で、位取りは 0 です。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが ceiling 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド コマンド set 関数 abs , floor , round , sign

char

説明 整数に相当する文字を返します。

構文 `char(integer_expr)`

パラメータ *integer_expr*
 整数値型 (`tinyint`、`smallint`、または `int`) のカラム名、変数、または 0 ~ 255 の定数式のいずれかです。

例 1

```
select char(42)
-
*
```

例 2

```
select xxx = char(65)
xxx
---
A
```

使用法

- `char` 文字列関数は、シングルバイト整数値を文字値に変換します (`char` は通常 `ascii` の逆変換として使用されます)。
- `char` は `char` データ型を返します。結果の値がマルチバイト文字の最初のバイトの場合、その文字は未定義であることがあります。
- *char_expr* が `NULL` の場合は、`NULL` を返します。

char を使った出力フォーマットの変更

- 連結値や `char` 値を使ってタブや復帰改行文字を追加し、出力フォーマットを変更できます。`char(10)` は復帰改行文字に変換され、`char(9)` はタブに変換されます。たとえば、次のようにします。

```
/* just a space */
select title_id + " " + title from titles where title_id = "T67061"
/* a return */
select title_id + char(10) + title from titles where title_id = "T67061"
/* a tab */
select title_id + char(9) + title from titles where title_id = "T67061"
-----
T67061 Programming with Curses
-----
T67061

Programming with Curses
-----
T67061      Programming with Curses
```

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが char 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 ascii , str

char_length

説明	式の文字数を返します。
構文	<code>char_length(char_expr uchar_expr)</code>
パラメータ	<p>char_expr 文字型のカラム名、変数、または (<code>char</code>、<code>varchar</code>、<code>nchar</code>、<code>text_locator</code>、<code>unitext_locator</code>、または <code>nvarchar</code> 型の) 定数式です。</p> <p>uchar_expr 文字型のカラム名、変数、または (<code>unichar</code> または <code>univarchar</code> 型の) 定数式です。</p>

例

例 1

```
select char_length(notes) from titles
       where title_id = "PC9999"
```

```
-----
                39
```

例 2

```
declare @var1 varchar(20), @var2 varchar(20), @char char(20)
select @var1 = "abcd", @var2 = "abcd  ", @char = "abcd"
select char_length(@var1), char_length(@var2),
       char_length(@char)
```

```
-----
                4                8                20
```

使用法

- `char_length` 文字列関数は、文字式やテキスト値の中の文字数を表す整数を返します。
- 圧縮されたラージ・オブジェクト (LOB) カラムに対して、`char_length` は元のプレーン・テキストの文字数を返します。
- 可変長のカラムおよび変数の場合、`char_length` は (カラムまたは変数の定義された長さではなく) 文字数を返します。明示的な後続ブランクが可変長変数に含まれている場合、それらは取り除かれませんが、リテラルや固定長カラムまたは固定長変数の場合、`char_length` は式から後続ブランクを削除しません (例 2 参照)。
- `unitext`、`unichar`、`univarchar` カラムの場合、`char_length` は Unicode 値 (16 ビット) の数を返します。このとき、1 つのサロゲート・ペアは 2 つの Unicode 値としてカウントされます。たとえば、`unitext` のカラム `ut` に `U+0041U+0042U+d800dc00` というロー値が保管されている場合、次の値が返されます。

```
select char_length(ut) from unitable
-----
                4
```

- マルチバイト文字セットの場合、式の文字数は、通常、バイト数より少なくなります。バイト数を指定するには、`datalength` 関数を使います。

- Unicode 式の場合、式の (バイト数ではなく) Unicode 値が返されます。サロゲート・ペアは 2 つの Unicode 値としてカウントされます。
- *char_expr* または *uchar_expr* が NULL の場合は、**char_length** は NULL を返します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが **char_length** 関数を実行できます。

参照

マニュアル Transact-SQL ユーザーズ・ガイド

関数 [datalength](#)

charindex

説明	式の開始位置を表す整数を返します。
構文	<code>charindex(expression1, expression2 [, start])</code>
パラメータ	<p>expression バイナリ型または文字型のカラム名、変数、または定数式です。char、varchar、nchar、nvarchar、unichar、univarchar、binary、text_locator、unitext_locator、image_locator、または varbinary を使用できます。</p> <p>start これが指定されると、expression1 の検索が expression2 で指定されたオフセットから開始します。start が指定されていない場合、検索は expression2 の始めから開始します。start には式も使用できますが、戻り値は整数値である必要があります。</p>
例	<p>例 1 titles テーブルの notes カラムにある文字式「wonderful」の開始位置を返します。</p> <pre>select charindex("wonderful", notes) from titles where title_id = "TC3218" ----- 46</pre> <p>例 2 このクエリは正常に実行され、ゼロ個のローが返ります。spt_values.name カラムは varchar(35) として定義されています。</p> <pre>select name from spt_values where charindex('NO', name, 1000) > 0</pre> <p>それに比較して、次のクエリでは start を使用しないで、titles テーブルの notes カラム内で文字式「wonderful」が開始する位置を返します。</p> <pre>select charindex("wonderful", notes) from titles where title_id = "TC3218" ----- 46</pre>
使用法	<ul style="list-style-type: none"> • charindex は、expression1 の最初の出現例を expression2 内で検索し、その開始位置を示す整数を返します。expression1 が見つからないときは charindex は 0 を返します。 • expression1 にワイルドカード文字が含まれている場合、charindex はこれらのワイルドカード文字をリテラルとして扱います。 • expression2 が NULL の場合は、0 を返します。

- `varchar` 式と `unichar` 式がそれぞれパラメータとして指定されている場合には、`varchar` 式は暗黙的に `unichar` に変換されます (変換時にトランケーションが発生することがあります)。
- `expression1` または `expression2` のどちらか一つのみがロケータの場合、もう一方の式のデータ型はロケータが参照している LOB のデータ型に暗黙的に変換できるデータ型でなければなりません。
- `expression1` がロケータの場合、参照先の LOB の最大長は 16KB です。
- `start` の値は検索対象のデータ型が `varchar`、`univarchar`、`text_locator`、または `unitext_locator` の場合は検索を開始する前に飛ばす文字数として解釈され、データ型が `binary` または `image_locator` の場合はバイト数として解釈されます。
- `expression1` の最大長は 16,384 バイトです。
- `varchar` 式と `unichar` 式がそれぞれパラメータとして指定されている場合には、`varchar` 式は暗黙的に `unichar` に変換されます (変換時にトランケーションが発生することがあります)。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `charindex` 関数を実行できます。

参照

マニュアル Transact-SQL ユーザーズ・ガイド

関数 [patindex](#)

coalesce

説明 条件付き SQL 式をサポートします。値式が使用できる場所であればどこでも使用できます。[case](#) 式の代わりに使用できます。

構文 `coalesce(expression, expression [, expression]...)`

パラメータ

coalesce

リストされた式を評価し、最初の null 以外の値を返します。すべての式が null 値の場合、**coalesce** は NULL 値を返します。

expression

カラム名、定数、関数、サブクエリ、またはこれらを算術演算子やビット処理演算子でつないだ任意の組み合わせです。式の詳細については、「[式](#)」([325 ページ](#))を参照してください。

例

例 1 discounts テーブルの **lowqty** カラムまたは **highqty** カラムのどちらかにある最初の null 以外の値を返します。

```
select stor_id, discount,
       coalesce (lowqty, highqty)
from discounts
```

例 2 例 1 の別の表記です。

```
select stor_id, discount,
       case
         when lowqty is not NULL then lowqty
         else highqty
       end
from discounts
```

使用法

- **coalesce** 式は、**when...then** 構造を使用する代わりに、簡単な探索条件を表すことによって、標準 SQL 式を簡略化したものです。
- **coalesce** 式は、SQL 内で式が使用できる場所であればどこでも使用できます。
- **coalesce** 式の 1 つ以上の結果が、null 以外の値を返す必要があります。次の例では、以下のエラー・メッセージが表示されます。

```
select price, coalesce (NULL, NULL, NULL)
from titles
```

All result expressions in a CASE expression must not be NULL.

- クエリがさまざまなデータ型を作成する場合は、**case** 式の結果のデータ型は、「混合モードの式のデータ型」(6 ページ)で説明されているとおり、データ型の階層によって決定されます。Adaptive Server で暗黙的に変換されないデータ型 (**char** や **int** など) を 2 つ指定した場合、クエリは正常に動作しません。
- coalesce** は、**case** 式を省略した形です。例 2 は、**coalesce** 文の別の記述方法を示します。
- coalesce** の後ろには、少なくとも 2 つの式が必要です。次の例では、以下のエラー・メッセージが表示されます。

```
select stor_id, discount, coalesce (highqty)
from discounts
```

A single coalesce element is illegal in a COALESCE expression.

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

coalesce は、すべてのユーザが実行できます。

参照

コマンド [case](#), [nullif](#), [select](#), [if...else](#), [where](#) 句

col_length

説明	カラムの定義済みの長さを返します。
構文	<code>col_length(object_name, column_name)</code>
パラメータ	<p>object_name データベース・オブジェクト (テーブル、ビュー、プロシージャ、トリガ、デフォルト、またはルールなど) の名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。名前は引用符で囲んでください。</p> <p>column_name カラムの名前です。</p>
例	<p>titles テーブル内の title カラムの長さを求めます。「x」は結果のカラム見出しです。</p> <pre>select x = col_length("titles", "title") x ----- 80</pre>
使用法	<ul style="list-style-type: none"> • col_length システム関数は、カラムの定義済みの長さを返します。 • 各ローに格納されているデータの実際の長さを求めるには、datalength を使います。 • text、unitext、image カラムの場合は、col_length は 16 を返します。これは、実際のテキスト・ページを示す binary(16) ポインタの長さです。 • unichar カラムの場合、定義されている長さは、(表されるバイトの数ではなく) カラムの定義時に宣言されている Unicode 値の数です。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが col_length 関数を実行できます。
参照	<p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 datalength</p>

col_name

説明	指定されたテーブルとカラム ID に対応するカラム名を返します。カラム名は、最長で 255 バイトです。
構文	<code>col_name(object_id, column_id [, database_id])</code>
パラメータ	object_id テーブル、ビュー、またはその他のデータベース・オブジェクトのオブジェクト ID を表す数値式です。この ID は <code>sysobjects</code> の <code>id</code> カラムに格納されています。 column_id カラムのカラム ID である数値式です。この式は <code>syscolumns</code> の <code>colid</code> カラムに格納されています。 database_id データベースの ID である数値式です。この ID は <code>sysdatabases</code> の <code>db_id</code> カラムに格納されています。
例	<pre>select col_name(208003772, 2) ----- title</pre>
使用法	<ul style="list-style-type: none">• <code>col_name</code> システム関数は、カラムの名前を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>col_name</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 db_id , object_id

compare

説明	代替照合規則をもとに、2つの文字列を直接比較できるようにします。
構文	<pre>compare ({char_expression1 uchar_expression1}, {char_expression2 uchar_expression2}), [{collation_name collation_ID}]</pre>
パラメータ	<p><i>char_expression1</i> または <i>uchar_expression1</i> <i>char_expression2</i> または <i>uchar_expression2</i> と比較する文字式です。</p> <p><i>char_expression2</i> または <i>uchar_expression2</i> <i>char_expression1</i> または <i>uchar_expression1</i> と比較する文字式です。 <i>char_expression1</i> および <i>char_expression2</i> は、次のいずれかです。</p> <ul style="list-style-type: none"> • 文字型 (char、varchar、nchar、または nvarchar) • 文字変数 • 文字定数の式 (一重引用符または二重引用符で囲む) <p><i>uchar_expression1</i> および <i>uchar_expression2</i> は、次のいずれかです。</p> <ul style="list-style-type: none"> • 文字型 (unicchar または univarchar) • 文字変数 • 文字定数の式 (一重引用符または二重引用符で囲む) <p><i>collation_name</i> 引用符で囲まれた文字列、または使用する照合動作を指定する文字変数です。表 2-2 (82 ページ) に、有効値を示します。</p> <p><i>collation_ID</i> 整数の定数、または使用する照合動作を指定する変数です。表 2-2 (82 ページ) に有効値を示します。</p>
例	<p>例 1 aaa と bbb を比較します。</p> <pre>1> select compare ("aaa","bbb") 2> go ----- -1 (1 row affected)</pre> <p>または、次のようなフォーマットでも、aaa と bbb を比較できます。</p> <pre>1> select compare (("aaa"),("bbb")) 2> go ----- -1 (1 row affected)</pre>

例 2 aaa と bbb を比較し、バイナリ・ソート順を指定します。

```
1> select compare ("aaa","bbb","binary")
2> go

-----
          -1
(1 row affected)
```

または、次のようなフォーマットで aaa と bbb を比較し、照合名の代わりに照合 ID を使用することもできます。

```
1> select compare ("aaa"), ("bbb"), (50)
2> go

-----
          -1
(1 row affected)
```

使用法

- **compare** 関数は、選択した照合規則をもとに次の値を返します。
 - 1 – *char_expression1* または *uchar_expression1* が *char_expression2* または *uchar_expression2* より大きいことを示します。
 - 0 – *char_expression1* または *uchar_expression1* が *char_expression2* または *uchar_expression2* と等しいことを示します。
 - -1 – *char_expression1* または *uchar_expression1* が *char_expression2* または *uchar_expression2* よりも小さいことを示します。
- **compare** は、それぞれの入力文字について 6 バイトの照合情報を生成します。したがって、**compare** の実行結果は、**varbinary** データ型の長さの制限値を超える可能性があります。この制限値を超える場合、結果は制限値に収まるようにトランケートされます。Adaptive Server からは警告メッセージが表示されますが、**compare** 関数を含むクエリまたはトランザクションは引き続き実行されます。この制限はサーバの論理ページ・サイズに依存するので、DOL テーブルおよび APL テーブルでは、結果の文字列が次のサイズより小さくなるまで、各入力文字の結果バイトはトランケートによって削除されます。

表 2-1: ローとカラムの最大長 - APL および DOL

ロック・スキーム	ページ・サイズ	ローの最大長	カラムの最大長
APL テーブル	2K (2048 バイト)	1962	1960 バイト
	4K (4096 バイト)	4010	4008 バイト
	8K (8192 バイト)	8106	8104 バイト
	16K (16384 バイト)	16298	16296 バイト
DOL テーブル	2K (2048 バイト)	1964	1958 バイト
	4K (4096 バイト)	4012	4006 バイト
	8K (8192 バイト)	8108	8102 バイト
	16K (16384 バイト)	16300	16294 バイト - テーブルに変長カラムがない場合
	16K (16384 バイト)	16300 (varlen = 8191 の最大開始オフセットによって変化)	8191-6-2 = 8183 バイト - 1 つ以上の可変長カラムがテーブルにある場合*

* このサイズには、ローのオーバヘッドの 6 バイトとローの長さのフィールドの 2 バイトが含まれる。

- *char_expression1* と *uchar_expression1*、および *char_expression2* と *uchar_expression2* は、サーバのデフォルト文字セットでエンコードされている文字である必要があります。
- *char_expression1* と *char_expression2*、または *uchar_expression1* と *uchar_expression2* では、いずれかまたは両方に空文字列を指定できます。
 - *char_expression2* または *uchar_expression2* が空文字列の場合は、1 が返されます。
 - 両方とも空文字列の場合は、等しいとみなされ、0 が返されます。
 - *char_expression1* または *uchar_expression1* が空文字列の場合は、-1 が返されます。

compare 関数では、空の文字列とスペースだけの文字列は等しいとみなされません。**compare** は、**sortkey** 関数を使用して、比較のための照合キーを生成します。したがって、まったく空の文字列、1 つのスペースからなる文字列、または 2 つのスペースからなる文字列は、等しいとはみなされません。

- *char_expression1* または *uchar_expression1*、あるいは *char_expression2* または *uchar_expression2* が NULL の場合、結果は NULL になります。
- **varchar** 式と **unichar** 式がそれぞれパラメータとして指定されている場合には、**varchar** 式は暗黙的に **unichar** に変換されます (変換時にトランケーションが発生することがあります)。
- *collation_name* または *collation_ID* に値を指定しない場合、**compare** はバイナリ照合とみなします。
- 表 2-2 は、*collation_name* と *collation_ID* の有効値のリストを示します。

表 2-2: 照合名と ID

説明	照合名	照合 ID
デフォルト Unicode マルチ言語	default	20
タイ語辞書順	thaidict	21
ISO14651 規格	iso14651	22
UTF-16 順 – UTF-8 バイナリ順と一致	utf8bin	24
CP 850 代替言語 – アクセント記号なし	altnoacc	39
CP 850 代替言語、小文字優先	altdict	45
CP 850 西欧言語 – 大文字/小文字の優先設定なし	altnocsp	46
CP 850 スカンジナビア語 – 辞書順	scandict	47
CP 850 スカンジナビア語辞書順 – 大文字と小文字を区別しない、優先度を付けた順位	scannoep	48
GB ピンイン	gbpinyin	なし
バイナリ・ソート	binary	50
Latin-1 英語、フランス語、ドイツ語辞書	dict	51
Latin-1 英語、フランス語、ドイツ語、大文字/小文字の区別なし	nocase	52
Latin-1 英語、フランス語、ドイツ語、大文字/小文字の優先設定なし	nocasep	53
Latin-1 英語、フランス語、ドイツ語、アクセント記号なし	noaccent	54
Latin-1 スペイン語辞書	espdict	55
Latin-1 スペイン語、大文字/小文字の区別なし	espnocs	56
Latin-1 スペイン語、アクセント記号なし	espnoac	57
ISO 8859-5 ロシア語辞書	rusdict	58
ISO 8859-5 ロシア語、大文字小文字の区別なし	rusnocs	59
ISO 8859-5 キリル語辞書	cyrdict	63
ISO 8859-5 キリル語、大文字小文字の区別なし	cyrnocs	64
ISO 8859-7 ギリシア語辞書	elldict	65
ISO 8859-2 ハンガリー語、辞書	hundict	69
ISO 8859-2 ハンガリー語、アクセント記号の区別なし	hunnoac	70
ISO 8859-2 ハンガリー語、大文字小文字の区別なし	hunnocs	71
ISO 8859-9 トルコ語辞書	turdict	72
ISO 8859-9 トルコ語、アクセント記号の区別なし	turknoac	73
ISO 8859-9 トルコ語、大文字小文字の区別なし	turknocs	74
CP932 バイナリ順	cp932bin	129
中国語、読み方順	dynix	130
GB2312 バイナリ順	gb2312bn	137
共通キリル語辞書	cyrdict	140
トルコ語辞書	turdict	155
EUCKSC バイナリ順	euckscbn	161
中国語、読み方順	gbpinyin	163
ロシア語辞書順	rusdict	165
SJIS バイナリ順	sjisbin	179

説明	照合名	照合 ID
EUCJIS バイナリ順	eucjisbn	192
BIG5 バイナリ順	big5bin	194
Shift-JIS バイナリ順	sjisbin	259

標準規格 ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション すべてのユーザが `compare` 関数を実行できます。

参照 関数 [sortkey](#)

convert

説明 指定した値を別のデータ型または異なる **datetime** 表示フォーマットに変換して返します。

構文 `convert (datatype [(length) | (precision[, scale])]
[null | not null], expression [, style])`

パラメータ

datatype

システムが提供するデータ型 (**char**(10)、**unichar** (10)、**varbinary** (50)、または **int** など) です。式はこのデータ型に変換されます。ユーザ定義データ型は使えません。

データベースで Java が実行可能な場合、**datatype** には、現在のデータベースの Java-SQL クラスも指定できます。

length

char、**nchar**、**unichar**、**univarchar**、**varchar**、**nvarchar**、**binary**、**varbinary** データ型とともに使用されるオプションのパラメータです。この引数を指定しないと、Adaptive Server は文字型のデータを 30 文字にトランケートし、バイナリ型のデータを 30 バイトにトランケートします。文字型データおよびバイナリ型データの最大長は 64K です。

precision

データ型が **numeric** または **decimal** のときの有効桁数です。**float** データ型のときは仮数内の有効バイナリ桁数です。**numeric** データ型および **decimal** のデータ型の場合、この引数を指定しないと、Adaptive Server はデフォルトの精度 18 を使います。

scale

データ型が **numeric** や **decimal** の場合の、小数点の右側の桁数です。この引数を指定しないと、Adaptive Server はデフォルトの位取り 0 を使います。

null | not null

結果式の **null** 入力可能性を指定します。**null** または **not null** のどちらも指定しない場合、変換された結果の **null** 入力可能性は、式と同じです。

expression

あるデータ型や日付フォーマットから別のデータ型や日付フォーマットに変換される値です。

データベースで Java が実行可能な場合、**expression** には、Java-SQL クラスに変換される値を指定できます。

変換先データ型として **unichar** が使用されており、長さが指定されていない場合には、デフォルトの長さである 30 Unicode 値が使用されます。

style

変換後のデータに使用する表示形式です。money データや smallmoney データを文字型に変換するときは、style を 1 にして 3 桁ごとにカンマを表示します。

datetime データや smalldatetime データを文字型に変換するときは、表 2-3 に示すスタイル番号を使って表示フォーマットを指定します。左端のカラムの値は 2 桁の年 (yy) を表します。4 桁の年 (yyyy) については、この左端のカラムの値に 100 を加えるか、または左から 2 番目のカラムにある値を使用します。

date データを文字型に変換するときは、表 2-3 に示すスタイル番号の 1 ~ 7 (101 ~ 107) または 10 ~ 12 (110 ~ 112) を使って表示フォーマットを指定します。デフォルト値は 100 (mon dd yyyy hh:miAM (または PM)) です。時刻部分を含むスタイルに date データを変換する場合は、時刻部分はデフォルト値の 0 になります。

time データを文字型に変換するときは、スタイル番号の 8 または 9 (108 または 109) を使って表示フォーマットを指定します。デフォルトは 100 (mon dd yyyy hh:miAM (または PM)) です。日付部分を含むスタイルに time データを変換する場合は、デフォルト日付の Jan 1, 1900 が表示されます。

表 2-3: style パラメータを使用した日付フォーマットの変換

世紀なし (yy)	世紀あり (yyyy)	規格	出力
-	0 または 100	デフォルト	mon dd yyyy hh:mm AM (または PM)
1	101	USA	mm/dd/yy
2	2	SQL 規格	yy.mm.dd
3	103	イギリス/フランス	dd/mm/yy
4	104	ドイツ	dd.mm.yy
5	105		dd-mm-yy
6	106		dd mon yy
7	107		mon dd, yy
8	108		HH:mm:ss
-	9 または 109	デフォルト + ミリ秒	mon dd yyyy hh:mm:ss AM (または PM)
10	110	USA	mm-dd-yy
11	111	日本	yy/mm/dd
12	112	ISO	yymmdd
13	113		dd/yy/mm
14	114		mm/yy/dd
14	114		hh:mi:ss:mmAM (または PM)
15	115		dd/yy/mm
-	16 または 116		mon dd yyyy HH:mm:ss
17	117		hh:mmAM

キー “mon” はスペルアウトした月名、“mm” は数字の月または分です。“HH” は 24 時間形式の値、“hh” は 12 時間形式の値です。最後のロー 23 には、フォーマット内で日付と時刻を区切るリテラル “T” が含まれます。

世紀なし (yy)	世紀あり (yyyy)	規格	出力
18	118		HH:mm
19			hh:mm:ss:zzzAM
20			hh:mm:ss:zzz
21			yy/mm/dd HH:mm:ss
22			yy/mm/dd HH:mm AM (または PM)
23			yyyy-mm-ddTHH:mm:ss

キー “mon” はスペルアウトした月名、“mm” は数字の月または分です。“HH” は 24 時間形式の値、“hh” は 12 時間形式の値です。最後のロー 23 には、フォーマット内で日付と時刻を区切るリテラル “T” が含まれます。

デフォルト値 (*style* が 0 か 100) のときと、*style* が 9 か 109 のときは、常に 4 桁表示 (yyyy) が返されます。smalldatetime から char や varchar に変換すると、秒やミリ秒を含むスタイルでは秒やミリ秒の位置に 0 が表示されます。

例

例 1

```
select title, convert(char(12), total_sales)
from titles
```

例 2

```
select title, total_sales
from titles
where convert(char(20), total_sales) like "1%"
```

例 3 現在の日付をスタイル 3 の dd/mm/yy に変換します。

```
select convert(char(12), getdate(), 3)
```

例 4 pubdate の値が null になる可能性がある場合は、char ではなく varchar を使ってください。そうしないと、エラーとなることがあります。

```
select convert(varchar(12), pubdate, 3) from titles
```

例 5 文字列「0x00000100」に相当する整数を返します。結果はプラットフォームにより異なることがあります。

```
select convert(integer, 0x00000100)
```

例 6 プラットフォーム固有のビット・パターンを Sybase のバイナリ型として返します。

```
select convert (binary, 10)
```

例 7 1.11 ドルに相当するビット文字列 1 を返します。

```
select convert(bit, $1.11)
```

例 8 データ型 char(100) の total_sales を持つ #tempsales を作成し、null 値は使用できないようにします。titles.total_sales が、null を使用できるように定義されている場合でも、作成される #tempsales には、null 値が許可されない #tempsales.total_sales があります。

使用法

```
select title, convert (char(100) not null, total_sales) into
#tempsales
from titles
```

- データ型変換関数 **convert** は各種データ型を変換し、表示のための日付／時刻データや通貨データのフォーマットを変更します。
- 圧縮されている場合、**convert** はデータ型を変換する前にラージ・オブジェクト (LOB) カラムを圧縮解除します。
- convert** - 指定した値を別のデータ型または異なる日付表示フォーマットに変換して返します。 **unitext** データを他の文字やバイナリのデータ型に変換する場合、変換結果は変換先のデータ型の最大長に制約されます。長さを指定しない場合、変換された値はデフォルト・サイズである 30 バイトになります。 **enable surrogate processing** を使用している場合、サロゲート・ペアは全体が返されます。たとえば、U+0041U+0042U+20acU+0043 (「AB €」を表す) というデータが保管されている **unitext** カラムを UTF-8 **varchar(3)** カラムに変換した場合、次の値が返されます。

```
select convert(varchar(3), ut) from untable
---
AB
```

- convert** 関数の引数が、この関数に定義されている範囲の外にある場合は、ドメイン・エラーが生成されます。このエラーが起こることはほとんどありません。
- ターゲット・カラムが **null** 入力可能かどうかを指定するには、**null** または **not null** を使用してください。これは特に、**select into** とともに使用すると、新しいテーブルを作成し、ソース・テーブル内の既存カラムのデータ型と **null** 入力可能性を変更することができます (前述の例 8 を参照)。

次の場合に、結果は未定義の値です。

- 変換される式の結果が **not null** になります。
- 式の値が **null** になる。

予測可能な結果を表す **NULL** 以外の既知の値を生成するには、次の **select** 文を使用します。

```
select convert(int not null, isnull(col2, 5)) from table1
```

- convert** 関数を使って、**image** カラムを **binary** または **varbinary** に変換できます。 **binary** データ型の最大長は制限されています。この最大長は、サーバの論理ページ・サイズの最大カラム・サイズによって決定します。長さを指定しない場合、変換された値はデフォルトの長さである 30 文字になります。
- unicar** 式を変換先データ型として使用できます。また、この式を別のデータ型へ変換できます。 **unicar** 式と、サーバで使用できる任意のデータ型との間で、明示的または暗黙的にデータ型を変換できます。

- `unicchar` が変換先の型として指定されているときに長さを指定しないと、デフォルトの長さである 30 Unicode 値が使用されます。変換先の型の長さが、指定された式を格納するのに十分でない場合は、エラー・メッセージが表示されます。

暗黙的変換

プライマリ・フィールドが一致しない場合、データ型の暗黙的な変換が行われると、データのトランケーション、デフォルト値の挿入、またはエラー・メッセージの発生のいずれかが生じる場合があります。たとえば、`datetime` 値を `date` 値に変換すると、時刻部分がトランケートされ、日付部分のみが残ります。`time` 値を `datetime` 値に変換すると、新しい `datetime` 値の日付部分に、デフォルト日付の Jan 1, 1900 が追加されます。`date` 値を `datetime` 値に変換すると、`datetime` 値の時刻部分に、デフォルト時刻の 00:00:00.000 が追加されます。

```
DATE -> VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME
TIME -> VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME
VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME -> DATE
VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME -> TIME
```

明示的変換

`date` を `datetime` に明示的に変換しようとしたとき、その値が `datetime` の範囲外 (たとえば「Jan 1, 1000」) の場合は、変換が許可されず、そのことを知らせるエラー・メッセージが表示されます。

```
DATE -> UNICHAR, UNIVARCHAR
TIME -> UNICHAR, UNIVARCHAR
UNICHAR, UNIVARCHAR -> DATE
UNICHAR, UNIVARCHAR -> TIME
```

Java クラスに関連する変換

- データベースで Java が実行可能な場合は、`convert` を使用して、次の方法でデータ型を変更できます。
 - Java オブジェクト・タイプを SQL データ型に変換する。
 - SQL データ型を Java タイプに変換する。
 - 式 (ソース・クラス) のコンパイル時のデータ型がターゲット・クラスのサブクラスまたはスーパークラスである場合、Adaptive Server にインストールされた任意の Java-SQL クラスを、Adaptive Server にインストールされた任意の他の Java-SQL クラスに変換する。

変換の結果は、現在のデータベースと関連付けられます。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `convert` 関数を実行できます。

参照

マニュアル 使用できるデータ型マッピングのリストと、Java クラスに関連するデータ型変換の詳細については、『Transact-SQL ユーザーズ・ガイド』、『Adaptive Server Enterprise における Java』を参照してください。

データ型 [ユーザ定義データ型](#)

関数 [hextoint](#), [inttohex](#)

COS

説明	指定した角度 (ラジアン) の余弦を返します。
構文	<code>cos(<i>angle</i>)</code>
パラメータ	<i>angle</i> 概数値型 (<code>float</code> 、 <code>real</code> 、または <code>double precision</code>) のカラム名、変数、または定数式のいずれかです。
例	<pre>select cos(44) 0.999843</pre>
使用法	<ul style="list-style-type: none">算術関数 <code>cos</code> は、指定した角度 (ラジアン) の余弦を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>cos</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 acos 、 degrees 、 radians 、 sin

cot

説明	ラジアンで指定された角度の余接を返します。
構文	<code>cot(<i>angle</i>)</code>
パラメータ	<i>angle</i> 概数値型 (<code>float</code> 、 <code>real</code> 、または <code>double precision</code>) のカラム名、変数、または定数式のいずれかです。
例	<pre>select cot(90) ----- -0.501203</pre>
使用法	<ul style="list-style-type: none">算術関数 <code>cot</code> は、指定した角度 (ラジアン) の余接を返します。
標準規格	ANSI SQL – 準拠レベル: Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>cot</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 degrees , radians , sin

count

説明	null 以外の値の数 (重複する値を除く) または選択したローの数を整数で返します。
構文	count([all distinct] expression)
パラメータ	all すべての値に count を適用します。デフォルト設定は all です。 distinct count を適用する前に重複する値を削除します。distinct はオプションです。 expression カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、expression はカラム名です。詳細については、「式」(325 ページ) を参照してください。
例	例 1 作者の居住都市の数を求めます。 <pre>select count(distinct city) from authors</pre> 例 2 titles テーブルにある種類をリストにしますが、該当する本が 1 冊だけの種類や該当する本がない種類は除きます。 <pre>select type from titles group by type having count (*) > 1</pre>
使用法	<ul style="list-style-type: none">• 集合関数 count は、カラムにある null 以外の値の数をカウントします。• distinct を指定すると、count はカラムにある null 以外のユニークな値の数をカウントします。count は、unicar などのすべてのデータ型で使用できますが、text と image では使用できません。カウントでは null 値は無視されます。• 空のテーブルや null 値だけが入っているカラムやグループでは、count(column_name) は値 0 を返します。• count(*) はローの数をカウントします。count(*) には引数がないので、distinct は指定できません。null 値の有無に関係なく、すべてのローがカウントされます。• 複数のテーブルをジョインするときは、「select リスト」に count(*) を含めて、ジョイン結果のロー数をカウントします。1 つのテーブルから基準に一致したローの数をカウントする場合は、count(column_name) を使います。

- `count` を使って、サブクエリ内にあるものが存在するかどうかを調べることができます。たとえば、次のようにします。

```
select * from tab where 0 <
    (select count(*) from tab2 where ...)
```

ただし、`count` は一致するすべての値をカウントするため、`exists` や `in` の方が結果が早く返されます。たとえば、次のようにします。

```
select * from tab where exists
    (select * from tab2 where ...)
```

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `count` 関数を実行できます。

参照

コマンド [compute 句](#), [group by 句](#) と [having 句](#), [select](#), [where 句](#)

マニュアル [Transact-SQL ユーザーズ・ガイド](#)

count_big

説明 null 以外の値の数 (重複する値を除く) または選択したローの数を bigint の値で返します。

構文 count_big([all | distinct] *expression*)

パラメータ

all

すべての値に count_big を適用します。デフォルトは all です。

distinct

count_big を適用する前に重複する値を削除します。distinct はオプションです。

expression

カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、expression はカラム名です。

例

systypes 内の *name* の数をカウントします。

```
1> select count_big(name) from systypes
2> go
-----
42
```

使用法

- count_big 集合関数は、カラムにある null 以外の値の数をカウントします。
- distinct を指定すると、count_big は、カラムにある null 以外のユニークな値の数をカウントします。カウントでは null 値は無視されます。
- 空のテーブルや null 値だけが入っているカラムやグループでは、count_big(*column_name*) は値 0 を返します。
- count_big(*) はローの数をカウントします。count_big(*) には引数がなく、distinct を指定することもできません。null 値の有無に関係なく、すべてのローがカウントされます。
- 複数のテーブルをジョインするときは、select リストに count_big(*) を含めて、ジョイン結果のロー数をカウントします。1 つのテーブルから基準と一致したローの数をカウントする場合は、count_big(*column_name*) を使用します。
- count_big を使用して、サブクエリ内にあるものが存在するかどうかを調べることができます。たとえば、次のようにします。

```
select * from tab where 0 <
(select count_big(*) from tab2 where ...)
```

ただし、count_big は一致するすべての値をカウントするため、exists や in の方が結果が早く返されます。たとえば、次のようにします。

```
select * from tab where exists
(select * from tab2 where ...)
```

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが count_big を実行できます。

参照

コマンド compute 句、group by 句および having 句、select、where 句

create_locator

説明	<p>指定された LOB のロケータを明示的に作成して、そのロケータを返します。</p> <p><code>create_locator</code> によって作成されたロケータは、<code>create_locator</code> を使用したクエリを含むトランザクションが終了するまで有効です。トランザクションが開始されなかった場合は、<code>create_locator</code> を含むクエリが実行を完了するまで有効です。</p>
構文	<code>create_locator (datatype, lob_expression)</code>
パラメータ	<p>datatype</p> <p>LOB ロケータのデータ型です。有効な値は次のとおりです。</p> <ul style="list-style-type: none">• <code>text_locator</code>• <code>unitext_locator</code>• <code>image_locator</code> <p>lob_expression</p> <p>データ型が <code>text</code>、<code>unitext</code>、または <code>image</code> の LOB の値です。</p>
例	<p>例 1 単純なテキスト式からテキスト・ロケータを作成します。</p> <pre>select create_locator(text_locator, convert (text, "abc"))</pre> <p>例 2 型が <code>text_locator</code> のローカル変数 <code>@v</code> を作成してから、<code>@v</code> を <code>my_table</code> の <code>textcol</code> カラムにある LOB へのハンドルとして使用してロケータを作成します。</p> <pre>declare @v text_locator select @v = create_locator(text_locator, textcol) from my_table where id=10</pre>
パーミッション	すべてのユーザが <code>create_locator</code> 関数を実行できます。
参照	<p>コマンド <code>deallocate locator</code>、<code>truncate lob</code></p> <p>Transact-SQL 関数 <code>locator_literal</code>、<code>locator_valid</code>、<code>return_lob</code></p>

current_bigdatetime

説明 現在の時刻を表すマイクロ秒の精度の `bigtime` 値を返します。現在の時刻部分の精度はシステム・クロックの精度によって制限されます。

構文 `current_bigdatetime()`

パラメータ なし。

例 **例 1** 現在の `bigdatetime` を求めます。

```
select current_bigdatetime()
-----
Nov 25 1995 10:32:00.010101AM
```

例 2 現在の `bigdatetime` を求めます。

```
select datepart(us, current_bigdatetime())
-----
010101
```

使用法 サーバ上にある現在の日付を取得します。

標準規格 ANSI SQL – 準拠レベル：初級レベル。

パーミッション すべてのユーザが `current_date` 関数を実行できます。

参照 **データ型** [日付と時刻のデータ型](#)

コマンド [select](#), [where 句](#)

関数 [dateadd](#)、[datediff](#)、[datepart](#)、[datetime](#)、[current_bigtime](#)

current_bigtime

説明	現在の時刻を表すマイクロ秒の精度の bigtime 値を返します。現在の時刻部分の精度はシステム・クロックの精度によって制限されます。
構文	current_bigtime()
パラメータ	なし。
例	<p>例 1 現在の bigtime を求めます。</p> <pre>select current_bigtime() ----- 10:32:00.010101AM</pre> <p>例 2 現在の bigtime を求めます。</p> <pre>select datepart(us, current_bigtime()) ----- 01010</pre>
使用法	サーバ上にある現在の日付を取得します。
標準規格	ANSI SQL – 準拠レベル：初級レベル。
パーミッション	すべてのユーザが current_date 関数を実行できます。
参照	<p>データ型 日付と時刻のデータ型</p> <p>コマンド select, where 句</p> <p>関数 dateadd、datediff、datepart、datetime、current_bigdatetime</p>

current_date

説明 現在の日付を返します。

構文 current_date()

パラメータ なし。

例 **例 1** datename を使用して現在の日付を確認します。

```
1> select datename(month, current_date())
2> go
```

```
-----
August
```

例 2 datepart を使用して現在の日付を確認します。

```
1> select datepart(month, current_date())
2> go
```

```
-----
8
```

```
(1 row affected)
```

使用法 サーバ上にある現在の日付を取得します。

標準規格 ANSI SQL – 準拠レベル：初級レベル。

パーミッション すべてのユーザーが current_date 関数を実行できます。

参照 **データ型** [日付と時刻のデータ型](#)

コマンド [select](#), [where 句](#)

関数 [dateadd](#), [datename](#), [datepart](#), [getdate](#)

current_time

説明	現在の時刻を返します。
構文	current_time()
パラメータ	なし。
例	<p>例 1 現在の時刻を検索します。</p> <pre>1> select current_time() 2> go</pre> <p>----- 12:29PM (1 row affected)</p> <p>例 2 datename と一緒に使用します。</p> <pre>1> select datename(minute, current_time()) 2> go</pre> <p>----- 45 (1 row affected)</p>
使用法	サーバ上にある現在の時刻を取得します。
標準規格	ANSI SQL – 準拠レベル：初級レベル。
パーミッション	すべてのユーザが current_time 関数を実行できます。
参照	<p>データ型 日付と時刻のデータ型</p> <p>コマンド select, where 句</p> <p>関数 dateadd, datename, datepart, getdate</p>

curunreservedpgs

説明	指定したディスク区分の空きページ数を表示します。
構文	curunreservedpgs (dbid, lstart, unreservedpgs)
パラメータ	<p>dbid データベースの ID です。この ID は <code>sysdatabases</code> の <code>db_id</code> カラムに格納されています。</p> <p>lstart 戻り値としてカウントされるページが入っているディスク区分内のページです。</p> <p>unreservedpgs 要求したデータベースの <code>dbtable</code> が現在ないときに返すデフォルト値です。</p>

例 1 デバイス・フラグメントごとに、データベース名、デバイス名、未使用ページ数を返します。

データベースが開いている場合、`curunreservedpgs` は値をメモリから取得します。データベースが使用されていない場合、値は `curunreservedpgs` に指定した 3 番目のパラメータから取得されます。この例では、値が `sysusages` テーブルの `unreservedpgs` カラムから取得されます。

```
select db_name(dbid), d.name,
       curunreservedpgs(dbid, lstart, unreservedpgs)
   from sysusages u, sysdevices d
  where u.vdevno=d.vdevno
        and d.status &2 = 2
```

```

                                     name
-----
master                               master          1634
tempdb                               master           423
model                                master           423
pubs2                                 master            72
sybssystemdb                          master           399
sybssystemprocs                       master          6577
sybsyntax                             master           359
```

(7 rows affected)

例 2 `dbid` のセグメント上の空きページ数を始点 `sysusages.lstart` から表示します。

```
select curunreservedpgs (dbid, sysusages.lstart, 0)
```

使用法	<ul style="list-style-type: none">システム関数 <code>curunreservedpgs</code> は、ディスク区分内の空きページ数を返します。データベースが開いている場合、<code>curunreservedpgs</code> から返される値はメモリから取得されます。データベースが使用されていない場合、値は <code>curunreservedpgs</code> に指定した3番目のパラメータから取得されます。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>curunreservedpgs</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 db_id, lct_admin

data_pages

説明 指定されたテーブル、インデックス、または特定のパーティションで使用されるページ数を返します。結果には、内部構造に使用するページは含まれません。この関数は、バージョン 15.0 より前の Adaptive Server の `data_pgs` および `ptn_data_pgs` を置き換えます。

構文 `data_pages(dbid, object_id [, indid [, ptnid]])`

パラメータ

dbid

データ・ページがあるデータベースのデータベース ID です。

object_id

テーブル、ビュー、またはその他のデータベース・オブジェクトのオブジェクト ID です。この ID は `sysobjects` の `id` カラムに格納されています。

indid

ターゲット・インデックスのインデックス ID です。

ptnid

ターゲット・パーティションのパーティション ID です。

例 **例 1** 指定されたデータベースにある、オブジェクト ID が 31000114 のオブジェクトで使用されるページ数を返します (インデックスも含む)。

```
select data_pages(5, 31000114)
```

例 2 (クラスタ環境で) クラスタード・インデックスがあるかどうかに関係なく、データ・レイヤに含まれるオブジェクトで使用されるページ数を返します。

```
select data_pages(5, 31000114, 0)
```

例 3 (クラスタ環境で) クラスタード・インデックスに対応するインデックス・レイヤのオブジェクトで使用されるページ数を返します。データ・レイヤで使用されるページは含まれません。

```
select data_pages(5, 31000114, 1)
```

例 4 指定されたパーティション (この例では 2323242432) のデータ・レイヤにあるオブジェクトで使用されるページ数を返します。

```
select data_pages(5, 31000114, 0, 2323242432)
```

使用法

- APL (全ページロック) テーブルにクラスタード・インデックスがある場合は、*indid* に次の値を指定します。
 - 0 - データ・ページをレポートします。
 - 1 - インデックス・ページをレポートします。

すべてのエラーで常に値 0 が返されます。たとえば、*object_id* が現在のデータベースに存在しない、ターゲットの *indid* または *ptnid* が見つからない、などの場合です。

- リソースを無駄遣いする代わりに、`data_pages` は既にキャッシュに存在しないオブジェクトの記述子を廃棄します。

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>data_pages</code> 関数を実行できます。
参照	関数 object_id, row_count システム・プロシージャ sp_spaceused

datachange

説明	<p><code>update statistics</code> が最後に実行された後にデータ分配で行われた変更の量を測定します。具体的には、特定のオブジェクト、パーティション、カラムで発生した <code>inserts</code>、<code>updates</code>、<code>deletes</code> の数が測定されます。これは、<code>update statistics</code> の実行がクエリ・プランに役立つかどうかを判断する目安になります。</p>
構文	<pre>datachange(object_name, partition_name, column_name)</pre>
パラメータ	<p>object_name 現在のデータベース内のオブジェクト名です。</p> <p>partition_name データ・パーティションの名前です。 <code>null</code> を指定できます。</p> <p>column_name <code>datachange</code> への要求の目的であるカラム名です。 <code>null</code> を指定できます。</p>
例	<p>例 1 <code>author_ptn</code> パーティション内の <code>au_id</code> カラムで行われた変更をパーセンテージで示します。</p> <pre>select datachange("authors", "author_ptn", "au_id")</pre> <p>例 2 <code>au_ptn</code> パーティション内の <code>authors</code> テーブルで行われた変更をパーセンテージで示します。 column_name パラメータに <code>null</code> 値を指定すると、ヒストグラム統計があるすべてのカラムがチェックされ、その中から最大のデータ変更値が取得されます。</p> <pre>select datachange("authors", "au_ptn", null)</pre>
使用方法	<ul style="list-style-type: none"> • <code>datachange</code> 関数には 3 つのパラメータをすべて指定する必要があります。 • <code>datachange</code> は <code>insert</code>、<code>delete</code>、および <code>update</code> を測定しますが、個別のカウントは行いません。 <code>datachange</code> は <code>update</code> を <code>delete</code> および <code>insert</code> としてカウントするため、1 つの <code>update</code> は <code>datachange</code> カウンタを 2 つ進めます。 • <code>datachange</code> 組み込み関数は、データ変更のカウントをロー数のパーセントとして返しますが、このときに分母となるロー数は、最初に存在したロー数ではなく現在残っているロー数です。たとえば、テーブルにローが 5 つあり、1 つのローを削除した場合、現在のロー数カウントが 4 で <code>datachange</code> カウンタが 1 であるため、<code>datachange</code> は 25% という値をレポートします。 • <code>datachange</code> は、テーブル (パーティションを指定した場合はパーティション) 内の合計ロー数のパーセンテージとして表されます。パーセンテージの値は 100 パーセントを超えることがあります。これは、オブジェクトの変更回数、テーブル内のロー数を大幅に超える場合があるからです。特に、1 つのテーブルで削除と更新が非常に多く行われるとこの状況になります。 • <code>datachange</code> が示す値は、メモリ内の値です。この値は、ディスク内の値と異なる可能性があります。ディスク内の値は、<code>sp_flushstats</code> を実行したときに (オブジェクト記述子がフラッシュされたときに) ハウスキーピングによって更新されます。

- 分割されたテーブルのグローバル・インデックスに対してヒストグラムが作成されたときには、**datachange** 値はリセットされません。
- リソースを無駄遣いする代わりに、**datachange** は既にキャッシュに存在しないオブジェクトの記述子を廃棄します。

datachange が 0 にリセットまたは初期化されるのは、以下の場合です。

- 新しいカラムが追加され、その **datachange** 値が初期化される時。
- 新しいパーティションが追加され、その **datachange** 値が初期化される時。
- データ・パーティションに固有のヒストグラムが作成、削除、または更新された時。この場合、これらの変更が加えられたカラムとパーティションに対応するヒストグラムの **datachange** 値がリセットされます。
- テーブルまたはパーティションのデータがトランケートされ、それらの **datachange** 値がリセットされた時。
- 他のコマンドの実行結果としてテーブルが直接または間接に再分割され、テーブルのすべてのパーティションとカラムの **datachange** 値がリセットされた時。
- テーブルの分割が解除され、テーブルのすべてのカラムの **datachange** 値がリセットされた時。

datachange には、次のような制限があります。

- **datachange** 統計は、システム tempdb、ユーザ定義 tempdb、システム・テーブル、またはプロキシ・テーブルのテーブルに保持されません。
- **datachange** の更新は、非トランザクション指向です。トランザクションをロールバックしても **datachange** 値はロールバックされません。値が不正確になる可能性があります。
- カラムレベル・カウンタにメモリを割り当てられなかった場合、カラムレベルの **datachange** 値ではなくパーティションレベルの値が追跡されます。
- カラムレベルの **datachange** 値が維持されない場合、カラムの **datachange** 値がリセットされるたびにパーティションレベルの **datachange** 値がリセットされます。

パーミッション

datachange は、すべてのユーザが実行できます。

datalength

説明	指定したカラムや文字列の実際の長さをバイト単位で返します。
構文	<code>datalength(expression)</code>
パラメータ	<p>expression</p> <p>カラム名、変数、定数式、またはこれらを任意に組み合わせたもので1つの値を計算するものです。<i>expression</i> には、任意のデータ型を使用できます。通常はカラム名を指定します。<i>expression</i> が文字定数のときは引用符で囲む。</p>
例	<p><code>publishers</code> テーブルの <code>pub_name</code> カラムの長さを計算します。</p> <pre>select Length = datalength(pub_name) from publishers length ----- 13 16 20</pre>
使用法	<ul style="list-style-type: none"> • <code>datalength</code> システム関数は、<i>expression</i> のバイト単位の長さを返します。 • <code>datalength</code> はカラムが圧縮されているときでも、圧縮されていないラージ・オブジェクト・カラムの長さを返します。 • Unicode データ型で定義されたカラムの場合、<code>datalength</code> は、各ローに保管されているデータの実際のバイト数を返します。たとえば、<code>unitext</code> のカラム <code>ut</code> に <code>U+0041U+0042U+d800dc00</code> というロー値が保管されている場合、次の値が返されます。 <pre>select datalength(ut) from unitable ----- 8</pre> • <code>datalength</code> 関数は、各ローに格納されているデータの実際の長さを計算します。<code>datalength</code> は、<code>varchar</code>、<code>univarchar</code>、<code>varbinary</code>、<code>text</code>、<code>image</code> データ型の場合に便利です。これは、これらのデータ型が可変長を保管し、後続ブランクを保管しないからです。<code>char</code> 値または <code>unichar</code> 値で <code>null</code> を入力可能であることを宣言すると、Adaptive Server 内部では、これらの値が <code>varchar</code> または <code>univarchar</code> として格納されます。他のすべてのデータ型の場合、<code>datalength</code> 関数は定義済みの長さをレポートします。 • <code>datalength</code> は <code>text_locator</code>、<code>unitext_locator</code>、および <code>image_locator</code> LOB データ型を受け入れます。 • <code>NULL</code> データに対して <code>datalength</code> 関数を実行すると、常に <code>NULL</code> が返されます。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>datalength</code> 関数を実行できます。
参照	関数 char_length 、 col_length

dateadd

説明	指定した日付または時刻に間隔を追加します。
構文	<code>dateadd(date_part, integer, date_expression)</code>
パラメータ	<p>date_part 日付要素または省略形です。Adaptive Server が認識する日付要素と省略形のリストについては、『Transact-SQL ユーザーズ・ガイド』を参照してください。</p> <p>integer 整数式です。</p> <p>date_expression datetime、smalldatetime、bigdatetime、bigtime、date、time 型の式、または datetime フォーマットの文字列です。</p>
例	<p>例 1 bigtime に 100 万マイクロ秒を加えます。</p> <pre>declare @a bigtime select @a = "14:20:00.010101" select dateadd(us, 1000000, @a) ----- 2:20:01.010101PM</pre> <p>例 2 bigdatetime に 25 時間を加えると日が増分します。</p> <pre>declare @a bigdatetime select @a = "apr 12, 0001 14:20:00 " select dateadd(hh, 25, @a) ----- Apr 13 0001 2:20PM</pre> <p>例 3 titles テーブル内のすべての本の出版日が 21 日ずれた場合、その新しい出版日を表示します。</p> <pre>select newpubdate = dateadd(day, 21, pubdate) from titles</pre> <p>例 4 日付に 1 日加えます。</p> <pre>declare @a date select @a = "apr 12, 9999" select dateadd(dd, 1, @a) ----- Apr 13 9999</pre> <p>例 5 時刻から 5 分を引きます。</p> <pre>select dateadd(mi, -5, convert(time, "14:20:00")) ----- 2:15PM</pre>

例 6 時刻に 1 日加えると時刻は変わりません。

```
declare @a time
select @a = "14:20:00"
select dateadd(dd, 1, @a)
-----
2:20PM
```

例 7 各 `date_part` には `datetime` 値のように制限がありますが、制限値よりも大きい値を加算して、次の有効なフィールドに値を反映できます。

```
--Add 24 hours to a datetime
select dateadd(hh, 24, "4/1/1979")
-----
Apr  2 1979 12:00AM

--Add 24 hours to a date
select dateadd(hh, 24, "4/1/1979")
-----
Apr  2 1979
```

使用法

- 日付関数 `dateadd` は、指定した日付に特定の間隔を追加します。ビューの詳細については、『*Transact-SQL ユーザーズ・ガイド*』を参照してください。
- `dateadd` 関数には日付要素、数字、日付の 3 つの引数があります。結果は、指定された日付に日付要素数を加算した `datetime` 値です。最後の引数が `bigtime` で、日付要素が年、月、または日の場合、結果は元の `bigtime` 引数です。

日付引数が `smalldatetime` 値の場合、結果も `smalldatetime` 値です。`dateadd` 関数を使用して `smalldatetime` に秒またはミリ秒を追加できますが、このような追加が意味を持つのは、`dateadd` 関数が返す結果の日付が 1 分以上変化するときだけです。

- 日付順の値ではなく文字列が引数として指定された場合、サーバは示された精度にかかわらず、その文字列を `datetime` 値として解釈します。デフォルトの動作は、設定パラメータ `builtin date strings` または設定オプション `builtin_date_strings` を設定することで変更できます。これらのオプションを設定すると、サーバは発生順の組み込み関数に与えられた文字列を `bigdatetime` として解釈します。詳細については、『*システム管理ガイド*』を参照してください。
- この組み込み関数にマイクロ秒の日付要素を与えると、文字列値は常に `bigdatetime` として解釈されます。

- Use the **datetime** データ型は、1753 年 1 月 1 日より後の日付だけに使います。**datetime** 値は一重引用符か二重引用符で囲んでください。0001 年 1 月 1 日から 9999 年 1 月 1 日までの日付には、**date** データ型を使用します。**date** は一重引用符か二重引用符で囲んでください。それよりも前の日付には、**char**、**nchar**、**varchar**、または **nvarchar** を使用します。Adaptive Server では、さまざまな種類の日付フォーマットを識別できません。詳細については、「[ユーザ定義データ型](#)」(42 ページ)と『Transact-SQL ユーザーズ・ガイド』を参照してください。

Adaptive Server は、必要に応じて (たとえば、文字値を **datetime** 値と比較するとき) 文字値と **datetime** 値とを自動的に変換します。

- 日付要素の **weekday** や **dw** を **dateadd** で使用するの論理的でなく、疑似結果が返されます。これらの代わりに **day** または **dd** を使ってください。

表 2-4: **date_part** として認識される省略形

日付要素	省略形	値
Year	yy	1753 ~ 9999 (datetime) 1900 ~ 2079 (smalldatetime) 0001 ~ 9999 (date)
Quarter	qq	1 ~ 4
Month	mm	1 ~ 12
Week	wk	1054
Day	dd	1 ~ 7
dayofyear	dy	1 ~ 366
Weekday	dw	1 ~ 7
Hour	hh	0 ~ 23
Minute	mi	0 ~ 59
Second	ss	0 ~ 59
millisecond	ms	0 ~ 999
microsecond	us	0 ~ 999999

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが **dateadd** 関数を実行できます。

参照

データ型 [日付と時刻のデータ型](#)

コマンド [select, where 句](#)

関数 [datediff, datename, datepart, getdate](#)

datediff

説明

指定した 2 つの日付間または時刻間で日付要素の数を計算します。

構文

`datediff(datepart, date expression1, date expression2)`

パラメータ

datepart

日付要素または省略形です。Adaptive Server が認識する日付要素と省略形のリストについては、『Transact-SQL ユーザーズ・ガイド』を参照してください。

date expression1

`datetime`、`smalldatetime`、`bigdatetime`、`bigtime`、`date`、`time` 型の式、または `datetime` フォーマットの文字列です。

date expression2

`datetime`、`smalldatetime`、`bigdatetime`、`bigtime`、`date`、`time` 型の式、または `datetime` フォーマットの文字列です。

例

例 1 2 つの `bigdatetimes` 間のマイクロ秒数を返します。

```
declare @a bigdatetime
declare @b bigdatetime
select @a = "apr 1, 1999 00:00:00.000000"
select @b = "apr 2, 1999 00:00:00.000000"
select datediff(us, @a, @b)
-----
86400000000
```

例 2 オーバフロー・エラーが返されます。

```
select datediff(ms, convert(bigdatetime, "4/1/1753"),
convert(bigdatetime, "4/1/9999"))
Msg 535, Level 16, State 0:
Line 2:
Difference of two datetime fields caused overflow at runtime.
Command has been aborted
```

例 3 `pubdate` から (`getdate` 関数で求めた) 現在の日付までの経過日数を求めます。

```
select newdate = datediff(day, pubdate, getdate())
from titles
```

例 4 2 つの時刻間の時間数を求めます。

```
declare @a time
declare @b time
select @a = "20:43:22"
select @b = "10:43:22"
select datediff(hh, @a, @b)
-----
-10
```

例 5 2つの日付間の時間数を求めます。

```
declare @a date
declare @b date
select @a = "apr 1, 1999"
select @b = "apr 2, 1999"
select datediff(hh, @a, @b)
-----
24
```

例 6 2つの時刻間の日数を求めます。

```
declare @a time
declare @b time
select @a = "20:43:22"
select @b = "10:43:22"
select datediff(dd, @a, @b)
-----
0
```

例 7 オーバフロー・エラーが返されます。

```
select datediff(ms, convert(date, "4/1/1753"), convert(date, "4/1/9999"))
Msg 535, Level 16, State 0:
Line 2:
Difference of two datetime fields caused overflow at runtime.
Command has been aborted
```

使用法

- **datediff** 関数には 3 つの引数があります。最初の引数は日付要素です。2 番目と 3 番目は発生順の値です。**dates**、**times**、**datetimes**、および **bigdatetimes** の場合、結果は日付要素の **date2** と **date1** に等しい符号付き整数値です。
 - 2 番目または 3 番目の引数が日付で、日付要素が時間、分、秒、ミリ秒、またはマイクロ秒の場合は、日付が午前 0 時として処理されます。
 - 2 番目または 3 番目の引数が時刻で、日付要素が年、月、または日の場合は、ゼロが返されます。
 - **datediff** 関数の結果が日付要素の整数倍でないときは、結果が丸められずトランケートされます。
 - より小さい時間単位では、オーバーフロー値が生じることがあります。次の制限を超えた場合は、関数からオーバーフロー・エラーが返されます。
- **datediff** 関数はデータ型 **int** の結果を生成しますが、結果が 2,147,483,647 を超えるとエラーになります。ミリ秒値の場合、これは約 24 日 20:31.846 時間です。秒値の場合、68 年 19 日 3:14:07 時間になります。
- **datediff** 関数の結果が日付要素の整数倍でないときは、結果は丸められず、常にトランケートされます。たとえば、日付要素として **hour** を使用すると、「4:00AM」と「5:50AM」の差は 1 になります。

日付要素として **day** を使うと、**datediff** 関数は指定した 2 つの時刻の間の午前 0 時の数をカウントします。たとえば、1992 年 1 月 1 日 23:00 時と 1992 年 1 月 2 日 01:00 時の間の差は 1 ですが、1992 年の 1 月 1 日 00:00 時と 1992 年 1 月 1 日の 23:59 の間の差は 0 です。

- 日付要素 **month** は、2 つの日付間の各月の第 1 日目の数をカウントします。たとえば、1 月 25 日と 2 月 2 日の差は 1 ですが、1 月 1 日と 1 月 31 日の差は 0 です。
- 日付要素 **week** を **datediff** 関数で使うと、2 つの日付の間 (2 番目の日付は含みますが、1 番目の日付は含みません) の日曜日の数をカウントします。たとえば、1 月 4 日の日曜日と 1 月 11 日の日曜日の間における週の数 は 1 です。
- **smalldatetime** 値を使用すると、この値は内部で計算用に **datetime** 値に変換されます。**smalldatetime** 値の秒やミリ秒は、差を計算するために自動的に 0 に設定されます。
- 2 番目または 3 番目の引数が日付であり、**datepart** が時間、分、秒、またはミリ秒の場合は、その日付は午前 0 時として処理されます。
- 2 番目または 3 番目の引数が時刻であり、**datepart** が年、月、または日の場合は、ゼロが返されます。
- **datediff** 関数の結果が日付要素の整数倍でないときは、結果は丸められずトランケートされます。
- 日付順の値ではなく文字列が引数として指定された場合、サーバは示された精度にかかわらず、その文字列を **datetime** 値として解釈します。デフォルトの動作は、設定パラメータ **builtin date strings** または設定オプション **builtin_date_strings** を設定することで変更できます。これらのオプションを設定すると、サーバは発生順の組み込み関数に与えられた文字列を **bigdatetime** として解釈します。詳細については、『システム管理ガイド』を参照してください。
- この組み込み関数にマイクロ秒の日付要素を与えると、文字列値は常に **bigdatetime** として解釈されます。
- より小さい **time** 単位では、オーバーフロー値が生じることがあります。次の制限を超えた場合は、関数からオーバーフロー・エラーが返されます。
 - マイクロ秒：約 3 日
 - ミリ秒：約 24 日
 - 秒約：約 68 年
 - 分：約 4083 年
 - その他：オーバーフロー制限なし

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが **datediff** 関数を実行できます。

参照

データ型 [日付と時刻のデータ型](#)

コマンド [select, where 句](#)

関数 [dateadd, datename, datepart, getdate](#)

datetimeame

説明	指定した <code>date</code> または <code>time</code> の指定した <code>datepart</code> を文字列として返します。
構文	<code>datetimeame(datepart, date expression)</code>
パラメータ	<p>datepart 日付要素または省略形です。Adaptive Server が認識する日付要素と省略形のリストについては、『Transact-SQL ユーザーズ・ガイド』を参照してください。</p> <p>date_expression <code>datetime</code>、<code>smalldatetime</code>、<code>bigdatetime</code>、<code>bigtime</code>、<code>time</code> 型の式、または <code>datetime</code> フォーマットの文字列です。</p>
例	<p>例 1 <code>bigdatetime</code> の月名を求めます。</p> <pre> declare @a bigdatetime select @a = "apr 12, 0001 00:00:000.010101" select datetimeame(mm, @a) ----- April </pre> <p>例 2 現在の日付が 2000 年 11 月 20 日だとします。</p> <pre> select datetimeame(month, getdate()) November </pre> <p>例 3 日付の月名を求めます。</p> <pre> declare @a date select @a = "apr 12, 0001" select datetimeame(mm, @a) ----- April </pre> <p>例 4 時刻の秒を求めます。</p> <pre> declare @a time select @a = "20:43:22" select datetimeame(ss, @a) ----- 22 </pre>
使用法	<ul style="list-style-type: none"> • 日付関数 <code>datetimeame</code> は、<code>datetime</code> 値または <code>smalldatetime</code> 値の指定した要素の名前 (月「June」など) を文字列として返します。結果が数値 (「23」日など) の場合でも、この値を文字列として返します。 • <code>date</code>、<code>time</code>、<code>bigdatetime</code>、<code>bigtime</code>、<code>datetime</code>、<code>smalldatetime</code> のいずれかの値を 2 番目の引数として指定します。 • <code>datetimeame</code> 関数で日付要素の <code>weekday</code> や <code>dw</code> を使うと、曜日 (Sunday、Monday など) が返されます。 • <code>smalldatetime</code> は分単位までしか正確でないため、<code>datetimeame</code> 関数で <code>smalldatetime</code> 値を使うときは、秒とミリ秒は常に 0 です。

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>datetime</code> 関数を実行できます。
参照	データ型 日付と時刻のデータ型 コマンド select , where 句 関数 dateadd , datetime , datepart , getdate

datepart

説明	日付式の指定した要素の整数値を返します。
構文	<code>datepart(date_part, date_expression)</code>
パラメータ	<p>date_part</p> <p>日付要素です。表 2-5 に、日付要素、datepart に指定できる省略形、有効値を示します。</p>

表 2-5: 日付要素とその有効値

日付要素	省略形	値
year	yy	1753 ~ 9999 (smalldatetime の場合は 2079)。date の場合は 0001 ~ 9999
quarter	qq	1 ~ 4
month	mm	1 ~ 12
week	wk	1 ~ 54
day	dd	1 ~ 31
dayofyear	dy	1 ~ 366
weekday	dw	1 ~ 7 (日曜 ~ 土曜)
hour	hh	0 ~ 23
minute	mi	0 ~ 59
second	ss	0 ~ 59
millisecond	ms	0 ~ 999
microsecond	us	0 ~ 999999
calweekofyear	cwk	1 ~ 53
calyearofweek	cyr	1753 ~ 9999 (smalldatetime の場合は 2079)。date の場合は 0001 ~ 9999
caldayofweek	cdw	1 ~ 7

2 桁の年 (yy) を入力する場合には、以下の点に注意してください。

- 50 未満の数値は 20yy として解釈されます。たとえば、01 は 2001、32 は 2032、49 は 2049 です。
- 50 以上の数値は 19yy として解釈されます。たとえば、50 は 1950、74 は 1974、99 は 1999 です。

datetime、smalldatetime、および time 型のミリ秒は、前にコロンかピリオドが付く場合があります。コロンに続ける場合、数字は 1 秒の 1000 分の 1 を意味します。ピリオドに続ける場合、1 桁は 1 秒の 10 分の 1、2 桁は 100 分の 1、3 桁は 1000 分の 1 を意味します。たとえば、「12:30:20:1」は 12 時 30 分から 20 秒と 1000 分の 1 秒が経過したことを示し、「12:30:20.1」は 12 時 30 分から 20 秒と 10 分の 1 秒が経過したことを示します。

ミリ秒には小数点を付けて端数を表す必要があります。

date_expression

datetime、*smalldatetime*、*bigdatetime*、*bigintime*、*date*、*time* 型の式、または *datetime* フォーマットの文字列です。

例

例 1 *bigdatetime* のマイクロ秒を求めます。

```
declare @a bigdatetime
select @a = "apr 12, 0001 12:00:00.000001"
select datepart(us, @a)
-----
000001
```

例 2 現在の日付が 1995 年 11 月 25 日であるとします。

```
select datepart(month, getdate())
-----
11
```

例 3 伝統料理の本の出版年を返します。

```
select datepart(year, pubdate) from titles
       where type = "trad_cook"
-----
1990
1985
1987
```

例 4

```
select datepart(cwk, '1993/01/01')
-----
53
```

例 5

```
select datepart(cyr, i1993/01/01i)
-----
1992
```

例 6

```
select datepart(cdw, i1993/01/01i)
-----
5
```

例 7 時刻の時間を求めます。

```
declare @a time
select @a = "20:43:22"
select datepart(hh, @a)
-----
20
```

例 8 `datetime` または `datepart` を使用して、`date` から時間、分、秒の部分と、結果はデフォルト時間のゼロになります。`datetime` または `datepart` を使用して、`time` から月、日、または年を求めると、結果はデフォルト日付の 1990 年 1 月 1 日になります。

```
--Find the hours in a date
declare @a date
select @a = "apr 12, 0001"
select datepart(hh, @a)
-----
0

--Find the month of a time
declare @a time
select @a = "20:43:22"
select datetime(mm, @a)
-----
January
```

`datetime` 関数にパラメータとして `null` 値を指定した場合は、`NULL` が返されます。

使用法

- 指定した `date` (2 番目の引数) の指定した `datepart` (最初の引数) を整数として返します。`date`、`time`、`datetime`、`bigdatetime`、`bigtime`、または `smalldatetime` のいずれかの値を 2 番目の引数として指定します。`datepart` が時間、分、秒、ミリ秒、またはマイクロ秒の場合は、結果は 0 になります。
- `datepart` は、ISO 標準 8601 に従った数字を返します。これは、週の最初の日と年の最初の週を定義します。`datepart` 関数に `calweekofyear`、`calyearofweek`、または `caldayorweek` のどの値が含まれるかによって、同じ時間単位について返される日付が異なることがあります。たとえば、Adaptive Server が英語 (U.S. English) をデフォルト言語として使用するよう設定されているとします。次の関数は 1988 を返します。

```
datepart(cyr, "1/1/1989")
```

一方、次の関数は 1989 を返します。

```
datepart(yy, "1/1/1989")
```

この違いは、ISO 標準が年の最初の週を、木曜日を含みかつ月曜日から始まる最初の週として定義するために発生します。

デフォルトの言語として英語を使用しているサーバでは、週の最初の曜日は日曜日で、年の最初の週は 1 月 4 日を含む週となります。

- `datepart` 関数で日付要素の `weekday` や `dw` を使うと、対応する番号が返されます。曜日の名前に対応する番号は `datefirst` の設定により異なります。一部の言語のデフォルト (`us_english` など) では、日曜日は 1、月曜日は 2 になり、他の言語では月曜日が 1、火曜日が 2 になります。デフォルト設定をセッションごとに変更するには、`set datefirst` を使用します。詳細については、`set` コマンドの `datefirst` オプションを参照してください。

- `calweekofyear` (省略形は `cwkw`) は、その年における週の順序数を返します。`calyearofweek` (省略形は `cyr`) は、その週が始まる年を返します。`caldayofweek` (省略形は `cdw`) は、その週におけるその日の順序数を返します。`dateadd`、`datediff`、`datename` では、`calweekofyear`、`calyearofweek`、`caldayofweek` を日付要素としては使用できません。
- `datetime` および `time` が正確なのは 300 分の 1 秒単位までなので、`datepart` 関数でこれらのデータ型の値を使用すると、ミリ秒は最も近い 300 分の 1 秒に丸められます。
- `smalldatetime` が正確なのは分単位までなので、`datepart` 関数で `smalldatetime` 値を使うときは、秒とミリ秒は常に 0 です。
- 曜日付要素の値は言語設定の影響を受けます。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `datepart` 関数を実行できます。

参照

データ型 [日付と時刻のデータ型](#)

コマンド [select](#), [where 句](#)

関数 [dateadd](#), [datediff](#), [datename](#), [getdate](#)

day

説明	指定した日付の <code>datepart</code> の日を表す整数を返します。
構文	<code>day(date_expression)</code>
パラメータ	<code>date_expression</code> datetime、smalldatetime、date 型の式、または datetime フォーマットの文字列です。
例	整数 02 を返します。 <pre>day("11/02/03") ----- 02</pre>
使用法	<code>day(date_expression)</code> は <code>datepart(dd,date_expression)</code> と同じです。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザーが <code>day</code> 関数を実行できます。
参照	データ型 datetime、smalldatetime、date、time 関数 datepart 、 month 、 year

db_attr

説明 指定したデータベースの `durability`、`dml_logging`、`template` の設定、および圧縮レベルを返します。

構文 `db_attr('database_name' | database_ID | NULL, 'attribute')`

パラメータ `database_name`
データベースの名前です。

`database_ID`
データベースの ID です。

NULL
含めた場合、`db_attr` が現在のデータベースについてレポートします。

attribute
次のいずれかです。

- `help` – `db_attr` の使用方法を表示します。
- `durability` – 特定のデータベースの持続性を返します。 `full`、`at_shutdown`、または `no_recovery`。
- `dml_logging` – 指定したデータベースのデータ操作言語 (DML: Data Manipulation Language) ロギングの `full` または `minimal` の値を返します。
- `template` – 指定したデータベースに使用されたテンプレート・データベースの名前を返します。テンプレートとして使用されたデータベースがない場合は、`NULL` を返します。
- `compression` – データベースの圧縮レベルを返します。

例 **例 1** `db_attr` の構文を返します。

```
select db_attr(0, "help")
-----
Usage: db_attr('dbname' | dbid | NULL, 'attribute')
List of options in attributes table:
      help
      durability
      dml_logging
      template
      4 : compression
```

例 2 名前、durability 設定、dml_logging 設定、および使用されたテンプレートを sysdatabases から選択します。

```
select name = convert(char(20), name),
durability = convert(char(15), db_attr(name, "durability")),
dml_logging = convert(char(15), db_attr(dbid, "dml_logging")),
template = convert(char(15), db_attr(dbid, "template"))
from sysdatabases
```

name	durability	dml_logging	template
master	full	full	NULL
model	full	full	NULL
tempdb	no_recovery	full	NULL
sybssystemdb	full	full	NULL
sybssystemprocs	full	full	NULL
repro	full	full	NULL
imdb	no_recovery	full	db1
db	full	full	NULL
at_shutdown_db	at_shutdown	full	NULL
db1	full	full	NULL
dml	at_shutdown	minimal	NULL

例 3 存在しない DoesNotExist データベースに対して db_attr を実行します。

```
select db_attr("DoesNotExist", "durability")
```

```
-----
NULL
```

例 4 ID が 12345 の存在しないデータベースに対して db_attr を実行します。

```
select db_attr(12345, "durability")
```

```
-----
NULL
```

例 5 存在しない属性に対して db_attr を実行します。

```
select db_attr(1, "Cmd Does Not Exist")
```

```
-----
NULL
```

使用法

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが db_attr 関数を実行できます。

参照

関数

db_id

説明	指定したデータベースの ID 番号を表示します。
構文	<code>db_id(database_name)</code>
パラメータ	database_name データベースの名前です。 <i>database_name</i> には、文字式を指定する必要があります。expression が文字定数のときは、引用符で囲んでください。
例	sybssystemprocs の ID 番号を返します。 <pre>select db_id("sybssystemprocs") ----- 4</pre>
使用法	<ul style="list-style-type: none">システム関数 <code>db_id</code> は、データベースの ID 番号を返します。<i>database_name</i> を指定しないと、<code>db_id</code> 関数は現在のデータベースの ID 番号を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>db_id</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 db_name , object_id

db_instanceid

説明	クラスタ環境のみ – 指定したローカル・テンポラリ・データベースの所有インスタンスの ID を返します。指定したデータベースがグローバル・テンポラリ・データベースまたは非テンポラリ・データベースである場合は NULL を返します。
構文	<pre>db_instanceid(database_id) db_instanceid(database_name)</pre>
パラメータ	<p>database_id データベースの ID です。</p> <p>database_name データベースの名前です。</p>
例	データベース ID 5 の所有インスタンスを返します。 <pre>select db_instanceid(5)</pre>
使用法	<ul style="list-style-type: none">ローカル・テンポラリ・データベースへのアクセスは所有インスタンスからのみ可能です。db_instanceid は、指定したデータベースがローカル・テンポラリ・データベースでローカル・テンポラリ・データベースの所有インスタンスかどうかを判別します。確認後、所有インスタンスに接続して、そのローカル・テンポラリ・データベースにアクセスできます。db_instanceid と一緒にパラメータを含める必要があります。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが sdsc_intempdbconfig を実行できます。

db_name

説明	ID 番号で指定されたデータベースの名前を表示します。
構文	<code>db_name([database_id])</code>
パラメータ	<code>database_id</code> (<code>sysdatabases.dbid</code> に格納されている) データベース ID の数値式です。
例	<p>例 1 現在のデータベースの名前を返します。</p> <pre>select db_name()</pre> <p>例 2 データベース ID が 4 のデータベースの名前を返します。</p> <pre>select db_name(4)</pre> <pre>----- sybssystemprocs</pre>
使用法	<ul style="list-style-type: none">システム関数 <code>db_name</code> はデータベース名を返します。<code>database_id</code> を指定しないときは、<code>db_name</code> は現在のデータベースの名前を返す。
標準規格	ANSI SQL - 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>db_name</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 col_name , db_id , object_name

db_recovery_status

説明	クラスタ環境のみ – 指定したデータベースのリカバリ・ステータスを返します。 <i>database_ID</i> または <i>database_name</i> の値を含めない場合は、現在のデータベースのリカバリ・ステータスを返します。
構文	<code>db_recovery_status([<i>database_ID</i> <i>database_name</i>])</code>
パラメータ	<i>database_ID</i> リカバリ・ステータスを要求するデータベースの ID です。 <i>database_name</i> リカバリ・ステータスを要求するデータベースの名前です。
例	例 1 現在のデータベースのリカバリ・ステータスを返します。 <pre>select db_recovery_status()</pre> 例 2 <code>test</code> という名前のデータベースのリカバリ・ステータスを返します。 <pre>select db_recovery_status("test")</pre> 例 3 データベース ID が 8 のデータベースのリカバリ・ステータスを返します。 <pre>select db_recovery_status(8)</pre>
使用法	戻り値 0 は、データベースがノードフェールオーバ・リカバリ状態にないことを示します。戻り値 1 は、データベースがノードフェールオーバ・リカバリ状態にあることを示します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>db_recovery_status</code> を実行できます。

degrees

説明	指定したラジアン角度を度数法の角度に変換します。
構文	<code>degrees(numeric)</code>
パラメータ	<i>numeric</i> 度数に変換されるラジアン数です。
例	<pre>select degrees(45) ----- 2578</pre>
使用法	<ul style="list-style-type: none"><code>degrees</code> 算術関数はラジアンを度数に変換します。結果は、数値式と同じ型になります。 numeric 型または decimal 型の式の場合、結果の内部精度は 77 で、位取りは式と同じです。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>degrees</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 radians

derived_stat

説明	指定したオブジェクトとインデックスに対して得られた統計を返します。
構文	<pre>derived_stat("object_name" object_id, index_name index_id, ["partition_name" partition_id,] "statistic")</pre>
パラメータ	<p>object_name 統計を取るオブジェクトの名前です。完全修飾オブジェクト名を指定しない場合は、derived_stat は現在のデータベースを検索します。</p> <p>object_id 統計を取るオブジェクトのオブジェクト ID で、object_name の代わりに指定できます。object_id は、現在のデータベース内に存在する必要があります。</p> <p>index_name 指定したオブジェクトに属している、統計を取るインデックスの名前です。</p> <p>index_id 指定したオブジェクトに属している、統計を取るインデックスのインデックス ID です。index_name の代わりに指定できます。</p> <p>partition_name 特定のパーティションに属している、統計を取るパーティションの名前です。partition_name は省略可能です。partition_name または partition_id を使用すると、Adaptive Server はオブジェクト全体ではなくターゲット・パーティションの統計を返します。</p> <p>partition_id 指定したオブジェクトに属している、統計を取るパーティションのパーティション ID です。partition_name の代わりに指定できます。partition_id は省略可能です。</p> <p>"statistic" 返される統計です。取得可能な統計値は、次のとおりです。</p> <ul style="list-style-type: none"> • data page cluster ratio または dpcr – オブジェクトとインデックスの組み合わせのデータ・ページ・クラスタ率 • index page cluster ratio または ipcr – オブジェクトとインデックスの組み合わせのインデックス・ページ・クラスタ率 • data row cluster ratio または drcr – オブジェクトとインデックスの組み合わせのデータ・ロー・クラスタ率 • large io efficiency または lgio – オブジェクトとインデックスの組み合わせの大容量 I/O 効率 • space utilization または sput – オブジェクトとインデックスの組み合わせの領域使用率

例

例 1 titles テーブルの titleidind インデックスの領域使用率を選択します。

```
select derived_stat("titles", "titleidind", "space utilization")
```

例 2 titles テーブルのインデックス ID 2 のデータ・ページ・クラスタ率を選択します。"dpcr" または "data page cluster ratio" のどちらでも使用できます。

```
select derived_stat("titles", 2, "dpcr")
```

例 3 パーティション ID もパーティション名も指定しないので、オブジェクト全体の統計がレポートされます。

```
1> select derived_stat(object_id("t1"), 2, "drcr")
2> go
```

```
-----
0.576923
```

例 4 パーティション tl_928003396 の統計をレポートします。

```
1> select derived_stat(object_id("t1"), 0, "tl_928003306", "drcr")
2> go
```

```
-----
1.000000
```

(1 row affected)

例 5 syspartitions からのデータを使用して、指定したテーブルのすべてのインデックスの抽出統計を選択します。

```
select convert(varchar(30), name) as name, indid,
       convert(decimal(5, 3), derived_stat(id, indid, 'sput')) as 'sput',
       convert(decimal(5, 3), derived_stat(id, indid, 'dpcr')) as 'dpcr',
       convert(decimal(5, 3), derived_stat(id, indid, 'drcr')) as 'drcr',
       convert(decimal(5, 3), derived_stat(id, indid, 'lgio')) as 'lgio'
from syspartitions where id = object_id('titles')
go
```

name	indid	sput	dpcr	drcr	lgio
titleidind_2133579608	1	0.895	1.000	1.000	1.000
titleind_2133579608	2	0.000	1.000	0.688	1.000

(2 rows affected)

例 6 分割されたテーブルのすべてのインデックスとパーティションの抽出統計を選択します。ここで、mymsgsr4 はラウンドロビン分割テーブルで、グローバル・インデックスとローカル・インデックスで作成されます。

```
1> select * into mymsgsr4 partition by roundrobin 4 lock datarows
2> from master..sysmessages
2> go
```

(7597 rows affected)

```

1> create clustered index mymsgs_rr4_clustind on mymsgs_rr4(error, severity)
2> go
1> create index mymsgs_rr4_ncind1 on mymsgs_rr4(severity)
2> go
1> create index mymsgs_rr4_ncind2 on mymsgs_rr4(langid, dlevel) local index
2> go

2> update statistics mymsgs_rr4
1>

2> select convert(varchar(10), object_name(id)) as name,
3>     (select convert(varchar(20), i.name) from sysindexes i
4>     where i.id = p.id and i.indid = p.indid),
5> convert(varchar(30), name) as ptnname, indid,
6> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'sput')) as 'sput',
7> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'dpcr')) as 'dpcr',
8> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'dr cr')) as 'dr cr',
9> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'lgio')) as 'lgio'
10> from syspartitions p
11> where id = object_id('mymsgs_rr4')

```

name	ptnname	indid	sput	dpcr	dr cr	lgio
mymsgs_rr4 mymsgs_rr4	mymsgs_rr4_786098810	0	0.90	1.000	1.00	1.000
mymsgs_rr4 mymsgs_rr4	mymsgs_rr4_802098867	0	0.90	1.000	1.00	1.000
mymsgs_rr4 mymsgs_rr4	mymsgs_rr4_818098924	0	0.89	1.000	1.00	1.000
mymsgs_rr4 mymsgs_rr4	mymsgs_rr4_834098981	0	0.90	1.000	1.00	1.000
mymsgs_rr4 mymsgs_rr4_clustind	mymsgs_rr4_clustind_850099038	2	0.83	0.995	1.00	1.000
mymsgs_rr4 mymsgs_rr4_ncind1	mymsgs_rr4_ncind1_882099152	3	0.99	0.445	0.88	1.000
mymsgs_rr4 mymsgs_rr4_ncind2	mymsgs_rr4_ncind2_898099209	4	0.15	1.000	1.00	1.000
mymsgs_rr4 mymsgs_rr4_ncind2	mymsgs_rr4_ncind2_914099266	4	0.88	1.000	1.00	1.000
mymsgs_rr4 mymsgs_rr4_ncind2	mymsgs_rr4_ncind2_930099323	4	0.877	1.000	1.000	1.000
mymsgs_rr4 mymsgs_rr4_ncind2	mymsgs_rr4_ncind2_946099380	4	0.945	0.993	1.000	1.000

例 7 現在のデータベースのページ・ロック・テーブルすべての抽出統計を選択します。

```

2> select convert(varchar(10), object_name(id)) as name
3>     (select convert(varchar(20), i.name) from sysindexes i
4>     where i.id = p.id and i.indid = p.indid),
5> convert(varchar(30), name) as ptnname, indid,
6> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'sput')) as 'sput',
7> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'dpcr')) as 'dpcr',
8> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'dr cr')) as 'dr cr',
9> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'lgio')) as 'lgio'
10> from syspartitions p
11> where lockscheme(id) = "allpages"
12> and (select o.type from sysobjects o where o.id = p.id) = 'U'

```

name	ptnname	indid	sput	dpcr	dr cr	lgio
-----	-----	-----	-----	-----	-----	-----

stores	stores	stores_18096074	0	0.276	1.000	1.000	1.000
discounts	discounts	discounts_50096188	0	0.075	1.000	1.000	1.000
au_pix	au_pix	au_pix_82096302	0	0.000	1.000	1.000	1.000
au_pix	tau_pix	tau_pix_82096302	255	NULL	NULL	NULL	NULL
blurbs	blurbs	blurbs_114096416	0	0.055	1.000	1.000	1.000
blurbs	tblurbs	tblurbs_114096416	255	NULL	NULL	NULL	NULL
tlapl	tlapl	tlapl_1497053338	0	0.095	1.000	1.000	1.000
tlapl	tlapl	tlapl_1513053395	0	0.082	1.000	1.000	1.000
tlapl	tlapl	tlapl_1529053452	0	0.095	1.000	1.000	1.000
tlapl	tlapl_ncind	tlapl_ncind_1545053509	2	0.149	0.000	1.000	1.000
tlapl	tlapl_ncind_local	tlapl_ncind_local_1561053566	3	0.066	0.000	1.000	1.000
tlapl	tlapl_ncind_local	tlapl_ncind_local_1577053623	3	0.057	0.000	1.000	1.000
tlapl	tlapl_ncind_local	tlapl_ncind_local_1593053680	3	0.066	0.000	1.000	1.000
authors	auidind	auidind_1941578924	1	0.966	0.000	1.000	1.000
authors	anmind	anmind_1941578924	2	0.303	0.000	1.000	1.000
publishers	pubind	pubind_1973579038	1	0.059	0.000	1.000	1.000
roysched	roysched	roysched_2005579152	0	0.324	1.000	1.000	1.000
roysched	titleidind	titleidind_2005579152	2	0.777	1.000	0.941	1.000
sales	salesind	salesind_2037579266	1	0.444	0.000	1.000	1.000
salesdetai	salesdetail	salesdetail_2069579380	0	0.614	1.000	1.000	1.000
salesdetai	titleidind	titleidind_2069579380	2	0.518	1.000	0.752	1.000
salesdetai	salesdetailind	salesdetailind_2069579380	3	0.794	1.000	0.726	1.000
titleautho	taind	taind_2101579494	1	0.397	0.000	1.000	1.000
titleautho	auidind	auidind_2101579494	2	0.285	0.000	1.000	1.000
titleautho	titleidind	titleidind_2101579494	3	0.223	0.000	1.000	1.000
titles	titleidind	titleidind_2133579608	1	0.895	1.000	1.000	1.000
titles	titleind	titleind_2133579608	2	0.402	1.000	0.688	1.000

(27 rows affected)

使用法

- `derived_stat` は倍精度値を返します。
- `derived_stat` が返す値は、`optdiag` ユーティリティで表示される値と一致します。
- 指定したオブジェクトまたはインデックスがない場合は、`derived_stat` は `NULL` を返します。
- 無効な統計タイプを指定すると、エラー・メッセージが発行されます。
- オプションの `partition_name` または `partition_id` を指定すると、`derived_stat` はターゲットのパーティションの要求された統計をレポートします。指定しない場合、`derived_stat` はオブジェクト全体の統計をレポートします。
- リソースを消費する代わりに、`derived_stat` はまだキャッシュに存在しないオブジェクトの記述子を廃棄します。

- 引数は次のように使用します。
 - 4つの引数 - `derived_stat` は、3番目の引数をパーティションとして使用し、4番目の引数に指定された抽出統計を返します。
 - 3つの引数 - `derived_stat` は、パーティションの指定が省略されたと解釈し、3番目の引数に指定された抽出統計を返します。

標準規格

ANSI SQL - 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

テーブルの所有者だけが `derived_stat` 関数を実行できます。

参照

マニュアル 『パフォーマンス&チューニング・ガイド』の次の項目を参照してください。

- 「1つのテーブルに対するアクセス・メソッドとクエリ・コスト計算」
- 「統計テーブルおよび`optdiag`を使った統計の表示」

ユーティリティ `optdiag`

difference

説明	2 つの soundex 値の差を返します。
構文	<code>difference(<i>expr1</i>,<i>expr2</i>)</code>
パラメータ	<p>expr1 文字型のカラム名、変数、または (<code>char</code>、<code>varchar</code>、<code>nchar</code>、<code>nvarchar</code>、または <code>unichar</code> 型の) 定数式です。</p> <p>expr2 もう 1 つの文字型のカラム名、変数、または (<code>char</code>、<code>varchar</code>、<code>nchar</code>、<code>nvarchar</code>、または <code>unichar</code> 型の) 定数式です。</p>
例	<p>例 1</p> <pre>select difference("smithers", "smothers") ----- 4</pre> <p>例 2</p> <pre>select difference("smothers", "brothers") ----- 2</pre>
使用法	<ul style="list-style-type: none"> 文字列関数 <code>difference</code> は 2 つの soundex 値の差を表す整数を返します。 <code>difference</code> 関数は 2 つの文字列を比較し、この 2 つの文字列の類似点を評価して、0 ~ 4 の値を返します。一致度が最も高い場合の値は 4 です。 文字列値として、一連の有効な半角または全角のローマ字からなる文字列を指定してください。 expr1 または expr2 が NULL の場合は、NULL を返します。 <code>varchar</code> 式と <code>unichar</code> 式をそれぞれパラメータとして指定した場合、<code>varchar</code> 式は暗黙的に <code>unichar</code> に変換されます (変換時にトランケーションが発生することがあります)。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>difference</code> 関数を実行できます。
参照	<p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 soundex</p>

dol_downgrade_check

説明	8191 バイトよりも長い可変長カラムを含む指定データベース内のデータオンリーロック (DOL) テーブルの数を返します。ワイドな可変長カラムがない場合やダウングレードを実行できる場合は、0 を返します。
構文	<code>dol_downgrade_check('database_name', target_version)</code>
パラメータ	database_name チェックするデータベースの名前または ID です。 database_name は修飾されたオブジェクト名場合があります (たとえば、 <code>mydb.dbo.mytable</code>)。 target_version ダウングレードする予定の Adaptive Server の整数バージョンです (たとえば、バージョン 15.0.3 は 1503 です)。
例	ワイド可変長カラムに対して <code>pubs2</code> データベース内の DOL テーブルをチェックし、バージョン 15.5 にダウングレードできるようにします。 <pre>select dol_downgrade_check('pubs2', 1550)</pre>
使用法	<ul style="list-style-type: none">ターゲット・バージョンが 15.7 以降の場合は 0 (正常) を返して、必要な作業がないことを示します。修飾されたテーブルを指定する場合に、そのテーブルが属するデータベースを示さないと、<code>dol_downgrade_check</code> は現在のデータベースをチェックします。

exp

説明	自然対数 e を指定した数値だけ累乗した値を返します。
構文	<code>exp(<i>approx_numeric</i>)</code>
パラメータ	<i>approx_numeric</i> 概数値型 (<code>float</code> 、 <code>real</code> 、または <code>double precision</code>) のカラム名、変数、または定数式のいずれかです。
例	<pre>select exp(3) ----- 20.085537</pre>
使用法	<ul style="list-style-type: none">算術関数 <code>exp</code> は、指定した値の指数値を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>exp</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 log 、 log10 、 power

floor

説明 指定された値以下の最大の整数を返します。

構文 `floor(numeric)`

パラメータ *numeric*

真数値 (numeric、dec、decimal、tinyint、smallint、int、または bigint 型)、概数値 (float、real、または double precision 型)、money 型のカラム、変数、定数式、またはこれらの組み合わせのいずれかです。

例 **例 1**

```
select floor(123)
-----
          123
```

例 2

```
select floor(123.45)
-----
          123
```

例 3

```
select floor(1.2345E2)
-----
    123.000000
```

例 4

```
select floor(-123.45)
-----
         -124
```

例 5

```
select floor(-1.2345E2)
-----
    -124.000000
```

例 6

```
select floor($123.45)
-----
         123.00
```

使用法

- 算術関数 `floor` は、指定した値以下の最大整数を返します。結果は、数値式と同じ型になります。

数値式や 10 進数式の場合、結果は式に等しく、位取りは 0 になります。

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが floor 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 abs , ceiling , round , sign

get_appcontext

説明	指定されたコンテキストの属性値を返します。get_appcontext は Application Context Facility (ACF) から提供されます。
構文	get_appcontext ("context_name", "attribute_name")
パラメータ	<p>context_name アプリケーション・コンテキスト名を示すローです。データ型 char(30) として保存されています。</p> <p>attribute_name アプリケーション・コンテキスト属性名を示すローです。データ型 char(30) として保存されています。</p>
例	<p>例 1 ATTR1 の値として VALUE1 が返されています。</p> <pre>select get_appcontext("CONTEXT1", "ATTR1") ----- VALUE1</pre> <p>ATTR1 は CONTEXT2 にはありません。</p> <pre>select get_appcontext("CONTEXT2", "ATTR1")</pre> <p>例 2 適切なパーミッションを持たないユーザがアプリケーション・コンテキストを取得しようとしたときの結果を示します。</p> <pre>select get_appcontext("CONTEXT1", "ATTR2", "VALUE1") Select permission denied on built-in get_appcontext, database dbid ----- -1</pre>
使用法	<ul style="list-style-type: none"> この関数は成功すると 0 を返し、失敗すると -1 を返します。 指定された属性がアプリケーション・コンテキスト内にない場合は、get_appcontext は NULL を返します。 get_appcontext は、属性を char データ型として格納します。属性値を他のデータ型と比較するアクセス・ルールを作成している場合は、アクセス・ルールで char データを適切なデータ型に変換します。 この関数では、すべての引数が必須です。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	パーミッションは、ACF に格納されているユーザ・プロファイルとアプリケーション・プロファイルに依存します。
参照	ACF の詳細については、『システム管理ガイド』の「第 11 章 ユーザ・パーミッションの管理」の「ロー・レベル・アクセス制御」を参照してください。
	関数 get_appcontext , list_appcontext , rm_appcontext , set_appcontext

getdate

説明	システムの現在の日付と時刻を返します。
構文	<code>getdate()</code>
パラメータ	なし。
例	<p>例 1 現在の日付と時刻が 1995 年 11 月 25 日、午前 10 時 32 分だとします。</p> <pre>select getdate() Nov 25 1995 10:32AM</pre> <p>例 2 現在の日付が 11 月だとします。</p> <pre>select datepart(month, getdate()) 11</pre> <p>例 3 現在の日付が 11 月だとします。</p> <pre>select datename(month, getdate()) November</pre>
使用法	<ul style="list-style-type: none">日付関数 <code>getdate</code> は、システムの現在の日付と時刻を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>getdate</code> 関数を実行できます。
参照	<p>データ型 日付と時刻のデータ型</p> <p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 dateadd, datediff, datename, datepart</p>

getutcdate

説明 日時を万国標準時 (UTC) で返します。getutcdate は、ローが挿入または選択されるたびに計算されます。

構文 getutcdate()

例

```
insert t1 (c1, c2, c3) select c1, getutcdate(),  
getdate() from t2)
```

参照 [関数 biginttohex](#), [convert](#)

has_role

説明	指定した役割がユーザに付与されているかどうかを示す情報を返します。
構文	<code>has_role ("role_name", option)</code>
パラメータ	<p>role_name システム定義またはユーザ定義の役割の名前です。</p> <p>option 返される情報の範囲を制限できます。現在サポートされているオプションは 1 だけで、監査を非表示にします。</p>
例	<p>例 1 ユーザがシステム管理者かどうかをチェックするプロシージャを作成します。</p> <pre>create procedure sa_check as if (has_role("sa_role", 0) > 0) begin print "You are a System Administrator." return(1) end</pre> <p>例 2 ユーザにシステム・セキュリティ担当者の役割が付与されているかどうかをチェックします。</p> <pre>select has_role("sso_role", 1)</pre> <p>例 3 ユーザにオペレータの役割が付与されているかどうかをチェックします。</p> <pre>select has_role("oper_role", 1)</pre>
使用法	<ul style="list-style-type: none"> • has_role は、proc_role と同じように機能します。Adaptive Server バージョン 15.0 から、Sybase は proc_role に代えて has_role をサポートしています。また、この関数の使用が推奨されています。ただし、既存の proc_role をすべて has_role に変換する必要はありません。 • システム関数 has_role は、指定した役割が呼び出し元ユーザに付与されてアクティブになっているかどうかをチェックします。 • ユーザの役割が次の場合に has_role は 0 を返します。 <ul style="list-style-type: none"> • 指定した役割がユーザに付与されていない場合。 • 指定した役割を含む役割がユーザに付与されていない場合。 • 指定した役割がユーザに付与されているが、アクティブ化されていない場合。 • 指定した役割が呼び出し元ユーザに対して付与されており、アクティブ化されている場合、has_role は 1 を返します。 • 呼び出し元ユーザのアクティブな役割に、指定の役割が含まれている場合、has_role は 2 を返します。

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが has_role 関数を実行できます。
参照	コマンド alter role , create role , drop role , grant , set , revoke マニュアル Transact-SQL ユーザーズ・ガイド 関数 mut_excl_roles , role_contain , role_id , role_name , show_role

hash

説明

固定長のハッシュ値式を生成します。

構文

`hash(expression, [algorithm])`

パラメータ

expression

ハッシュする値です。カラム名、変数、定数式、またはこれらを任意に組み合わせたものを指定すると、1つの値が生成されます。image、text、unitext、またはロー外の Java データ型を指定することはできません。expression は、通常はカラム名です。expression が文字定数のときは常に引用符で囲みます。

algorithm

ハッシュ値の生成に使用されるアルゴリズムです。md5 または sha1、2 (md5 バイナリを意味する) または 3 (sha1 バイナリを意味する) のいずれかの値を取ることができる文字リテラル (変数名またはカラム名ではない) です。省略すると、md5 が使用されます。

アルゴリズム	結果
<code>hash(expression, 'md5')</code>	varchar 32 バイトの文字列。 md5 (メッセージ・ダイジェスト・アルゴリズム 5) は、128 ビット・ハッシュ値を使用した暗号化ハッシュ関数です。
<code>hash(expression)</code>	varchar 32 バイトの文字列
<code>hash(expression, 'sha1')</code>	varchar 40 バイトの文字列 sha1 (セキュア・ハッシュ・アルゴリズム) は、160 ビット・ハッシュ値を使用した暗号化ハッシュ関数です。
<code>hash(expression, 2)</code>	varbinary 16 バイト値 (md5 アルゴリズムの使用)
<code>hash(expression, 3)</code>	varbinary 20 バイト値 (sha1 アルゴリズムの使用)

例

次の例では、seal の実装方法を示します。「atable」というテーブルと id、sensitive_field、tamper seal カラムの存在。

```
update atable set tamper_seal=hash(convert(varchar(30),
id) + sensitive_field+@salt, 'sha1')
```

使用法

文字リテラルとして指定されている場合、algorithm に大文字と小文字の区別はありません。“md5”、“Md5”、“MD5” はすべて同じです。ただし、expression が文字データ型として指定されている場合、値に大文字と小文字の区別があります。“Time”、“TIME”、“time” はそれぞれ異なるハッシュ値を生成します。

algorithm が文字リテラルの場合は、varchar 型の文字列が生成されます。“md5”の場合、ハッシュ計算の 128 ビット結果の 16 進数表現を含んでいる 32 バイト文字列です。“sha1”の場合、ハッシュ計算の 160 ビット結果の 16 進数表現を含んでいる 40 バイト文字列です。

algorithm が整数リテラルの場合、**varbinary** 値が生成されます。2 の場合、ハッシュ計算の 128 ビット結果を含んでいる 16 バイト値が生成されます。3 の場合、ハッシュ計算の 160 ビット結果を含んでいる 20 バイト値が生成されます。

注意 後続の null 値は、**varbinary** カラムに挿入されるときに Adaptive Server によってトリムされます。

expression を構成する個々のバイトは、メモリ内での表示順でハッシュ・アルゴリズムに渡されます。多くのデータ型について、バイト順序は大きな意味を持ちます。たとえば、4 バイト INT 値 1 のバイナリ表現は MSB-first (ビッグエンディアン) プラットフォームの場合は 0x00, 0x00, 0x00, 0x01 となり、LSB-first (リトルエンディアン) プラットフォームの場合は 0x01, 0x00, 0x00, 0x00 となります。バイト・ストリームはプラットフォーム間で異なるため、ハッシュ値も異なります。プラットフォームに依存しないハッシュ値を生成するには、**hashbytes** 関数を使用します。

注意 ハッシュ・アルゴリズム MD5 および SHA1 は、完全にセキュアではなくなったという認識が広がっています。したがって、セキュリティが重要なコンテキストで MD5 または SHA1 を使用する場合は、そのリスクに留意してください。

標準規格

SQL92 および SQL99 対応

パーミッション

すべてのユーザが **hash** 関数を実行できます。

参照

プラットフォームに依存しないハッシュ値については、「**hashbytes**」も参照してください。

hashbytes

説明

固定長のハッシュ値式を生成します。

構文

`hashbytes(algorithm, expression [, expression...] [, using options])`

パラメータ

`expression [, expression...]`

ハッシュする値です。この値には、カラム名、変数、定数式、またはこれらを組み合わせて指定することで、1つの値を生成できます。image、text、unitext、またはロー外の Java データ型を指定することはできません。

algorithm

ハッシュ値の生成に使用されるアルゴリズムです。“md5”、“sha”、“sha1”、“ptn”のいずれかの値を使用できる文字リテラル(変数名やカラム名ではない)です。

アルゴリズム	説明
Md5	メッセージ・ダイジェスト・アルゴリズム 5 は、128 ビットのハッシュ値を使用する暗号化ハッシュ・アルゴリズムです。hashbytes('md5', <i>expression</i> [...]) は varbinary 16 バイト値になります。
Sha-Sha1	セキュア・ハッシュ・アルゴリズムは、160 ビットのハッシュ値を使用する暗号化ハッシュ・アルゴリズムです。hashbytes('sha1', <i>expression</i> [...]) は varbinary 20 バイト値になります。
Ptn	32 ビット・ハッシュ値を使用するパーティション・ハッシュ・アルゴリズムです。'ptn' アルゴリズムの場合、using 句は無視されます。hashbytes('ptn', <i>expression</i> [...]) は unsigned int 4 バイト値になります。
using	プラットフォームに関係なく、バイト順序を決定します。オプションの using 句は、次の option 文字列の前に配置できます。 <ul style="list-style-type: none"> lsb — すべてのバイト順序依存データは、ハッシュ処理前にリトルエンディアンバイト順序に正規化されます。 msb — すべてのバイト順序依存データは、ハッシュ処理前にビッグエンディアンバイト順序に正規化されます。 unicode — 文字データは、ハッシュ処理前に unicode (UTF-16) に正規化されます。 <hr/> <p>注意 UTF-16 文字列は、単精度整数の配列と似た文字列です。この文字列はバイト順序に依存するので、プラットフォーム独立性を確保するために、lsb または msb を UNICODE と一緒に使用することをおすすめします。</p> <hr/> <ul style="list-style-type: none"> unicode_lsb — unicode と lsb の組み合わせ unicode_msb — unicode と msb の組み合わせ

例 **例 1** テーブルの各ローを改ざんから守ります。次の例では、“xtable” および col1、col2、col3、tamper_seal というユーザ・テーブルが存在すると想定します。

```
update xtable set tamper_seal=hashbytes('shal', col1,
col2, col4, @salt)
--
declare @nparts unsigned int
select @nparts= 5
select hashbytes('ptn', col1, col2, col3) % nparts from xtable
```

例 2 col1、col2、col3 がローを使用して 5 つのパーティションに分割する方法を示します。

```
alter table xtable partition by hash(col1, col2, col3) 5
```

使用法

algorithm パラメータに大文字と小文字の区別はありません。“md5”、“Md5”、“MD5” はすべて同じです。ただし、**expression** が文字データ型として指定されている場合は、値に大文字と小文字の区別があります。“Time”、“TIME”、“time” はそれぞれ異なるハッシュ値を生成します。

注意 後続の null 値は、**varbinary** カラムに挿入されるときに Adaptive Server によってトリムされます。

using 句を使用しない場合、**expression** を構成するバイトは、メモリ内での表示順でハッシュ・アルゴリズムに渡されます。多くのデータ型について、バイト順序は大きな意味を持ちます。たとえば、4 バイト INT 値 1 のバイナリ表現は MSB-first (ビッグエンディアン) プラットフォームの場合は 0x00、0x00、0x00、0x01 となり、LSB-first (リトルエンディアン) プラットフォームの場合は 0x01、0x00、0x00、0x00 となります。バイト・ストリームはプラットフォームごとに異なるので、ハッシュ値も異なります。

using 句を使用した場合、**expression** を構成するバイトは、プラットフォームに関係なくハッシュ・アルゴリズムに渡されます。また、**using** 句を使用すると、Unicode に文字データを変換できるので、ハッシュ値はサーバの文字設定に依存しなくなります。

注意 ハッシュ・アルゴリズム MD5 および SHA1 は、完全にセキュアではなくなったという認識が広がっています。MD5 または SHA1 をセキュリティが重要なコンテキストで使用する場合は、そのリスクに留意してください。

標準規格

SQL92 および SQL99 対応

パーミッション

すべてのユーザが **hashbyte** 関数を実行できます。

参照

プラットフォームに依存するハッシュ値については、「hash」も参照してください。

hextobigint

説明	16 進文字列に相当する bigint 値を返す。
構文	hextobigint (<i>hexadecimal_string</i>)
パラメータ	hexadecimal_string big integer に変換される 16 進数値です。文字型カラム、変数名、または有効な 16 進文字列 (“0x” プレフィックスの有無に関係なく引用符で囲む) のいずれかである必要があります。
例	次の例は、0x7fffffffffffffff という 16 進文字列を bigint 値に変換します。 <pre>1> select hextobigint("0x7fffffffffffffff") 2> go ----- 9223372036854775807</pre>
使用法	<ul style="list-style-type: none">データ型変換関数 hextobigint は、16 進文字列に相当する、プラットフォームに依存しない整数を返します。hextobigint 関数は、プラットフォームに依存しないで 16 進データを整数に変換するために使用します。hextobigint は、有効な 16 進文字列 (“0x” プレフィックスの有無に関係なく引用符で囲む)、文字型のカラム名、または変数名を受け取ります。 <p>hextobigint は、16 進文字列に相当する bigint を返します。この関数は、実行するプラットフォームに関係なく、16 進文字列に相当する同じ bigint を常に返します。</p>
参照	関数 biginttohex , convert , inttohex , hextoint

hextoint

説明	指定した 16 進文字列に相当する、プラットフォームに依存しない整数値を返します。
構文	<code>hextoint(hexadecimal_string)</code>
パラメータ	hexadecimal_string 整数に変換される 16 進数値です。文字型カラム、変数名、または有効な 16 進文字列 (“0x” プレフィクスの有無に関係なく引用符で囲む) のいずれかである必要があります。
例	16 進文字列 “0x00000100” に相当する整数値を返します。この関数が実行されるプラットフォームに関係なく、結果は常に 256 です。 <pre>select hextoint("0x00000100")</pre>
使用法	<ul style="list-style-type: none">データ型変換関数 hextoint は、16 進文字列に相当する、プラットフォームに依存しない整数を返します。hextoint 関数は、16 進数データから整数へのプラットフォームに依存しない変換に使用します。hextoint は、“0x” プレフィクスの有無に関係なく、引用符で囲まれた有効な 16 進文字列か、文字型のカラムまたは変数の名前を受け入れます。 hextoint 関数は 16 進文字列に相当する整数値を返します。この関数は、実行するプラットフォームに関係なく、16 進文字列に相当する同一整数値を常に返します。データ型変換の詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが hextoint 関数を実行できます。
参照	関数 biginttohex , convert , inttohex

host_id

説明	クライアント・コンピュータのオペレーティング・システムで使用されている、現在の Adaptive Server クライアントのプロセス ID を返します。
構文	host_id()
パラメータ	なし。
例	<p>この例では、クライアント・コンピュータの名前が “ephemeris” で、コンピュータ “ephemeris” で Adaptive Server クライアント・プロセスに使用されているプロセス ID が 2309 です。</p> <pre>select host_name(), host_id() ----- ephemeris 2309</pre> <p>次は、コンピュータ “ephemeris” 上で、UNIX の <code>ps</code> コマンドを使用して収集したプロセス情報です。この例のクライアントは “isql” であり、プロセス ID が 2309 であることを示しています。</p> <pre>2309 pts/2 S 0:00 /work/as125/OCS-12_5/bin/isql</pre>
使用法	<ul style="list-style-type: none">システム関数 <code>host_id</code> は、(サーバ・プロセスではなく) クライアント・プロセスのホスト・プロセス ID を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>host_id</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 host_name

host_name

説明	クライアント・プロセスの現在のホスト・コンピュータ名を表示します。
構文	host_name()
パラメータ	なし。
例	<pre>select host_name() ----- violet</pre>
使用法	<ul style="list-style-type: none">システム関数 host_name は、(サーバ・プロセスではなく)クライアント・プロセスの現在のホスト・コンピュータ名を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが host_name 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 host_id

instance_id

説明	クラスタ環境のみ – 指定したインスタンスの ID、または <i>name</i> の値を指定しない場合は、発行元のインスタンスの ID を返します。
構文	<code>instance_id([<i>name</i>])</code>
パラメータ	<i>name</i> ID を調べるキャッシュのインスタンス名です。
例	ローカル・インスタンスの ID を返します。 <pre>select instance_id()</pre>
使用法	“myserver1” というインスタンスの ID を返します。 <pre>select instance_id(myserver1)</pre>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>instance_id</code> を実行できます。

identity_burn_max

説明 指定したテーブルの破棄される identity 値の最大値を追跡します。この関数は値を返すだけで、更新は実行しません。

構文 `identity_burn_max(table_name)`

パラメータ *table_name*
選択したテーブルの名前です。

例

```
select identity_burn_max("t1")

t1
-----
51
```

使用法 `identity_burn_max` は、指定したテーブルの ID 消去最大値を追跡します。

パーミッション テーブル所有者、システム管理者、データベース管理者のみがこのコマンドを発行できます。

index_col

説明	指定されたテーブルまたはビューにあるインデックス付きのカラム名を表示します。カラム名は、最長で 255 バイトです。
構文	<code>index_col(object_name, index_id, key_#, user_id)</code>
パラメータ	<p>object_name テーブルまたはビューの名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。名前は引用符で囲んでください。</p> <p>index_id <code>object_name</code> のインデックス番号です。この番号は <code>sysindexes.indid</code> の値と同じです。</p> <p>key_# インデックス内のキーです。この値は、クラスタード・インデックスでは 1 と <code>sysindexes.keycnt</code> の間、ノンクラスタード・インデックスでは 1 と <code>sysindexes.keycnt+1</code> の間の値です。</p> <p>user_id <code>object_name</code> の所有者です。<code>user_id</code> を指定しないと、デフォルトの呼び出し元のユーザ ID になります。</p>
例	<p>テーブル <code>t4</code> のクラスタード・インデックス内のキーの名前を検索します。</p> <pre> declare @keycnt integer select @keycnt = keycnt from sysindexes where id = object_id("t4") and indid = 1 while @keycnt > 0 begin select index_col("t4", 1, @keycnt) select @keycnt = @keycnt - 1 end </pre>
使用法	<ul style="list-style-type: none"> システム関数 <code>index_col</code> は、インデックス・カラムの名前を返します。 <code>object_name</code> がテーブル名またはビュー名でない場合には、<code>index_col</code> 関数は NULL を返します。
標準規格	ANSI SQL - 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>index_col</code> 関数を実行できます。
参照	<p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 object_id</p> <p>システム・プロシージャ sp_helpindex</p>

index_colorder

説明	カラム順を返します。
構文	<code>index_colorder(object_name, index_id, key_#, user_id)</code>
パラメータ	<p>object_name テーブルまたはビューの名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。名前は引用符で囲んでください。</p> <p>index_id object_name のインデックス番号です。この番号は <code>sysindexes.indid</code> の値と同じです。</p> <p>key_# インデックス内のキーです。有効な値は 1 と、インデックス内のキーの数です。キーの数は、<code>sysindexes.keycnt</code> に格納されます。</p> <p>user_id object_name の所有者です。user_id を指定しないと、デフォルトの呼び出し元のユーザ ID になります。</p>
例	<p><code>sales</code> テーブルの <code>salesind</code> インデックスは降順であるため、“DESC” を返します。</p> <pre>select name, index_colorder("sales", indid, 2) from sysindexes where id = object_id ("sales") and indid > 0 name ----- salesind DESC</pre>
使用法	<ul style="list-style-type: none"> システム関数 <code>index_colorder</code> は、昇順のカラムの場合に “ASC”、降順のカラムの場合に “DESC” を返します。 <code>object_name</code> がテーブル名ではないか、または <code>key_#</code> が有効なキー番号ではない場合、<code>index_colorder</code> は NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>index_colorder</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド

index_name

説明	インデックス ID、データベース ID、およびインデックスが定義されているオブジェクトを指定すると、インデックスの名前を返します。
構文	<code>index_name(dbid, objid, indid)</code>
パラメータ	<p>dbid インデックスを定義するデータベースの ID です。</p> <p>objid インデックスを定義するテーブル (指定したデータベース内) の ID です。</p> <p>indid 名前を知りたいインデックスの ID です。</p>
例	<p>例 1 この関数の通常の使用法を示します。</p> <pre>select index_name(db_id("testdb"), object_id("testdb..tab_ap1"),1) -----</pre> <p>例 2 データベース ID が NULL で現在のデータベース ID を使用する場合の出力を示します。</p> <pre>select index_name(NULL,object_id("testdb..tab_ap1"),1) -----</pre> <p>例 3 インデックス ID が 0 で、データベース ID とオブジェクト ID が有効な場合は、テーブル名が表示されます。</p> <pre>select index_name(db_id("testdb"), object_id("testdb..tab_ap1"),0) -----</pre>
使用法	<ul style="list-style-type: none"> • dbid パラメータで NULL の値を渡した場合、index_name は現在のデータベース ID を使用します。 • dbid パラメータで NULL の値を渡した場合、index_name は NULL を返します。 • インデックス ID が 0 で、オブジェクト ID とデータベース ID に有効な入力値を渡した場合、index_name はオブジェクト名を返します。
パーミッション	この関数は、すべてのユーザが実行できます。
参照	db_id , object_id

instance_name

説明	クラスタ環境のみ – 指定した ID の Adaptive Server の名前、または <i>id</i> の値を指定しない場合は、発行元の Adaptive Server の名前を返します。
構文	<code>instance_name([id])</code>
パラメータ	<i>id</i> 調べる Adaptive Server の ID です。
例	データベース ID が 12 のインスタンスの名前を返します。 <pre>select instance_name(12)</pre>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>instance_name</code> 関数を実行できます。

inttohex

説明	指定された整数値に相当する、プラットフォームに依存しない 16 進文字列を返す。
構文	<code>inttohex(integer_expression)</code>
パラメータ	<i>integer_expression</i> 16 進文字列に変換される整数値です。
例	<pre>select inttohex (10) ----- 0000000A</pre>
使用法	<ul style="list-style-type: none">データ型変換関数 <code>inttohex</code> は、指定された整数値に相当する、プラットフォームに依存しない 16 進文字列を、プレフィクス “0x” を付けずに返します。<code>inttohex</code> 関数は、プラットフォームに依存しない、整数から 16 進文字列への変換に使用します。<code>inttohex</code> は、整数に評価される式を受け入れます。実行するプラットフォームに関係なく、指定の式と同じ 16 進数の値を常に返します。データ型変換の詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>inttohex</code> 関数を実行できます。
参照	関数 convert , hextobigint , hextoint

is_quiesced

説明

データベースが **quiesce database** モードにあるかどうかを示します。
is_quiesced は、データベースがクワイス (静止) 状態の場合に 1、クワイス状態でない場合に 0 を返します。

構文

```
is_quiesced(dbid)
```

パラメータ

dbid

データベースのデータベース ID です。

例

例 1 test データベースを使用します。データベース ID は 4 で、クワイス状態ではありません。

```
1> select is_quiesced(4)
2> go
```

```
-----
                0
```

(1 row affected)

例 2 quiesce database を実行してアクティビティをサスペンドしたあとで、**test** データベースを使用します。

```
1> quiesce database tst hold test
2> go
1> select is_quiesced(4)
2> go
```

```
-----
                1
```

(1 row affected)

例 3 quiesce database を使用してアクティビティを再開したあとで、**test** データベースを使用します。

```
1> quiesce database tst release
2> go
1> select is_quiesced(4)
2> go
```

```
-----
                0
```

(1 row affected)

例 4 無効なデータベース ID を使用する `is_quiesced` を指定して `select` 文を実行します。

```
1>select is_quiesced(-5)
2> go
```

```
-----
      NULL
```

```
(1 row affected)
```

使用法

- `is_quiesced` にデフォルト値はありません。データベースを指定しないで `is_quiesced` 関数を実行すると、エラーになります。
- `is_quiesced` は、存在しないデータベース ID が指定された場合に NULL を返します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

`is_quiesced` は、すべてのユーザが実行できます。

参照

コマンド [quiesce database](#)

is_sec_service_on

説明 特定のセキュリティ・サービスが有効化されているかどうかを確認する。サービスが有効な場合は 1、サービスが有効ではない場合は 0 を返す。

構文 `is_sec_service_on(security_service_nm)`

パラメータ `security_service_nm`
セキュリティ・サービスの名前です。

例

```
select is_sec_service_on("unifiedlogin")
```

使用法

- セッション中に、特定のセキュリティ・サービスが有効かどうかを調べるには、`is_sec_service_on` を使います。
- セキュリティ・サービスの有効な名前を調べるには、次のクエリを実行します。

```
select * from syssecmechs
```

次のような結果が出力されます。

```
sec_mech_name available_service
-----
dce            unifiedlogin
dce            mutualauth
dce            delegation
dce            integrity
dce            confidentiality
dce            detectreplay
dce            detectseq
```

`available_service` カラムには、Adaptive Server がサポートするセキュリティ・サービスが表示されます。

標準規格 ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション すべてのユーザが `is_sec_service_on` 関数を実行できます。

参照 関数 [show_sec_services](#)

is_singleusermode

説明 Adaptive Server がシングルユーザ・モードで実行されていない場合は 0 を返します。Adaptive Server がシングルユーザ・モードで実行されている場合は 1 を返します。

構文 is_singleusermode()

パラメータ is_singleusermode にはパラメータが含まれていません。

例 次の例は、シングルユーザ・モードで実行しているサーバを示します。

```
select is_singleusermode()
```

```
-----
```

```
1
```

パーミッション すべてのユーザが is_singleusermode を実行できます。

isdate

説明

入力式が有効な **datetime** 値かどうかを確認します。

構文

```
isdate(character_expression)
```

パラメータ

character_expression

文字型の変数、定数式、またはカラム名です。

例

例 1 文字列 12/21/2005 が有効な **datetime** 値であるかどうか確認するには、次のように入力します。

```
select isdate('12/21/2005')
```

例 2 sales テーブル内の **stor_id** と **date** が有効な **datetime** 値かどうかを確認するには、次のように入力します。

```
select isdate(stor_id), isdate(date) from sales
-----
0      1
```

store_id は有効な **datetime** 値ではありませんが、**date** は有効です。

使用法

式が有効な **datetime** 値である場合は 1 を返し、有効でない場合は 0 を返します。NULL の入力の場合は 0 を返します。

isnull

説明	<i>expression1</i> の値が NULL のとき、 <i>expression2</i> に指定した値と置き換えます。
構文	<code>isnull(<i>expression1</i>, <i>expression2</i>)</code>
パラメータ	<i>expression</i> カラム名、変数、定数式、またはこれらを任意に組み合わせたもので 1 つの値を計算するものです。unichar など、任意のデータ型を使用できます。 <i>expression</i> は通常、カラム名です。 <i>expression</i> が文字定数のときは引用符で囲む。
例	<code>price</code> の null 値を 0 に置き換えて、 <code>titles</code> テーブルのすべてのローを返します。 <pre>select isnull(price,0) from titles</pre>
使用法	<ul style="list-style-type: none">システム関数 <code>isnull</code> は、<i>expression1</i> の値が NULL の場合、<i>expression2</i> に指定した値と置き換えます。システム関数の一般的な情報については、『Transact-SQL ユーザーズ・ガイド』を参照してください。式のデータ型が暗黙的に変換されるか、または <code>convert</code> 関数を使用する必要があります。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>isnull</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 convert

isnumeric

説明	式が有効な numeric データ型かどうかを確認します。
構文	<code>isnumeric (character_expression)</code>
パラメータ	character_expression 文字型の変数、定数式、またはカラム名です。
例	<p>例 1 authors テーブルの postalcode カラムの値に有効な numeric データ型が含まれているかどうか確認するには、次のように入力します。</p> <pre>select isnumeric(postalcode) from authors</pre> <p>例 2 値 \$100.12345 が有効な numeric データ型かどうか確認するには、次のように入力します。</p> <pre>select isnumeric("\$100.12345")</pre>
使用法	<ul style="list-style-type: none">入力式が有効な整数型、浮動小数点型、通貨型、10 進数型のいずれかである場合は 1 を返し、有効でない場合または入力が NULL 値の場合は 0 を返します。戻り値が 1 の場合は、式をこれらの numeric 型のいずれかに変換できます。入力には通貨記号を含めることができます。

lc_id

説明	クラスタ環境のみ – 指定した名前の論理クラスタの ID、または名前を指定しない場合は現在の論理クラスタの ID を返します。
構文	<code>lc_id(logical_cluster_name)</code>
パラメータ	<i>logical_cluster_name</i> 論理クラスタの名前です。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>lc_id</code> を実行できます。

lc_name

説明	クラスタ環境のみ – 指定した ID の論理クラスタの名前、または ID を指定しない場合は現在の論理クラスタを返します。
構文	<code>lc_name([logical_cluster_ID])</code>
パラメータ	<i>logical_cluster_ID</i> 論理クラスタの ID です。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>lc_name</code> 関数を実行できます。

lct_admin

説明	ラストチャンス・スレッシュールド (LCT) を管理します。LCT の現在の値を返して、LCT に達したトランザクション・ログに含まれるトランザクションをアポートします。
構文	<pre>lct_admin({"lastchance" "logfull" "reserved_for_rollbacks"}, database_id "reserve", {log_pages 0 } "abort", process-id [, database-id])</pre>
パラメータ	<p>lastchance 指定されたデータベースに LCT を作成します。</p> <p>logfull 指定されたデータベースが LCT を超えた場合には 1 を返し、超えていない場合には 0 を返します。</p> <p>reserved_for_rollbacks データベースがロールバック用に現在確保しているページ数を判断します。</p> <p>database_id データベースを指定します。</p> <p>reserve LCT の現在の値、または指定サイズのトランザクション・ログのダンプに必要なログ・ページ数を取得します。</p> <p>log_pages LCT を決定するために使用されるページ数です。</p> <p>0 LCT の現在の値を返します。独立したログとデータ・セグメントを持つデータベース内の LCT のサイズは、動的に変化しません。この値は、トランザクション・ログのサイズに基づいて固定しています。混合ログとデータ・セグメントを持つデータベースでは、LCT は動的に変化します。</p> <p>abort トランザクション・ログがラストチャンス・スレッシュールドに達したデータベース内のトランザクションをアポートします。ログ中断モードのトランザクションだけをアポートできます。</p> <p>logsegment_freepages ログ・セグメント用に使用可能な空き領域を示します。これは、空き領域の合計値であり、ディスクごとの値ではありません。</p> <p>process-id ログ中断モードのプロセスの ID (<i>spid</i>) です。ラストチャンス・スレッシュールド (LCT) に達したトランザクション・ログ内にオープン・トランザクションがある場合、プロセスはログ中断モードになります。</p> <p>database-id トランザクション・ログが LCT に達したデータベースの ID です。<i>process-id</i> が 0 の場合は、指定データベース内のオープン・トランザクションすべてが終了します。</p>

例 **例 1** dbid が 1 のデータベースのログ・セグメントのラストチャンス・スレッシュヨールドが作成されます。新しいスレッシュヨールドが作成されたページの数を返します。前回のラストチャンス・スレッシュヨールドが検出されると、置き換えられます。

```
select lct_admin("lastchance", 1)
```

例 2 dbid が 6 のデータベースのラストチャンス・スレッシュヨールドを超えている場合は 1 を返し、そうでない場合は 0 を返します。

```
select lct_admin("logfull", 6)
```

例 3 64 ページのトランザクション・ログをダンプするために必要となるログ・ページ数を計算して返します。

```
select lct_admin("reserve", 64)
```

16

例 4 コマンドの発行元であるデータベース内のトランザクション・ログの、現在のラストチャンス・スレッシュヨールドを返します。

```
select lct_admin("reserve", 0)
```

例 5 プロセス 83 に属するトランザクションをアポートします。このプロセスは、ログ中断モードである必要があります。LCT に達したトランザクション・ログ内のトランザクションだけが終了します。

```
select lct_admin("abort", 83)
```

例 6 dbid が 5 のデータベース内にあるすべてのオープン・トランザクションをアポートします。これにより、ログ・セグメントのラストチャンス・スレッシュヨールドで中断されている可能性のあるプロセスがすべて起動されます。

```
select lct_admin("abort", 0, 5)
```

例 7 dbid が 5 である、pubs2 データベース内でロールバック用に確保されているページ数を判断します。

```
select lct_admin("reserved_for_rollback", 5, 0)
```

例 8 dbid が 4 のデータベースで使用可能な空き領域を示します。

```
select lct_admin("logsegment_freepages", 4)
```

使用方法

- システム関数 `lct_admin` は、ログ・セグメントのラストチャンス・スレッシュヨールドを管理します。システム関数の一般的な情報については、『Transact-SQL ユーザーズ・ガイド』を参照してください。
- `lct_admin("lastchance", dbid)` が 0 を返すときは、ログはこのデータベース内の別のセグメントにはないため、ラストチャンス・スレッシュヨールドは存在しません。

- 別のログ・セグメントを持つデータベースを作成すると、サーバでは常に、[sp_thresholdaction](#) をデフォルトで呼び出すラストチャンス・スレッシュールドが作成されます。サーバ上に [sp_thresholdaction](#) プロシージャがない場合にも、このようにラストチャンス・スレッシュールドが作成されます。

ログがラストチャンス・スレッシュールドを超えると、Adaptive Server は活動を中断して [sp_thresholdaction](#) を呼び出そうとします。このプロシージャが存在していないことが判明すると、エラーを生成し、ログがトランケート可能になるまでプロセスを中断します。

- LCT に達したトランザクション・ログ内で最も古いオープン・トランザクションを終了するには、そのトランザクションを開始したプロセスの ID を入力します。
- LCT に達したトランザクション・ログ内のすべてのオープン・トランザクションを終了するには、*process-id* として 0 を入力し、*database-id* パラメータにデータベース ID を指定します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

システム管理者だけが `lct_admin abort` を実行できます。すべてのユーザはその他の `lct_admin` オプションを実行できます。

参照

マニュアル 『システム管理ガイド』を参照してください。

コマンド [dump transaction](#)

関数 [curunreservedpgs](#)

システム・プロシージャ [sp_thresholdaction](#)

left

説明	文字列の左端から指定された数の文字を返します。
構文	<code>left(character_expression, integer_expression)</code>
パラメータ	<p>character_expression 左側にある文字が選択される文字列です。</p> <p>integer_expression 返される文字数を指定する正の整数値です。<i>integer_expression</i> が負の場合は、エラーが返されます。</p>
例	<p>例 1 各本のタイトルの左端の 5 文字を返します。</p> <pre>use pubs select left(title, 5) from titles order by title_id ----- The B Cooki You C Sushi (18 row(s) affected)</pre> <p>例 2 文字列 "abcdef" の左端の 2 文字を返します。</p> <pre>select left("abcdef", 2) ----- ab (1 row(s) affected)</pre>
使用法	<ul style="list-style-type: none"> character_expression には、<code>varchar</code> または <code>nvarchar</code> への暗黙的な変換が可能な任意のデータ型 (<code>text</code> と <code>image</code> を除く) を使用できます。character_expression は、定数、変数、またはカラム名にできます。character_expression は、<code>convert</code> を使用して明示的に変換できます。 <code>left</code> は <code>substring(character_expression, 1, integer_expression)</code> と同じです。この関数の詳細については、substring (269 ページ) を参照してください。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>left</code> 関数を実行できます。
参照	<p>データ型 <code>varchar</code>、<code>nvarchar</code></p> <p>関数 len、str_replace、substring</p>

len

説明	後続ブランクを除き、指定した文字列式の文字数 (バイト数ではない) を返します。
構文	<code>len(string_expression)</code>
パラメータ	string_expression 評価する文字列式です。
例	title_id エントリ “PC9999” の文字数を返します。 <pre>select len(notes) from titles where title_id = "PC9999" ----- 39</pre>
使用法	この関数は、 char_length(string_expression) と同じです。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>len</code> 関数を実行できます。
参照	データ型 char、nchar、varchar、または nvarchar 関数 char_length 、 left 、 str_replace

license_enabled

説明	機能のライセンスが有効になっている場合は 1 を返し、有効になっていない場合は 0 を返します。また、無効なライセンス名が指定された場合は NULL を返します。
構文	<code>license_enabled("ase_server" "ase_ha" "ase_dtm" "ase_java" "ase_asm")</code>
パラメータ	ase_server Adaptive Server のライセンスを指定します。 ase_ha Adaptive Server の高可用性機能のライセンスを指定します。 ase_dtm Adaptive Server 分散トランザクション管理機能のライセンスを指定します。 ase_java Adaptive Server 機能の Java のライセンスを指定します。 ase_asm Adaptive Server の高度セキュリティ・メカニズムのライセンスを指定します。
例	Adaptive Server 分散トランザクション管理機能が有効であることを示します。 <pre>select license_enabled("ase_dtm") ----- 1</pre>
使用法	<ul style="list-style-type: none">Adaptive Server 機能のライセンス・キーをインストールする方法については、『インストール・ガイド』を参照してください。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>license_enabled</code> 関数を実行できます。
参照	マニュアル 使用しているプラットフォームの『インストール・ガイド』 システム・プロシージャ sp_configure

list_appcontext

説明	現在のセッション内にある全コンテキストの属性をすべてリストします。 list_appcontext は ACF から提供されます。
構文	list_appcontext(["context_name"])
パラメータ	<p>context_name</p> <p>セッション内にあるすべてのアプリケーション・コンテキストの属性を指定するとき使用するオプションの引数です。</p>
例	<p>例 1 適切なパーミッションを持つユーザがアプリケーション・コンテキストをリストしようとしたときの結果を示します。</p> <pre>select list_appcontext ([context_name]) Context Name: (CONTEXT1) Attribute Name: (ATTR1) Value: (VALUE2) Context Name: (CONTEXT2) Attribute Name: (ATTR1) Value: (VALUE1)</pre> <p>例 2 適切なパーミッションを持たないユーザがアプリケーション・コンテキストをリストしようとしたときの結果を示します。</p> <pre>select list_appcontext () Select permission denied on built-in list_appcontext, database DBID ----- -1</pre>
使用法	<ul style="list-style-type: none"> この関数は成功すると 0 を返します。 組み込み関数が複数の結果セットを返すことはないので、クライアント・アプリケーションは list_appcontext の戻り値をメッセージとして受け取ります。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	パーミッションは、ACF に格納されているユーザ・プロファイルとアプリケーション・プロファイルに依存します。
参照	<p>ACF の詳細については、『システム管理ガイド』の「第 11 章 ユーザ・パーミッションの管理」の「ロー・レベル・アクセス制御」を参照してください。</p> <p>関数 get_appcontext, list_appcontext, rm_appcontext, set_appcontext</p>

locator_literal

説明	バイナリ値をロケータのリテラルとして表します。
構文	<code>locator_literal(<i>locator_type</i>, <i>literal_locator</i>)</code>
パラメータ	<i>locator_type</i> ロケータのタイプです。 <code>text_locator</code> 、 <code>image_locator</code> 、または <code>unitext_locator</code> のいずれかになります。 <i>literal_locator</i> LOB ロケータの実際のバイナリ値です。
例	次の例では、メモリに保存されて、 <code>my_table</code> の <code>imagecol</code> カラムのロケータで特定される <code>image</code> の LOB を挿入します。 <code>locator_literal</code> 関数を使用して、Adaptive Server がバイナリ値を LOB ロケータとして正しく解釈していることを確認します。 <pre>insert my_table (imagecol) values (locator_literal(image_locator, 0x9067ef450100000000010000000040100400800000000))</pre>
使用法	Adaptive Server がリテラル・ロケータ値を正しく識別し、この値を <code>image</code> や他のバイナリとして誤って解釈していないことを確認するために、 <code>locator_literal</code> を使用します。
パーミッション	すべてのユーザが <code>locator_literal</code> 関数を実行できます。
参照	コマンド <code>deallocate locator</code> 、 <code>truncate lob</code> Transact-SQL 関数 <code>locator_valid</code> 、 <code>return_lob</code> 、 <code>create_locator</code>

locator_valid

説明	LOB ロケータが有効かどうかを指定します。
構文	<code>locator_valid (locator_descriptor)</code>
パラメータ	locator_descriptor LOB ロケータの有効な表現です。ロケータのホスト変数、ローカル変数、またはリテラル・バイナリ値です。
例	ロケータ値 <code>0x9067ef4501000000001000000040100400800000000</code> を検証します。 <pre>locator_valid (0x9067ef4501000000001000000040100400800000000) ----- 1</pre>
使用法	<ul style="list-style-type: none">• <code>locator_valid</code> は指定されたロケータが有効な場合は 1 を返します。それ以外の場合は 0 (ゼロ) を返します。• <code>deallocate lob</code> コマンドによって無効にされた場合、またはトランザクションの終了時にロケータは無効になります。
パーミッション	すべてのユーザが <code>locator_valid</code> 関数を実行できます。
参照	コマンド <code>deallocate locator</code> 、 <code>truncate lob</code> Transact-SQL 関数 <code>locator_literal</code> 、 <code>return_lob</code> 、 <code>create_locator</code>

lockscheme

説明	指定されたオブジェクトのロック・スキームを文字列として返します。
構文	<pre>lockscheme(object_name) lockscheme(object_id [, db_id])</pre>
パラメータ	<p>object_name ロック・スキームを返すオブジェクトの名前です。 <i>object_name</i> には完全修飾名も指定できます。</p> <p>db_id <i>object_id</i> によって指定されたデータベースの ID です。</p> <p>object_id ロック・スキームを返すオブジェクトの ID です。</p>
例	<p>例 1 現在のデータベースの titles テーブルのロック・スキームを選択します。</p> <pre>select lockscheme("titles")</pre> <p>例 2 データベース ID が 4 のデータベース (pubs2 データベース) の <i>object_id</i> 224000798 のオブジェクト (この例では titles テーブル) のロック・スキームを選択します。</p> <pre>select lockscheme(224000798, 4)</pre> <p>例 3 titles テーブルのロック・スキームを返します (この例では、<i>object_name</i> には完全修飾名を指定しています)。</p> <pre>select lockscheme(tempdb.ownerjoe.titles)</pre>
使用法	<ul style="list-style-type: none">lockscheme() は varchar(11) を返します。このコマンドでは、NULL を使用できません。次の場合は、lockscheme のデフォルトは現在のデータベースになります。<ul style="list-style-type: none">完全に修飾された <i>object_name</i> を指定していない。<i>db_id</i> を指定していない。<i>db_id</i> に null を指定した。指定されたオブジェクトがテーブルでない場合は、lockscheme は文字列 “not a table” を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが lockscheme 関数を実行できます。

log

説明	指定した数値の自然対数を計算します。
構文	<code>log(<i>approx_numeric</i>)</code>
パラメータ	<i>approx_numeric</i> 概数値型 (<code>float</code> 、 <code>real</code> 、または <code>double precision</code>) のカラム名、変数、または定数式のいずれかです。
例	<pre>select log(20) ----- 2.995732</pre>
使用法	<ul style="list-style-type: none">算術関数 <code>log</code> は、指定した値の自然対数を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>log</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 log10 、 power

log10

説明	指定した数の常用対数 (底が 10 の対数) を計算します。
構文	<code>log10(<i>approx_numeric</i>)</code>
パラメータ	<i>approx_numeric</i> 概数値型 (<code>float</code> 、 <code>real</code> 、または <code>double precision</code>) のカラム名、変数、または定数式のいずれかです。
例	<pre>select log10(20) ----- 1.301030</pre>
使用法	<ul style="list-style-type: none">算術関数 <code>log10</code> は、指定した値の常用対数 (底が 10 の対数) を返します。
標準規格	ANSI SQL – 準拠レベル: Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>log10</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 log 、 power

lower

説明	大文字から小文字に変換します。
構文	<code>lower(char_expr uchar_expr)</code>
パラメータ	char_expr 文字型のカラム名、変数、または (char、varchar、nvarchar、または nvarchar 型の) 定数式です。 uchar_expr 文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。
例	<pre>select lower(city) from publishers ----- boston washington berkeley</pre>
使用法	<ul style="list-style-type: none">文字列関数 <code>lower</code> は大文字を小文字に変換して、文字値を返します。<code>lower</code> 関数は <code>upper</code> 関数の逆変換です。<code>char_expr</code> または <code>uchar_expr</code> が NULL の場合は、NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>lower</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 upper

lprofile_id

説明 指定したログイン・プロファイル名のログイン・プロファイル ID、または現在のログインもしくは指定したログイン名に関連付けられたログイン名・プロファイルのログイン・プロファイル ID を返します。

構文 lprofile_id(*name*)

パラメータ

name

(オプション) ログイン・プロファイルまたはログイン名を削除します。

ログイン・プロファイル名を指定した場合、lprofile_id は対応するログイン・プロファイル ID を返します。ログイン名を指定した場合、lprofile_id は関連付けられた (存在する場合) ログイン・プロファイル ID を返します。

name を指定しない場合、lprofile_id は現在のログインのログイン・プロファイル ID を返します。

lprofile_name

説明 指定したログイン・プロファイル ID のログイン・プロファイル名、または現在のログインもしくは指定した `suid` に関連付けられたログイン・プロファイルのログイン・プロファイル名を返します。

構文 `lprofile_id(ID)`

パラメータ

ID

(オプション) ログイン・プロファイル ID またはログイン `suid`。

ログイン・プロファイル ID を指定した場合、`lprofile_name` は対応するログイン・プロファイル名を返します。ログイン `suid` を指定した場合、`lprofile_name` は関連付けられた (存在する場合) ログイン・プロファイル名を返します。

ID を指定しない場合、`lprofile_name` は現在のログインのログイン・プロファイル名を返します。

ltrim

説明	指定した式から先行空白を削除します。
構文	<code>ltrim(char_expr uchar_expr)</code>
パラメータ	char_expr 文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。 uchar_expr 文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。
例	<pre>select ltrim(" 123") ----- 123</pre>
使用法	<ul style="list-style-type: none">文字列関数 <code>ltrim</code> は文字式から先行空白を削除します。現在の文字セットのスペース文字に相当する値だけを削除します。<code>char_expr</code> または <code>uchar_expr</code> が NULL の場合は、NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>ltrim</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 rtrim

max

説明	式内の最大値を返します。
構文	<code>max(expression)</code>
パラメータ	<p>expression</p> <p>カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。</p>
例	<p>例 1 salesdetail テーブルの <code>discount</code> カラム内の最大値を、新しいカラムとして返します。</p> <pre>select max(discount) from salesdetail</pre> <p>-----</p> <p style="text-align: center;">62.200000</p> <p>例 2 salesdetail テーブルの <code>discount</code> カラム内の最大値を、新しいローとして返します。</p> <pre>select discount from salesdetail compute max(discount)</pre>
使用法	<ul style="list-style-type: none"> • 集合関数 max はカラムまたは式の最大値を検索します。集合関数の一般的な情報については、『Transact-SQL ユーザーズ・ガイド』を参照してください。 • max 関数は、真数値、概数値、文字、datetime のカラムに対して使用できます。bit カラムに対しては使用できません。文字型のカラムの場合、max は照合順の最大値を検索します。max は、null 値を無視します。max 関数は、char データ型を varchar に、unichar データ型を univarchar に暗黙的に変換し、後続ブランクをすべて削除します。 • unichar データは、デフォルトの Unicode ソート順序に従って照合されます。 • max は varbinary データの後続のゼロを保持します。 • max は varbinary データ型を binary データのクエリから返します。 • 集合カラムにインデックスがあるときには、次に示す状況の場合を除き、Adaptive Server はインデックスの終わりに直接移動して max の最終ローを検索します。 <ul style="list-style-type: none"> • expression がカラムでない場合。 • カラムがインデックスの先頭カラムでない場合。 • クエリに別の集合関数がある場合。 • group by 句または where 句がある場合。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが max 関数を実行できます。
参照	<p>コマンド compute 句, group by 句 と having 句, select, where 句</p> <p>関数 avg, min</p>

min

説明	カラム内の最小値を返します。
構文	<code>min(expression)</code>
パラメータ	expression カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、 <code>expression</code> はカラム名です。詳細については、「式」(325 ページ)を参照してください。
例	<pre>select min(price) from titles where type = "psychology" ----- 7.00</pre>
使用法	<ul style="list-style-type: none">• 集合関数 <code>min</code> はカラム内の最小値を検索します。• <code>min</code> 関数は、数値、文字、<code>time</code> カラム、<code>datetime</code> カラムに対して使用できます。<code>bit</code> カラムに対しては使用できません。文字型のカラムの場合、<code>min</code> は、ソート順での最小値を検索します。<code>min</code> 関数は、<code>char</code> データ型を <code>varchar</code> に、<code>unichar</code> データ型を <code>univarchar</code> に暗黙的に変換し、後続空白をすべて削除します。<code>min</code> は、<code>null</code> 値を無視します。<code>distinct</code> は意味がないため、<code>min</code> では使用できません。• <code>min</code> は <code>varbinary</code> データの後続のゼロを保持します。• <code>min</code> は <code>varbinary</code> データ型を <code>binary</code> データのクエリから返します。• <code>unichar</code> データは、デフォルトの Unicode ソート順序に従って照合されます。• 集合カラムにインデックスがあるときは、次に示す状況の場合を除き、Adaptive Server は <code>min</code> 関数の条件を満たす最初のローに直接移動します。<ul style="list-style-type: none">• <code>expression</code> がカラムでない場合。• カラムがインデックスの先頭カラムでない場合。• クエリに別の集合関数がある場合。• <code>group by</code> 句がある場合。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>min</code> 関数を実行できます。
参照	コマンド compute 句 、 group by 句 と having 句 、 select 、 where 句 マニュアル Transact-SQL ユーザーズ・ガイド 関数 avg 、 max

month

説明	指定した日付の <code>datepart</code> の月を表す整数を返します。
構文	<code>month(date_expression)</code>
パラメータ	<code>date_expression</code> <code>datetime</code> 、 <code>smalldatetime</code> 、 <code>date</code> 型の式、または <code>datetime</code> フォーマットの文字列です。
例	整数 11 を返します。 <pre>day("11/02/03") ----- 11</pre>
使用法	<code>month(date_expression)</code> は <code>datepart(mm, date_expression)</code> と同じです。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>month</code> 関数を実行できます。
参照	データ型 <code>datetime</code> 、 <code>smalldatetime</code> 、 <code>date</code> 関数 <code>datepart</code> 、 <code>day</code> 、 <code>year</code>

mut_excl_roles

説明	2つの役割の間の相互排他性についての情報を返します。
構文	<code>mut_excl_roles (role1, role2 [membership activation])</code>
パラメータ	<p><i>role1</i> 相互排他性の関係にある、ユーザ定義の役割です。</p> <p><i>role2</i> 相互排他性の関係にある、もう1つのユーザ定義の役割です。</p> <p><i>level</i> 指定した役割が排他関係になるレベル (membership または activation) です。</p>
例	<p><code>admin</code> と <code>supervisor</code> の役割がお互いに排他関係であることを示しています。</p> <pre>alter role admin add exclusive membership supervisor select mut_excl_roles("admin", "supervisor", "membership") ----- 1</pre>
使用法	<ul style="list-style-type: none"> システム関数 <code>mut_excl_roles</code> は、2つの役割の間の相互排他性についての情報を返します。システム・セキュリティ担当者が、<code>role1</code> を <code>role2</code> または <code>role2</code> に直接含まれている役割と相互に排他的であると定義すると、<code>mut_excl_roles</code> は 1 を返します。これらの役割が相互に排他的ではないときは、<code>mut_excl_roles</code> は 0 を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>mut_excl_roles</code> 関数を実行できます。
参照	<p>コマンド alter role, create role, drop role, grant, set, revoke</p> <p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 proc_role, role_contain, role_id, role_name</p> <p>システム・プロシージャ sp_activeroles, sp_displayroles, sp_role</p>

newid

説明 指定した引数に基づいて、人間が判読できるグローバルにユニークな識別子 (GUID) を 2 とおりの異なるフォーマットで生成します。人間が判読できる GUID 値のフォーマットの長さは、32 バイト (ダッシュなし) または 36 バイト (ダッシュ付き) のいずれかです。

構文 `newid([optionflag])`

パラメータ

option flag

- 0 または値の指定なし - 生成される GUID は、人間が判読できる値 (`varchar`) ですが、ダッシュは含まれません。この引数はデフォルトであり、値を `varbinary` に変換するのに便利です。
- -1 - 生成される GUID は、人間が判読できる値 (`varchar`) であり、ダッシュが含まれます。
- -0x0 - GUID を `varbinary` として返します。
- `newid` のその他の値はすべて NULL を返します。

例 **例 1** 32 バイト長の `varchar` カラムを持つテーブルを作成し、`insert` 文で引数なしの `newid` を使います。

```
create table t (UUID varchar(32))
go
insert into t values (newid())
insert into t values (newid())
go
select * from t

UUID
-----
f81d4fae7dec11d0a76500a0c91e6bf6
7cd5b7769df75cfe040800208254639
```

例 2 ダッシュ付きの GUID を生成します。

```
select newid(1)
-----
b59462af-a55b-469d-a79f-1d6c3c1e19e3
```

例 3 ダッシュなしの GUID フォーマットを `varbinary(16)` カラムに変換するデフォルトを作成します。

```
create table t (UUID_VC varchar(32), UUID varbinary(16))
go
create default default_guid
as
strtobin(newid())
go
sp_bindefault default_guid, "t.UUID"
go
insert t (UUID_VC) values (newid())
go
```

例 4 クエリで返されるローのそれぞれに **varbinary** 型の新しい GUID を作成して返します。

```
select newid(0x0) from sysobjects
```

例 5 **newid** に **varbinary** データ型を使用します。

```
sp_addtype binguid, "varbinary(16)"
create default binguid_dflt
as
newid(0x0)
sp_bindefault "binguid_dflt","binguid"
create table T1 (empname char(60), empid int, emp_guid binguid)
insert T1 (empname, empid) values ("John Doe", 1)
insert T1 (empname, empid) values ("Jane Doe", 2)
```

使用法

- **newid** は、**newid** に渡された引数に基づいて、2 通りの GUID 値を生成します。デフォルトの引数では、ダッシュなしの GUID が生成されます。デフォルトで **newid** は、フィルタされたローごとに新しい値を返します。
- **newid** は、他の関数と同様に、デフォルト、ルール、トリガで使用できます。
- ダッシュなしの GUID フォーマットでは **varchar** カラムの長さが 32 バイト以上あることを、ダッシュ付きの GUID フォーマットでは 36 バイト以上あることを確認してください。カラム長がこれらの最低限必要な長さで宣言されていない場合は、カラム長がトランケートされます。カラム長がトランケートされると、重複値が生じる可能性があります。
- 引数ゼロはデフォルトと同じです。
- ダッシュなしの GUID フォーマットを **strtobin** 関数とともに使うと、GUID 値を 16 バイトのバイナリ・データに変換できます。一方、**strtobin** 関数でダッシュ付きの GUID 形式を使用すると、結果は NULL 値になります。
- GUID はグローバルにユニークなため、ドメイン間でトランスポートしても、値が重複することはありません。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが **newid** 関数を実行できます。

next_identity

説明	次の insert に使用可能な次の identity 値を取得します。
構文	<code>next_identity(table_name)</code>
パラメータ	table_name 使用するテーブルを指定します。
例	テーブル t1 の identity 値は 10 まで利用されています。次に使用可能な値は 11 です。 <pre>select next_identity ("t1") t1 ----- 11</pre>
使用法	<ul style="list-style-type: none">• <code>next_identity</code> は、このタスクによって挿入される次の値を返します。複数のユーザが同じテーブルに値を挿入しようとしている場合、次の値として報告された値でも、他のユーザがその間に値を挿入してしまうと、実際に挿入される値と異なる場合があります。• <code>next_identity</code> は、任意の精度の identity カラムをサポートする <code>varchar</code> 文字を返します。テーブルがプロキシ・テーブル、ユーザ・テーブル以外のテーブル、または identity プロパティを持たないテーブルの場合は、NULL が返されます。
パーミッション	テーブル所有者、システム管理者、データベース管理者のみがこのコマンドを発行できます。

nullif

説明 条件値を使用するための SQL 式を記述できます。nullif 式は、値式が使用できる場所であればどこでも case 式の代わりに使用できます。

構文 nullif(expression, expression)

パラメータ nullif

2 つの式の値を比較します。最初の式が 2 番目の式と同じであれば、nullif は NULL を返します。最初の式と 2 番目の式が異なれば、nullif は最初の式を返します。

expression

カラム名、定数、関数、サブクエリ、またはこれらを算術演算子やビット処理演算子でつないだ任意の組み合わせです。式の詳細については、「[式 \(325 ページ\)](#)」を参照してください。

例 titles テーブルから titles と type を選択します。本のタイプが UNDECIDED の場合、nullif は NULL 値を返します。

```
select title,
       nullif(type, "UNDECIDED")
from titles
```

または、次のように記述することもできます。

```
select title,
       case
         when type = "UNDECIDED" then NULL
         else type
       end
from titles
```

使用法

- nullif 式は case の代替式として使用されます。
- nullif 式は、when...then 構成を使用する代わりに探索条件を単純な比較として表すことによって、標準 SQL 式を簡素化します。
- nullif 式は、SQL 内で式が使用可能な任意の場所で使用できます。
- case 式の 1 つ以上の結果が null 以外の値を返す必要があります。たとえば、次の場合はエラー・メッセージが表示されます。

```
select price, coalesce (NULL, NULL, NULL)
from titles
All result expressions in a CASE expression must not be NULL.
```

- クエリがさまざまなデータ型を作成する場合は、「[混合モードの式のデータ型](#)」(6 ページ)で説明されているとおり、データ型の階層によってcase 式の結果のデータ型が決定されます。Adaptive Server で暗黙的に変換されないデータ型 (char や int など) を 2 つ指定した場合、クエリは正常に動作しません。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが nullif 関数を実行できます。

参照

コマンド [case](#), [coalesce](#), [select](#), [if...else](#), [where](#) 句

object_attr

説明 セッション、テーブル設定、データベースワイド設定に応じて、テーブルの現在のロギング・モードをレポートします。

構文 `object_attr(table_name, string)`

パラメータ **table_name**
テーブルの名前です。

string

クエリ対象のテーブルのプロパティ名です。サポートされている文字列値は以下のとおりです。

- **dml_logging** – 要求した有効なオブジェクトの DML ロギング・レベルを、明示的に設定されたテーブルまたはデータベースの DML ロギング・レベルに基づいて返します。
- **dml_logging for session** – 現在のセッションの DML ロギング・レベルを、**object_attr** を実行しているユーザ、テーブルのスキーマ、複数文のトランザクションに関するルールなどを考慮して返します。この引数の戻り値は、ユーザによって異なる場合があります。同じユーザでも文やトランザクションごとに異なる可能性があります。
- **compression** – 要求されたオブジェクトの圧縮タイプを返します。
- **help** – サポートされている文字列引数のリストを出力します。

例 **例 1** どのプロパティをクエリできるかを確認するには、次のコマンドを実行します。

```
select object_attr('sysobjects', 'help')
Usage: object_attr('tablename', 'attribute')
```

```
List of options in attributes table:
help
dml_logging
"dml_logging for session"
3 : compression
```

dml_logging は、オブジェクトの統計的に定義された **dml_logging** レベルをレポートし、**dml_logging for session** は、データベース固有の設定とセッションの設定に応じてオブジェクトに選択された実行時ロギング・レベルをレポートします。

例 2 持続性を **full** に設定したテーブルのデフォルトのロギング・モード：

```
select object_attr("pubs2..authors",
"dml_logging")
```

```
Returns: FULL
```

例 3 セッションですべてのテーブルのロギングを無効にしている場合、このユーザが所有するテーブルに対して返されるロギング・モードは **minimal** です。

```
select object_attr("pubs2..authors",
                  "dml_logging")
```

Returns: FULL

```
SET DML_LOGGING MINIMAL
go
```

```
select object_attr("pubs2..authors",
                  "dml_logging for session")
```

Returns: MINIMAL

例 4 テーブルが最低限のロギングを明示的に選択するように変更されている場合は、セッションおよびデータベース全体のロギングが **FULL** でも、**object_attr** は **minimal** の値を返します。

```
create database testdb WITH DML_LOGGING = FULL
go
```

```
create table non_logged_table (...)
WITH DML_LOGGING=MINIMAL
go
```

```
select object_attr("non_logged_table",
                  "dml_logging")
```

Returns: MINIMAL

例 5 テーブルのロギングを **full** から **minimal** に変更します。full ロギングを使用して明示的にテーブルを作成した場合は、テーブルの所有者または **sa_role** のユーザであれば、セッション中にロギングを **minimal** に再設定できます。

- 1 最低限のロギングを使用して **testdb** データベースを作成します。

```
create database testdb
with dml_logging = minimal
```

- 2 **dml_logging** を **full** に設定してテーブルを作成します。

```
create table logged_table(...)
with dml_logging = full
```

- 3 セッションのロギングを **minimal** に再設定します。

```
set dml_logging minimal
```

- 4 テーブルのロギングは最小限です。

```
select object_attr("logged_table",
                  "dml_logging for session")
-----
minimal
```

例 6 ロギング・モードを指定せずにテーブルを作成した場合に、セッションのロギング・モードを変更すると、テーブルのロギング・モードも変更されます。

- テーブル `normal_table` を作成します。

```
create table normal_table
```

- セッションのロギングをチェックします。

```
select object_attr("normal_table", "dml_logging")
-----
FULL
```

- セッションのロギングを `minimal` に設定します。

```
set dml_logging minimal
```

- テーブルのロギングを `minimal` に設定します。

```
select object_attr("normal_table",
                  "dml_logging for session")
-----
minimal
```

例 7 `object_attr` から返されるロギング・モードは、それを実行するテーブルによって異なります。この例では、ユーザ `joe` がスクリプトを実行しますが、Adaptive Server が返すロギング・モードが変わります。テーブル `joe.own_table` と `mary.other_table` は、`full` ロギング・モードを使用します。

```
select object_attr("own_table", "dml_logging")
-----
FULL
```

`joe` が `mary.other_table` に `object_attr` を実行すると、このテーブルも `full` に設定されます。

```
select object_attr("mary.other_table", "dml_logging")
-----
FULL
```

`joe` が `dml_logging` を `minimal` に変更した場合は、`joe` が所有しているテーブルのロギング・モードのみが影響を受けます。

```
set dml_logging minimal
select object_attr("own_table", "dml_logging for session")
-----
MINIMAL
```

他のユーザが所有しているテーブルは、それぞれのデフォルトのロギング・モードで処理を続行します。

```
Select object_attr("mary.other_table", "dml_logging for
session")
-----
FULL
```


例 8 新しい `show_exec_info` をロギングする実行時の選択を識別し、それを SQL バッチで使用します。

- 1 `set showplan` を有効にします。

```
set showplan on
```

- 2 `set` コマンドを有効にします。

```
set show_exec_info on
```

- 3 `dml_logging` を `minimal` に設定し、`object_attr` でロギングをチェックします。

```
set dml_logging minimal
select object_attr("logged_table", "dml_logging for
session")
```

- 4 テーブルからローを削除します。

```
delete logged_table
```

Adaptive Server は実行時にテーブルのロギング・モードを `show_exec_info` パラメータでレポートします。

使用法

- 戻り型は `varchar` で、クエリ対象のプロパティに応じてプロパティの値 (たとえば、`on` または `off`) を適切に返します。
- 拡張から `showplan` 出力にレポートされるロギング・モードは、DML の実行前、同じバッチにテーブルのロギング・モードを変更する `set` 文があると、実行時に影響を受ける可能性があります。
- 不明のプロパティの戻り値は値 `NULL` です (文字列 “`NULL`” ではありません)。
- 特殊な文字列パラメータ、`help` はセッションの出力に、現在サポートされている `object_attr` のプロパティすべてを出力します。そのため、`object_attr` がどのプロパティをサポートしているかを即座に確認できます。

object_id

説明

指定したオブジェクトのオブジェクト ID を返します。

構文

```
object_id(object_name)
```

パラメータ**object_name**

データベース・オブジェクト (テーブル、ビュー、プロシージャ、トリガ、デフォルト、またはルールなど) の名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。**object_name** は引用符で囲んでください。

例**例 1**

```
select object_id("titles")
-----
      208003772
```

例 2

```
select object_id("master..sysobjects")
-----
      1
```

使用法

- システム関数 **object_id** はオブジェクトの ID を返します。オブジェクト ID は **sysobjects** の **id** カラムに格納されています。
- リソースを消費する代わりに、**object_id** はまだキャッシュに存在しないオブジェクトの記述子を廃棄します。

標準規格

ANSI SQL – 準拠レベル: *Transact-SQL* 拡張機能です。

パーミッション

すべてのユーザが **object_id** 関数を実行できます。

参照

マニュアル *Transact-SQL ユーザーズ・ガイド*

関数 [col_name](#), [db_id](#), [object_name](#)

システム・プロシージャ [sp_help](#)

object_name

説明	指定したオブジェクト ID を持つオブジェクトの名前を返します。名前は最長で 255 バイトです。
構文	<code>object_name(object_id[, database_id])</code>
パラメータ	<p>object_id (テーブル、ビュー、プロシージャ、トリガ、デフォルト、ルールなどの)データベース・オブジェクトのオブジェクト ID です。オブジェクト ID は <code>sysobjects</code> の <code>id</code> カラムに格納されています。</p> <p>database_id オブジェクトが現在のデータベースにない場合に使用するデータベースの ID です。データベース ID は <code>sysdatabases</code> の <code>db_id</code> カラムに格納されています。</p>
例	<p>例 1</p> <pre>select object_name(208003772) ----- titles</pre> <p>例 2</p> <pre>select object_name(1, 1) ----- sysobjects</pre>
使用法	システム関数 <code>object_name</code> は、オブジェクトの名前を返します。
標準規格	ANSI SQL - 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>object_name</code> 関数を実行できます。
参照	<p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 <code>col_name</code>, <code>db_id</code>, <code>object_id</code></p> <p>システム・プロシージャ <code>sp_help</code></p>

object_owner_id

説明	オブジェクトの所有者 ID を返します。
構文	<code>object_owner_id(object_id[, database_id])</code>
パラメータ	object_id 調べるオブジェクトの ID です。 database_id オブジェクトが存在するデータベースの ID です。
例	ID が 1 のデータベース (マスタ・データベース) で ID が 1 のオブジェクトの所有者の ID を返します。 <pre>select object_owner_id(1,1)</pre>
パーミッション	すべてのユーザが <code>object_owner_id</code> を実行できます。

pagesize

説明 指定されたオブジェクトのページ・サイズをバイト数で返します。

構文

```
pagesize(object_name[, ])
pagesize(object_id[, db_id[, index_id]])
```

パラメータ *object_name*
この関数によってページ・サイズが返されるオブジェクトの名前です。

index_name
返されるページ・サイズのインデックスの名前です。

object_id
この関数によってページ・サイズが返されるオブジェクトの ID です。

db_id
オブジェクトのデータベース ID です。

index_id
返されるオブジェクトのインデックス ID です。

例 **例 1** 現在のデータベースの *title_id* インデックスのページ・サイズを返します。

```
select pagesize("title", "title_id")
```

例 2 *db_id* が 2 のデータベースで、*object_id* が 1234 のオブジェクトのデータ・レイヤのページ・サイズを返します (前の例は、デフォルトとして現在のデータベースになります)。

```
select pagesize(1234,2, null)
select pagesize(1234.2)
select pagesize(1234)
```

例 3 次の例は、すべて現在のデータベースのデフォルトになります。

```
select pagesize(1234, null, 2)
select pagesize(1234)
```

例 4 pubs2 データベース (*db_id* 4) の *titles* テーブル (*object_id* 224000798) のページ・サイズを返します。

```
select pagesize(224000798, 4)
```

例 5 現在のデータベースにある、ノンクラスタード・インデックスのページ・テーブル *mytable* のページ・サイズを返します。

```
pagesize(object_id(僭ytable1), NULL, 2)
```

例 6 現在のデータベースから、オブジェクト *titles_clustindex* のページ・サイズを返します。

```
select pagesize("titles", "titles_clustindex")
```

使用法

- インデックス名または *index_id* を指定しない場合 (たとえば、`select pagesize("t1")`)、またはパラメータに“null”を使用した場合 (たとえば、`select pagesize("t1", null)`) は、**pagesize** の対象はデフォルトでデータ・レイヤになります。
- 指定されたオブジェクトがページ用の物理データ記憶領域を必要としないオブジェクトの場合は (たとえば、ビューの名前を指定した場合)、**pagesize** は 0 を返します。
- 指定したオブジェクトが存在していない場合、**pagesize** は NULL を返します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが **pagesize** 関数を実行できます。

partition_id

説明	指定したデータまたはインデックス・パーティション名のパーティション ID を返します。
構文	<code>partition_id(table_name, partition_name[, index_name])</code>
パラメータ	<p>table_name テーブルの名前です。</p> <p>partition_name テーブル・パーティションまたはインデックス・パーティションのパーティション名です。</p> <p>index_name 対象となるインデックスの名前です。</p>
例	<p>例 1 パーティション名 <code>testtable_ptn1</code> とインデックス ID 0 (ベース・テーブル) に対応するパーティション ID を返します。 <code>testtable</code> は、現在のデータベースに存在している必要があります。</p> <pre>select partition_id("testtable", "testtable_ptn1")</pre> <p>例 2 インデックス名 <code>clust_index1</code> のパーティション名 <code>testtable_clust_ptn1</code> に対応するパーティション ID を返します。 <code>testtable</code> は、現在のデータベースに存在している必要があります。</p> <pre>select partition_id("testtable", "testtable_clust_ptn1", "clust_index1")</pre> <p>例 3 これは前の例と同じですが、ターゲット・テーブルが現在のデータベースにある必要はありません。</p> <pre>select partition_id("mydb.dbo.testtable", "testtable_clust_ptn1", "clust_index1")</pre>
使用法	<code>table_name</code> 、 <code>partition_name</code> 、 <code>index_name</code> は引用符で囲む必要があります。
参照	関数 <code>data_pages</code> 、 <code>object_id</code> 、 <code>partition_name</code> 、 <code>reserved_pages</code> 、 <code>row_count</code> 、 <code>used_pages</code>

partition_name

説明	新しいパーティションの明示的な名前 <code>partition_name</code> は、指定されたデータのパーティション名またはインデックス・パーティション ID を返します。
構文	<code>partition_name(indid, ptnid [, dbid])</code>
パラメータ	indid ターゲット・パーティションのインデックス ID です。 ptnid ターゲット・パーティションの ID です。 dbid ターゲット・パーティションのデータベース ID です。このパラメータを指定しない場合、ターゲット・パーティションは現在のデータベースにあると解釈されます。
例	例 1 ベース・テーブル (インデックス ID 0) に属する特定のパーティション ID を持つパーティション名を返します。ここではデータベース ID を指定しないため、検索は現在のデータベースを対象として行われます。 <pre>select partition_name(0, 1111111111)</pre> 例 2 <code>testdb</code> データベース内で、クラスタード・インデックス (インデックス ID に 1 を指定) に属する特定のパーティション ID を持つパーティション名を返します。 <pre>select partition_name(1, 1212121212, db_id("testdb"))</pre>
使用法	<ul style="list-style-type: none">ターゲット・パーティションが見つからない場合、NULL が返されます。
参照	関数 data_pages , object_id , partition_id , reserved_pages , row_count

partition_object_id

説明	指定されたパーティション ID とデータベース ID のオブジェクト ID を返します。
構文	<code>partition_object_id(partition_id [, database_id])</code>
パラメータ	<p>partition_id 取得するオブジェクト ID が設定されているパーティションの ID です。</p> <p>database_id パーティションのデータベース ID です。</p>
例	<p>例 1 パーティション ID が 2 であるパーティションのオブジェクト ID を表示するには、次のように入力します。</p> <pre>select partition_object_id(2)</pre> <p>例 2 パーティション ID が 14 でデータベース ID が 7 であるパーティションのオブジェクト ID を表示するには、次のように入力します。</p> <pre>select partition_object_id(14,7)</pre> <p>例 3 次の場合は、関数に NULL 値が渡されているため、このデータベース ID に対して NULL 値を返します。</p> <pre>select partition_object_id(1424005073, NULL)</pre> <pre>----- NULL (1 row affected)</pre>
使用法	<ul style="list-style-type: none"> • データベース ID を含まない場合、<code>partition_object_id</code> は現在のデータベース ID を使用します。 • <code>partition_id</code> に NULL 値を使用すると、<code>partition_object_id</code> は NULL を返します。 • データベース ID に NULL 値を指定すると、<code>partition_object_id</code> は NULL 値を返します。 • 無効または存在しない <code>partition_id</code> または <code>database_id</code> を指定すると、<code>partition_object_id</code> は NULL を返します。

patindex

説明

指定したパターンが最初に検出された先頭位置を返します。

構文

```
patindex("%pattern%", char_expr|uchar_expr[, using
{bytes | characters | chars}])
```

パラメータ

pattern

データ型が `char` または `varchar` の文字式です。この式には、Adaptive Server で使用可能なパターン一致ワイルドカード文字を含めることができます。*pattern* の前後には、% ワイルドカード文字を挿入してください (最初の文字または最後の文字を検索する場合は除く)。ワイルドカード文字については、「ワイルドカード文字によるパターン一致」(344 ページ) を参照してください。

char_expr

文字型のカラム名、変数、または (`char`, `varchar`, `nchar`, `nvarchar`, `text_locator`, または `unitext_locator` 型の) 定数式です。

uchar_expr

文字型のカラム名、変数、または (`unichar` または `univarchar` 型の) 定数式です。

using

先頭位置のフォーマットを指定します。

bytes

オフセットをバイト数で返します。

chars または *characters*

オフセットを文字数で返します (デフォルト)。

例

例 1 作者 ID と `copy` カラムの “circus” という語の先頭文字の位置を選択します。

```
select au_id, patindex("%circus%", copy)
from blurbs

au_id
-----
486-29-1786      0
648-92-1872      0
998-72-3567      38
899-46-2035      31
672-71-3249      0
409-56-7008      0
```

例 2

```
select au_id, patindex("%circus%", copy,
using chars)
from blurbs
```

例 3 sysobjects 内で、“sys” で始まり、4 番目の文字が “a”、“b”、“c”、または “d” であるすべてのローを検索します。

```
select name
from sysobjects
where patindex("sys[a-d]%", name) > 0

name
-----
sysalternates
sysattributes
syscharsets
syscolumns
syscomments
sysconfigures
sysconstraints
syscurconfigs
sysdatabases
sysdepends
sysdevices
```

使用法

- 文字列関数 `patindex` は、指定した文字式内で `pattern` が最初に検出された先頭位置を表す整数を返します。`pattern` がないときは 0 を返します。
- `patindex` 関数では、`text` データや `image` データを含むすべての文字データを使用できます。
- `text`、`unitext`、および `image` データの場合、`ciphertext` を 1 に設定すると、`patindex` はサポートされません。エラー・メッセージが表示されます。
- `text`、`unitext`、および `image` データの場合、`ciphertext` を 0 に設定すると、プレーン・テキスト内のパターンのバイトまたは文字インデックスが返されます。
- `unichar`、`univarchar`、`unitext` の場合、`patindex` は、Unicode 文字内のオフセットを返します。パターン文字列は、比較の前に UTF-16 に暗黙的に変換されます。この変換は `default unicode sort order` の設定に基づいて実行されます。たとえば、`unitext` カラムに `U+0041U+0042U+d800U+dc00U+0043` というロー値が保管されている場合、次の値が返されます。

```
select patindex("%C%", ut) from unitable
-----
4
```

- デフォルトでは、`patindex` 関数はオフセットを文字数で返します。オフセットをバイト数 (マルチバイト文字列) で返すには、`using bytes` 句を指定します。
- `pattern` 引数の前後にはパーセント記号を指定してください。カラム内で、`pattern` に指定した値で始まる文字列を検索するには、この引数の前に % を指定しないでください。カラム内で、`pattern` に指定した値で終わる文字列を検索するには、この引数の後に % を指定しないでください。

- *char_expr* または *uchar_expr* が NULL の場合は、*patindex* は 0 を返します。
- *varchar* 式と *unichar* 式をそれぞれパラメータとして指定した場合、*varchar* 式は暗黙的に *unichar* に変換されます (変換時にトランケーションが発生することがあります)。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが *patindex* 関数を実行できます。

参照

マニュアル Transact-SQL ユーザーズ・ガイド

関数 [charindex](#), [substring](#)

pi

説明	定数値 (円周率) 3.1415926535897936 を返します。
構文	pi()
パラメータ	なし。
例	<pre>select pi() ----- 3.141593</pre>
使用法	算術関数 pi は、定数値 3.1415926535897931 を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが pi 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 degrees , radians

power

説明	指定した数字を指定の累乗で計算した結果の値を返します。
構文	<code>power(value, power)</code>
パラメータ	value 数値です。 power 真数値、概数値、または通貨値です。
例	<pre>select power(2, 3) ----- 8</pre>
使用法	<ul style="list-style-type: none">算術関数 power は、<i>value</i> の値を <i>power</i> 乗した結果の値を返します。結果の型は引数 <i>value</i> と同じ型になります。 numeric 型または decimal 型の式の場合、この関数は精度 38 と位取り 18 を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが power 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 exp , log , log10

proc_role

説明

指定した役割がユーザに付与されているかどうかを示す情報を返します。

注意 Sybase は、`has_role` に代えて `proc_role` をサポートしています。また、この関数の使用を推奨しています。ただし、既存の `proc_role` をすべて `has_role` に変換する必要はありません。

構文

```
proc_role("role_name")
```

パラメータ

role_name

システム定義またはユーザ定義の役割の名前です。

例

例 1 ユーザがシステム管理者かどうかをチェックするプロシージャを作成します。

```
create procedure sa_check as
if (proc_role("sa_role") > 0)
begin
    print "You are a System Administrator."
    return(1)
end
```

例 2 ユーザにシステム・セキュリティ担当者の役割が付与されているかどうかをチェックします。

```
select proc_role("sso_role")
```

例 3 ユーザにオペレータの役割が付与されているかどうかをチェックします。

```
select proc_role("oper_role")
```

使用法

- 名前が “sp_” で始まるプロシージャを指定して `proc_role` を使用すると、エラーが返されます。
- システム関数 `proc_role` は、指定した役割が呼び出し元ユーザに対して付与されているかどうかをチェックします。
- ユーザの役割が次の場合に `proc_role` は 0 を返します。
 - 指定した役割がユーザに付与されていない場合。
 - 指定した役割を含む役割がユーザに付与されていない場合。
 - 指定した役割がユーザに付与されているが、アクティブ化されていない場合。
- 指定した役割が呼び出し元ユーザに対して付与されており、アクティブ化されている場合、`proc_role` は 1 を返します。
- 呼び出し元ユーザのアクティブな役割に、指定の役割が含まれている場合、`proc_role` は 2 を返します。

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが proc_role 関数を実行できます。
参照	コマンド alter role , create role , drop role , grant , set , revoke マニュアル Transact-SQL ユーザーズ・ガイド 関数 mut_excl_roles , role_contain , role_id , role_name , show_role

pssinfo

説明	Adaptive Server プロセス・ステータス構造 (pss) から情報を返します。
構文	<code>pssinfo(spид 0, 'pss_field')</code>
パラメータ	<p>spид プロセス ID です。0 を指定すると現在のプロセスが使用されます。</p> <p>pss_field プロセス・ステータス構造体フィールドです。有効な値は次のとおりです。</p> <ul style="list-style-type: none"> • dn – LDAP 認証を使用する場合の識別名。 • extusername – PAM や LDAP のような外部認証を使用する場合、extusername は、使用される外部 PAM ユーザ名または LDAP ユーザ名を返します。 • ipaddr – クライアントの IP アドレス。 • ipport – クエリ対象のユーザ・タスクに関連するクライアント接続に使用されるクライアント IP ポート番号。 • isolation_level – 現在のセッションの分離レベル。 • tempdb_pages – 使用される tempdb ページ数。
例	<p>spид 14 番のポート番号を表示します。</p> <pre>select pssinfo(14,'ipport') ----- 52039</pre>
使用法	<ul style="list-style-type: none"> • pssinfo 関数には、外部ユーザ名と識別名を表示するオプションもあります。 • ipport 出力と ipaddr 出力を組み合わせると、Adaptive Server とクライアント間のネットワーク・トラフィックを一意に識別できます。

radians

説明	指定した度数の角度の大きさをラジアンで返します。
構文	<code>radians(numeric)</code>
パラメータ	<i>numeric</i> (<i>numeric</i> , <i>dec</i> , <i>decimal</i> , <i>tinyint</i> , <i>smallint</i> , または <i>int</i> 型の) 真数値、(<i>float</i> , <i>real</i> , または <i>double precision</i> 型の) 概数値、 <i>money</i> 型のカラム、変数、定数式、またはこれらの組み合わせのいずれかです。
例	<pre>select radians(2578) ----- 44</pre>
使用法	<ul style="list-style-type: none">算術関数 radians は、度数をラジアン数に変換します。結果は、<i>numeric</i> と同じ型になります。 <i>numeric</i> データ型または <i>numeric</i> データ型を表すため、この関数は精度 38 と位取り 18 を返します。 通貨データ型を使用すると、<i>float</i> 型への内部変換で精度ロスが発生することがあります。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが radians 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 degrees

rand

説明	指定された (オプションの) 整数値を基礎値として使用し、0 と 1 の間の浮動小数の乱数値を返します。
構文	<code>rand([integer])</code>
パラメータ	integer 整数値型 (<code>tinyint</code> 、 <code>smallint</code> 、または <code>int</code> 型) のカラム名、変数、定数式、またはこれらの組み合わせのいずれかです。
例	例 1 <pre>select rand() ----- 0.395740</pre> 例 2 <pre>declare @seed int select @seed=100 select rand(@seed) ----- 0.000783</pre>
使用法	<ul style="list-style-type: none">算術関数 rand は、オプションの整数値を基数値として使用し、0 と 1 の間の浮動小数の乱数値を返します。rand 関数は、32 ビット整数の疑似乱数値ジェネレータの出力を使用します。整数値を 32 ビットの最大整数値で除算し、0.0 ~ 1.0 の間の 2 倍の値を求めます。rand 関数の基数値はサーバの起動時に無作為に決まるので、ユーザが最初に定数の基数値でこの関数を初期化しないかぎり、同じ乱数順になることはほとんどありません。rand 関数はグローバル・リソースです。rand 関数を呼び出している複数のユーザは、一連の疑似乱数値にそって処理されます。一連の乱数を繰り返したいときは、ユーザはその関数の基数値を最初に同じ値にし、繰り返したい間は他のユーザが rand 関数を呼び出さないことを確かめてください。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが rand 関数を実行できます。
参照	データ型 概数値データ型 マニュアル Transact-SQL ユーザーズ・ガイド 関数 rand2

rand2

説明	指定したシード値を使用して生成され、 select リストで使用されたときに返された各ローに対して計算された 0 と 1 の間のランダム値を返します。
構文	<code>rand2([integer])</code>
パラメータ	<i>integer</i> 整数値型 (<code>tinyint</code> 、 <code>smallint</code> 、または <code>int</code> 型) のカラム名、変数、定数式、またはこれらの組み合わせのいずれかです。
例	t に n 個のローある場合、次の <code>select</code> 文によって 1 つだけではなく n 個の異なる乱数値が返されます。 <pre>select rand2() from t -----</pre>
使用法	<ul style="list-style-type: none">算術関数 <code>rand2</code> は、オプションの整数値を基数値として使用し、0 と 1 の間の浮動小数の乱数値を返します。<code>rand</code> とは異なり、<code>select</code> リストで使用される場合は、返されたローごとに計算されます。<code>rand2</code> の動作は、現在 <code>select</code> リストを除いて未定義です。32 ビット整数の擬似乱数ジェネレータの詳細については、rand の「使用法」を参照してください。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>rand</code> 関数を実行できます。
参照	データ型 概数値データ型 マニュアル Transact-SQL ユーザーズ・ガイド 関数 rand

replicate

説明	指定した回数か、または 16 KB の記憶領域に保存可能な最大回数のうち、少ない回数で繰り返された式を含む文字列を返します。
構文	<code>replicate(char_expr uchar_expr, integer_expr)</code>
パラメータ	char_expr 文字型のカラム名、変数、または (char、varchar、nvarchar、または nvarchar 型の) 定数式です。 uchar_expr 文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。 integer_expr 整数値型 (tinyint、smallint、または int) のカラム名、変数、または定数式のいずれかです。
例	<pre>select replicate("abcd", 3) ----- abcdabcdabcd</pre>
使用法	<ul style="list-style-type: none">文字列関数 replicate は、指定した回数か、または 16 KB の記憶領域に保存可能な最大回数のうち、少ない回数で繰り返された式を含む文字列を返します。返される文字列のデータ型は char_expr または uchar_expr と同じデータ型です。char_expr または uchar_expr が NULL のときは、NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが replicate 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 stuff

reserve_identity

説明

`reserve_identity` では、あるプロセスが `identity` 値のブロックを予約し、そのプロセスがこの値を利用できるようにします。

プロセスが `reserve_identity` を呼び出して値のブロックを予約した後、このプロセスが必要とする後続の `identity` 値は、この予約済みプールから導出されます。これらの予約済み数字が使い終わるか、別のテーブルにデータを挿入すると、既存の `identity` オプションが適用されます。`reserve_identity` には `identity` 値の複数のブロックを保持できます。そのため、別のテーブルに挿入されたデータが1つのプロセスによってインタリーブされると、テーブルの予約済みブロック内の次の値が使用されます。

具体的なテーブルの `identity` 値の特定のサイズ・ブロックを予約します。これらの `identity` 値は、呼び出し元プロセスによって排他的に使用されます。予約済みの開始番号が返されます。このプロセスによって、指定されたテーブルに対してその後 `insert` が行われると、これらの値が使用されます。プロセスが終了すると、未使用の値は排除されます。

構文

```
reserve_identity (table_name, number_of_values)
```

パラメータ

table_name

予約するテーブルの名前です。完全修飾名を指定できます。つまり、`database_name`、`owner_name`、`object_name` を含めた名前を引用符で囲んで指定できます。

number_of_values

このプロセスに予約されている `identity` 連続値の数です。この数値には、いずれの予約値も `identity` カラムのデータ型の最大値を超えないような正の数値を指定してください。

例

`reserve_identity` の代表的な使用シナリオについて説明します。ここでは、`table1` に `col1` (データ型は `int`) と `col2` (データ型 `int` の `identity` カラム) が含まれていると想定します。これは `spid 3` のプロセスです。

```
select reserve_identity( table1, 5 )
-----
10
```

`spid 3` および `4` の値を挿入します。

```
Insert table1 values(56) -> spid 3
Insert table1 values(48) -> spid 3
Insert table1 values(96) -> spid 3
Insert table1 values(02) -> spid 4
Insert table1 values(84) -> spid 3
```

テーブル `table1` から選択します。

```
select * from table1

Col1          col2
-----
3             1-> spid 3 reserved 1-5
3             2-> spid 3
3             3-> spid 3
4             6<= spid 4 gets next unreserved value
3             4<= spid 3 continues with reservation
```

`spid 3` が `identity` 値 1 ~ 5 を予約したこと、また `spid 4` が次の未予約の値を受け取っていること、さらに `spid 3` が次の `identity` 値を予約していることが結果セットに表示されます。

使用法

- 戻り値 ***start_value*** は、予約済み `identity` 値のブロックの開始値です。呼び出し元プロセスは、この値を使用して指定されたテーブルに次の挿入を行います。
- `reserve_identity` を使用すると、次の操作ができます。
 - `insert` 文を発行せずに `identity` 値を予約する。
 - `insert` 文を発行する前に予約された値を把握する。
 - 必要に応じて、`identity` 値のさまざまなサイズ・ブロックを「取得」する。
 - 必要な値だけを予約して「余分なギャップ」を制御する(つまり、事前設定されたサーバ取得サイズによってギャップが制限されることがありません)。
- 値は、`insert` 構文を変更しないで自動的に使用されます。
- 次の場合は、`NULL` 値が返されます。
 - 負の値またはゼロがブロック・サイズとして指定された。
 - テーブルが存在しない。
 - テーブルに `identity` カラムが含まれていない。
- このプロセスがこれらの `identity` 値をすでに予約しているテーブルに対して `reserve_identity` を発行すると、この関数は正常に実行され、最新の値グループが使用されます。
- `reserve_identity` を使用してプロキシ・テーブル上で `identity` 値を予約することはできません。ローカル・サーバが `reserve_identity` を呼び出すリモート・プロシージャを呼び出すと、このローカル・サーバは、リモート・テーブルについて `reserve_identity` 関数を使用できます。これらの `reserved` 値はリモート・サーバに保存されますが、その後にリモート・テーブルにデータを挿入するときに、ローカル・サーバのセッションでこの `reserved` 値が使用されます。

- `identity_gap` が予約済みブロック・サイズより小さい場合、値の指定ブロック・サイズ (`identity_gap` サイズではない) を予約することで予約を正常に実行できます。これらの値がプロセスによって使用されない場合、`identity_gap` 設定に関係なく、指定されたブロック・サイズを上限としたギャップが生成される可能性があります。

パーミッション

`identity` 値を予約するには、`insert` パーミッションが必要です。

reserved_pages

説明 データベース、オブジェクト、またはインデックス用に予約されているページ数をレポートします。レポートには、内部構造に使用するページが含まれます。この関数は、バージョン 15.0 より前の Adaptive Server で使用された古い `reserved_pgs` 関数を置き換えるものです。

構文 `reserved_pages(dbid, object_id[, indid[, ptnid]])`

パラメータ

dbid

ターゲット・オブジェクトが存在するデータベースのデータベース ID です。

object_id

テーブルのオブジェクト ID です。

indid

ターゲット・インデックスのインデックス ID です。

ptnid

ターゲット・パーティションのパーティション ID です。

例 **例 1** 指定されたデータベースにある、オブジェクト ID が 31000114 のオブジェクトに予約されているページ数を返します (インデックスも含む)。

```
select reserved_pages(5, 31000114)
```

例 2 クラスタード・インデックスがあるかどうかに関係なく、データ・レイヤに含まれるオブジェクトで予約されているページ数を返します。

```
select reserved_pages(5, 31000114, 0)
```

例 3 クラスタード・インデックスに対応するインデックス・レイヤのオブジェクトで予約されているページ数を返します。データ・レイヤで使用されるページは含まれません。

```
select reserved_pages(5, 31000114, 1)
```

例 4 指定されたパーティション (この例では 2323242432) のデータ・レイヤにあるオブジェクトで予約されているページ数を返します。

```
select reserved_pages(5, 31000114, 0, 2323242432)
```

例 5 以下の 3 とおりの方法のいずれかを使用して、`reserved_pages` でデータベースのスペースを計算します。

- `case` 式を使用して、調べるインデックスに適した値を選択し、このデータベースの `sysindexes` のログ以外のインデックスを選択します。このクエリでは、以下のとおりになります。
 - データに “index 0” の値があり、文 `when sysindexes.indid = 0` または `sysindexes.indid = 1` を含めるときに使用できます。
 - 1 より大きい `indid` 値はインデックスです。このクエリはデータ領域をインデックス・カウントに合計しないため、`indid` が 0 のページ・カウントは含んでいません。

- 各オブジェクトには0か1のインデックス・エントリがあり、両方が含まれていることはありません。
- このクエリはテーブル1つにつきインデックス0を1度だけカウントします。

```
select
'data rsvd' = sum( case
    when indid > 1 then 0
    else reserved_pages(db_id(), id, 0)
end ),
'index rsvd' = sum( case
    when indid = 0 then 0
    else reserved_pages(db_id(), id, indid)
end )
from sysindexes
where id != 8
data rsvd    index rsvd
-----
812          1044
```

- **sysindexes** を複数回クエリして、すべてのクエリが完了した跡で結果を表示します。

```
declare @data int,
@dbsize int,
@dataused int,
@indices int,
@indused int
select @data = sum( reserved_pages(db_id(), id, 0) ),
       @dataused = sum( used_pages(db_id(), id, 0) )
from sysindexes
where id != 8
and indid <= 1
select @indices = sum( reserved_pages(db_id(), id, indid) ),
       @indused = sum( used_pages(db_id(), id, indid) )
from sysindexes
where id != 8 and indid > 0
select @dbsize as 'db size',
       @data as 'data rsvd'
db size    data rsvd
-----
NULL       820
```

- データ領域情報には**sysobjects**、インデックス情報には **sysindexes** をクエリします。**sysobjects** からテーブルのオブジェクト [S]ystem または [U]ser を選択します。

```
declare @data int,
@dbsize int,
@dataused int,
@indices int,
@indused int
```

```

select @data = sum( reserved_pages(db_id(), id, 0) ),
@dataused = sum( used_pages(db_id(), id, 0) )
from sysobjects
where id != 8
and type in ('S', 'U')
select @indices = sum( reserved_pages(db_id(), id, indid) ),
       @indused = sum( used_pages(db_id(), id, indid) )
from sysindexes
where id != 8
and indid > 0
select  @dbsize as 'db size',
        @data as 'data rsvd',
        @dataused as 'data used',
        @indices as 'index rsvd',
        @indused as 'index used'

```

db size	data rsvd	data used	index rsvd	index used
NULL	812	499	1044	381

使用法

- クラストード・インデックスが、すべてのページがロックされたテーブルにある場合、インデックス ID 0 を渡すと予約済みデータ・ページがレポートされ、インデックス ID 1 を渡すと予約済みインデックス・ページがレポートされます。すべてのエラーで常に値 0 が返されます。
- reserved_pages** は指定したアイテムをカウントします。有効なデータベース、オブジェクト、インデックス (データはすべてのテーブルで「インデックス 0」) を指定すると、このデータベース、オブジェクト、またはインデックス用に予約された領域が返されます。ただし、データベース、オブジェクト、またはインデックスを複数回カウントすることもできます。複数のインデックスがあるテーブルのすべてのインデックスについてデータ領域をカウントさせる場合は、各インデックスにつきデータ領域を 1 回カウントさせます。これらの結果を合計すると、オブジェクトの合計データページ数ではなく、インデックス数に合計データ領域を掛けた数が得られます。
- リソースを消費する代わりに、**reserved_pages** はまだキャッシュに存在しないオブジェクトの記述子を廃棄します。
- Adaptive Server バージョン 15.0 以降では、**reserved_pages** 関数が **reserved_pgs** 関数に取って代わります。**reserved_pages** と **reserved_pgs** には違いがあります。
- Adaptive Server バージョン 12.5 以前では、データとインデックスの OAM ページは **sysindexes** に保存されました。Adaptive Server バージョン 15.0 以降では、この情報は **syspartitions** にパーティションごとに保存されます。この情報の保存方法の違いにより、**reserved_pages** と **reserved_pgs** は別のパラメータを必要とし、結果セットも異なります。

- `reserved_pgs` にはページ ID が必要でした。一致する `sysindexes` ローがない値を指定する場合、入力する ID は 0 でした (たとえば、ノンクラスタード・インデックス・ローのデータ OAM ページ)。0 は有効な OAM ページではないため、0 のページ ID を入力した場合、`reserved_pgs` は 0 を返しました。入力値が無効なため、`reserved_pgs` は何もカウントできませんでした。

一方、`reserved_pages` にはインデックス ID が必要で、0 は有効なインデックス ID です (たとえば、すべてのテーブルでデータが “index 0”)。`reserved_pages` は、`indid 0` または 1 を除いてインデックス・ローのデータ領域を再カウントする必要がないことをコンテキストから判別できないため、インデックス ID として 0 を渡すたびにデータ領域を数えます。`reserved_pages` はこのデータ領域を各ローにつき 1 度カウントするため、真の値の何倍もの合計を出します。

これらの違いを以下に説明します。

- OAM ページ入力のページ ID の値として 0 を指定した場合、`reserved_pgs` は合計に影響せず、0 の値を返すだけです。
- インデックス ID を 0 の値にして `reserved_pages` を指定した場合は、データ領域をカウントします。データをカウントしたい場合にのみ `reserved_pages` を発行してください。そうしないと、合計に影響します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `reserved_pgs` 関数を実行できます。

参照

コマンド [update statistics](#)

関数 [data_pages](#), [reserved_pages](#), [row_count](#), [used_pages](#)

return_lob

説明	ロケータの参照を解除し、そのロケータによって参照されていた LOB を返します。
構文	<code>return_lob (datatype, locator_descriptor)</code>
パラメータ	<i>datatype</i> LOB のデータ型です。有効なデータ型は次のとおりです。 <ul style="list-style-type: none">• text• unitext• image <i>locator_descriptor</i> LOB ロケータの有効な表現です。ロケータのホスト変数、ローカル変数、またはリテラル・バイナリ値です。
例	この例では、ロケータの参照を解除しリテラル・ロケータ値 <code>0x9067ef450100000000100000004010040080000000</code> によって参照される LOB を返します。 ロケータ値 <code>0x9067ef450100000000100000004010040080000000</code> を検証します。
使用法	<code>return_lob</code> 関数は <code>set send_locator on</code> コマンドを無効にし、常に LOB を返します。
パーミッション	すべてのユーザが <code>return_lob</code> 関数を実行できます。
参照	コマンド <code>deallocate locator</code> 、 <code>truncate lob</code> Transact-SQL 関数 <code>locator_literal</code> 、 <code>locator_valid</code> 、 <code>create_locator</code>

reverse

説明	指定した文字列の文字順を逆にして返します。
構文	<code>reverse(expression uchar_expr)</code>
パラメータ	expression 文字型またはバイナリ型のカラム名、変数、または (<code>char</code> 、 <code>varchar</code> 、 <code>nchar</code> 、 <code>nvarchar</code> 、 <code>binary</code> 、または <code>varbinary</code> 型の) 定数式です。 uchar_expr 文字型のカラム名、変数、または (<code>unichar</code> または <code>univarchar</code> 型の) 定数式です。
例	例 1 <pre>select reverse("abcd") ----- dcba</pre> 例 2 <pre>select reverse(0x12345000) ----- 0x00503412</pre>
使用法	<ul style="list-style-type: none">文字列関数 <code>reverse</code> は、expression の文字順を逆にした文字列を返します。expression が <code>NULL</code> の場合、<code>reverse</code> は <code>NULL</code> を返します。サロゲート・ペアは分割不可能なデータとして処理されるため、文字順が逆にされることはありません。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>reverse</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 lower 、 upper

right

説明 文字式またはバイナリ式の一部を、指定した数値を始点としてその右側を返します。文字式と同じデータ型の値を返します。

構文 `right(expression, integer_expr)`

パラメータ *expression*

文字型またはバイナリ型のカラム名、変数、または (char、varchar、nchar、unicar、nvarchar、univarchar、binary、または varbinary 型の) 定数式です。

integer_expr

整数値型 (tinyint、smallint、または int) のカラム名、変数、または定数式のいずれかです。

例

例 1

```
select right("abcde", 3)
--
cde
```

例 2

```
select right("abcde", 2)
--
de
```

例 3

```
select right("abcde", 6)
----
abcde
```

例 4

```
select right(0x12345000, 3)
-----
0x345000
```

例 5

```
select right(0x12345000, 2)
-----
0x5000
```

例 6

```
select right(0x12345000, 6)
-----
0x12345000
```

使用法	<ul style="list-style-type: none">文字列関数 right は、文字式やバイナリ式の右端から、指定した文字数の部分を返します。指定した右端部分が、サロゲート・ペアの 2 番目のサロゲート (下位サロゲート) の場合、次の完全文字から始まる文字列値が返されます。したがって、指定の文字数より 1 文字少ない文字数の文字列が返されます。戻り値のデータ型は、文字式やバイナリ式と同じデータ型になります。expression が NULL のとき、right は NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが right 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 rtrim , substring

rm_appcontext

説明	特定のアプリケーション・コンテキストまたはすべてのアプリケーション・コンテキストを削除します。rm_appcontext は Application Context Facility (ACF) が提供する関数です。
構文	rm_appcontext("context_name", "attribute_name")
パラメータ	<p>context_name アプリケーション・コンテキスト名を指定するローです。データ型 char(30) として格納されます。</p> <p>attribute_name アプリケーション・コンテキスト属性名を指定するローです。データ型 char(30) として格納されます。</p>
例	<p>例 1 一部またはすべての属性を指定して、アプリケーション・コンテキストを削除します。</p> <pre> select rm_appcontext("CONTEXT1", "*") ----- 0 select rm_appcontext(" ", "*") ----- 0 select rm_appcontext("NON_EXISTING_CTX", "ATTR") ----- -1 </pre> <p>例 2 適切なパーミッションを持たないユーザがアプリケーション・コンテキストを削除しようとしたときの結果を示します。</p> <pre> select rm_appcontext("CONTEXT1", "ATTR2") ----- -1 </pre>
使用法	<ul style="list-style-type: none"> この関数は、成功すると常に 0 を返します。 この関数では、すべての引数が必須です。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	パーミッションは、ACF に格納されているユーザ・プロファイルとアプリケーション・プロファイルに依存します。
参照	ACF の詳細については、『システム管理ガイド』の「第 11 章 ユーザ・パーミッションの管理」の「ロー・レベル・アクセス制御」を参照してください。
	関数 get_appcontext , list_appcontext , set_appcontext

role_contain

説明	指定した役割が別の指定した役割に含まれているかどうかを確認します。
構文	<code>role_contain("role1", "role2")</code>
パラメータ	role1 システム定義またはユーザ定義の役割の名前です。 role2 もう1つのシステム定義またはユーザ定義の役割の名前です。
例	例 1 <pre>select role_contain("intern_role", "doctor_role") ----- 1</pre> 例 2 <pre>select role_contain("specialist_role", "intern_role") ----- 0</pre>
使用法	システム関数 <code>role_contain</code> は、 role1 が role2 に含まれている場合には 1 を返します。それ以外の場合は、 <code>role_contain</code> は 0 を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>role_contain</code> 関数を実行できます。
参照	マニュアル 含まれている役割と役割階層の詳細については、『システム管理ガイド』を参照してください。システム関数については、『Transact-SQL ユーザーズ・ガイド』を参照してください。 関数 mut_excl_roles , proc_role , role_id , role_name コマンド alter role システム・プロシージャ sp_activeroles , sp_displayroles , sp_role

role_id

説明	指定した役割名の役割 ID を返す。
構文	<code>role_id("role_name")</code>
パラメータ	<p>role_name</p> <p>システム定義またはユーザ定義の役割の名前です。役割名と役割 ID は、<code>sysssrvroles</code> システム・テーブルに格納されています。</p>
例	<p>例 1 <code>sa_role</code> のシステム役割 ID を返します。</p> <pre>select role_id("sa_role") ----- 0</pre> <p>例 2 “<code>intern_role</code>” のシステム役割 ID を返します。</p> <pre>select role_id("intern_role") ----- 6</pre>
使用法	<ul style="list-style-type: none"> システム関数 <code>role_id</code> は、システム役割 ID (<code>srid</code>) を返します。システム役割 ID は、<code>sysssrvroles</code> システム・テーブルの <code>srid</code> カラムに格納されています。 <code>role_name</code> がシステムの有効な役割でない場合、Adaptive Server は NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>role_id</code> を実行できます。
参照	<p>マニュアル 詳細</p> <ul style="list-style-type: none"> 役割 – 『システム管理ガイド』を参照してください。 システム関数 – 『Transact-SQL ユーザーズ・ガイド』を参照してください。 <p>関数 <code>mut_excl_roles</code>, <code>proc_role</code>, <code>role_contain</code>, <code>role_name</code></p>

role_name

説明	指定した役割 ID の役割名を返す。
構文	<code>role_name(role_id)</code>
パラメータ	role_id 役割のシステム役割 ID (srid) です。役割名は <code>sysssvroles</code> に格納されています。
例	<pre>select role_name(01) ----- sso_role</pre>
使用法	システム関数 <code>role_name</code> は役割名を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>role_name</code> を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 mut_excl_roles , proc_role , role_contain , role_id

round

説明	指定した数値を、指定した小数桁数に丸めた値を返します。
構文	<code>round(number, decimal_places)</code>
パラメータ	<p>number 真数値 (numeric、dec、decimal、tinyint、smallint、int、または bigint 型)、概数値 (float、real、または double precision 型)、money 型のカラム、変数、定数式、またはこれらの組み合わせのいずれかです。</p> <p>decimal_places 丸めたあとの小数桁数値です。</p>
例	<p>例 1</p> <pre>select round(123.4545, 2) ----- 123.4500</pre> <p>例 2</p> <pre>select round(123.45, -2) ----- 100.00</pre> <p>例 3</p> <pre>select round(1.2345E2, 2) ----- 123.450000</pre> <p>例 4</p> <pre>select round(1.2345E2, -2) ----- 100.000000</pre>
使用法	<ul style="list-style-type: none"> • 算術関数 round は <i>number</i> を有効小数桁数 <i>decimal_places</i> まで丸めます。 • <i>decimal_places</i> が正の数値の場合は有効小数桁数を示し、<i>decimal_places</i> が負の数値の場合は有効整数桁数を示します。 • 結果の型は <i>number</i> と同じ型になります。numeric 型または decimal 型の式では、結果の内部精度は第 1 引数に 1 を加算した精度になり、位取りは <i>number</i> と同じになります。 • round 関数は常に値を 1 つだけ返します。<i>decimal_places</i> が負の数値であり、<i>number</i> に指定された有効桁数を超えている場合、Adaptive Server は 0 を返します (小数点以下の 0 の数が numeric の位取りと同じ場合は、この値は 0.00 の形式で表されます)。たとえば、次の式は、0.00 の値を返します。 <pre>select round(55.55, -3)</pre>

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが round 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 abs , ceiling , floor , sign , str

row_count

説明	指定されたテーブルの推定ロー数を返す。
構文	<code>row_count(<i>dbid</i>, <i>object_id</i> [, <i>ptnid</i>] [, "option"])</code>
パラメータ	<i>dbid</i> ターゲット・オブジェクトが存在するデータベースの ID です。 <i>object_id</i> テーブルのオブジェクト ID です。 <i>ptnid</i> 対象のパーティション ID です。
例	例 1 指定したテーブル内のローの予測数を返します。 <pre>select row_count(5, 31000114)</pre> 例 2 オブジェクト ID 31000114 のオブジェクトにある指定のパーティション (パーティション ID 2323242432) に存在するローの予測数を返します。 <pre>select row_count(5, 31000114, 2323242432)</pre>
使用法	<ul style="list-style-type: none">• すべてのエラーで常に値 0 が返されます。• リソースを消費する代わりに、<code>row_count</code> はまだキャッシュに存在しないオブジェクトの記述子を廃棄します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>row_count</code> を実行できます。
参照	関数 reserved_pages , used_pages

rtrim

説明	指定した式から後続空白を削除します。
構文	<code>rtrim(char_expr uchar_expr)</code>
パラメータ	char_expr 文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。 uchar_expr 文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。
例	<pre>select rtrim("abcd ") ----- abcd</pre>
使用法	<ul style="list-style-type: none">文字列関数 <code>rtrim</code> は、後続空白を削除します。Unicode では、空白は Unicode 値 U+0020 として定義されています。<code>char_expr</code> または <code>uchar_expr</code> が NULL の場合は、NULL を返します。現在の文字セットのスペース文字に相当する値だけを削除します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>rtrim</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 ltrim

sdc_intempdbconfig

説明	クラスタ環境のみ – システムが現在、テンポラリ・データベース設定モードにある場合は 1 を返し、それ以外の場合は 0 を返します。
構文	<code>sdc_intempdbconfig()</code>
例	<code>select sdc_intempdbconfig()</code>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>sdc_intempdbconfig</code> を実行できます。

set_appcontext

説明 ユーザ・セッション用のアプリケーション・コンテキスト名、属性名、属性値を設定します。それらは、指定したアプリケーションの属性によって定義されます。`set_appcontext` は、Application Context Facility (ACF) から提供される関数です。

構文 `set_appcontext("context_name", "attribute_name", "attribute_value")`

パラメータ **context_name**
アプリケーション・コンテキスト名を指定するローです。データ型 `char(30)` として格納されます。

attribute_name
アプリケーション・コンテキストの属性名を指定するローです。データ型 `char(30)` として格納されます。

attribute_value
アプリケーションの属性値を指定するローです。データ型 `char(30)` として格納されます。

例 **例 1** `CONTEXT1` という名前のアプリケーション・コンテキストを作成し、それに属性 `ATTR1` とその値 `VALUE1` を設定します。

```
select set_appcontext("CONTEXT1", "ATTR1", "VALUE1")
```

```
-----  
0
```

既存のアプリケーション・コンテキストを上書きしようとする、次のようになります。

```
select set_appcontext("CONTEXT1", "ATTR1", "VALUE1")
```

```
-----  
-1
```

例 2 値のデータ型変換を含む `set_appcontext` を示します。

```
declare @numericvarchar varchar(25)  
select @numericvar = "20"  
select set_appcontext ("CONTEXT1", "ATTR2",  
convert(char(20), @numericvar))
```

```
-----  
0
```

例 3 適切なパーミッションを持たないユーザがアプリケーション・コンテキストを設定しようとしたときの結果を示します。

```
select set_appcontext("CONTEXT1", "ATTR2", "VALUE1")
```

```
-----  
-1
```

使用法	<ul style="list-style-type: none">• <code>set_appcontext</code> は、成功すると 0 を返し、失敗すると -1 を返します。• 現在のセッションにすでに存在する値を設定すると、<code>set_appcontext</code> は -1 を返します。• この関数では、既存のアプリケーション・コンテキストの値は上書きできません。コンテキストに新しい値を割り当てるには、コンテキストを削除してから、新しい値を使用してコンテキストを再作成してください。• <code>set_appcontext</code> は、属性を <code>char</code> データ型として格納します。属性値を他のデータ型と比較する必要があるアクセス・ルールを作成する場合は、アクセス・ルールで <code>char</code> データを適切なデータ型に変換します。• この関数では、すべての引数が必須です。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	パーミッションは、ACF に格納されているユーザ・プロファイルとアプリケーション・プロファイルに依存します。
参照	マニュアル ACF の詳細については、『システム管理ガイド』の「第 11 章 ユーザ・パーミッションの管理」の「ロー・レベル・アクセス制御」を参照してください。 関数 get_appcontext , list_appcontext , rm_appcontext

setdata

説明	一部またはすべてのラージ・オブジェクト (LOB) を上書きします。
構文	<code>setdata(locator_name, offset_value, new_value)</code>
パラメータ	locator_name 修正する LOB 値を参照するロケータです。 offset_value <i>locator_name</i> が示す LOB 内の位置です。Adaptive Server はこの位置から <i>new_value</i> の内容を記述します。 offset_value の値は text_locator と unitext_locator では文字単位で image_locator ではバイト単位です。LOB の最初の文字またはバイトの offset_value 値は 1 です。 new_value 古いデータを上書きするデータです。
例	この例の最後の <code>select</code> 文では元の文字列 “Sybase “ASE/IQ/ASA” の代わりに “Sybase ABC/IQ/ASA” を返します。 <pre>declare @v text_locator select @v = create_locator (text_locator, convert(text, "Sybase ASE/IQ/ASA") select setdata(@v, 8, "ABC") select return_lob(text, @v)</pre>
使用法	<ul style="list-style-type: none">• setdata は現在の LOB 値を修正します。つまり、Adaptive Server では修正されるまで LOB をコピーしません。• <i>new_value</i> の長さが offset_value をスキップした後の LOB の残りの長さを超えた場合、Adaptive Server では LOB を拡張して <i>new_value</i> の全体の長さを保持します。• <i>new_value</i> と offset_value の合計が LOB の長さに満たない場合、Adaptive Server では LOB の末尾のデータを変更またはトランケートしません。• offset_value の方が更新する LOB 値よりも長い場合、setdata は NULL を返します。
パーミッション	すべてのユーザが setdata 関数を実行できます。
参照	コマンド <code>deallocate locator</code> 、 <code>truncate lob</code> Transact-SQL 関数 <code>locator_valid</code> 、 <code>return_lob</code> 、 <code>create_locator</code>

show_cached_plan_in_xml

説明	<p>ステートメント・キャッシュ内のクエリの実行クエリ・プランを XML で表示します。</p> <p><code>show_cached_plan_in_xml</code> は <code>showplan</code> のセクションを XML フォーマットで返します。</p>
構文	<pre>show_cached_plan_in_xml(statement_id, plan_id, [level_of_detail])</pre>
パラメータ	<p>statement_id ライトウェイト・プロシージャのオブジェクト ID です (Adaptive Server の内部で作成して呼び出すことができるプロシージャ)。これは <code>monCachedStatement</code> の SSQLID です。</p> <p>plan_id プランのユニークな識別子です。これは <code>monCachedProcedures</code> の PlanID です。<code>plan_id</code> のゼロの値は示された SSQLID についてキャッシュされたすべてのプランの <code>showplan</code> 出力を表示します。</p> <p>level_of_detail <code>show_cached_plan_in_xml</code> が返す詳細度を示す 0～6 の値です (表 2-6 参照)。<code>level_of_detail</code> は <code>show_cached_plan_in_xml</code> によって返される <code>showplan</code> のセクションを決定します。デフォルト値は 0 です。</p> <p><code>show_cached_plan_in_xml</code> の出力には <code>plan_id</code> と次のセクションが含まれています。</p> <ul style="list-style-type: none"> • <code>parameter</code> – パフォーマンスを低下させるクエリとパラメータ値のコンパイルに使用するパラメータ値が含まれます。コンパイル・パラメータは <code><compileParameters></code> タグと <code></compileParameters></code> タグで示されます。最も遅いパラメータ値は <code><execParameters></code> タグと <code></execParameters></code> タグで示されます。パラメータごとに、<code>show_cached_plan_in_xml</code> では次の項目が表示されます。 <ul style="list-style-type: none"> • 番号 • データ型 • Value – 500 バイトより大きな値と <code>insert-value</code> 文が表示されない値です。すべてのパラメータの値を格納するために使用する合計メモリは 2 つのパラメータ・セットあたり 2KB です。 • <code>opTree</code> – クエリ・プランとオブティマイザによる見積もりが含まれています。<code>opTree</code> セクションは <code><opTree></code> タグと <code></opTree></code> タグで示されます。 • <code>execTree</code> – クエリプランと <code>lava</code> 演算子の詳細が含まれています。<code>execTree</code> セクションは <code><execTree></code> タグと <code></execTree></code> タグで示されます。

例

XML で表示されたクエリ・プランの例を次に示します。

```

select show_cache_plan_in_xml(1328134997,0)
go
-----

<?xml version="1.0" encoding="UTF-8"?>
<query>
  <statementId>1328134997</statementId>
<text>
  <![CDATA[SQL Text: select name from sysobjects where id =
10]]>
</text>
<plan>
  <planId>11</planId>
  <planStatus> available </planStatus>
  <execCount>1371</execCount>
  <maxTime>3</maxTime>
  <avgTime>0</avgTime>
  <compileParameters/>
  <execParameters/>
  <opTree>
    <Emit>
      <VA>1</VA>
      <est>
        <rowCnt>10</rowCnt>
        <lio>0</lio>
        <pio>0</pio>
        <rowSz>22.54878</rowSz>
      </est>
      <act>
        <rowCnt>1</rowCnt>
      </act>
      <arity>1</arity>
      <IndexScan>
        <VA>0</VA>
        <est>
          <rowCnt>10</rowCnt>
          <lio>0</lio>
          <pio>0</pio>
          <rowSz>22.54878</rowSz>
        </est>
        <act>
          <rowCnt>1</rowCnt>
          <lio>3</lio>
          <pio>0</pio>
        </act>
      <varNo>0</varNo>
      <objName>sysobjects</objName>
      <scanType>IndexScan</scanType>
      <indName>csysobjects</indName>
      <indId>3</indId>
    </Emit>
  </opTree>
</plan>

```

```

        <scanOrder> ForwardScan </scanOrder>
        <positioning> ByKey </positioning>
        <perKey>
            <keyCol>id</keyCol>
            <keyOrder> Ascending </keyOrder>
        </perKey>
        <indexIOSizeInKB>2</indexIOSizeInKB>
        <indexBufReplStrategy> LRU
    </indexBufReplStrategy>
        <dataIOSizeInKB>2</dataIOSizeInKB>
        <dataBufReplStrategy> LRU
    </dataBufReplStrategy>
        </IndexScan>
    </Emit>
</opTree>
</plan>

```

使用法

- `show_cached_plan_in_xml` を使用する前に、ステートメント・キャッシュを有効にします。
- `show_cached_plan_in_xml` はキャッシュされた文にのみ使用します。
- プランは使用中は印刷されません。ステータスが `available` のプランは詳細が印刷されます。ステータスが `in use` のプランはプロセス ID のみが表示されます。
- 表 2-6 は、`level_of_detail` 値を示す `show_cached_plan_in_xml` セクションを表示します。

表 2-6: `level_of_detail` 値を表示するセクション

<code>level_of_detail</code>	<code>parameter</code>	<code>opTree</code>	<code>execTree</code>
0 (デフォルト)	X	X	
1	X		
2		X	
3			X
4		X	X
5	X		X
6	X	X	X

show_dynamic_params_in_xml

説明 動的 SQL クエリのパラメータ情報 (準備文) を XML フォーマットで返します。

構文 show_dynamic_params_in_xml(*object_id*)

パラメータ *object_id*

調査対象となる動的 SQL またはライトウェイト・ストアド・プロシージャの ID。通常は @@plwpid グローバル変数の戻り値です。

例 この例では、最初にオブジェクト ID を検索します。

```
select @@plwpid
-----
707749902
```

show_dynamic_params_in_xml の入力パラメータとして ID を使用します。

```
select show_dynamic_params_in_xml(707749902)

<?xml version="1.0" encoding="UTF-8"?>
<query>
  <parameter>
    <number>1</number>
    <type>INT</type>
    <column>tab.coll</column>
  </parameter>
</query>
```

パラメータ	値	定義
number	1	動的パラメータは文の冒頭にあります。
type	INT	テーブルは int データ型を使用します。
column	tab.coll	クエリは tab テーブルの col1 カラムを使用します。

使用法

- show_dynamic_params_in_xml では where 句、update の set 句、および insert の values リストで動的パラメータを使用できます。
- where 句の場合、show_dynamic_params_in_xml はカラムのある式、関係演算子、パラメータのある式を含んだ最小のサブツリーに従って関連付けを決定します。たとえば、次のようにします。

```
select * from tab where coll + 1 = ?
```

クエリにサブツリーがない場合、show_dynamic_params_in_xml は <column> 要素を省略します。たとえば、次のようにします。

```
select * from tab where ? < 1000
```

- show_dynamic_params_in_xml は複数のカラムが関与する式に検出する最初のカラムを選択します。

```
delete tab where coll + col2 > ?
```

- 関連付けは update ... set 文では明白です。たとえば、次のようにします。

```
update tab set coll = ?
```


show_role

説明	ログインの現在有効なシステム定義役割を表示します。
構文	<code>show_role()</code>
パラメータ	なし。
例	<p>例 1</p> <pre>select show_role() sa_role sso_role oper_role replication_role</pre> <p>例 2</p> <pre>if charindex("sa_role", show_role()) >0 begin print "You have sa_role" end</pre>
使用法	<ul style="list-style-type: none">システム関数 <code>show_role</code> は、ログインの現在有効なシステム定義の役割 (<code>sa_role</code>、<code>sso_role</code>、<code>oper_role</code>、または <code>replication_role</code>) があればそれを返します。ログインに役割がない場合は、<code>show_role</code> は NULL を返します。データベース所有者が <code>setuser</code> を実行したあとで <code>show_role</code> 関数を呼び出すと、<code>show_role</code> 関数は <code>setuser</code> で指定したユーザではなく、データベース所有者の有効な役割を表示します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザがシステム関数 <code>show_role</code> を実行できます。
参照	コマンド alter role 、 create role 、 drop role 、 grant 、 set 、 revoke マニュアル Transact-SQL ユーザーズ・ガイド 関数 proc_role 、 role_contain システム・プロシージャ sp_activeroles 、 sp_displayroles 、 sp_role

show_sec_services

説明	セッションに有効なセキュリティ・サービスをリストします。
構文	show_sec_services()
パラメータ	なし。
例	ユーザの現在のセッションがデータの暗号化とリプレイの検出のチェックを実行していることを示します。 <pre>select show_sec_services() encryption, replay_detection</pre>
使用法	<ul style="list-style-type: none">セッション中の有効なセキュリティ・サービスをリストするには、show_sec_services 関数を使います。有効なセキュリティ・サービスがないときは、show_sec_services 関数は NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが show_sec_services 関数を実行できます。
参照	関数 is_sec_service_on

sign

説明	指定した値の符号 (正の 1、0、または負の -1) を返します。
構文	<code>sign(<i>numeric</i>)</code>
パラメータ	<i>numeric</i> 真数値 (<code>numeric</code> 、 <code>dec</code> 、 <code>decimal</code> 、 <code>tinyint</code> 、 <code>smallint</code> 、 <code>int</code> 、または <code>bigint</code> 型)、概数値 (<code>float</code> 、 <code>real</code> 、または <code>double precision</code> 型)、 <code>money</code> 型のカラム、変数、定数式、またはこれらの組み合わせのいずれかです。
例	<p>例 1</p> <pre>select sign(-123) ----- -1</pre> <p>例 2</p> <pre>select sign(0) ----- 0</pre> <p>例 3</p> <pre>select sign(123) ----- 1</pre>
使用法	<ul style="list-style-type: none">算術関数 <code>sign</code> は、正 (1)、ゼロ (0)、または負 (-1) を返します。結果は、引数の数値式と同じ型、精度、位取りになります。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>sign</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 abs 、 ceiling 、 floor 、 round

sin

説明	指定した角度 (ラジアン) の正弦を返します。
構文	<code>sin(<i>approx_numeric</i>)</code>
パラメータ	<i>approx_numeric</i> 概数値型 (<code>float</code> 、 <code>real</code> 、または <code>double precision</code>) のカラム名、変数、または定数式のいずれかです。
例	<pre>select sin(45) ----- 0.850904</pre>
使用法	算術関数 <code>sin</code> は、指定した角度 (ラジアン) の正弦を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>sin</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 cos 、 degrees 、 radians

sortkey

説明 照合動作をもとに、結果を並べ替えるために使用する値を生成します。これにより、デフォルト・セットのラテン文字以外の文字セットについても、辞書順、大文字と小文字の区別、アクセントの区別をもとにした文字列照合動作が可能になります。

構文 `sortkey(char_expression | uchar_expression)[, {collation_name | collation_ID}]`

パラメータ **char_expression**
文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expression
文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

collation_name
引用符で囲まれた文字列、または使用する照合動作を指定する文字変数です。[表 2-8 \(250 ページ\)](#) に、有効値を示します。

collation_ID
整数の定数、または使用する照合動作を指定する変数です。[表 2-8 \(250 ページ\)](#) に有効値を示します。

例 **例 1** 西欧言語辞書順のソートを示します。

```
select * from cust_table where cust_name like "TI%" order by
(sortkey(cust_name, "dict"))
```

例 2 中国語 (簡体字) の読み方順のソートを示します。

```
select * from cust_table where cust_name like "TI%" order by
(sortkey(cust_name, "gbpinyin"))
```

例 3 インライン・オプションを使用した西欧言語辞書順のソートを示します。

```
select * from cust_table where cust_name like "TI%" order by cust_french_sort
```

例 4 既存のキーを使用した中国語 (簡体字) の読み方順のソートを示します。

```
select * from cust_table where cust_name like "TI%" order by
cust_chinese_sort.
```

使用法

- システム関数 `sortkey` は、照合動作をもとに、結果を並べ替えるために使用する値を生成します。これにより、デフォルト・セットのラテン文字以外の文字セットについても、辞書順、大文字と小文字の区別、アクセントの区別をもとにした文字列照合動作が可能になります。戻り値は `varbinary` データ型値で、`sortkey` 関数から返される入力文字列に対応するコード化照合情報が含まれています。

たとえば、`sortkey` によって返された値を、ソース文字列を含むカラムに格納できます。文字列データを希望の順序で検索する場合は、`select` 文に、`sortkey` の実行結果を含むカラムに関する `order by` 句を含めます。

sortkey により、特定の照合基準のセットに対して返される値が、varbinary データ型について実行されるバイナリ比較に使用できるようになります。

- sortkey は、それぞれの入力文字について 6 バイトの照合情報を生成します。したがって、sortkey の実行結果は、varbinary データ型の長さの制限値である 255 バイトを超える可能性があります。この制限値を超える場合、結果は制限値に収まるようにトランケートされます。この制限はサーバの論理ページ・サイズに依存するので、DOL テーブルおよび APL テーブルでは、結果の文字列が次のサイズより小さくなるまで、各入力文字の結果バイトはトランケーションによって削除されます。

表 2-7: ローとカラムの最大長 – APL テーブルおよび DOL テーブル

ロック・スキーム	ページ・サイズ	ローの最大長	カラムの最大長
APL テーブル	2K (2048 バイト)	1962	1960 バイト
	4K (4096 バイト)	4010	4008 バイト
	8K (8192 バイト)	8106	8104 バイト
	16K (16384 バイト)	16298	16296 バイト
DOL テーブル	2K (2048 バイト)	1964	1958 バイト
	4K (4096 バイト)	4012	4006 バイト
	8K (8192 バイト)	8108	8102 バイト
	16K (16384 バイト)	16300	16294 バイト テーブルに変長カラムがない場合
	16K (16384 バイト)	16300 (varlen = 8191 の最大開始 オフセットによって変化)	8191-6-2 = 8183 バイト 1 つ以上の可変長カラムが テーブルにある場合*

* このサイズには、ローのオーバヘッドの 6 バイトとローの長さのフィールドの 2 バイトが含まれる。

このような状態が発生した場合は、Adaptive Server によって警告メッセージが表示されますが、sortkey 関数を含むクエリまたはトランザクションは引き続き実行されます。

- *char_expression* または *uchar_expression* は、サーバのデフォルトの文字セットでコード化されている文字の組み合わせでなければなりません。
- *char_expression* または *uchar_expression* は、空文字列でもかまいません。空の文字列の場合、sortkey は長さがゼロの varbinary 値を返し、空の文字列にブランクを格納します。

空文字列は、データベース・カラムの NULL 文字列とは異なる照合値を持ちます。

- *char_expression* または *uchar_expression* が NULL の場合、sortkey は null を返します。
- unicode 式にソート順序が指定されていない場合は、binary ソート順序が使用されます。

- `collation_name` または `collation_ID` に値を指定しない場合、`sortkey` はバイナリ照合とみなします。
- `sortkey` 関数によって生成されるバイナリ値は、Adaptive Server のメジャー・バージョン間で異なる可能性があります (たとえば、バージョン 12.0 と 12.5、バージョン 11.9.2 と 12.0 など)。現在のバージョンの Adaptive Server にアップグレードする場合は、キーを再生成して隠しカラムを再配置してから、バイナリ比較を実行する必要があります。

注意 バージョン 12.5 から 12.5.0.1 へのアップグレードでは、この手順は不要で、キーを再生成しなくてもエラー・メッセージや警告メッセージは表示されません。隠しカラムを持つクエリは正常に実行されますが、その比較結果はアップグレード前のサーバとは異なる可能性があります。

照合テーブル

マルチ言語ソートを実行するときには、次の 2 種類の照合テーブルを使用できます。

- 1 `sortkey` 関数で作成された「組み込み」照合テーブル。この関数は Adaptive Server 11.5.1 以降のバージョンに組み込まれています。照合名または照合 ID を使用して、組み込みテーブルを指定できます。
- 2 Unilib ライブラリ・ソート機能を使用する外部照合テーブル。外部テーブルを指定するときには、照合名を使用します。これらのファイルは、`$$SYBASE/collate/unicode` に格納されています。

どちらのテーブルを使用しても適切に機能します。ただし、「組み込み」テーブルは Adaptive Server データベースに関連付けられていますが、外部テーブルは関連付けられていません。Adaptive Server データベースを使用する場合は、組み込みテーブルを使用するとパフォーマンスを最大限に引き出すことができます。どちらのテーブルでも、英語、西欧言語、アジア言語を組み合わせて使用できます。

`sortkey` は、次の 2 とおりの方法で使用できます。

- 1 インライン `order by` 句に `sortkey` を指定する方法。変更内容を最小限に抑えて既存のアプリケーションを更新する場合に役立ちます。ただし、この方法では実行中にソート・キーが生成されるため、レコード数が 1000 を超える大きなデータセットでは、最適なパフォーマンスを実現できません。

- 2 既存のキー – 新規顧客名などの新しいレコードで、テーブルに多言語ソート機能を追加する必要がある場合に **sortkey** を呼び出す方法。データベース内に隠しカラム (**binary** または **varbinary** 型) を設定する必要があります。この場合、使用するソート順 (フランス語、中国語の順など) ごとに 1 つのカラムを、同一テーブル内に設定することをすすめます。クエリの出力をソートする必要がある場合は、**order by 句** に隠しカラムを 1 つ指定します。この方法では、パフォーマンスを最大限に引き出すことができます。これは、キーがすでに生成、ソートされており、バイナリ値に基づいて比較を迅速に実行できるためです。

使用可能な照合ルールの一覧を表示できます。この一覧を出力するには、**sp_helpsort** を実行するか、または **syscharsets** の **name**、**id**、**description** を問い合わせで選択します (**type** は 2003 ~ 2999 の間です)。

- **表 2-8** は、**collation_name** と **collation_ID** の有効値の一覧を示します。

表 2-8: 照合名と ID

説明	照合名	照合 ID
デフォルト Unicode マルチ言語	default	20
タイ語辞書順	thaidict	21
ISO14651 規格	iso14651	22
UTF-16 順 – UTF-8 バイナリ順と一致	utf8bin	24
CP 850 代替言語 – アクセント記号なし	altnoacc	39
CP 850 代替言語、小文字優先	altdict	45
CP 850 西欧言語 – 大文字/小文字の優先設定なし	altnocsp	46
CP 850 スカンジナビア語 – 辞書順	scandict	47
CP 850 スカンジナビア語辞書順 – 大文字と小文字を区別しない、優先度を付けた順位	scannocp	48
GB ピンイン	gbpinyin	該当なし
バイナリ・ソート	binary	50
Latin-1 英語、フランス語、ドイツ語辞書	dict	51
Latin-1 英語、フランス語、ドイツ語、大文字/小文字の区別なし	nocase	52
Latin-1 英語、フランス語、ドイツ語、大文字/小文字の優先設定なし	nocasep	53
Latin-1 英語、フランス語、ドイツ語、アクセント記号なし	noaccent	54
Latin-1 スペイン語辞書	espdict	55
Latin-1 スペイン語、大文字/小文字の区別なし	espnoacs	56
Latin-1 スペイン語、アクセント記号なし	espnoac	57
ISO 8859-5 ロシア語辞書	rusdict	58
ISO 8859-5 ロシア語、大文字小文字の区別なし	rusnoacs	59
ISO 8859-5 キリル語辞書	cyrdict	63
ISO 8859-5 キリル語、大文字小文字の区別なし	cyrnoacs	64
ISO 8859-7 ギリシア語辞書	elldict	65
ISO 8859-2 ハンガリー語、辞書	hundiect	69

説明	照合名	照合 ID
ISO 8859-2 ハンガリー語、アクセント記号の区別なし	hunnoac	70
ISO 8859-2 ハンガリー語、大文字小文字の区別なし	hunnocs	71
ISO 8859-9 トルコ語辞書	turdict	72
ISO 8859-9 トルコ語、アクセント記号の区別なし	turknoac	73
ISO 8859-9 トルコ語、大文字小文字の区別なし	turknocs	74
CP932 バイナリ順	cp932bin	129
中国語、読み方順	dynix	130
GB2312 バイナリ順	gb2312bn	137
共通キリル語辞書	cyrdict	140
トルコ語辞書	turdict	155
EUCKSC バイナリ順	euckscbn	161
中国語、読み方順	gbpinyin	163
ロシア語辞書順	rusdict	165
SJIS バイナリ順	sjisbin	179
EUCJIS バイナリ順	eucljlsbn	192
BIG5 バイナリ順	big5bin	194
Shift-JIS バイナリ順	sjisbin	259

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザーが `sortkey` 関数を実行できます。

参照

関数 [compare](#)

soundex

説明	式の発音を表す 4 文字コードを返します。
構文	<code>soundex(char_expr uchar_expr)</code>
パラメータ	char_expr 文字型のカラム名、変数、または (<code>char</code> 、 <code>varchar</code> 、 <code>nchar</code> 、または <code>nvarchar</code> 型の) 定数式です。 uchar_expr 文字型のカラム名、変数、または (<code>unichar</code> または <code>univarchar</code> 型の) 定数式です。
例	<pre>select soundex ("smith"), soundex ("smythe") ----- S530 S530</pre>
使用法	<ul style="list-style-type: none">文字列関数 <code>soundex</code> は、一連の有効なシングルバイトまたはダブルバイトのローマ字からなる文字列の 4 文字 <code>soundex</code> コードを返します。<code>soundex</code> 関数は、アルファベット文字列を 4 桁コードに変換します。このコードは、発音が似ている単語や名前を検索するために使用されます。文字列の先頭文字以外のすべての母音は無視されます。<code>char_expr</code> または <code>uchar_expr</code> が <code>NULL</code> のときは、<code>NULL</code> を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>soundex</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 difference

space

説明	指定した数のシングルバイト・スペースからなる文字列を返します。
構文	<code>space(integer_expr)</code>
パラメータ	<i>integer_expr</i> 整数値型 (<code>tinyint</code> 、 <code>smallint</code> 、または <code>int</code>) のカラム名、変数、または定数式のいずれかです。
例	<pre>select "aaa", space(4), "bbb" ----- aaa bbb</pre>
使用法	文字列関数 <code>space</code> は、指定した数のシングルバイト・スペースからなる文字列を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>space</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 isnull 、 rtrim

spid_instance_id

説明	クラスタ環境のみ – 指定したプロセス ID (spid) を実行しているインスタンスの ID を返します。
構文	<code>spid_instance_id(spид_value)</code>
パラメータ	<i>spид_value</i> インスタンス ID を要求する spid 番号
例	プロセス ID 番号 27 をインスタンスの ID を返します。 <pre>select spid_instance_id(27)</pre>
使用法	<ul style="list-style-type: none">• spid の値を含めない場合、spid_instance_id は NULL を返します。• 無効な値や存在しないプロセス ID 値を入力すると、spid_instance_id は NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが spid_instance_id を実行できます。

square

説明	指定した値の平方を、float で表したものを返します。
構文	<code>square(numeric_expression)</code>
パラメータ	<i>numeric_expression</i> float 型の数値式です。
例	<p>例 1 整数カラムの平方を返します。</p> <pre>select square(total_sales) from titles ----- 16769025.00000 15023376.00000 350513284.00000 ... 16769025.00000 (18 row(s) affected)</pre> <p>例 2 通貨カラムの平方を返します。</p> <pre>select square(price) from titles ----- 399.600100 142.802500 8.940100 NULL ... 224.700100 (18 row(s) affected)</pre>
使用法	この関数は <code>power(numeric_expression,2)</code> と同じですが、int 型ではなく float 型を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが square 関数を実行できます。
参照	関数 power データ型 <code>exact_numeric</code> 、 <code>approximate_numeric</code> 、 <code>money</code> 、 <code>float</code>

sqrt

説明	指定した数値の平方根を返します。
構文	<code>sqrt(<i>approx_numeric</i>)</code>
パラメータ	<i>approx_numeric</i> 概数値型 (<code>float</code> 、 <code>real</code> 、または <code>double precision</code> 型) のカラム名、変数、または正の数になる定数式のいずれかです。
例	<pre>select sqrt(4) 2.000000</pre>
使用法	<ul style="list-style-type: none">算術関数 <code>sqrt</code> は、指定した値の平方根を返します。負の数値の平方根を求めようとすると、Adaptive Server は次のようなエラー・メッセージを返します。<pre>Domain error occurred.</pre>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>sqrt</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 power

stddev

説明

1 つの数値式で構成される標本標準偏差を `double` 型として計算します。

注意 `stddev` と `stdev` は、`stddev_samp` のエイリアスです。詳細については、[stddev_samp \(261 ページ\)](#) を参照してください。

stdev

説明

1つの数値式で構成される標本標準偏差を `double` 型として計算します。

注意 `stddev` と `stdev` は、`stddev_samp` のエイリアスです。詳細については、[stddev_samp \(261 ページ\)](#) を参照してください。

stdevp

説明

1つの数値式で構成される母標準偏差を `double` 型として計算します。

注意 `stdevp` は、`stddev_pop` のエイリアスです。詳細については、[stddev_pop \(260 ページ\)](#) を参照してください。

stddev_pop

説明 1つの数値式で構成される母標準偏差を `double` 型として計算します。 `stdevp` は、 `stddev_pop` のエイリアスで、同じ構文を使用します。

構文 `stddev_pop ([all | distinct] expression)`

パラメータ `all`

すべての値に `stddev_pop` を適用します。デフォルト設定は `all` です。

`distinct`

`stddev_pop` を適用する前に重複する値を削除します。

expression

その母集団ベースの標準偏差がローのセットに対して計算される式です (通常はカラム名)。

例 次の文は、 `pubs2` データベースにおける書籍の各種類の前払い金の平均と標準分散を示します。

```
select type, avg(advance) as "avg", stddev_pop(advance)
as "stddev" from titles group by type order by type
```

使用法

グループの各ロー (`distinct` が指定されている場合、重複が削除された後に残る各ロー) に対して評価される、指定された値式の母標準偏差を計算します。これは、母分散の平方根として定義されます。

図 2-1: 母集団に関する統計集合関数の計算式

平均が μ (`var_pop`) のサイズ n の母分散を定義する計算式は、次のとおりです。母標準偏差値 deviation (`stddev_pop`) は、この数値の正の平方根です。

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n}$$

σ^2 = 分散
 n = 母集団のサイズ
 $\mu = x_i$ 値の平均

標準規格

ANSI SQL – 準拠レベル: Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `stddev_pop` を実行できます。

参照

マニュアル Transact-SQL ユーザーズ・ガイド

関数 [stddev_samp](#), [var_pop](#), [var_samp](#)

stddev_samp

説明 1つの数値式で構成される標本標準偏差を `double` 型として計算します。`stdev` と `stddev` は、`stddev_samp` のエイリアスで、同じ構文を使用します。

構文 `stddev_samp ([all | distinct] expression)`

パラメータ

all

すべての値に `stddev_samp` を適用します。デフォルト設定は `all` です。

distinct

`stddev_samp` を適用する前に重複する値を削除します。

expression

任意の numeric データ型 (`float`、`real`、または `double precision`) の式です。

例

次の文は、`pubs2` データベースにおける書籍の各種類の前払い金の平均と標準分散を示します。

```
select type, avg(advance) as "avg",
       stddev_samp(advance) as "stddev" from titles
       where total_sales > 2000 group by type order by type
```

使用法

グループの各ロー (`distinct` が指定されている場合、重複が削除された後に残る各ロー) に対して評価される、指定された値式の標本標準偏差を計算します。これは、標本分散の平方根として定義されます。

図 2-2: 標本に関する統計集合関数の計算式

平均 \bar{x} (`var_samp`) を持つサイズ n の標本からの母分散の不偏推定値を定義する計算式は、次のとおりです。標本標準偏差 (`stddev_samp`) は、この数値の正の平方根です。

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

$s^2 =$ 分散
 $n =$ 標本サイズ
 $\bar{x} = x_i$ 値の平均

標準規格

ANSI SQL – 準拠レベル: Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `stddev_samp` を実行できます。

参照

マニュアル Transact-SQL ユーザーズ・ガイド

関数 [stddev_pop](#), [var_pop](#), [var_samp](#)

str

説明 指定した数値に相当する文字列を返し、指定された長さになるように出力に文字または数値を埋め込みます。

構文 `str(approx_numeric[, length [, decimal]])`

パラメータ `approx_numeric`
概数値型 (float、real、または double precision) のカラム名、変数、または定数式のいずれかです。

length

返す文字の数 (小数点、小数桁、整数桁、ブランクを含む) を指定します。デフォルトは 10。

decimal

返す小数桁数を指定します。デフォルトは 0 です。指定した長さになるように出力に文字または数値を埋め込むときにも使用できます。

文字または数値をリテラル文字列に指定する場合、文字または数値をフィールドの埋め込みに使用できます。数値を指定する場合は、小数桁数を設定します。デフォルトは 0 です。**decimal** を設定しない場合、フィールドには **length** で指定された値になるようにブランクが埋め込まれます。

例 1 **decimal** をリテラル文字列 '0' に設定した場合、10 スペースの長さになるようにフィールドには 0 が埋め込まれます。

```
select str(5,10,'0')
-----
0000000005
```

例 2 **decimal** を 5 に設定した場合、小数桁数は 5 桁に設定されます。

```
select str(5,10,5)
-----
5.00000
```

例 3 **decimal** を文字 '_' に設定した場合、元の値が保持され長さが 16 スペースになるようにフィールドには指定した文字が埋め込まれます。

```
select str(12.34500,16,'_')
-----
_____12.34500
```

例 4 **decimal** を設定しない場合、浮動小数点数がゼロに設定され長さが 16 スペースになるようにフィールドにはブランクが埋め込まれます。

```
select str(12.34500e,16,7)
-----
12
```

例 5 *decimal* を数値に設定した場合、浮動小数点数が小数点以下 7 桁まで処理され長さが 16 スペースになるようにフィールドにはブランクが埋め込まれます。

```
select str(12.34500e,16,7)
-----
12.3450000
```

例 6 以下の例を使用して、プレフィクス文字を指定した小数桁数まで浮動小数点数を処理します。

```
select str(convert(numeric(10,2),12.34500e),16,'-')
-----
-----12.35

select str(convert(numeric(10,8),12.34500e),16,'-')
-----
-----12.34500000
```

使用法

- *length* と *decimal* はオプションです。使用する場合は、負ではない値を指定してください。str 関数は、結果が指定した長さになるよう数値の小数部を丸めます。長さは小数点や数字の符号(負数の場合)が十分入る長さにしてください。結果の小数部分は、指定された長さに収まるように丸められます。ただし、数字の整数部分が指定された長さを超える場合、str は指定された長さ分のアスタリスクのローを返します。たとえば、次のようにします。

```
select str(123.456, 2, 4)
--
**
```

- *approx_numeric* が NULL の場合、NULL を返します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが str 関数を実行できます。

参照

マニュアル Transact-SQL ユーザーズ・ガイド

関数 [abs](#), [ceiling](#), [floor](#), [round](#), [sign](#)

str_replace

説明

最初の文字列式 (*string_expression1*) 内に出現する 2 番目の文字列式 (*string_expression2*) のすべてのインスタンスを、3 番目の式 (*string_expression3*) で置き換える。

構文

```
str_replace("string_expression1", "string_expression2", "string_expression3")
```

パラメータ

string_expression1

検索される側のソース文字列または文字列式です。char、varchar、unichar、univarchar、varbinary、または binary データ型として表されます。

string_expression2

最初の式 (*string_expression1*) 内で検索するパターン文字列または文字列式です。*string_expression2* は char、varchar、unichar、univarchar、varbinary、または binary データ型として表されます。

string_expression3

2 番目の式を置き換える文字列式です。char、varchar、unichar、univarchar、binary、または varbinary データ型として表されます。

例

例 1 文字列 *cdefghi* 内の文字列 *def* を *yyy* に置き換えます。

```
str_replace("cdefghi", "def", "yyy")
-----
cyyyghi
(1 row(s) affected)
```

例 2 すべてのスペースを “toyota” に置き換えます。

```
select str_replace("chevy, ford, mercedes", " ", "toyota")
-----
chevy, toyotaford, toyotamercedes
(1 row(s) affected)
```

注意 Adaptive Server では、空の文字列を NULL 値と区別するために、空の文字列定数は自動的に 1 つのスペース文字列に変換されます。

例 3 “*abcdefghijklm*” を返します。

```
select str_replace("abcdefghijklm", "def", NULL)
-----
abcdefghijklm
(1 row affected)
```

使用法	<ul style="list-style-type: none"> • <i>string_expression</i> (1、2、または 3) が char または varchar の場合は、varchar データを返します。 • <i>string_expression</i> (1、2、または 3) が unichar または univarchar の場合は、univarchar データを返します。 • <i>string_expression</i> (1、2、または 3) が binary または varbinary の場合は、varbinary データを返します。 • すべての引数を同じデータ型にしてください。 • 3 つの引数のいずれかが NULL の場合は、null を返します。 <p><i>str_replace</i> では、3 つ目のパラメータに NULL を使用できるようになりました。この場合、<i>string_expression2</i> が NULL に置き換えられます。この方法で <i>str_replace</i> を使用して文字列を削除する操作を行うことができます。たとえば、次の例は “abcdefghiklm” を返します。</p> <pre style="margin-left: 40px;">str_replace("abcdefghiklm", "def", NULL)</pre> <ul style="list-style-type: none"> • 結果の長さは、式がコンパイルされるときに、引数値が既知かどうかによって異なります。すべての引数が既知の定数値を持つ変数の場合は、結果の長さは、Adaptive Server によって次のように計算されます。 <pre style="margin-left: 40px;">result_length = ((s/p)*(r-p)+s) ここで、 s = length of source string p = length of pattern string r = length of replacement string if (r-p) <= 0, result length = s</pre> <ul style="list-style-type: none"> • ソース文字列 (<i>string_expression1</i>) がカラムで、<i>string_expression2</i> と <i>string_expression3</i> がコンパイル時に既知の定数値である場合は、結果の長さは上記の計算式を使って計算されます。 • 式のコンパイル時に引数値がわからないため Adaptive Server が結果の長さを計算できない場合は、トレース・フラグ 244 が on でないかぎり、結果の長さは 255 になります。トレース・フラグ 244 が on の場合は、結果の長さは 16384 になります。 • <i>result_len</i> が 16384 を超えることはありません。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <i>str_replace</i> 関数を実行できます。
参照	データ型 char、varchar、binary、varbinary、unichar、univarchar 関数 length

strtobin

説明 一連の英数字をそれに相当する 16 進数に変換します。

構文 `select strtobin("string of valid alphanumeric characters")`

パラメータ *string of valid alphanumeric characters*
[1 - 9]、[a - f]、および [A - F] からなる有効な英数字の文字列です。

例 1 英数字「723ad82fe」を 16 進の数列に変換します。

```
select strtobin("723ad82fe")
go

-----
0x0723ad82fe
```

この例では、英数字文字列とそれに相当する 16 進数のメモリ内表現は次のようになります。

英数字文字列 (9 バイト)

0	7	2	3	a	d	8	2	f	e
---	---	---	---	---	---	---	---	---	---

16 進数 (5 バイト)

0	7	2	3	a	d	8	2	f	e
---	---	---	---	---	---	---	---	---	---

この関数は文字を右から左へと処理します。この例では、入力の数文字数は奇数です。そのため、16 進の数列に「0」のプレフィクスが付いて出力に反映されます。

例 2 `@str_data` というローカル変数の英数字文字列を「723ad82fe」の値に相当する 16 進の数列に変換します。

```
declare @str_data varchar(30)
select @str_data = "723ad82fe"
select strtobin(@str_data)
go

-----
0x0723ad82fe
```

使用法

- 入力の無効な文字は、出力として NULL になります。
- 16 進数の入力シーケンスには“0x”のプレフィクスが必要です。
- NULL の入力結果は NULL の出力になります。

標準規格 ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション strtobin は、すべてのユーザが実行できます。

参照 関数 [bintostr](#)

stuff

説明	1つの文字列から指定した数の文字を削除して、代わりに別の文字列を入れて作成した文字列を返します。
構文	<code>stuff(char_expr1 uchar_expr1, start, length, char_expr2 uchar_expr2)</code>
パラメータ	char_expr1 文字型のカラム名、変数、または (char、varchar、nvarchar、または nvarchar 型の) 定数式です。 uchar_expr1 文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。 start 削除する文字の先頭位置を指定します。 length 削除する文字の数を指定します。 char_expr2 別の文字型のカラム名、変数、または (char、varchar、nvarchar、または nvarchar 型の) 定数式です。 uchar_expr2 別の文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。
例	例 1 <pre>select stuff("abc", 2, 3, "xyz") ---- axyz</pre> 例 2 <pre>select stuff("abcdef", 2, 3, null) go --- aef</pre> 例 3 <pre>select stuff("abcdef", 2, 3, "") ---- a ef</pre>
使用法	<ul style="list-style-type: none">文字列関数 <code>stuff</code> は、<code>char_expr1</code> または <code>uchar_expr1</code> の位置 <code>start</code> から <code>length</code> の数の文字を削除し、次に <code>char_expr2</code> または <code>uchar_expr2</code> を、<code>char_expr1</code> または <code>uchar_expr2</code> の <code>start</code> の位置に挿入します。文字列関数の一般的な情報については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

- 開始位置または長さが負の場合は、NULL 文字列が返されます。先頭位置がゼロまたは *expr1* より長い場合、NULL 文字列を返します。長さが *expr1* より長いときは、*expr1* は最後の文字まで削除されます (例 1 を参照)。
- 指定した先頭位置がサロゲート・ペアの中央にある場合、先頭位置は指定の位置より 1 文字少ない位置に調整されます。指定した長さの終わりの位置がサロゲート・ペアの中央にある場合、長さは指定の長さより 1 文字少なくなります。
- `stuff` 関数を使って文字を削除するには、*expr2* には空の引用符ではなく NULL を使います。“ ” を使用して null 文字を指定すると、スペースに置き換わります (例 2 と 3 を参照)。
- *char_expr1* または *uchar_expr1* が NULL のとき、`stuff` は NULL を返しません。*char_expr1* または *uchar_expr1* が文字列値で *char_expr2* または *uchar_expr2* が NULL の場合、`stuff` は、削除された文字の代わりに何も挿入しません。
- `varchar` 式と `unichar` 式をそれぞれパラメータとして指定した場合、`varchar` 式は暗黙的に `unichar` に変換されます (変換時にトランケーションが発生することがあります)。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `stuff` 関数を実行できます。

参照

関数 [replicate](#), [substring](#)

substring

説明	指定した数の文字を文字列から取り出して作った文字列を返します。
構文	<code>substring(expression, start, length)</code>
パラメータ	<p>expression バイナリ型または文字型のカラム名、変数、または定数式です。char、nchar、unicar、varchar、univarchar、または nvarchar のデータ、binary または varbinary を使用できます。</p> <p>start 部分文字列の先頭文字の位置を指定します。</p> <p>length 部分文字列の文字数を指定します。</p>
例	<p>例 1 “Bennet A” のように、各作者の姓と名前の頭文字を表示します。</p> <pre>select au_lname, substring(au_fname, 1, 1) from authors</pre> <p>例 2 作者の姓を大文字に変換し、最初の 3 文字を表示します。</p> <pre>select substring(upper(au_lname), 1, 3) from authors</pre> <p>例 3 pub_id と title_id を連結して作成された文字列の最初の 6 文字を表示します。</p> <pre>select substring((pub_id + title_id), 1, 6) from titles</pre> <p>例 4 各桁が 2 つのバイナリ値を表すバイナリ・フィールドから、下 4 桁を取り出します。</p> <pre>select substring(xactid, 5, 2) from syslogs</pre>
使用法	<ul style="list-style-type: none"> 文字列関数 substring は、文字列またはバイナリ文字列の一部を返しません。文字列関数の一般的な情報については、『Transact-SQL ユーザーズ・ガイド』を参照してください。 substring の 2 番目の引数が NULL の場合、結果は NULL です。substring の最初および 3 番目の引数が NULL の場合、結果はブランクです。 uchar_expr1 の式で指定の先頭位置がサロゲート・ペアの中央にある場合、start は指定の位置より 1 文字少ない位置に調整されます。uchar_expr1 の式で長さの終わりの位置がサロゲート・ペアの中央にある場合、length は指定の長さより 1 文字少なくなります。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが substring 関数を実行できます。
参照	関数 charindex 、 patindex 、 stuff

sum

説明 値の合計を返します。

構文 `sum([all | distinct] expression)`

パラメータ `all`

すべての値に `sum` を適用します。デフォルト設定は `all` です。

`distinct`

`sum` を適用する前に、重複する値を削除します。`distinct` はオプションです。

`expression`

カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、`expression` はカラム名です。詳細については、「[式](#)」(325 ページ)を参照してください。

例 **例 1** すべてのビジネス雑誌について前払い金の平均と総売上額を計算します。これらの各集合関数は、検索したすべてのローに対して 1 つの合計値を返します。

```
select avg(advance), sum(total_sales)
from titles
where type = "business"
```

例 2 `group by` 句を指定すると、集合関数はテーブル全体ではなくグループごとに 1 つの値を計算します。この文では、本の種類ごとに合計値を計算します。

```
select type, avg(advance), sum(total_sales)
from titles
group by type
```

例 3 `titles` テーブルは出版社別のグループに分けられ、前払い総額が 25,000 ドルを超え、本の平均価格が 15 ドルを超える出版社のグループだけが合計されます。

```
select pub_id, sum(advance), avg(price)
from titles
group by pub_id
having sum(advance) > $25000 and avg(price) > $15
```

使用法

- 集合関数 `sum` は、カラム内のすべての値の合計を求めます。`sum` 関数を使えるのは、数値 (整数、浮動小数点、または通貨) データ型だけです。`null` 値は合計では無視されます。
- 整数データを合計すると、カラムのデータ型が `smallint` または `tinyint` であっても、Adaptive Server では結果が `int` 値として扱われます。`bigint` のデータを合計すると、結果は `bigint` 値として扱われます。DB-Library プログラムでのオーバーフロー・エラーを避けるために、平均や合計の結果が格納されるすべての変数を適切に宣言してください。
- バイナリ・データ型では `sum` 関数は使えません。
- この関数では数値型だけを定義します。Unicode 式に対してこの関数を実行すると、エラーが発生します。

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>sum</code> 関数を実行できます。
参照	コマンド <code>compute 句</code> , <code>group by 句</code> と <code>having 句</code> , <code>select</code> , <code>where 句</code> マニュアル Transact-SQL ユーザーズ・ガイド 関数 <code>count</code> , <code>max</code> , <code>min</code>

suser_id

説明 syslogins テーブルからサーバ・ユーザの ID 番号を返します。

構文 suser_id([server_user_name])

パラメータ **server_user_name**
Adaptive Server のログイン名です。

例 **例 1**

```
select suser_id()  
-----  
1
```

例 2

```
select suser_id("margaret")  
-----  
5
```

使用法

- システム関数 **suser_id** は、**syslogins** からサーバ・ユーザの ID 番号を返します。システム関数の一般的な情報については、『Transact-SQL ユーザーズ・ガイド』を参照してください。
- 特定のデータベース内のユーザ ID を **sysusers** テーブルから検索するには、システム関数 **user_id** を使用します。
- **server_user_name** を指定しない場合、**suser_id** は現在のユーザのサーバ ID を返します。

標準規格 ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション すべてのユーザが **suser_id** 関数を実行できます。

参照 マニュアル Transact-SQL ユーザーズ・ガイド

関数 [suser_name](#), [user_id](#)

suser_name

説明	現在のサーバ・ユーザの名前、または指定したサーバ ID を持つユーザの名前を返します。
構文	<code>suser_name([server_user_id])</code>
パラメータ	<code>server_user_id</code> Adaptive Server のユーザ ID です。
例	例 1 <pre>select suser_name() ----- sa</pre> 例 2 <pre>select suser_name(4) ----- margaret</pre>
使用法	システム関数 <code>suser_name</code> は、サーバ・ユーザの名前を返します。サーバ・ユーザ ID は <code>syslogins</code> に格納されています。 <code>server_user_id</code> を指定しない場合、 <code>suser_name</code> は現在のユーザの名前を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>suser_name</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 suser_id, user_name

syb_quit

説明 接続を終了します。

構文 `syb_quit()`

例 関数が実行されている接続を終了し、エラー・メッセージを返します。

```
select syb_quit()  
-----  
CT-LIBRARY error:  
  ct_results(): network packet layer:  
internal net library error: Net-Library operation terminated  
due to disconnect
```

使用法 `isql` プリプロセッサ・コマンド `exit` でエラーが発生した場合は、`syb_quit` を使用するとスクリプトを終了できます。

パーミッション すべてのユーザが `syb_quit` 関数を実行できます。

syb_sendmsg

説明	UNIX のみ UDP (ユーザ・データグラム・プロトコル) ポートにメッセージを送信します。
構文	<code>syb_sendmsg ip_address, port_number, message</code>
パラメータ	<p>ip_address UDP アプリケーションが稼働しているマシンの IP アドレスです。</p> <p>port_number UDP ポートのポート番号です。</p> <p>message 送信されるメッセージです。最大 255 文字数までです。</p>
例	<p>例 1 “Hello” というメッセージを IP アドレス 120.10.20.5 のポート 3456 に送信します。</p> <pre>select syb_sendmsg("120.10.20.5", 3456, "Hello")</pre> <p>例 2 IP アドレスとポート番号をユーザ・テーブルから読み取り、送信するメッセージの変数を使用します。</p> <pre>declare @msg varchar(255) select @msg = "Message to send" select syb_sendmsg (ip_address, portnum, @msg) from sendports where username = user_name()</pre>
使用法	<ul style="list-style-type: none"> UDP のメッセージ機能を使用できるようにするには、システム・セキュリティ担当者が allow sendmsg パラメータを 1 に設定する必要があります。 syb_sendmsg では、セキュリティ・チェックは実行されません。ネットワーク経由で重要な情報を送信する目的には syb_sendmsg を使用しないよう強くおすすめます。この機能を有効にした場合、ユーザは、このメッセージ機能を使用したために発生するあらゆるセキュリティ上の問題を承認したことになります。 UDP ポートを作成するサンプル C プログラムについては、「sp_sendmsg」を参照してください。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが syb_sendmsg 関数を実行できます。
参照	システム・プロシージャ sp_sendmsg

sys_tempdbid

説明	クラスタ環境のみ – 指定したインスタンスの有効なローカル・システム・テンポラリ・データベースの ID を返します。 <i>instance_id</i> を指定しない場合は、現在のインスタンスの有効なローカル・システム・テンポラリ・データベースの ID を返します。
構文	<code>sys_tempdbid(instance_id)</code>
パラメータ	<i>instance_id</i> インスタンスの ID です。
例	インスタンス ID が 3 のインスタンスの有効なローカル・システム・テンポラリ・データベースの ID を返します。 <pre>select sys_tempdbid(3)</pre>
使用法	インスタンス ID を指定しない場合、 <code>sys_tempdbid</code> は現在のインスタンスの有効なローカル・システム・テンポラリ・データベースの ID を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>sys_tempdbid</code> を実行できます。

tan

説明	指定した角度 (ラジアン) の正接を返します。
構文	<code>tan(<i>angle</i>)</code>
パラメータ	<i>angle</i> カラム名、変数、または (float、real、double precision、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 式として表される角度の大きさ (ラジアン) です。
例	<pre>select tan(60) ----- 0.320040</pre>
使用法	算術関数 tan は、指定した角度 (ラジアン単位) の正接を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが tan 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 atan , atn2 , degrees , radians

tempdb_id

説明	指定のセッションが割り当てられているテンポラリ・データベースをレポートします。 tempdb_id 関数の入力サーバ・プロセス ID であり、出力はそのプロセスが割り当てられているテンポラリ・データベースです。サーバ・プロセスを指定しない場合、 tempdb_id は、現在のプロセスに割り当てられているテンポラリ・データベースの dbid をレポートします。
構文	tempdb_id()
例	指定したテンポラリ・データベースに割り当てられているすべてのサーバ・プロセスを検索するには、次を実行します。 <pre>select spid from master..sysprocesses where tempdb_id(spid) = db_id("tempdatabase")</pre>
使用法	select tempdb_id は、 select @@tempdbid と同じ結果になります。
参照	コマンド select

textptr

説明	text、image、または unitext カラムの先頭ページを指すポインタを返します。
構文	textptr(column_name)
パラメータ	<p>column_name text カラムの名前です。</p>
例	<p>例 1 textptr 関数を使用して、作者に関する blurbs テーブル内の au_id 486-29-1786 に対応する、text カラムの copy を検索します。テキスト・ポインタはローカル変数 @val に配置され、readtext コマンドのパラメータとして指定されます。このコマンドは2番目のバイト(オフセット 1)から5バイト分を返します。</p> <pre>declare @val binary(16) select @val = textptr(copy) from blurbs where au_id = "486-29-1786" readtext blurbs.copy @val 1 5</pre> <p>例 2 blurbs テーブルから title_id カラムと、copy カラムの 16 バイト・テキスト・ポインタを返します。</p> <pre>select au_id, textptr(copy) from blurbs</pre>
使用法	<ul style="list-style-type: none"> • テキストおよびイメージ関数 textptr は、テキスト・ポインタ値 (16 バイトの varbinary 値) を返します。 • 転送ロー内に格納された LOB カラムに対して返される textptr 値は変更されず転送後も有効です。 • text、unitext、または image カラムが、null 以外の insert 文や任意の update 文により初期化されていないと、textptr 関数は NULL ポインタを返します。テキスト・ポインタが存在するかどうかは、textvalid 関数を使ってチェックします。有効なテキスト・ポインタがない場合は、writetext または readtext を使用できません。
	<hr/> <p>注意 varbinary 値がテーブルに格納される場合、この値の後続の f はトランケートされます。テキスト・ポインタ値をテーブルに格納する場合は、カラムのデータ型として binary を使用します。</p> <hr/>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが textptr 関数を実行できます。
参照	<p>データ型 text、image、unitext データ型</p> <p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 textvalid</p> <p>コマンド insert、update、readtext、writetext</p>

textvalid

説明	指定した text カラム、 unitext カラム、ロー内とロー外の LOB カラムを指すポインタが有効な場合は 1 を返し、無効な場合は 0 を返します。
構文	<code>textvalid("table_name.column_name", textpointer)</code>
パラメータ	table_name.column_name テーブルの名前とその text カラムです。 textpointer テキスト・ポインタ値です。
例	texttest テーブルの blurb カラムの各値に、有効なテーブル・ポインタがあるかどうかをレポートします。 <pre>select textvalid ("texttest.blurb", textptr(blurb)) from texttest</pre>
使用法	<ul style="list-style-type: none">• textvalid は特定のポインタが有効であるかどうかをチェックします。ポインタが有効な場合は 1 を返し、無効な場合は 0 を返します。• text カラムや image カラムの識別子には、テーブル名を含めてください。• テキストおよびイメージ関数の一般的な情報については、『Transact-SQL ユーザーズ・ガイド』を参照してください。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが textvalid 関数を実行できます。
参照	データ型 text 、 image 、 unitext データ型 マニュアル Transact-SQL ユーザーズ・ガイド 関数 textptr

to_unichar

説明	指定した整数式の値が含まれた unichar 式を返します。
構文	<code>to_unichar(integer_expr)</code>
パラメータ	<i>integer_expr</i> 整数値型 (tinyint、smallint、または int) のカラム名、変数、または定数式のいずれかです。
使用法	<ul style="list-style-type: none">• 文字列関数 to_unichar は、Unicode 整数値を Unicode 文字値に変換します。• unichar 式がサロゲート・ペアのいずれか 1 つのサロゲートだけを参照している場合、エラー・メッセージが表示されて処理がアボートされます。• <i>integer_expr</i> が NULL の場合、to_unichar は NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが to_unichar 関数を実行できます。
参照	データ型 text 、 image 、 unitext データ型 マニュアル Transact-SQL ユーザーズ・ガイド 関数 char

tran_dumpable_status

説明 dump transaction を使用できるかどうかを示す true/false の値を返します。

構文 tran_dumpable_status("database_name")

パラメータ database_name
ターゲット・データベースの名前です。

例 pubs2 データベースをダンプできるかどうかをチェックします。

```
1> select tran_dumpable_status("pubs2")
2> go
```

```
-----
          106
```

```
(1 row affected)
```

この例では、pubs2 をダンプできません。リターン・コード 106 は、一致した条件 (2、8、32、64) すべての合計です。リターン・コードについては、「使用法」の項を参照してください。

使用法 tran_dumpable_status を使用して、データベースでダンプ・トランザクションが許可されるかどうかをコマンドを実行せずに確認できます。

tran_dumpable_status は、ダンプ・トランザクションの実行時に Adaptive Server で行われるすべてのチェックを実行します。

tran_dumpable_status が 0 を返す場合は、データベースに dump transaction コマンドを実行できます。その他の値を返す場合は、実行できません。0 以外の値の意味は次のとおりです。

- 1 - 指定した名前のデータベースは存在しない。
- 2 - ログが別のデバイスに存在しない。
- 4 - ログの最初のページがデータ専用ディスク・フラグメントの範囲にある。
- 8 - データベースに trunc log on chkpt オプションが設定されている。
- 16 - ログを取らない書き込みがデータベースで実行された。
- 32 - トランケート・オンリーの dump tran が、ダンプ・デバイスへの連続したダンプに割り込みをかけた。
- 64 - データベースが新規に作成されたか、アップグレードされた。トランザクション・ログは、dump database が実行されるまでダンプされません。
- 128 - データベースの持続性はトランザクションのダンプを許可していない。

- 256 – データベースは読み込み専用である。dump transaction がトランザクションを開始しましたが、読み込み専用データベースでは許可されていません。
- 512 – データベースはスタンバイ・アクセスのためオンラインである。dump transaction がトランザクションを開始しましたが、ローカル・シーケンスを妨害するため、スタンバイ・アクセスのデータベースでは許可されていません。
- 1024 – データベースは、dump transaction をサポートしていないアーカイブ・データベースである。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

この関数は、すべてのユーザが実行できます。

参照

コマンド [dump transaction](#)

tsequal

説明	ローがブラウズ用に選択されたあとで変更された場合に、ローを更新できないようにするため <code>timestamp</code> 値を比較します。
構文	<code>tsequal(browsed_row_timestamp, stored_row_timestamp)</code>
パラメータ	<p><code>browsed_row_timestamp</code> ブラウズされたローの <code>timestamp</code> カラムです。</p> <p><code>stored_row_timestamp</code> 保管されたローの <code>timestamp</code> カラムです。</p>
例	<p><code>publishers</code> テーブルの現在のバージョンから <code>timestamp</code> カラムを検索し、保管された <code>timestamp</code> カラムの値と比較します。<code>timestamp</code> カラムを追加するには、次のコマンドを実行します。</p> <pre>alter table publishers add timestamp</pre> <p>2つの <code>timestamp</code> カラムの値が等しい場合、<code>tsequal</code> はローを更新します。等しくない場合、<code>tsequal</code> は次のエラー・メッセージを返します。</p> <pre>update publishers set city = "Springfield" where pub_id = "0736" and tsequal(timestamp,0x0001000000002ea8) Msg 532, Level 16, State 2: Server 'server_name', Line 1: The timestamp (changed to 0x0001000000002ea8) shows that the row has been updated by another user. Command has been aborted. (0 rows affected)</pre>
使用法	<ul style="list-style-type: none"> システム関数 <code>tsequal</code> は、<code>timestamp</code> カラムの値を比較して、ブラウズ用に選択されたあとで変更されたローの更新ができないようにします。システム関数の一般的な情報については、『<i>Transact-SQL ユーザーズ・ガイド</i>』を参照してください。 <code>tsequal</code> 関数を使うと、DB-Library の <code>dbequal</code> 関数を呼び出さずにブラウズ・モードを使うことができます。ブラウズ・モードでは、データの表示中に更新を実行できます。ブラウズ・モードは <code>Open Client</code> や <code>ホスト・プログラミング言語</code> を使う <code>フロントエンド・アプリケーション</code> で使います。ローにタイムスタンプが記録されているテーブルをブラウズできます。 <code>フロントエンド・アプリケーション</code> でテーブルをブラウズするには、<code>Adaptive Server</code> に送信する <code>select</code> 文の最後に <code>for browse</code> キーワードを追加します。たとえば、次のようにします。

```
Start of select statement in an Open Client application
```

```
...
```

```
    for browse
```

```
Completion of the Open Client application routine
```

- `tsequal` 関数は、`select` 文の `where` 句では使用しないでください。 `where` 句でこの関数を使用できるのは、`insert` 文や `update` 文で `where` 句以降が1つのユニークなローに一致する場合だけです。

`timestamp` カラムを検索句として使うときは、通常の `varbinary` カラムのように、`timestamp1 = timestamp2` として比較してください。

ブラウズ用の新規テーブルにタイムスタンプを設定する

- ブラウズ用に新しいテーブルを作成するときは、`timestamp` というカラムをテーブル定義に指定してください。このカラムには自動的に `timestamp` データ型が割り当てられるので、データ型を指定する必要はありません。たとえば、次のようにします。

```
create table newtable(col1 int, timestamp, col3 char(7))
```

ローを挿入または更新するたびに、Adaptive Server は `timestamp` カラムにユニークな `varbinary` 値を自動的に割り当て、タイムスタンプを設定します。

既存のテーブルにタイムスタンプを設定する

- ブラウズ用に既存のテーブルを準備するには、`alter table` を使用して `timestamp` カラムを追加します。たとえば、NULL の値を持つ `timestamp` カラムを既存のローに追加するには、次のクエリを使用します。

```
alter table oldtable add timestamp
```

タイムスタンプを生成するには、新しいカラム値を指定しないで、既存の各ローを更新します。

```
update oldtable  
set col1 = col1
```

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `tsequal` 関数を実行できます。

参照

データ型 [timestamp データ型](#)

uhighsurr

説明	<p><code>start</code> の位置の Unicode 値が、サロゲート・ペアの上位サロゲート (ペアの中で最初に指定されている) の場合は 1 を返します。それ以外の場合は、0 を返す。この関数ではサロゲート処理のための明示的なコードを作成できる。</p>
構文	<code>uhighsurr(<i>uchar_expr</i>, <i>start</i>)</code>
パラメータ	<p><i>uchar_expr</i></p> <p>文字型のカラム名、変数、または (unicar または univarchar 型の) 定数式です。</p> <p><i>start</i></p> <p>検査する文字位置を指定します。</p>
使用法	<ul style="list-style-type: none">文字列関数 <code>uhighsurr</code> を使用すれば、サロゲート処理のための明示的なコードを作成できます。特に、<code>uhighsurr</code> が <code>true</code> である Unicode 文字で部分文字列が始まる場合、少なくとも 2 つの Unicode 値を持つ部分文字列を抽出します (<i>substr</i> はサロゲート・ペアのいずれか 1 つのサロゲートを抽出しません)。<i>uchar_expr</i> が <code>NULL</code> の場合、<code>uhighsurr</code> は <code>NULL</code> を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>uhighsurr</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド

関数 [ulowsurr](#)

ulowsurr

説明	<p><i>start</i> の位置の Unicode 値が、サロゲート・ペアの下位サロゲート (ペアの中で 2 番目に指定されている) の場合は、1 を返します。それ以外の場合は 0 を返します。この関数を使用すると、substr()、stuff()、right() によって実行される調整を明示的にコード化できます。</p>
構文	<pre>ulowsurr(<i>uchar_expr</i>, <i>start</i>)</pre>
パラメータ	<p><i>uchar_expr</i> 文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。</p> <p><i>start</i> 検査する文字位置を指定します。</p>
使用法	<ul style="list-style-type: none">文字列関数 ulowsurr を使用すれば、substr、stuff、right によって実行される調整処理の明示的なコードを作成できます。特に、ulowsurr が true である Unicode 値で部分文字列が終わる場合、1 少ない (または 1 多い) 部分文字列が抽出されます。substr では、対になっていないサロゲート・ペアが含まれている文字列は抽出されません。<i>uchar_expr</i> が NULL のとき、ulowsurr は NULL を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが ulowsurr 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 uhighsurr

upper

説明	指定した小文字文字列に相当する大文字を返します。
構文	<code>upper(char_expr)</code>
パラメータ	char_expr 文字型のカラム名、変数、または (<code>char</code> 、 <code>unichar</code> 、 <code>varchar</code> 、 <code>nchar</code> 、 <code>nvarchar</code> 、または <code>univarchar</code> 型の) 定数式です。
例	<pre>select upper("abcd") ----- ABCD</pre>
使用法	<ul style="list-style-type: none">文字列関数 <code>upper</code> は、小文字を大文字に変換して文字値を返します。<code>char_expr</code> または <code>uchar_expr</code> が <code>NULL</code> の場合、<code>upper</code> は <code>NULL</code> を返します。対応する大文字がない文字は、変換されません。作成される <code>unichar</code> 式にサロゲート・ペアのいずれか 1 つのサロゲートだけが含まれる場合は、エラー・メッセージが表示され、処理がアボートされます。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>upper</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 lower

uscalar

説明	式の最初の Unicode 文字に対する Unicode スカラ値を返します。
構文	<code>uscalar(<i>uchar_expr</i>)</code>
パラメータ	<i>uchar_expr</i> 文字型のカラム名、変数、または (<code>unichar</code> または <code>univarchar</code> 型の) 定数式です。
使用法	<ul style="list-style-type: none">文字列関数 <code>uscalar</code> は、式の最初の Unicode 文字の Unicode 値を返します。<i>uchar_expr</i> が NULL の場合は、NULL を返します。対になっていない上位または下位のサロゲートが含まれている <i>uchar_expr</i> に対して <code>uscalar</code> が呼び出されると、エラーが発生して処理がアボートされます。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>uscalar</code> 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 ascii

used_pages

説明

テーブル、インデックス、または特定のパーティションで使用されているページ数をレポートします。`data_pages`とは異なり、`used_pages`は、内部構造に使用されるページを数に含めません。この関数は、バージョン 15.0 より前の Adaptive Server で使用された古い `used_pgs` 関数を置き換えるものです。

構文

```
used_pages(dbid, object_id[, indid[, ptnid]])
```

パラメータ

dbid

ターゲット・オブジェクトが存在するデータベースの ID です。

object_id

使用ページを表示するテーブルのオブジェクト ID です。インデックスが使用するページを表示するには、そのインデックスが設定されているテーブルのオブジェクト ID を指定してください。

indid

対象のインデックス ID です。

ptnid

対象のパーティション ID です。

例

例 1 指定されたデータベースにある、オブジェクト ID が 31000114 のオブジェクトで使用されるページ数を返します (インデックスも含む)。

```
select used_pages(5, 31000114)
```

例 2 クラスタード・インデックスがあるかどうかに関係なく、データ・レイヤに含まれるオブジェクトで使用されるページ数を返します。

```
select used_pages(5, 31000114, 0)
```

例 3 インデックス ID が 2 のインデックス・レイヤにあるオブジェクトが使用しているページの数返します。これには、データ・レイヤが使用するページは含まれていません (式の「使用法」の最初の項を参照してください)。

```
select used_pages(5, 31000114, 2)
```

例 4 指定されたパーティション (この例では 2323242432) のデータ・レイヤにあるオブジェクトで使用されるページ数を返します。

```
select used_pages(5, 31000114, 0, 2323242432)
```

使用法

- クラスタード・インデックス付きの全ページ・ロック・テーブルでは、最後のパラメータの値によって、使用されたどのページが返されるかが決まります。
 - `used_pages(dbid, objid, 0)` – インデックス ID として明示的に 0 を渡し、データ・レイヤで使用されているページのみを返します。
 - `used_pages(dbid, objid, 1)` – インデックス・レイヤで使用されているページと、データ・レイヤで使用されているページを返します。

クラスタード・インデックスを持つ全ページロック・テーブルの場合、インデックス・レイヤで使用されているページを取得するには、`used_pages(dbid, objid, 1)` から `used_pages(dbid, objid, 0)` を減算します。

- リソースを消費する代わりに、`used_pages` はまだキャッシュに存在しないオブジェクトの記述子を廃棄します。
- クラスタード・インデックスを持つ全ページ・ロック・テーブルでは、`indid = 0` の値に対して `used_pages` はデータ・レイヤの使用ページ数のみを渡されます。`indid=1` を渡した場合、以前のバージョンと同じく、データ・レイヤとクラスタード・インデックス・レイヤの使用ページ数が返されます。
- `used_pages` は、以前の `used_pgs(objid, doampg, ioampg)` 関数に似ています。
- すべてのエラーで常に戻り値 0 が返されます。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザが `used_pgs` 関数を実行できます。

参照

関数 [data_pages](#), [object_id](#)

user

説明 現在のユーザの名前を返します。

構文 user

パラメータ なし。

例

```
select user
-----
dbo
```

使用法

- システム関数 **user** はユーザの名前を返します。
- **sa_role** が有効な場合は、どのデータベースを使っているか、自動的にそのデータベースの所有者になります。データベース内では、データベース所有者のユーザ名は常に“dbo”です。

標準規格 ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション すべてのユーザが **user** 関数を実行できます。

参照 マニュアル Transact-SQL ユーザーズ・ガイド

関数 [user_name](#)

user_id

説明	指定したユーザの ID、またはデータベース内の現在のユーザの ID を返します。
構文	<code>user_id([user_name])</code>
パラメータ	user_name ユーザの名前です。
例	<p>例 1</p> <pre>select user_id() ----- 1</pre> <p>例 2</p> <pre>select user_id("margaret") ----- 4</pre>
使用法	<ul style="list-style-type: none"> システム関数 user_id は、ユーザの ID 番号を返します。システム関数の一般的な情報については、『Transact-SQL ユーザーズ・ガイド』を参照してください。 user_id 関数は、現在のデータベース内の sysusers から番号を返します。user_name を指定しないと、user_id 関数は現在のユーザの ID を返します。Adaptive Server のすべてのデータベースで同じ番号であるサーバ・ユーザ ID を検索するには、suser_id 関数を使います。 データベース内では“guest”ユーザ ID は常に 2 です。 データベース内では、データベース所有者の user_id は常に 1 です。sa_role が有効な場合は、どのデータベースを使っても、自動的にそのデータベースの所有者になります。実際のユーザ ID へ戻るには、set sa_role off を実行してから user_id を実行します。データベースの有効なユーザでない場合には、set sa_role off を実行すると、Adaptive Server からエラーが返されます。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	自分以外の user_name でこの関数を使えるのは、システム管理者とシステム・セキュリティ担当者だけです。
参照	<p>コマンド setuser</p> <p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>関数 suser_id, user_name</p>

user_name

説明	指定したユーザまたは現在のユーザの、データベース内の名前を返します。
構文	<code>user_name([<i>user_id</i>])</code>
パラメータ	<i>user_id</i> ユーザの ID です。
例	例 1 <pre>select user_name() ----- dbo</pre> 例 2 <pre>select user_name(4) ----- margaret</pre>
使用法	<ul style="list-style-type: none">システム関数 <code>user_name</code> は、現在のデータベース内のユーザ ID をもとに、ユーザの名前を返します。<code>user_id</code> を指定しないと、<code>user_name</code> 関数は現在のユーザの名前を返します。<code>sa_role</code> が有効な場合は、どのデータベースを使っても、自動的にそのデータベースの所有者になります。データベース内のデータベース所有者の <code>user_name</code> は常に “dbo” です。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	自分以外の <code>user_id</code> でこの関数を使えるのは、システム管理者とシステム・セキュリティ担当者だけです。
参照	マニュアル Transact-SQL ユーザーズ・ガイド 関数 suser_name , user_id

valid_name

説明	指定された文字列が有効な識別子でない場合は、0 を返します。有効な識別子の場合は、0 以外の値を返します。文字列の最大長は 255 バイトです。
構文	<code>valid_name(character_expression[, maximum_length])</code>
パラメータ	character_expression 文字型のカラム名、変数、または (char、varchar、nvarchar、または nvarchar 型の) 定数式です。定数式は、引用符で囲んでください。 maximum_length 0 より大きく 255 以下の整数です。デフォルト値は 30 です。識別子の長さが 2 番目の引数より大きい場合に、 valid_name は 0 を返し、識別子の長さが有効な場合に 0 より大きな値を返します。
例	識別子が有効であることを検証するプロシージャを作成します。 <pre>create procedure chkname @name varchar(30) as if valid_name(@name) = 0 print "name not valid"</pre>
使用法	<ul style="list-style-type: none">システム関数 valid_name は、character_expression が有効な識別子でない (不正な文字列、30 バイトを超える文字列や予約語) 場合は 0 を返し、有効な識別子の場合は 0 以外の数を返します。Adaptive Server の識別子の最大長は、使用されている文字がシングルバイト文字でもダブルバイト文字でも、16384 バイトです。識別子の先頭文字は、現在の文字セットに定義されている文字か、アンダースコア () 文字を使用してください。この規則の例外として、シャープ記号 (#) で始まるテンポラリ・テーブル名と at 記号 (@) で始まるローカル変数名があります。識別子がシャープ記号 (#) と at 記号 (@) で始まる場合に valid_name 関数は 0 を返します。
標準規格	ANSI SQL - 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが valid_name 関数を実行できます。
参照	マニュアル Transact-SQL ユーザーズ・ガイド システム・プロシージャ sp_checkreswords

valid_user

説明	指定した ID が、Adaptive Server 上の少なくとも 1 つのデータベースで有効なユーザまたはエイリアスの場合は 1 を返し、そうでない場合は 0 を返します。
構文	<code>valid_user(server_user_id [, database_id])</code>
パラメータ	<p>server_user_id サーバ・ユーザ ID です。サーバ・ユーザ ID は <code>syslogins</code> の <code>suid</code> カラムに格納されています。</p> <p>database_id ユーザが有効かどうかを決定するデータベースの ID です。データベース ID は <code>sysdatabases</code> の <code>dbid</code> カラムに格納されています。</p>
例	<p>例 1 <code>suid</code> が 4 のユーザは少なくとも 1 つのデータベースで有効なユーザかエイリアスです。</p> <pre>select valid_user(4) ----- 1</pre> <p>例 2 <code>suid</code> が 4 のユーザは ID が 6 のデータベースで有効なユーザかエイリアスです。</p> <pre>select valid_user(4,6) ----- 1</pre>
使用法	<ul style="list-style-type: none"> 指定された <code>server_user_id</code> が指定された <code>database_id</code> で有効なユーザがエイリアスの場合、<code>valid_user</code> は 1 を返します。 <code>database_id</code> を指定しない場合またはこの値が 0 の場合、<code>valid_user</code> はユーザが少なくとも 1 つのデータベースで有効なユーザかエイリアスかを決定します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	自分以外の <code>server_user_id</code> でこの関数を使えるのは、システム管理者とシステム・セキュリティ担当者だけです。
参照	<p>マニュアル Transact-SQL ユーザーズ・ガイド</p> <p>システム・プロシージャ sp_addlogin, sp_adduser</p>

var

説明

1つの数値式で構成される標本の統計分散を `double` 型として計算し、数値セットの分散を返します。

注意 `var` と `variance` は、`var_samp` のエイリアスです。詳細については、[var_samp \(299 ページ\)](#) を参照してください。

var_pop

説明	1つの数値式で構成される母統計分散を double 型として計算します。varp は、var_pop のエイリアスで、同じ構文を使用します。
構文	var_pop ([all distinct] expression)
パラメータ	<p>all すべての値に var_pop を適用します。デフォルト設定は all です。</p> <p>distinct var_pop を適用する前に重複する値を削除します。</p> <p>expression その母集団ベースの分散がローのセットに対して計算される式です (通常はカラム名)。</p>
例	<p>pubs2 データベースにおける書籍の各種類の前払い金の平均と分散を示します。</p> <pre>select type, avg(advance) as "avg", var_pop(advance) as "variance" from titles group by type order by type</pre>
使用法	グループの各ロー (distinct が指定されている場合は、重複が削除された後に残る各ロー) に対して評価される、指定された値式の母分散を計算します。これは、値式と値式の平均の差の2乗和を、グループまたはパーティション内のローの数で割った値として定義されます。

図 2-3: 母集団に関する統計集合関数の計算式

平均が μ (var_pop) のサイズ n の母分散を定義する計算式は、次のとおりです。母標準偏差値 deviation (stddev_pop) は、この数値の正の平方根です。

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n}$$

σ^2 = 分散
 n = 母集団のサイズ
 $\mu = x_i$ 値の平均

標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが var_pop を実行できます。
参照	集合関数の一般情報については、『ASE リファレンス・マニュアル：ビルディング・ブロック』の「集合関数」を参照してください。
関数	stddev_pop, stddev_samp, var_samp

var_samp

説明	1つの数値式で構成される標本統計分散を <code>double</code> 型として計算し、数値セットの分散を返します。 <code>var</code> と <code>variance</code> は、 <code>var_samp</code> のエイリアスで、同じ構文を使用します。
構文	<code>var_samp ([all distinct] expression)</code>
パラメータ	<p><code>all</code> すべての値に <code>var_samp</code> を適用します。デフォルト設定は <code>all</code> です。</p> <p><code>distinct</code> <code>var_samp</code> を適用する前に重複する値を削除します。</p> <p><code>expression</code> 任意の numeric データ型 (<code>float</code>、<code>real</code>、または <code>double</code>) の式です。</p>
例	<p><code>pubs2</code> データベースにおける書籍の各種類の前払い金の平均と分散を示します。</p> <pre>select type, avg(advance) as "avg", var_samp(advance) as "variance" from titles where total_sales > 2000 group by type order by type</pre>
使用法	<code>var_samp</code> は、結果を倍精度浮動小数点数のデータ型で返します。空のセットが指定された場合の結果は <code>NULL</code> になります。

図 2-4: 標本に関する統計集合関数の計算式

平均 \bar{x} (`var_samp`) を持つサイズ n の標本からの母分散の不偏推定値を定義する計算式は、次のとおりです。標本標準偏差 (`stddev_samp`) は、この数値の正の平方根です。

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

$s^2 =$ 分散
 $n =$ 標本サイズ
 $\bar{x} = x_i$ 値の平均

標準規格	ANSI SQL - 準拠レベル: Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>var_samp</code> を実行できます。
参照	集合関数の一般情報については、『ASE リファレンス・マニュアル: ビルディング・ブロック』の「集合関数」を参照してください。
関数	<code>stddev_pop</code> , <code>stddev_samp</code> , <code>var_pop</code>

variance

説明

1つの数値式で構成される標本の統計分散を `double` 型として計算し、数値セットの分散を返します。

注意 `var` と `variance` は、`var_samp` のエイリアスです。詳細については、[var_samp \(299 ページ\)](#) を参照してください。

varp

説明

1 つの数値式で構成される母統計分散を `double` 型として計算します。

注意 `varp` は、`var_pop` のエイリアスです。詳細については、[var_pop \(298 ページ\)](#) を参照してください。

workload_metric

説明	クラスタ環境のみ – 指定したインスタンスの現在の負荷測定基準のクエリを実行するか、指定したインスタンスの測定基準を更新します。
構文	<code>workload_metric(instance_id instance_name [, new_value])</code>
パラメータ	instance_id インスタンスの ID です。 instance_name インスタンスの名前です。 new_value 新しい測定基準を表す浮動小数値です。
例	例 1 現在のインスタンスのユーザ測定基準を見ます。 <pre>select workload_metric()</pre> 例 2 インスタンス “ase2” のユーザ測定基準を見ます。 <pre>select workload_metric("ase2")</pre> 例 3 “ase3” から 27.54 までのユーザ測定基準の値を設定します。 <pre>select workload_metric("ase3", 27.54)</pre>
使用法	<ul style="list-style-type: none">• NULL 値は現在のインスタンスを示します。• new_value に値を指定した場合は、その値が現在のユーザ測定基準になります。new_value に値を指定しなかった場合は、現在の負荷測定基準が返されます。• new_value の値は 0 以上でなければなりません。• new_value に値を指定すると、<code>workload_metric</code> は処理に成功した場合にその値を返します。それ以外の場合、<code>workload_metric</code> は -1 を返します。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	<code>workload_metric</code> を実行するには、 <code>sa_role</code> または <code>ha_role</code> が必要です。

xa_bqual

説明 ASCII XA トランザクション ID の `bqual` コンポーネントのバイナリ・バージョンを返します。

構文 `xa_bqual(xactname, 0)`

パラメータ `xid`
`systransactions` の `xactname` カラム、または `sp_transactions` で取得される Adaptive Server トランザクションの ID です。

0
 今後のために予約済みです。

例 **例 1** Adaptive Server トランザクション ID
 “0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0” の分岐修飾子をバイナリ変換した “0x227f06ca80” を返します。Adaptive Server トランザクション ID は、最初は `sp_transactions` を使用して次のように取得されます。

```
1> sp_transactions

xactkey          type      coordinator starttime      state
connection dbid  spid  loid  failover  srvname  namelen  xactname
-----
-----
0x531600000600000017e4885b0700 External XA      Dec 9 2005  5:15PM In
Command Attached    7    20    877 Resident Tx  NULL      39
0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0

1> select xa_bqual("0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0", 0)
2> go

...
-----
0x227f06ca80
```

例 2 `xa_bqual` は、`xa_gtrid` と一緒に使用されることが多くあります。この例は、`systransactions` のすべてのローからのグローバル・トランザクション ID と分岐修飾子を返します。`coordinator` カラムの値は “3” です。

```
1> select gtrid=xa_gtrid(xactname,0),
        bqual=xa_bqual(xactname,0)
        from systransactions where coordinator = 3
2> go

gtrid

bqual

-----
0xb1946cdc52464a61cba42fe4e0f5232b
0x227f06ca80
```

使用法

Adaptive Server で外部トランザクションがブロックされており、ブロックされているトランザクションを `sp_lock` および `sp_transactions` を使用して識別する場合、XA トランザクション・マネージャを使用してグローバル・トランザクションを終了させることができます。ただし、`sp_transactions` を実行する場合、返される `xactname` の値は ASCII 文字列フォーマットです。一方、XA Server は、復号化されていないバイナリ値を使用します。`xa_bqual` を使用すると、XA トランザクション・マネージャが解読できる形式のトランザクション名の `bqual` 部分を特定できます。

`xa_bqual` は、以下を返します。

- この文字列の変換バージョンで、2 番目の “_” (アンダースコア) より後、3 番目の “_” または end-of-string 値のうち先に出現した方の前にあります。
- トランザクション ID が復号化できないか予想外の形式の場合は NULL。

注意 `xa_bqual` は、`xid` に対して検証チェックを実行せず、単に変換された文字列を返します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザは、`xa_bqual` を使用できます。

参照

関数 [xa_gtrid](#)

ストアド・プロシージャ `sp_lock`、`sp_transactions`

xa_gtrid

説明 ASCII XA トランザクション ID の `gtrid` コンポーネントのバイナリ・バージョンを返します。

構文 `xa_gtrid(xactname, int)`

パラメータ `xid`
`systransactions` の `xactname` カラム、または `sp_transactions` で取得される Adaptive Server トランザクションの ID です。

0
 今後のために予約済みです。

例 1 一般的な状況では、Adaptive Server トランザクション ID “0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0” に対して “0x227f06ca80” (分岐修飾子のバイナリ変換)、“0xb1946cdc52464a61cba42fe4e0f5232b” (グローバル・トランザクション ID) が返されます。

```
1> select xa_gtrid("0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0", 0)
2> go
```

...

```
-----
0xb1946cdc52464a61cba42fe4e0f5232b
```

(1 row affected)

例 2 `xa_bqual` は、`xa_gtrid` と一緒に使用されることが多くあります。この例は、`systransactions` のすべてのローからのグローバル・トランザクション ID と分岐修飾子を返します。`coordinator` カラムの値は “3” です。

```
1> select gtrid=xa_gtrid(xactname,0),
        bqual=xa_bqual(xactname,0)
        from systransactions where coordinator = 3
2> go
```

gtrid

bqual

```
-----
0xb1946cdc52464a61cba42fe4e0f5232b
```

```
0x227f06ca80
```

使用法

Adaptive Server で外部トランザクションがブロックされており、ブロックされているトランザクションを `sp_lock` および `sp_transactions` を使用して識別する場合、XA トランザクション・マネージャを使用してグローバル・トランザクションを終了させることができます。ただし、`sp_transactions` を実行する場合、返される `xactname` の値は ASCII 文字列フォーマットです。一方、XA Server は、復号化されていないバイナリ値を使用します。`xa_gtrid` を使用することにより、XA トランザクション・マネージャが解読できる形式のトランザクション名の `gtrid` 部分を特定できます。

`xa_gtrid` は、以下を返します。

- 最初の “_” (アンダースコア) より後にあり、2 番目の “_” または end-of-string 値のうち先に出現した方の前にある、この文字列の変換バージョン。
- トランザクション ID が復号化できないか予想外の形式の場合は NULL。

注意 `xa_gtrid` は、`xid` に対して検証チェックを実行せず、単に変換された文字列を返します。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。

パーミッション

すべてのユーザは、`xa_gtrid` を使用できます。

参照

関数 [xa_bqual](#)

ストアド・プロシージャ `sp_lock`、`sp_transactions`

xact_connmigrate_check

説明	クラスタ環境のみ – 現在の接続が外部トランザクションを処理できるかどうかを確認できます。
構文	<code>xact_connmigrate_check("txn_name")</code>
パラメータ	txn_name トランザクション ID です。このパラメータはオプションです。
例	<p>例 1 インスタンス “ase1” で実行する XA トランザクション “txn_name”。</p> <pre>select xact_connmigrate_check("txn_name") ----- 1</pre> <p>例 2 インスタンス “ase2” で実行する XA トランザクション “txn_name”。接続はマイグレートできます。</p> <pre>select xact_connmigrate_check("txn_name") ----- 1</pre> <p>例 3 インスタンス “ase2” で実行する XA トランザクション “txn_name”。接続はマイグレートできません。</p> <pre>select xact_connmigrate_check("txn_name") ----- 0</pre>
使用法	<ul style="list-style-type: none"> • XID を指定した場合、<code>xact_connmigrate_check</code> は以下を返します。 <ul style="list-style-type: none"> • 指定したトランザクションを実行するインスタンスへの接続の場合、またはマイグレートできる別のインスタンスへの接続の場合は 1。 • 接続またはトランザクション ID が存在しない場合、またはマイグレートできない別のインスタンスへの接続の場合は 0。 • XID を指定しない場合、<code>xact_connmigrate_check</code> は以下を返します。 <ul style="list-style-type: none"> • 接続がマイグレートできる場合は、1。 • 接続が存在しないか、またはマイグレートできない場合は 0。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが、 <code>xact_connmigrate_check</code> を実行できます。
参照	関数 xact_owner_instance

xact_owner_instance

説明	クラスタ環境のみ – 分散トランザクションを実行しているインスタンスの ID を返します。
構文	<code>xact_owner_instance("txn_name")</code>
パラメータ	<code>txn_name</code> トランザクション ID です。
例	<p>例 1 インスタンス “ase1” で実行する XA トランザクション “txn_name”。</p> <pre>select xact_owner_instance(txn_name) ----- 1</pre> <p>例 2 XA トランザクション “txn_name” は実行していません。</p> <pre>select xact_owner_instance(txn_name) ----- NULL</pre>
使用法	<ul style="list-style-type: none">• <code>xact_owner_instance</code> は以下を返します。<ul style="list-style-type: none">• トランザクションを実行するインスタンスのインスタンス ID、または• トランザクションが実行していない場合は Null。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>xact_owner_instance</code> を実行できます。
参照	関数 xact_connmigrate_check

xmlextract

- 説明** XML ドキュメントに XML クエリ式を適用し、指定された結果を返します。XML タグ付きまたはタグなしの情報を返すことができます。
- 参照** `xmlextract` およびデータベースで XML をサポートする他のすべての Transact-SQL 関数の構文、例、使用方法については、XML サービスを参照してください。

xmlparse

説明

パラメータとして渡される XML ドキュメントを解析し、ドキュメントの解析済み形式を含む **image** (デフォルト)、**binary**、または **varbinary** 値を返します。

参照

xmlparse およびデータベースで XML をサポートする他のすべての Transact-SQL 関数の構文、例、使用方法については、XML サービスを参照してください。

xmlrepresentation

- 説明** 式の `image` パラメータを調べて、パラメータに解析済み XML データが含まれるか、他の種類の `image` データが含まれるかを示す整数値を返します。
- 参照** `xmlrepresentation` およびデータベースで XML をサポートする他のすべての Transact-SQL 関数の構文、例、使用方法については、XML サービスを参照してください。

xmtable

説明

XML ドキュメントからデータを抽出し、SQL テーブルとして返します。

参照

xmtable およびデータベースで XML をサポートする他のすべての Transact-SQL 関数の構文、例、使用方法については、XML サービスを参照してください。

xmltest

説明

XML クエリ式を評価する SQL 述部です。XML ドキュメント・パラメータを参照でき、ブール値の結果を返します。xmltest は SQL like 述部に似ています。

参照

xmltest およびデータベースで XML をサポートする他のすべての Transact-SQL 関数の構文、例、使用方法については、XML サービスを参照してください。

xmlvalidate

説明

XML ドキュメントを検証します。

参照

xmlvalidate およびデータベースで XML をサポートする他のすべての Transact-SQL 関数の構文、例、使用方法については、XML サービスを参照してください。

year

説明	指定した日付の <code>datepart</code> の年を表す整数を返します。
構文	<code>year(date_expression)</code>
パラメータ	<code>date_expression</code> datetime、smalldatetime、date、または time 型の式、または datetime フォーマットの文字列です。
例	整数 03 を返します。 <pre>year("11/02/03") ----- 03 (1 row(s) affected)</pre>
使用法	<code>year(date_expression)</code> は <code>datepart(yy, date_expression)</code> と同じです。
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能です。
パーミッション	すべてのユーザが <code>year</code> 関数を実行できます。
参照	データ型 datetime、smalldatetime、date 関数 datepart 、 day 、 month

トピック名	ページ
Adaptive Server のグローバル変数	317

Adaptive Server のグローバル変数

グローバル変数は、システムの実行中に Adaptive Server によって継続的に更新されるシステム定義変数です。一部のグローバル変数はセッション固有の変数で、それ以外はサーバ・インスタンス固有の変数です。たとえば、`@@error` には、特定のユーザ接続に対してシステムによって生成された最新のエラー番号が保管されます。

アプリケーション・コンテキストの変数を指定するには、[get_appcontext](#) および [set_appcontext](#) を参照してください。

グローバル変数の値を表示するには、次のように入力します。

```
select variable_name
```

次に例を示します。

```
select @@char_convert
```

最後に Adaptive Server が起動してから発生したシステム・アクティビティについて報告するグローバル変数が多数あります。`sp_monitor` は、いくつかのグローバル変数の現在値を表示します。

表 3-1 に、Adaptive Server で使用できるグローバル変数をリストします。

表 3-1: Adaptive Server のグローバル変数

グローバル変数	定義
<code>@@active_instances</code>	クラスタ内のアクティブなインスタンスの数を返す。
<code>@@authmech</code>	ユーザの認証に使用するメカニズムを指定する読み込み専用変数。
<code>@@bootcount</code>	Adaptive Server が起動された回数を返す。
<code>@@boottime</code>	Adaptive Server が最後に起動された日付と時刻を返す。
<code>@@bulkarraysize</code>	バルク・コピー・インタフェースを使用して転送される前にローカル・サーバ・メモリにバッファリングされるローの数を返す。コンポーネント統合サービスで <code>select into</code> を使ってリモート・サーバにローを転送する場合にのみ使用する。『コンポーネント統合サービス・ユーザズ・ガイド』を参照してください。

グローバル変数	定義
@@bulkbatchsize	バルク・インタフェースを使用して <code>select into proxy_table</code> によってリモート・サーバに転送されるローの数を返す。コンポーネント統合サービスで <code>select into</code> を使ってリモート・サーバにローを転送する場合にのみ使用する。『コンポーネント統合サービス・ユーザーズ・ガイド』を参照してください。
@@char_convert	文字セット変換が有効でない場合は 0 を返す。文字セット変換が有効な場合は 1 を返す。
@@cis_rpc_handling	<code>cis rpc handling</code> がオフの場合は 0 を返す。 <code>cis rpc handling</code> がオンの場合は 1 を返す。『コンポーネント統合サービス・ユーザーズ・ガイド』を参照してください。
@@cis_version	コンポーネント統合サービスの日付とバージョンを返す。
@@client_csexpansion	サーバの文字セットをクライアントの文字セットに変換するときに使用する拡張係数を返す。たとえば、値が 2 の場合、サーバの文字セットの文字は、クライアントの文字セットに変換後、最大 2 倍までのバイト数を使用できる。
@@client_csid	クライアントの文字セットが初期化されていない場合は -1 を返す。クライアントの文字セットが初期化されている場合は <code>syscharsets</code> から接続用のクライアントの文字セット ID を返す。
@@client_csname	クライアントの文字セットが初期化されていない場合は、NULL を返す。クライアントの文字セットが初期化されている場合は、接続の文字セットの名前を返す。
@@clusterboottime	クラスタの起動を最初に開始したインスタンスが停止している場合でも、クラスタが最初に起動した日時を返す。
@@clustercoorid	現在のクラスタ・コーディネータのインスタンス ID を返す。
@@clustermode	文字列 “shared-disk cluster” を返す。
@@clustername	クラスタの名前を返す。
@@cmpstate	高可用性環境における Adaptive Server の現在のモードを返す。高可用性環境以外では使用されない。
@@connections	ユーザ・ログインが試行された回数を返す。
@@cpu_busy	Adaptive Server が最後に起動してから、CPU が Adaptive Server の作業に費やした時間 (チック単位) を返す。
@@cursor_rows	スクロール可能カーソルのために設計されたグローバル変数。カーソル結果セットに含まれるローの総数を示す。次の値が返される。 <ul style="list-style-type: none"> • -1 - カーソルが次のような場合： <ul style="list-style-type: none"> • 動的 - 動的カーソルにはすべての変更が反映されるため、カーソルを構成するローの数は常に変動する。構成するローがすべて取得されたかどうかは確認できない。 • <code>semi_sensitive</code> であり、スクロール可能だが、スクロールのワーク・テーブルにまだ値が完全には格納されていない - カーソルを構成するロー数は、この値を取得した時点では不明となる。 • 0 - カーソルが開いていない、最後に開いたカーソルを構成するローがない、または最後に開いたカーソルが閉じられたか割り付け解除された。 • <i>n</i> - 最後に開いた、またはフェッチしたカーソル結果セットに値が完全に格納された。返される値は、カーソル結果セット内のローの総数である。
@@curlid	現在のセッションのロック所有者 ID を返す。

グローバル変数	定義
<code>@@datefirst</code>	set datefirst <i>n</i> を使用して設定する。ここで <i>n</i> は 1 ~ 7 の値を示す。 <code>@@datefirst</code> で表される各週の指定された最初の曜日を示す tinyint の現在の値を返す。 Active Server のデフォルト値は日曜日 (us_language のデフォルトに基づく) で、set datefirst 7 を指定して設定する。設定と値の詳細については、set コマンドの datefirst オプションを参照。
<code>@@dbts</code>	現在のデータベースのタイムスタンプを返す。
<code>@@error</code>	システムによって生成された最新のエラー番号を返す。
<code>@@errorlog</code>	Adaptive Server エラー・ログが置かれているディレクトリを、\$SYBASE ディレクトリ (NT では %SYBASE%) の相対パスとしてフル・パスで返す。
<code>@@failedoverconn</code>	プライマリ・コンパニオンへの接続がフェールオーバーして、セカンダリ・コンパニオン・サーバが実行中の場合は、0 より大きい値を返す。この変数は高可用性環境でのみ使用し、セッション固有のものである。
<code>@@fetch_status</code>	戻り値： <ul style="list-style-type: none"> • 0 - フェッチが成功した。 • -1 - フェッチが成功しなかった。 • -2 - 今後のために予約済み。
<code>@@guestuserid</code>	guest ユーザの ID を返す。
<code>@@hacmpservername</code>	高可用性設定のコンパニオン・サーバの名前を返す。
<code>@@haconnection</code>	接続でフェールオーバー・プロパティが有効になっている場合は、0 より大きい値を返す。これはセッション固有のプロパティである。
<code>@@heapmemsize</code>	ヒープ・メモリ・プールのサイズ (バイト単位) を返す。ヒープ・メモリの詳細については、『システム管理ガイド』を参照。
<code>@@identity</code>	生成された最新の IDENTITY カラム値を返す。
<code>@@idle</code>	Adaptive Server が最後に起動してからアイドル状態になっていた時間 (チック単位) を返す。
<code>@@instanceid</code>	実行されたインスタンスの ID を返す。
<code>@@instancename</code>	実行されたインスタンスの名前を返す。
<code>@@invaliduserid</code>	無効なユーザ ID に対して -1 の値を返す。
<code>@@io_busy</code>	Adaptive Server が I/O 動作に費やした時間 (チック単位) を返す。
<code>@@isolation</code>	現在の Transact-SQL プログラムのセッション固有の独立性レベルの値 (0, 1, または 3) を返す。
<code>@@jsinstanceid</code>	Job Scheduler が実行されている、または有効化後に実行されるインスタンスの ID。
<code>@@kernel_addr</code>	カーネル領域を含む最初の共有メモリ領域の開始アドレスを返す。結果は 0xaddress pointer value のフォームになる。
<code>@@kernel_size</code>	最初の共有メモリ領域の一部であるカーネル領域のサイズを返す。
<code>@@kernelmode</code>	Adaptive Server が設定されているモード (スレッドまたはプロセス) を返す。
<code>@@langid</code>	syslanguages.langid に指定されている、現在使用中の言語のサーバワイドの言語 ID を返す。
<code>@@language</code>	syslanguages.name に指定されている、使用中の言語名を返す。
<code>@@lastkpgendate</code>	sp_passwordpolicy の "keypair regeneration period" ポリシー・オプションに従って、最後のキー・ペアが生成された日時を返す。

グローバル変数	定義
<code>@@lastlogindate</code>	各ユーザ・ログイン・セッションで使用可能。 <code>@@lastlogindate</code> には <code>datetime</code> データ型が入り、その値は、現在のセッションが確立される前のそのログイン・アカウントの <code>lastlogindate</code> カラムとなる。この変数は各ログイン・セッションに特有のもので、そのアカウントへの前回のログイン日時を知るために各セッションで使用できる。以前に使用されたことのないアカウントの場合、または “ <code>sp_passwordpolicy 'set', enable last login updates</code> ” が 0 に設定されている場合、 <code>@@lastlogindate</code> の値は NULL となる。
<code>@@lock_timeout</code>	<code>set lock wait n</code> を使用して設定する。現在の <code>lock_timeout</code> 設定 (ミリ秒) を返す。 <code>@@lock_timeout</code> は n の値を返す。デフォルト値はタイムアウトなし。セッションの開始時に <code>set lock wait n</code> が実行されない場合、 <code>@@lock_timeout</code> は -1 を返す。
<code>@@lwpid</code>	最後から 2 番目に実行されたライトウェイト・プロシージャのオブジェクト ID を返す。
<code>@@max_connections</code>	現在のコンピュータ環境の Adaptive Server に対して、同時に確立できる接続の最大数を返す。 <code>number of user connections</code> 設定パラメータを使用して、 <code>@@max_connections</code> の値以下の任意の接続数に対応するように、Adaptive Server を設定できる。
<code>@@max_precision</code>	サーバによって設定された <code>decimal</code> と <code>numeric</code> データ型で使用する精度レベルを返す。この値は定数 38。
<code>@@maxcharlen</code>	Adaptive Server のデフォルト文字セット中の 1 文字の最大長 (バイト単位) を返す。
<code>@@maxgroupid</code>	最大のグループ・ユーザ ID を返す。最大値は 1048576。
<code>@@maxpagesize</code>	サーバの論理ページ・サイズを返す。
<code>@@maxspid</code>	<code>spid</code> の最大有効値を返す。
<code>@@maxsuid</code>	最大のサーバ・ユーザ ID を返す。デフォルト値は 2147483647。
<code>@@maxuserid</code>	最大のユーザ ID を返す。最大値は 2147483647。
<code>@@mempool_addr</code>	グローバル・メモリ・プールのテーブルのアドレスを返す。結果は <code>0xaddress pointer value</code> のフォームになる。この変数は内部で使用される。
<code>@@min_poolsize</code>	名前付きキャッシュ・プールの最小サイズ (キロバイト) を返す。 <code>DEFAULT_POOL_SIZE (256)</code> と <code>max database page size</code> の現在値に基づいて計算される。
<code>@@mingroupid</code>	最小のグループ・ユーザ ID を返す。最小値は 16384。
<code>@@minspid</code>	<code>spid</code> の最小値である 1 を返す。
<code>@@minsuid</code>	最小のサーバ・ユーザ ID を返す。最小値は -32768。
<code>@@minuserid</code>	最小のユーザ ID を返す。最小値は -32768。
<code>@@monitors_active</code>	<code>sp_sysmon</code> によって表示されるメッセージ数を減らす。
<code>@@ncharsize</code>	現在のサーバのデフォルト文字セット中の 1 文字の最大長 (バイト単位) を返す。
<code>@@nestlevel</code>	現在のネスト・レベルを返す。
<code>@@nextkpgendate</code>	<code>sp_passwordpolicy</code> の “ <code>keypair regeneration period</code> ” ポリシー・オプションに従って、次のキー・ペアの生成予定日時を返す。
<code>@@nodeid</code>	現在のインストール環境の 48 ビットのノード識別子を返す。Adaptive Server は、マスタ・デバイスを最初に使用するときノード ID を生成し、Adaptive Server インストール環境をユニークに識別する。
<code>@@optgoal</code>	クエリの最適化に使用される現在の <code>optimization goal</code> 設定を返す。
<code>@@options</code>	セッションの <code>set</code> オプションの 16 進表現を返す。
<code>@@optlevel</code>	現在の最適化レベルの設定を返す。
<code>@@opttimeoutlimit</code>	クエリの最適化に使用される現在の <code>optimization timeout limit</code> 設定を返す。
<code>@@ospid</code>	(スレッド・モードのみ) サーバのオペレーティング・システム ID を返す。
<code>@@pack_received</code>	Adaptive Server が読み込んだ入力パケット数を返す。

グローバル変数	定義
<code>@@pack_sent</code>	Adaptive Server が書き込みを行った出力パケット数を返す。
<code>@@packet_errors</code>	パケットの読み込みおよび書き込み中に Adaptive Server が検出したエラーの数を返す。
<code>@@pagesize</code>	サーバの仮想ページ・サイズを返す。
<code>@@parallel_degree</code>	現在の最大並列度の設定を返す。
<code>@@plwpid</code>	最も最近に準備されたライトウェイト・プロシージャのオブジェクト ID を返す。
<code>@@probesuid</code>	プローブ・ユーザ ID に対して 2 の値を返す。
<code>@@procid</code>	現在実行中のプロシージャのストアド・プロシージャ ID を返す。
<code>@@quorum_physname</code>	クォーラム・デバイスの物理パスを返す。
<code>@@recovery_state</code>	次のいずれかの値を返し、これらに基づいて Adaptive Server がリカバリ中かどうかを示す。 <ul style="list-style-type: none"> NOT_IN_RECOVERY – Adaptive Server は起動時のリカバリ中またはフェールオーバー・リカバリ中ではない。リカバリは完了しており、オンラインにできるすべてのデータベースがオンラインになっている。 RECOVERY_TUNING – Adaptive Server はリカバリ中 (起動時またはフェールオーバー) であり、最適なりカバリ・タスク数をチューニング中である。 BOOTIME_RECOVERY – Adaptive Server は起動時のリカバリ中で、最適なタスク数のチューニングを完了した。すべてのデータベースがリカバリされたわけではない。 FAILOVER_RECOVER – Adaptive Server は HA フェールオーバー時にリカバリ中であり、最適なりカバリ・タスク数のチューニングを完了した。すべてのデータベースがオンラインになっているわけではない。
<code>@@remotestate</code>	高可用性環境におけるプライマリ・コンパニオンの現在のモードを返す。返される値については、『高可用性システムにおける Sybase フェールオーバーの使用』を参照。
<code>@@repartition_degree</code>	現在の動的な <code>repartitioning degree</code> 設定を返す。
<code>@@resource_granularity</code>	クエリの最適化に使用される最大リソース使用量ヒントの設定を返す。
<code>@@rowcount</code>	最後のクエリによる影響を受けたローの数を返す。 <code>@@rowcount</code> の値は、指定されたカーソルが前方スクロールのみか、スクロール可能であるかで異なる。 カーソルがデフォルトの非スクロール可能カーソルであれば、 <code>@@rowcount</code> の値は、結果セットにフェッチされたロー数に等しくなるまで、前方のみに 1 つずつ増える。これらのローは、基本となるテーブルからクライアントへフェッチされる。 <code>@@rowcount</code> の最大値は、結果セットに含まれるロー数に等しい。 デフォルト・カーソルでは、ローを返さないかローに影響を与えないコマンドが実行されると、 <code>@@rowcount</code> が 0 に設定される。たとえば、 <code>if</code> コマンドや <code>set</code> コマンド、または <code>update</code> 文や <code>delete</code> 文は、ローに影響を与えない。 スクロール可能カーソルの場合、 <code>@@rowcount</code> に最大値はない。値は <code>fetch</code> のたびに方向に関係なく増加し、最大値はない。スクロール可能カーソルの <code>@@rowcount</code> 値には、基本となるテーブルからではなく、結果セットからクライアントにフェッチされたローの数が反映される。
<code>@@scan_parallel_degree</code>	ノンクラスタード・インデックス・スキャンの現在の最大並列度の設定を返す。
<code>@@servername</code>	Adaptive Server の名前を返す。
<code>@@setrowcount</code>	<code>set rowcount</code> の現在の値を返す。

グローバル変数	定義
<code>@@shmem_flags</code>	共有メモリ領域のプロパティを返す。この変数は内部で使用される。合計 13 の異なるプロパティ値があり、それぞれ整数の 13 ビットに対応している。有効な値は、低いビットから順に、MR_SHARED、MR_SPECIAL、MR_PRIVATE、MR_READABLE、MR_WRITABLE、MR_EXECUTABLE、MR_HWCOHERENCY、MR_SWCOHERENC、MR_EXACT、MR_BEST、MR_NAIL、MR_PSUEDO、MR_ZERO である。
<code>@@spid</code>	現在のプロセスのサーバ・プロセス ID を返す。
<code>@@sqlstatus</code>	<code>fetch</code> 文の実行結果であるステータス情報 (例外の警告) を返す。
<code>@@ssl_ciphersuite</code>	現在の接続で SSL が使用されていない場合に NULL を返し、それ以外の場合に現在の接続での SSL ハンドシェイクに選択された暗号スイートの名前を返す。
<code>@@stringsize</code>	<code>toString()</code> メソッドから返される文字データの量を返す。デフォルトは 50。最大値は 2GB。0 の値を指定すると、デフォルト値となる。詳細については、『コンポーネント統合サービス・ユーザーズ・ガイド』を参照。
<code>@@sys_tempdbid</code>	実行中のインスタンスの有効なローカル・システム・テンポラリ・データベースのデータベース ID を返す。
<code>@@system_busy</code>	Adaptive Server でシステム・タスクが実行されていた時間 (チック単位)。 ¹
<code>@@system_view</code>	セッション固有のシステム・ビュー設定として “instance” または “cluster” を返す。
<code>@@tempdbid</code>	セッションに割り当てられたテンポラリ・データベースの有効なテンポラリ・データベース ID (dbid) を返す。
<code>@@textcolid</code>	<code>@@textptr</code> が参照するカラムのカラム ID を返す。
<code>@@textdataptmid</code>	<code>@@textptr</code> によって参照されるカラムが含まれるテキスト・パーティションのパーティション ID を返す。
<code>@@textdbid</code>	<code>@@textptr</code> が参照するカラムを持つオブジェクトを格納しているデータベース ID を返す。
<code>@@textobjid</code>	<code>@@textptr</code> が参照するカラムを持つオブジェクトのオブジェクト ID を返す。
<code>@@textptmid</code>	<code>@@textptr</code> から参照されるカラムが含まれているデータ・パーティションのパーティション ID を返す。
<code>@@textptr</code>	プロセスが最後に挿入または更新した、 <code>text</code> 、 <code>unitext</code> 、または <code>image</code> カラムのテキスト・ポインタを返す (<code>textptr</code> 関数とは異なる)。
<code>@@textptr_parameters</code>	<code>textptr_parameters</code> 設定パラメータの現在のステータスがオフの場合は 0 を返す。 <code>textptr_parameters</code> の現在のステータスがオンの場合は 1 を返す。詳細については、『コンポーネント統合サービス・ユーザーズ・ガイド』を参照。
<code>@@textsize</code>	<code>select</code> が返す <code>text</code> 、 <code>unitext</code> 、または <code>image</code> データのバイト数に対する制限を返す。isql のデフォルトの制限値は 32K バイト。デフォルト値は、クライアント・ソフトウェアによって異なる。デフォルトごとに <code>set textsize</code> を使用して変更できる。
<code>@@textts</code>	<code>@@textptr</code> が参照するカラムのテキスト・タイムスタンプを返す。
<code>@@thresh_hysteresis</code>	スレッシュホールドをアクティブ化するために必要な空き領域の減少量を返す。ヒステリシス値と呼ばれるこの値は、2 KB のデータベース・ページ単位で測定される。これは、データベース・セグメント上にスレッシュホールドをどのくらい接近させて配置できるかを判別する。
<code>@@timeticks</code>	チックごとのマイクロ秒数を返す。1 チックの長さはマシンによって異なる。
<code>@@total_errors</code>	読み込みおよび書き込み中に Adaptive Server が検出したエラーの数を返す。
<code>@@total_read</code>	Adaptive Server によるディスク読み込み回数を返す。
<code>@@total_write</code>	Adaptive Server によるディスク書き込み回数を返す。

グローバル変数	定義
<code>@@tranchained</code>	Transact-SQL プログラムの現在のトランザクション・モードが非連鎖モードの場合は、0 を返す。Transact-SQL プログラムの現在のトランザクション・モードが連鎖モードである場合は 1 を返す。
<code>@@trancount</code>	現在のユーザ・セッションにおけるトランザクションのネスト・レベルを返す。
<code>@@transactional_rpc</code>	リモート・サーバへの RPC がトランザクション指向の場合は 0 を返す。リモート・サーバへの RPC が非トランザクション指向の場合は 1 を返す。詳細については、『リファレンス・マニュアル』の <code>enable xact coordination</code> および <code>set option transactional_rpc</code> を参照。また、『コンポーネント統合サービス・ユーザーズ・ガイド』も参照。
<code>@@transtate</code>	現在のユーザ・セッションで文を実行した後のトランザクションの現在の状態を返す。
<code>@@unicharsize</code>	<code>unichar</code> の文字サイズである 2 を返す。
<code>@@user_busy</code>	Adaptive Server でユーザ・タスクが実行されていた時間 (チック単位)。 ¹
<code>@@version</code>	Adaptive Server の現在のリリースの日付、バージョン文字列などを返す。
<code>@@version_as_integer</code>	「Adaptive Server の現在のリリースの最新のアップグレード・バージョン番号を整数で返す。たとえば、Adaptive Server バージョン 12.5、12.5.0.3、または 12.5.1 を実行している場合、 <code>@@version_as_integer</code> は 12500 を返す。
<code>@@version_number</code>	Adaptive Server の現在のリリースのバージョン全体を整数として返す。

¹`@@user_busy` と `@@system_busy` の値の合計は、`@@cpu_busy` の値と一致します。

クラスタ環境でのグローバル変数の使用

`@@servername` を使用する場合、Cluster Edition ではインスタンス名ではなくクラスタの名前が返されます。インスタンスの名前を返すには、`@@instancename` を使用します。

ノンクラスタの Adaptive Server では、レコードが挿入されるたびに `@@identity` の値が変わります。挿入された最新のレコードに、IDENTITY プロパティが指定されたカラムが入っている場合、`@@identity` はこのカラムの値に設定されます。そうでない場合は、“0” (無効な値) に設定されます。この変数がセッション固有である場合は、このセッションで最後に行われた挿入に基づいて値が設定されます。

クラスタ環境では、テーブルに対して複数のノードで挿入が実行されるので、セッション固有の動作は `@@identity` に保持されません。クラスタ環境では、`@@identity` の値は、クラスタに最後に挿入されたレコードではなく、現在のセッションでノードに最後に挿入されたレコードによって決まります。

この章では、Transact-SQL の式、有効な識別子、ワイルドカード文字について説明します。

この章では、次の項目について説明します。

トピック名	ページ
式	325
識別子	335
ワイルドカード文字によるパターン一致	344

式

式は、1つまたは複数の定数、リテラル、関数、カラム識別子、変数またはその組み合わせを演算子で区切ったもので、式は1つの値を返します。式には、「算術式」、「関係式」、「論理式(またはブール式)」、および「文字列式」など、いくつかの種類があります。Transact-SQL 句には、式の中にサブクエリを使用できるものがあります。式では、**case** 式を使用できます。

表 4-1 は、Adaptive Server の構文で使用できる各種の式のリストを示します。

表 4-1: 構文で使用されている式の種類

使用法	定義
<i>expression</i>	定数、リテラル、関数、カラム識別子、変数、パラメータが含まれる。
<i>logical expression</i>	TRUE (真)、FALSE (偽)、UNKNOWN (未知または認識できない) のいずれかを返す。
<i>constant expression</i>	常に同じ値を返す。たとえば "5+3" または "ABCDE" など。
<i>float_expr</i>	任意の浮動小数の式、または暗黙的に浮動小数の値に変換される式。
<i>integer_expr</i>	任意の整数式、または暗黙的に整数の値に変換される式。
<i>numeric_expr</i>	1つの値を返す数値式。
<i>char_expr</i>	1つの文字値を返す式。
<i>binary_expression</i>	1つの binary 値または varbinary 値を返す式。

式のサイズ

バイナリ・データまたは文字データを返す式の長さは、最大 16384 バイトです。しかし、以前のバージョンの Adaptive Server では、式の長さは最大で 255 バイトまででした。以前のリリースの Adaptive Server からアップグレードしたときに、ストアド・プロシージャまたはスクリプトで 255 バイトまでの結果文字列が格納されている場合は、残りの部分はトランケートされます。式の追加された長さに対応するには、これらのストアド・プロシージャとスクリプトの書き換えが必要になる場合があります。

算術式と文字式

算術式および文字式の一般的なパターンを次に示します。

```
{constant | column_name | function | (subquery)
 | (case_expression)}
  [{arithmetic_operator | bitwise_operator |
   string_operator | comparison_operator }
 {constant | column_name | function | (subquery)
 | case_expression}]...
```

関係式と論理式

論理式または関係式は、TRUE (真)、FALSE (偽)、UNKNOWN (未知または認識できない) のいずれかを返します。一般的なパターンを示します。

```
expression comparison_operator [any | all] expression
expression [not] in expression
[not]exists expression
expression [not] between expression and expression
expression [not] like "match_string" [escape "escape_character "]
not expression like "match_string" [escape "escape_character "]
expression is [not] null
not logical_expression
logical_expression {and | or} logical_expression
```

演算子の優先度

演算子の優先度レベルを次に示します。1 が最も高く、6 が最も低いレベルになります。

- 1 単項 (単独の引数) - + ~
- 2 * / %
- 3 2項 (2つの引数) + - & | ^
- 4 not
- 5 and
- 6 or

式のすべての演算子が同じ優先度であれば、左から右に実行されます。実行の順序は、カッコを使用して変更できます。最も内側にネストされた式が最初に処理されます。

算術演算子

Adaptive Server は、次の算術演算子を使用します。

表 4-2: 算術演算子

演算子	意味
+	加算
-	減算
*	乗算
/	除算
%	モジュロ (Transact-SQL 拡張機能)

加算、減算、除算、乗算は、真数、概数、通貨型のカラムで使用できます。

モジュロ演算子は `smallmoney`、`money`、`numeric`、`float`、または `real` カラムには使用できません。モジュロは、2つの整数を使用する除算の整数の剰余を検出します。たとえば、`21 % 11 = 10` は、21 割る 11 は、商が 1 で余りが 10 になることを表します。

`float` と `int` など、混合データ型に対して算術演算を行うと、Adaptive Server は特定の規則に従って結果の型を決定します。詳細については、「[第1章 システム・データ型とユーザ定義データ型](#)」を参照してください。

ビット処理演算子

ビット処理演算子は、整数値型のデータで使用する Transact-SQL 拡張機能です。これらの演算子は、それぞれの整数のオペランドをバイナリ表現に変換し、次にカラムごとにオペランドを評価します。値 1 は true に対応します。値 0 は false に対応します。

表 4-3 に、0 と 1 のオペランドの結果を示します。どちらかのオペランドが NULL の場合は、ビット処理演算子は NULL を返します。

表 4-3: ビット処理演算の真理値表

& (and)	1	0
1	1	0
0	0	0
(or)	1	0
1	1	1
0	1	0
^ (exclusive or)	1	0
1	0	1
0	1	0
~ (not)		
1	FALSE	
0	0	

表 4-4 の例は、次の 2 つの tinyint 引数を使用します。A = 170 (バイナリ形式では 10101010) および B = 75 (バイナリ形式では 01001011) を使用しています。

表 4-4: ビット処理演算の例

オペレーション	バイナリ形式	結果	説明
(A & B)	10101010 01001011 ----- 00001010	10	A と B の両方が 1 のときは、結果カラムには 1 が表示される。それ以外の場合は、結果カラムには 0 が表示される。
(A B)	10101010 01001011 ----- 11101011	235	A と B のいずれか片方または両方が 1 のときは、結果カラムには 1 が表示される。それ以外の場合は、結果カラムには 0 が表示される。
(A ^ B)	10101010 01001011 ----- 11100001	225	A か B の片方だけが 1 のときは、結果カラムには 1 が表示される。
(~A)	10101010 ----- 01010101	85	すべての 1 は 0 に変更され、すべての 0 は 1 に変更される。

文字列連結演算子

複数の文字式やバイナリ式を連結するには、文字列演算子 `+` や `||` (二重パイプ) を使用します。次の例では、著者の名前をカラム見出し `Name` の下に姓、名前の順で表示します。姓のあとにはカンマを置きます。たとえば、“Bennett, Abraham” のようになります。

```
select Name = (au_lname + ", " + au_fname)
from authors
```

次の例の結果は、“abcdef”, “abcdef” になります。

```
select "abc" + "def", "abc" || "def"
```

次の演算では文字列 “abc def” を返します。`char`、`varchar`、`unichar`、`nchar`、`nvarchar`、`text` の連結文と、`varchar`、`univarchar` の挿入文と代入文では、空の文字列は 1 つのスペースとして解釈されます。

```
select "abc" + " " + "def"
```

文字式でもバイナリ式でもない式を連結するには、必ず `convert` を使用します。

```
select "The date is " +
convert(varchar(12), getdate())
```

NULL と連結された文字列は、その文字列の値として評価されます。これは、NULL で連結された文字列は、NULL に評価すると定められている SQL 標準では例外です。

比較演算子

Adaptive Server が使用する比較演算子を表 4-5 に示します。

表 4-5: 比較演算子

演算子	意味
=	等しい
>	より大きい
<	より小さい
>=	以上
<=	以下
<>	等しくない
!=	(Transact-SQL 拡張機能) 等しくない
!>	(Transact-SQL 拡張機能) より大きくない
!<	(Transact-SQL 拡張機能) より小さくない

文字データの比較では、< はサーバのソート順の初めの方に近いことを意味し、> はソート順の終わりの方に近いことを意味します。大文字と小文字を区別しないソート順では、ある文字の大文字と小文字は同じ文字として扱われます。使用している Adaptive Server のソート順を表示するには、[sp_helpsort](#) を使用します。比較の場合、後続ブランクは無視されます。たとえば、“Dirk” と “Dirk ” は同じです。

データの比較では、< はより前を意味し、> はより後を意味します。

比較演算子とともに使用するすべての文字および `datetime` データは、一重引用符か二重引用符で囲みます。

```
= "Bennet"
> "May 22 1947"
```

標準以外の演算子

次に示す演算子は Transact-SQL 拡張機能です。

- モジュロ演算子：%
- 否定の比較演算子：!>、!<、!=
- ビット処理演算子：~、^、|、&
- ジョイン演算子：*= および =*

any、all、および in の使用

any は <、>、または = のいずれかと、サブクエリとともに使用します。これは、サブクエリ内で検索された値のいずれかが、外側の文の **where** 句または **having** 句の値と一致すると、結果を返します。詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

all は <か > のいずれかと、サブクエリとともに使用します。これは、サブクエリ内で検索されたすべての値が、外側の文の **where** 句または **having** 句の値より小さい (<) か、または大きい (>) と、結果を返します。詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

in は、2番目の式が返した値のいずれかが、1番目の式の値と一致する場合に、結果を返します。2番目の式は、サブクエリ、またはカッコで囲んだ値のリストです。**in** は **= any** と等しくなります。詳細については、『リファレンスマニュアル：コマンド』の「**where 句**」を参照してください。

否定と存在確認

not はキーワードの意味または論理式の意味を否定します。

特定の結果が存在するかどうかをテストするには、**exists** の後にサブクエリを指定します。

範囲

between は範囲開始のキーワードです。**and** は範囲終了のキーワードです。次の範囲は包括的範囲です。

```
where column1 between x and y
```

次の範囲は包括的範囲ではありません。

```
where column1 > x and column1 < y
```

式での null の使用

is null と **is not null** は、**null** 値が使用可能であると定義されたカラムのクエリで使用します。

ビット処理演算子と算術演算子のある式は、いずれかのオペランドが **null** である場合は、**NULL** と評価されます。次の式は **column1** が **NULL** であれば **NULL** と評価されます。

```
1 + column1
```

TRUE を返す比較

一般に、null の値を比較した結果は、UNKNOWN になります。これは、NULL が特定の値や他の NULL に等しいか (または等しくないか) どうかを判断できないためです。ただし、次のような場合には、*expression* が、NULL と評価されるカラム、変数、リテラル、またはその組み合わせであると、TRUE が返されます。

- *expression is null*
- *expression = null*
- *expression = @x* (この *@x* は NULL を含む変数またはパラメータです。この例外によって、null のデフォルト・パラメータを使用したストアド・プロシージャの作成が簡単になります。)
- *expression != n* (この *n* は NULL を含まないリテラルです。この場合 *expression* は NULL に評価されます。)

これらの式の否定は、式が NULL と評価されないときに TRUE を返します。

- *expression is not null*
- *expression != null*
- *expression != @x*

注意 これらの式の右端の値は、リテラルの null であるか、NULL を含む変数またはパラメータです。比較の一番右側が式 (*@nullvar + 1* など) の場合、式全体が NULL に評価されます。

この規則に従って、null のカラムの値は、他の null のカラムの値に結合されません。1 つの **where 句** のなかで、null のカラムの値を他の null のカラムの値と比較すると、どの比較演算子を使った場合にも、null の値には UNKNOWN が返され、結果にはローは含まれません。たとえば次のクエリは、両方のテーブルにおいて column1 に NULL が含まれている場合、ローを返しません (ただし他のローを返すことはあります)。

```
select column1
from table1, table2
where table1.column1 = table2.column1
```

FALSE と UNKNOWN の違い

FALSE も UNKNOWN も値を返しません、FALSE と UNKNOWN には論理的に重要な違いがあります。これは、false の反対 (“not false”) が true であるためです。たとえば、“1 = 2” は false と評価され、その反対の “1 != 2” は true と評価されます。しかし、“not unknown” は、unknown のままです。比較に null 値が含まれている場合は、式を否定して反対のローの集合や反対の真理値を取得することはできません。

文字列としての“NULL”の使用

create table 文内で NULL が指定され、ユーザが明示的に NULL を (引用符を付けずに) 挿入したカラム、またはデータが挿入されていないカラムだけが、null 値を含みます。文字カラムには、文字列 “NULL” (引用符付き) をデータとして入力しないでください。入力すると、混乱を招きます。代わりに “N/A” や “none” などの値を使用してください。値 NULL を明示的に入力したい場合は、一重または二重の引用符を使用しないでください。

NULL と空の文字列の比較

空の文字列 (“ ” や ‘ ’) は、常にシングル・スペースとして変数やカラム・データに格納されます。次の連結文は、“abc def” と等価ですが、“abcdef” とは等価ではありません。

```
"abc" + " " + "def"
```

空文字列は NULL として評価されることはありません。

式の接続

and は 2 つの式を接続し、両方が true の場合に結果を返します。or は 2 つ以上の条件を接続し、いずれかの条件が true であれば、結果を返します。

1 つの文中で複数の論理演算子が使用されるときは、or の前に and が評価されます。実行の順序はカッコを使用して変更できます。

表 4-6 に、null 値を使用するものも含めて、論理演算の結果を示します。

表 4-6: 論理式の真理値表

and	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
NULL	UNKNOWN	FALSE	UNKNOWN
or	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
NULL	TRUE	UNKNOWN	UNKNOWN
not			
TRUE	FALSE		
FALSE	TRUE		
NULL	UNKNOWN		

UNKNOWN という結果は、1 つ以上の式が NULL と評価されて、演算の結果が TRUE であるか FALSE であるか判別できないことを示します。詳細については、「式での null の使用」(331 ページ) を参照してください。

式でのカッコの使用

式で要素をグループにまとめるにはカッコを使用します。構文内で「式」が変数として指定されている場合は、単純式が想定されます。1つの論理式だけが受け入れ可能な場合は、「論理式」が指定されます。

文字式の比較

文字定数式は、`varchar` として扱われます。`varchar` 以外の変数やカラム・データと比較する場合、比較にはデータ型の優先度の規則が適用されます(つまり、優先度の低いデータ型が優先度の高いデータ型に変換されます)。暗黙的なデータ型の変換がサポートされていない場合は、`convert` 関数を使用する必要があります。

`char` 式と `varchar` 式との比較は、データ型の優先度の規則に従います。「低い」データ型は「高い」データ型に変換されます。すべての `varchar` 式は、比較のために `char` に変換されます(つまり、後続ブランクが追加されます)。`unichar` 式と `char` (`varchar`、`nchar`、`nvarchar`) 式を比較する場合、`char` 式が `unichar` に変換されます。

空文字列の使用

空文字列 ("") または (' ') は、`varchar` または `univarchar` データの `insert` 文や代入文では1つのブランクとして解釈されます。`varchar`、`char`、`nchar`、`nvarchar` データの連結では、空の文字列は1つのスペースとして解釈されます。たとえば、次の例では、“abc def”として格納されます。

```
"abc" + "" + "def"
```

空文字列は `NULL` として評価されることはありません。

文字式に含む引用符

`char` または `varchar` エントリ内にリテラル引用符を指定するには、2つの方法があります。1つ目の方法は、引用符を2つ使用する方法です。たとえば一重引用符で文字の入力を始めて、入力の一部に一重引用符を使いたい場合には、一重引用符を2つ使います。

```
'I don''t understand.'
```

二重引用符の場合には、次のようにします。

```
"He said, ""It's not really confusing."""
```

2つ目の方法は、片方の引用符を別の引用符で囲む方法です。つまり、二重引用符を含むエントリを単重引用符で (または単重引用符を含むエントリを二重引用符で) 囲みます。例を示します。

```
'George said, "There must be a better way."'
"Isn't there a better way?"
'George asked, "Isn"t there a better way?"'
```

継続文字の使用

画面上で次の行に文字列を続けるには、改行する前に円記号(¥)を入力します。

識別子

識別子は、データベース、テーブル、ビュー、カラム、インデックス、トリガ、プロシージャ、デフォルト、ルール、カーソルなどのデータベース・オブジェクトの名前です。

オブジェクト名または識別子の長さに対する制限は、通常の識別子の場合に 255 バイト、区切り識別子の場合に 253 バイトです。この制限は、テーブル名、カラム名、インデックス名などのほとんどのユーザ定義識別子に適用されます。制限が拡張されたため、一部のシステム・テーブル (カタログ) および組み込み関数も拡張されました。

変数では “@” が 1 文字としてカウントされるため、変数名は最大で 254 文字です。

次のリストに、この制限の影響を受ける識別子、システム・テーブル、組み込み関数を示します。

次の識別子の最大長は 255 バイトになりました。

- テーブル名
- カラム名
- インデックス名
- ビュー名
- ユーザ定義データ型
- トリガ名
- デフォルト名
- ルール名
- 制約名

- プロシージャ名
- 変数名
- JAR 名
- LWP または動的文の名前
- 関数名
- 時間範囲の名前
- アプリケーション・コンテキスト名

Adaptive Server のほとんどのユーザ定義識別子では、使われている文字がシングルバイト文字でもダブルバイト文字でも、最大長は 255 バイトです。それ以外の場合は、最大 30 バイトです。255 バイトの識別子と 30 バイトの識別子のリストについては、『Transact-SQL ユーザーズ・ガイド』を参照してください。

識別子の先頭文字は、現在の文字セットに定義されている文字か、アンダースコア () 文字を使用してください。

注意 この規則の例外として、シャープ記号 (#) で始まるテンポラリー・テーブル名と、at 記号 (@) で始まるローカル変数名があります。

後続の文字として、英字、数字、記号 (#、@、_)、ドル記号 (\$)、円記号 (¥)、または通貨ポンド記号 (£) などの通貨記号を使用できます。識別子には、!、%、^、&、*、. などの特別な記号や、埋め込みスペースを使うことはできません。

Transact-SQL コマンドなどの予約語も、識別子には使用できません。予約語の完全なリストについては、[第 5 章「予約語」](#)を参照してください。

ダッシュ記号 (-) は識別子として使用できません。

短い識別子

次の識別子の最大長は 30 バイトです。

- カーソル名
- サーバ名
- ホスト名
- ログイン名
- パスワード
- ホスト・プロセス ID
- アプリケーション名
- 初期言語名

- 文字セット名
- ユーザ名
- グループ名
- データベース名
- 論理デバイス名
- セグメント名
- セッション名
- 実行クラス名
- エンジン名
- クワイス・タグ名
- キャッシュ名

で始まるテーブル (テンポラリ・テーブル)

名前がシャープ記号(#)で始まるテーブルは、テンポラリ・テーブルです。別の種類のオブジェクトを作成するときに、シャープ記号で始まる名前を指定することはできません。

Adaptive Server は、セッションごとにユニークな名前を付けるために、テンポラリ・テーブル名に対して特別な操作を行います。テンポラリ・テーブルを 238 バイトより短い名前で作成すると、**sysobjects** は **tempdb** 内でテーブル名をユニークにするために名前に 17 バイトを追加します。テーブル名が 238 バイトより長い場合、**sysobjects** は最初の 238 バイトだけを使用し、それに 17 バイトを付加してテーブル名をユニークにします。

15.0 より前のバージョンの Adaptive Server では、**sysobjects** のテンポラリ・テーブル名は 30 バイトでした。13 バイトより短いテーブル名を使用していた場合は、長さが 13 バイトになるまで名前にアンダースコア(_)が付加され、さらに別の 17 バイトの文字が追加されて 30 バイトになりました。

大文字と小文字の区別と識別子

識別子やデータの大文字と小文字の区別は、ご使用の Adaptive Server にインストールされているソート順に応じて決まります。1 バイト文字セットの大文字と小文字の区別の指定を変更するには、Adaptive Server のソート順の設定を変更します。詳細については、『システム管理ガイド』を参照してください。ユーティリティ・プログラム・オプションでは、大文字と小文字の区別が重要になります。

Adaptive Server が、大文字と小文字を区別しないソート順でインストールされている場合は、MyTable や mytable のような名前のテーブルがすでに存在していると、MYTABLE という名前のテーブルを作成することはできません。同様に、次のコマンドは MYTABLE、MyTable、mytable、あるいは大文字と小文字を任意に組み合わせた同じ名前のテーブルから、ローを返します。

```
select * from MYTABLE
```

オブジェクト名の一意性

データベースでは、オブジェクト名を一意(ユニーク)にする必要はありません。しかし、テーブルの内部のカラム名とインデックス名は、一意にする必要があります。また、他のオブジェクト名も、データベース内のそれぞれの所有者に対して一意にする必要があります。Adaptive Server では、データベース名を一意にしてください。

区切り識別子の使用

「区切り識別子」は、二重引用符で囲まれたオブジェクト名です。区切り識別子を使用することによって、オブジェクト名に関する一定の制限を回避できます。以前のバージョンの Adaptive Server では、テーブル、ビュー、カラム名のみを引用符で区切ることができ、ほかのオブジェクト名はできませんでした。これは Adaptive Server バージョン 15.7 から改善されましたが、機能の有効化には設定パラメータの設定が必要です。

区切り識別子には予約語を使用でき、最初の文字に英字以外のものを使用できます。また、規則では認められていない文字も使用できます。253 バイトを超えることはできません。

警告! すべてのフロント・エンド・アプリケーションで、区切り識別子が認識されるとは限りません。区切り識別子をシステム・プロシージャへのパラメータとして使用しないでください。

区切り識別子を作成または参照する前に、以下の文を実行してください。

```
set quoted_identifier on
```

文で区切り識別子を使用するときには、必ずこの識別子を二重引用符で囲んでください。たとえば、次のようにします。

```
create table "lone"(coll char(3))
create table "include spaces" (coll int)
create table "grant"("add" int)
insert "grant"("add") values (3)
```


`quoted_identifier` オプションが `on` に設定されている間は、二重引用符ではなく一重引用符で文字やデータ文字列を囲んでください。これらの文字列を二重引用符で区切ると、Adaptive Server はこれを識別子として扱います。たとえば *table* の *coll* に文字列を挿入するには、次のように指定します。

```
insert "lone"(coll) values ('abc')
```

次のようには指定しないでください。

```
insert "lone"(coll) values ("abc")
```

カラムに一重引用符を挿入するには、連続する2つの一重引用符を使用してください。たとえば、*coll* に “a”b” を挿入するには、次のようにします。

```
insert "lone"(coll) values ('a"b')
```

引用符を含む構文

文の構文で識別子を引用符付き文字列に入れるように要求されている場合、`quoted_identifier` オプションが `on` に設定されていれば、識別子の前後に二重引用符を使用する必要はありません。たとえば、次のようにします。

```
set quoted_identifier on
create table 'lone' (c1 int)
```

ただし、`object_id()` では文字列が必須のため、テーブル名を引用符で囲んで情報を選択します。

```
select object_id('lone')
-----
      896003192
```

引用符を2つ使用すると、引用符付き識別子に埋め込み二重引用符を含めることができます。

```
create table "embedded"quote" (c1 int)
```

ただし、文の構文でオブジェクト名を文字列として表現することが要求されている場合には、引用符を2つ使用する必要はありません。

```
select object_id('embedded"quote')
```

インデックス名およびシステム・プロシージャ・パラメータの引用符付き識別子の有効化

次の引用符付き識別子を使用できます。

- テーブル
- ビュー
- カラム名
- インデックス名 (Adaptive Server version 15.7 以降)
- システム・プロシージャ・パラメータ (Adaptive Server version 15.7 以降)

Adaptive Server で引用符付き識別子を使用できるようにするには、次の手順に従います。

- 1 **enable functionality group** または **quoted identifier enhancement** 設定パラメータを 1 に設定します。次に例を示します。

```
sp_configure "enable functionality group", 1
```

- 2 現在のセッションの **quoted_identifier** をオンにします。

```
set quoted_identifier on
```

この機能を有効にすると、引用符付き識別子を含めたときに、クエリ・プロセッサにより、オブジェクト定義からデリミタおよび後続のスペースが削除されます。たとえば、`"ident"`、`[ident]`、`ident` は同一であるとみなされます。この機能が無効になっていると、`"ident"` は、ほかの 2 つとは異なる識別子とみなされます。

この機能を有効にして Adaptive Server を起動すると、次の処理が行われます。

- **enable functionality group** 設定パラメータを有効にして Adaptive Server を再起動する「前に」引用符付き識別子で作成したオブジェクトは、このパラメータを有効にしてサーバを起動した「後に」引用符付き識別子を使用しても自動的にアクセス可能になりません。逆も同様です。つまり、Adaptive Server では自動的にすべてのデータベース・オブジェクトの名前が変更されません。

ただし、**sp_rename** を使用して、オブジェクトの名前を手動で変更することはできます。たとえば、`"ident"` という名前のオブジェクトを作成し、この機能を有効にして Adaptive Server を再起動する場合は、次のコマンドを発行してオブジェクトの名前を変更します。

```
sp_rename '"ident"', 'ident'
```

- この機能を有効にして起動した Adaptive Server では、`[tab.dba.ident]` および `"tab.dba.ident"` を完全に修飾された名前として扱います。
- オブジェクトの識別子を受け入れる Transact-SQL 文、関数、システム・プロシージャ、ストアド・プロシージャも区切り識別子を使用します。

- この機能を有効にして Adaptive Server を起動すると、`valid_name` 関数では、通常のルールで識別子に有効な文字列と、区切り識別子のルールで有効な文字列が区別され、有効な名前であることを示す戻り値としてゼロ以外の値が使用されます。

たとえば、`valid_name('ident/v1')` は、引用符付き識別子のルールを使用した場合に有効な名前について `true` (つまり、ゼロ以外の値) を返します。通常のルールを使用した場合に有効な識別子についても `true` (ゼロ以外) を返します。引用符付き識別子ルールを使用した場合のみ有効な識別子は、値 128 (0x80) によって区別されます。

- 引用符付き識別子を使用すると、通常は 255 バイトまで可能な Adaptive Server の識別子は 253 バイトに制限され、通常は 30 バイトに制限されている識別子は 28 バイトに制限されます。
- Adaptive Server 以外の項目に関連する一部の識別子には、ほかの制約がある可能性があります。たとえば、サーバ名は外部の文字セットおよび構文ルールが適用される場合があるディレクトリ・サービスとともに使用されるので (`$$SYBASE/interfaces`、LDAP など)、サーバ名には ASCII 英数字のみを使用することをおすすめします。

修飾されたオブジェクト名によるテーブルまたはカラムの識別

テーブルやカラムを一意に識別するには、これらを修飾する別の名前を追加します。たとえばデータベース名、所有者名、(カラムについては) テーブル名やビュー名で修飾できます。それぞれの修飾子はピリオドで区切ります。たとえば、次のようにします。

```
database.owner.table_name.column_name
database.owner.view_name.column_name
```

命名規則は次のようになります。

```
[[database.]owner.]table_name
[[database.]owner.]view_name
```

オブジェクト名の中での区切り識別子の使用

`set quoted_identifier on` を使用すると、修飾されたオブジェクト名の個々の部分を二重引用符で囲むことができます。引用符が必要なそれぞれの修飾子について、引用符を一組ずつ使います。たとえば、次のようにします。

```
database.owner."table_name"."column_name"
```

次のような使い方はできません。

```
database.owner."table_name.column_name"
```

所有者名の省略

オブジェクトを識別するための十分な情報がシステムに提供されている場合には、名前の中間部分を省略し、省略した場所をドットで示すことができます。

```
database..table_name
```

```
database..view_name
```

現在のデータベース内のユーザ自身が所有するオブジェクトの参照

現在のデータベース内にある、ユーザ自身が所有するオブジェクトを参照するときに、データベース名や所有者名を使う必要はありません。*owner* のデフォルト値は現在のユーザで、*database* のデフォルト値は現在のデータベースです。

データベース名と所有者名で修飾されていないオブジェクトを参照する場合は、Adaptive Server は現在のデータベースで、ユーザが所有しているオブジェクトの中から、そのオブジェクトを見つけようとします。

データベース所有者が所有しているオブジェクトの参照

所有者名を省略し、その名前オブジェクトを所有していない場合は、Adaptive Server はデータベース所有者が所有している同名のオブジェクトを検索します。データベース所有者が所有しているオブジェクトを参照する場合に、そのオブジェクトを修飾する必要があるのは、同じ名前のオブジェクトをユーザが所有している場合だけです。しかし、他のユーザが所有しているオブジェクトについては、ユーザ自身が同じ名前のオブジェクトを所有しているかどうかとは関係なく、他のユーザの名前でオブジェクトを修飾する必要があります。

修飾された識別子の一貫した使用

同じ文でカラム名とテーブル名を修飾する場合には、それぞれに同じ修飾式を使用するようにしてください。これらは文字列として評価され、一致する必要があります。そうでない場合はエラーが返されます。例 2 は正しくない使い方です。この例では、カラム名の構文スタイルが、テーブル名で使用している構文スタイルと一致していません。

```
例 1
select demo.mary.publishers.city
from demo.mary.publishers

city
-----
Boston
Washington
Berkeley
```

```
例 2
select demo.mary.publishers.city
from demo..publishers
```

カラム・プレフィクス "demo.mary.publishers" が、クエリで使用されるテーブル名またはエイリアス名と一致していません。

識別子の有効性の確認

文字セットを変更したあと、またはテーブルやビューを作成する前に、システム関数 `valid_name` を使用して、オブジェクト名が Adaptive Server で使用できるかどうかを確認してください。構文は次のとおりです。

```
select valid_name("Object_name")
```

`object_name` が有効な識別子でない場合 (たとえば無効な文字が入っていたり、30 バイトの長さを超えていたりする場合は、Adaptive Server は 0 を返します。`object_name` が有効な識別子である場合は、Adaptive Server はゼロ以外の数字を返します。

データベース・オブジェクト名の変更

`sp_rename` でユーザ・オブジェクト (ユーザ定義データ型など) の名前を変更します。

警告! テーブルやカラムの名前を変更した場合は、名前が変更されたオブジェクトを使用するプロシージャ、トリガ、ビューもすべて定義し直してください。

マルチバイト文字セットの使用

マルチバイト文字セットでは、さまざまな文字を識別子に使うことができます。たとえば日本語がインストールされたサーバでは、識別子の最初の文字に、全角と半角のカタカナ、ひらがな、漢字、ローマ字、ギリシャ文字、キリル文字、ASCII などの文字を使用できます。

半角カタカナは日本語システムの識別子では有効ですが、異機種間システムでの使用はおすすりできません。半角カタカナは EUC-JIS 文字セットと Shift-JIS 文字セットの間では変換できません。

これは 8 ビット欧州文字のいくつかにもあてはまります。たとえば、Macintosh の文字セット (コードポイント 0xCE) 中にある OE の合字です。この文字は、ISO 8859-1 (iso_1) 文字セットには含まれていません。Macintosh から ISO 8859-1 文字セットに変換するデータに O と E の合字があると、変換エラーが発生します。

変換できない文字がオブジェクト識別子に含まれていると、クライアントはそのオブジェクトに直接アクセスできなくなります。

like パターン一致

Adaptive Server バージョンでは、like パターン一致アルゴリズム内の角カッコを個別に処理できます。

たとえば、以前のバージョンの Adaptive Server で ‘[XX]’ のローを一致させるには、次を使用しました。

```
select * from t1 where f1 like '[][XX[]]'
```

ただし、次を使用することもできます。

```
select * from t1 where f1 like '[][XX]'
```

like パターン一致を使用すると、次のようになります。

- 右角カッコ (“]”) を左角カッコ (“[”) のすぐ後に置くことで、右角カッコ自体を表すことができます。つまり、パターン “[]” は、文字列 “[]” と一致します。
- 最初の脱字記号 (“^”) は、すべての文字範囲内の意味を反転させます。つまり、パターン “[^]” は、“]” ではない 1 つの文字列と一致します。
- 右カッコ (“]”) は、どの位置にあっても、文字範囲の末尾を示します。

次の表は like パターン一致のパターンと一致を示します。

パターン	一致
“[]”	“[]”
“[]”	“]”
“]”	“]”
“[][XX]”	“[XX]”
“[][XX[]”	“[XX]”

ワイルドカード文字によるパターン一致

match_string では、ワイルドカード文字は、1 文字以上の文字を表すか、または文字の範囲を表します。*match_string* は、式の中で検索するパターンからなる文字列です。次のように、定数、変数、カラム名、連結した式などの任意の組み合わせを使うことができます。

```
like @variable + "%".
```

一致する文字列が定数の場合は、必ず一重引用符か二重引用符で囲んでください。

特定のパターンと一致する文字や日付文字列を検索するには、キーワード `like` とワイルドカード文字を使います。ただし、秒とミリ秒を検索する場合は、`like` を使うことはできません。詳細については、[「datetime データでのワイルドカード文字の使用」\(351 ページ\)](#)を参照してください。

一致する文字列に似た (*like*) 文字または日付/時刻の情報、あるいは似ていない (*not like*) 文字または日付/時刻の情報を検索するには、*where* と *having* 句で次のようなワイルドカード文字を使用します。

```
{where | having} [not]
  expression [not] like match_string
  [escape "escape_character"]
```

この *expression* には、文字値を持つ関数、定数、カラム名の任意の組み合わせを指定できます。

ワイルドカード文字を *like* とともに使用しない場合、ワイルドカード文字に特別な意味はありません。たとえば次のクエリでは、“415%” の 4 文字で始まる任意の電話番号が検索されます。

```
select phone
from authors
where phone = "415%"
```

like パターン一致

Adaptive Server バージョンでは、*like* パターン一致アルゴリズム内の角カッコを個別に処理できます。

たとえば、以前のバージョンの Adaptive Server で “[XX]” のローを一致させるには、次を使用しました。

```
select * from t1 where f1 like '[[XX]]'
```

Adaptive Server バージョン 15.7 では、以下も使用できます。

```
select * from t1 where f1 like '[[]XX]'
```

完全互換が必要であるため、この機能は次のコマンドを有効にすることによって Adaptive Server version 15.7 でのみ使用できます。

```
sp_configure "enable functionality group", 1
```

この機能を有効にしないと、角カッコの *like* パターン一致の動作は前のリリースのようになります。

この機能を有効にすると、次のようになります。

- *like* パターン一致を使用すると、右角カッコ (“]”) を左角カッコ (“[”) のすぐ後に置くことで、右角カッコ自体を表すことができます。つまり、パターン “[]” は、文字列 “[” と一致します。
- 最初の脱字記号 (“^”) は、すべての文字範囲内の意味を反転させます。つまり、パターン “[^]” は、“]” ではない 1 つの文字列と一致します。
- 右カッコ (“]”) は、どの位置にあっても、文字範囲の末尾を示します。

-T4086 を有効にしたときに機能するパターンは次のとおりです。

パターン	一致
"[]"	"["
"[]]"	"]"
"j]"	"]"
"[[]XX]"	"[XX]"
"[[]XX []]"	"[XX]"

not likeの使用

`not like` は、特定のパターンと一致しない文字列を見つけるために使用します。次に示す2つのクエリは、同じことを意味します。いずれも `authors` テーブルから、地域コード 415 以外で始まるすべての電話番号を探します。

```
select phone
from authors
where phone not like "415%"
```

```
select phone
from authors
where not phone like "415%"
```

たとえば次のクエリは、データベース内の名前が“sys”で始まるシステム・テーブルを検索します。

```
select name
from sysobjects
where name like "sys%"
```

システム・テーブルでないすべてのオブジェクトを検索するには、次のように指定します。

```
not like "sys%"
```

オブジェクトの合計数が 32 で、`like` でパターンが一致する名前が 13 個見つかったとすると、`not like` ではパターンに一致しないオブジェクトが 19 個見つかります。

`not like` を否定のワイルドカード文字 `[^]` とともに使うと、異なった結果になることがあります (「[脱字記号 \(^\) ワイルドカード文字](#)」(349 ページ) を参照)。`not like` のパターンで得られた結果は、`like` と `^` によって得られた結果と異なることがあります。これは、`not like` は `like` のパターン全体と一致しない項目を検索しますが、`like` と否定のワイルドカード文字の組み合わせでは 1 文字ずつ順に評価されるためです。

like “[^s][^y][^s]” のようなパターンでは、同じ結果が得られない可能性があります。この場合、19 個ではなく、14 個だけが検出されることがあります。ここでは、“s” で始まる名前、“y” を 2 番目の文字として持つ名前、“s” を 3 番目の文字として持つ名前がすべて、システム・テーブル名と同様に検索結果から外されます。これは、否定のワイルドカードでは、一致文字列が 1 文字ずつ順に評価されるためです。評価のある時点で失敗した一致は、削除されます。

大文字と小文字の区別とアクセントの区別の無視

Adaptive Server で使用しているソート順が大文字と小文字を区別しないものである場合は、*expression* と *match_string* を比較する際に、大文字と小文字の区別は無視されます。たとえば、大文字と小文字を区別しない Adaptive Server では、次の句を指定すると、“Smith”、“smith”、“SMITH”などが返されます。

```
where col_name like "Sm%"
```

ご使用の Adaptive Server がアクセントについても無視する場合は、大文字でも小文字でも、アクセントのある文字はどれもアクセントのない文字と同様に扱われます。システム・プロシージャ `sp_helpsort` では、同じ文字として取り扱われた文字を“=”で結んで表示します。

ワイルドカード文字の使用

一致文字列を各種ワイルドカードと一緒に使うことができます。以降の項では、このワイルドカード文字について詳しく説明します。表 4-7 に、ワイルドカード文字を示します。

表 4-7: ワイルドカード文字の使用

記号	意味
%	0 個以上の文字の文字列
_	任意の 1 文字
[]	指定の範囲 ([a-f]) またはセット ([abcdef]) 内の任意の 1 文字
[^]	指定した範囲 [^a-f] またはセット [^abcdef] に含まれない任意の 1 文字

ワイルドカード文字と一致文字列は、一重または二重引用符で囲みます。(like “[dD]eFr_nce”)

パーセント記号 (%) ワイルドカード文字

0 文字以上の文字列を表すには、% ワイルドカード文字を使用します。たとえば、authors テーブルで局番が 415 で始まるすべての電話番号を検索するには、次のように指定します。

```
select phone
from authors
where phone like "415%"
```

“en” という文字が含まれている文字列 (Bennet、Green、McBadden) を検索するには、次のように指定します。

```
select au_lname
from authors
where au_lname like "%en%"
```

like 句では、“%” に続く複数の後続空白は切り捨てられ 1 つの後続空白になります。たとえば、“%” の後に 2 つのスペースが続く場合、この “%” は、“X ” (スペース 1 つ)、“X ” (スペース 2 つ)、“X ” (スペース 3 つ)、または任意の後続スペースと一致します。

アンダースコア (_) ワイルドカード文字

1 文字を表すためには、アンダースコア (_) ワイルドカード文字を使用します。たとえば “heryl” で終わる 6 文字の名前 (Cheryl など) をすべて検索する場合は、次のように指定します。

```
select au_fname
from authors
where au_fname like "_heryl"
```

角カッコ ([]) ワイルドカード文字

[a-f] のような文字の範囲、または [a2Br] のような文字セットを指定するには、これを角カッコで囲みます。範囲を指定した場合は、ソート順で *rangespec1* と *rangespec2* の間に含まれるすべての値が返されます (両端の文字を含みます)。たとえば、“[0-z]” は、7 ビット ASCII での 0-9、A-Z、a-z (およびその他の句読点記号) と一致します。

“inger” で終わり、M から Z までの 1 文字で始まる名前を検索するには、次のように指定します。

```
select au_lname
from authors
where au_lname like "[M-Z]inger"
```

“DeFrance” と “deFrance” の両方を検索するには、次のように指定します。

```
select au_lname
from authors
where au_lname like "[dD]eFrance"
```

オブジェクトを作成するときに、create table [*table_name*] や create database [*dbname*] のように角カッコ識別子を使用するときは、少なくとも 1 つの有効な文字を含める必要があります。

角カッコ識別子内のすべての後続スペースは、オブジェクト名から削除されます。たとえば、次の `create table` コマンドの実行結果はすべて同じになります。

- `create table [tab1<space><space>]`
- `create table [tab1]`
- `create table [tab1<space><space><space>]`
- `create table tab1`

この規則は、角カッコ識別子を使用して作成できるすべてのオブジェクトに適用されます。

脱字記号 (^) ワイルドカード文字

脱字記号は、否定のワイルドカード文字です。特定のパターンに一致しない文字列を検索するときに、この記号を使います。たとえば、“`[^a-f]`”と指定すると、`a`～`f`の範囲にない文字列が検索されます。また、“`[^a2bR]`”と指定すると、“`a`”、“`2`”、“`b`”、または“`R`”でない文字列が検索されます。

“`M`”で始まり、2番目の文字が“`c`”でない名前を見つけるには、次のように指定します。

```
select au_lname
from authors
where au_lname like "M[^c]%"
```

範囲を指定した場合は、ソート順で *rangespec1* と *rangespec2* の間に含まれるすべての値が返されます(両端の文字を含みます)。次に例を示します。“`[0-z]`”は7ビットASCIIでの0～9、A～Z、a～z、およびその他の句読点記号と一致します。

マルチバイト・ワイルドカード文字の使用

ご使用の Adaptive Server でマルチバイト文字セットが設定されていて、ワイルドカード文字 `_`、`%`、`[`、`]`、`^` に対応する2バイトの文字が定義されている場合には、対応する文字を一致文字列で使用できます。アンダースコアは、一致文字列のシングルバイト文字またはダブルバイト文字を表します。

リテラル文字としてのワイルドカード文字の使用

文字列の中の `%`、`_`、`[`、`]`、`^` そのものを検索したい場合には、エスケープ文字を使用します。エスケープ文字とともにワイルドカード文字を使うと、Adaptive Server ではワイルドカード文字が他の文字を示すものとは解釈せず、この文字をリテラル文字として解釈します。

Adaptive Server には次の 2 種類のエスケープ文字があります。

- 角カッコ (Transact-SQL 拡張機能)
- `escape` 句の直後の 1 文字 (SQL 標準への準拠)

エスケープ文字としての角カッコ ([]) の使用

パーセント記号、アンダースコア、左カッコのエスケープ文字として、角カッコを使います。右カッコには、エスケープ文字は不要です。右カッコをそのまま使うことができます。ハイフンをリテラル文字として使う場合は、1 組の角カッコの中の最初の文字として使います。

表 4-8 は、`like` で角カッコをエスケープ文字として使用する例を示します。

表 4-8: 角カッコを使用したワイルドカード文字の検索

like 述部	意味
<code>like "5%"</code>	後に 0 文字以上の文字列が続く 5
<code>like "5[%]"</code>	5%
<code>like "_n"</code>	an、in、on など
<code>like "[_]n"</code>	_n
<code>like "[a-cdf]"</code>	a、b、c、d、または f
<code>like "[-acdf]"</code>	、a、c、d、または f
<code>like "[[]]"</code>	[
<code>like "]"</code>]
<code>like "[[]ab]"</code>	[]ab

`escape` 句の使用

エスケープ文字を指定するには、`escape` 句を使用します。サーバのデフォルトの文字セットにある任意の 1 文字を、エスケープ文字として指定できます。複数の文字をエスケープ文字として使おうとすると、Adaptive Server は例外を生成します。

次のような理由から、既存のワイルドカード文字をエスケープ文字として指定しないでください。

- アンダースコア (`_`) やパーセント記号 (`%`) をエスケープ文字として指定すると、その `like` 述部では特別な意味を失ってしまい、エスケープ文字としてしか使えなくなります。
- 左または右のカッコ (`[` と `]`) をエスケープ文字として指定すると、`like` 述部ではカッコの Transact-SQL 機能としての意味が無効になります。
- ハイフン (`-`) または脱字記号 (`^`) をエスケープ文字として指定すると、特別な意味を失い、エスケープ文字としてしか使用できなくなります。

アンダースコア、パーセント記号、開きカッコなどは異なり、エスケープ文字は角カッコの中でも、その特別な意味を失いません。

エスケープ記号は、指定された `like` 述部の中だけで有効で、同じ文の他の `like` 述部では無効です。エスケープ記号のあとで有効な文字は、ワイルドカード文字 (`_`、`%`、`[`、`]`、`[^]`) とエスケープ文字そのものだけです。エスケープ文字はその直後の1文字だけに有効で、そのあとに続く文字には影響しません。

パターンに、たまたまエスケープ文字として指定した文字がリテラル文字として2文字ある場合は、その文字列ではエスケープ文字を4つ続けて使う必要があります。パターンがエスケープ文字によって1文字または2文字に分割されない場合は、Adaptive Server はエラー・メッセージを返します。表 4-9 は、`like` で `escape` 句を使用した例を示します。

表 4-9: `escape` 句の使用

like 述部	意味
<code>like "5@%" escape "@"</code>	5%
<code>like "*_n" escape "**"</code>	<code>_n</code>
<code>like "%80@%" escape "@"</code>	80% を含む文字列
<code>like "*_sql**%" escape "**"</code>	<code>_sql*</code> を含む文字列
<code>like "%#####_#%" escape "#"</code>	<code>##_%</code> を含んでいる文字列

`datetime` データでのワイルドカード文字の使用

`datetime` の値で `like` を使用すると、Adaptive Server は日付を標準の `datetime` フォーマットに変換し、次に `varchar` に変換します。標準の格納フォーマットには秒やミリ秒は含まれていないため、`like` とパターンを使用して秒やミリ秒を検索することはできません。

`datetime` のエントリにはさまざまな日付要素が含まれている可能性があるため、`datetime` 値を検索するときは `like` を使用することをおすすめします。たとえば、`arrival_time` カラムに、“9:20” という値と現在の日付を入力する場合は、この値は次の句では検索できません。

```
where arrival_time = '9:20'
```

これは、Adaptive Server が入力された内容を “Jan 1 1900 9:20AM” に変換するためです。しかし、次の句ではこの値が検出されます。

```
where arrival_time like '%9:20%'
```


予約語

予約語として知られるキーワードは、特別な意味を持つワードです。この章では、Transact-SQL と ANSI SQL のキーワードについて説明します。

この章では、次の項目について説明します。

トピック名	ページ
Transact-SQL 予約語	353
ANSI SQL 予約語	354
ANSI SQL の予約語となる可能性のあるワード	356

Transact-SQL 予約語

表 5-1 に、Adaptive Server によって予約されているキーワード (SQL コマンドの構文の一部) を示します。これらの予約語は、データベース、テーブル、ルール、デフォルトなどのデータベース・オブジェクトの名前として使用することはできません。ただし、これらの予約語をローカル変数名、ストアド・プロシージャのパラメータ名として使用することはできます。

予約語が使われている既存のオブジェクト名を検索するには、『ASE リファレンス・マニュアル：プロシージャ』の [sp_checkreswords](#) を使用します。

表 5-1: Transact-SQL 予約語のリスト

	予約語
<i>A</i>	add, all, alter, and, any, arith_overflow, as, asc, at, authorization, avg
<i>B</i>	begin, between, break, browse, bulk, by
<i>C</i>	cascade, case, char_convert, check, checkpoint, close, clustered, coalesce, commit, compute, confirm, connect, constraint, continue, controlrow, convert, count, count_big, create, current, cursor
<i>D</i>	database, dbcc, deallocate, declare, decrypt, default, delete, desc, deterministic, disk distinct, drop, dummy, dump
<i>E</i>	else, encrypt, end, endtran, errlvl, errordata, errexit, escape, except, exclusive, exec, execute, exists, exit, exp_row_size, external
<i>F</i>	fetch, fillfactor, for, foreign, from
<i>G</i>	goto, grant, group
<i>H</i>	having, holdlock
<i>I</i>	identity, identity_gap, identity_start, if, in, index, inout, insensitive, insert, install, intersect, into, is, isolation
<i>J</i>	jar, join
<i>K</i>	key, kill
<i>L</i>	level, like, lineno, load, lock

	予約語
<i>M</i>	materialized, max, max_rows_per_page, min, mirror, mirrorexist, modify
<i>N</i>	national, new, noholdlock, nonclustered, not, null, nullif, numeric_truncation
	<p>注意 “new” は Transact-SQL の予約語ではありませんが、今後予約語となる可能性があるため、使用を避ける (データベース・オブジェクトの名前などに使用しない) ことをおすすめします。“New” は、spt_values テーブルに含まれ、また sp_checkreswords では予約語として表示されるため、特殊な例です (他の予約語の詳細については、「ANSI SQL の予約語となる可能性のあるワード」(356 ページ) を参照してください)。</p>
<i>O</i>	of, off, offsets, on, once, online, only, open, option, or, order, out, output, over
<i>P</i>	partition, perm, permanent, plan, prepare, primary, print, privileges, proc, procedure, processexist, proxy_table, public
<i>Q</i>	quiesce
<i>R</i>	raiserror, read, readpast, readtext, reconfigure, references, remove, reorg, replace, replication, reservepagegap, return, returns, revoke, role, rollback, rowcount, rows, rule
<i>S</i>	save, schema, scroll, select, semi_sensitive, set, setuser, shared, shutdown, some, statistics, stringsize, stripe, sum, syb_identity, syb_restree, syb_terminate
<i>T</i>	table, temp, temporary, textsize, to, tracefile, tran, transaction, trigger, truncate, tsequal
<i>U</i>	union, unique, unpartition, update, use, user, user_option, using
<i>V</i>	values, varying, view
<i>W</i>	waitfor, when, where, while, with, work, writetext
<i>X</i>	xmlextract, xmlparse, xmltest

ANSI SQL 予約語

Adaptive Server は、初級レベル ANSI SQL の機能を備えています。完全実装の ANSI SQL は、コマンド構文として次の表に示すワードを含んでいます。識別子のアップグレードが複雑になることがあるため、ユーザの便を図ってこのキーワード・リストを作成しました。しかし、このリストを提供しても、Sybase がこれら ANSI SQL の全機能を今後のリリースで提供することを保証するものではありません。また、今後のリリースでは、このリストにはないキーワードが含まれる可能性もあります。

表 5-2 は、ANSI SQL のキーワードを示します (Transact-SQL の予約語ではありません)。

表 5-2: ANSI SQL 予約語のリスト

	予約語
<i>A</i>	absolute, action, allocate, are, assertion
<i>B</i>	bit, bit_length, both
<i>C</i>	cascaded, case, cast, catalog, char, char_length, character, character_length, coalesce, collate, collation, column, connection, constraints, corresponding, cross, current_date, current_time, current_timestamp, current_user
<i>D</i>	date, day, dec, decimal, deferrable, deferred, describe, descriptor, diagnostics, disconnect, domain
<i>E</i>	end-exec, exception, extract
<i>F</i>	false, first, float, found, full
<i>G</i>	get, global, go
<i>H</i>	hour
<i>I</i>	immediate, indicator, initially, inner, input, insensitive, int, integer, interval
<i>J</i>	join
<i>L</i>	language, last, leading, left, local, lower
<i>M</i>	match, minute, module, month
<i>N</i>	names, natural, nchar, next, no, nullif, numeric
<i>O</i>	octet_length, outer, output, overlaps
<i>P</i>	pad, partial, position, preserve, prior
<i>R</i>	real, relative, restrict, right
<i>S</i>	scroll, second, section, semi_sensitive, session_user, size, smallint, space, sql, sqlcode, sqlerror, sqlstate, substring, system_user
<i>T</i>	then, time, timestamp, timezone_hour, timezone_minute, trailing, translate, translation, trim, true
<i>U</i>	unknown, upper, usage
<i>V</i>	value, varchar
<i>W</i>	when, whenever, write, year
<i>Z</i>	zone

ANSI SQL の予約語となる可能性のあるワード

ISO/IEC 9075:1989 標準を使用する場合は、次のリストに示すワードの使用は避けてください。これらのワードは、ANSI SQL の予約語になる可能性があります。

表 5-3: ANSI SQL の予約語となる可能性のあるワードのリスト

	予約語
<i>A</i>	after, alias, async
<i>B</i>	before, boolean, breadth
<i>C</i>	call, completion, cycle
<i>D</i>	data, depth, dictionary
<i>E</i>	each, elseif, equals
<i>G</i>	general
<i>I</i>	ignore
<i>L</i>	leave, less, limit, loop
<i>M</i>	modify
<i>N</i>	new, none
<i>O</i>	object, oid, old, operation, operators, others
<i>P</i>	parameters, pendant, preorder, private, protected
<i>R</i>	recursive, ref, referencing, resignal, return, returns, routine, row
<i>S</i>	savepoint, search, sensitive, sequence, signal, similar, sqlexception, structure
<i>T</i>	test, there, type
<i>U</i>	under
<i>V</i>	variable, virtual, visible
<i>W</i>	wait, without

この章では、Adaptive Server の SQLSTATE ステータス・コードと各コードに関連付けられているメッセージについて説明します。

この章では、次の項目について説明します。

トピック名	ページ
警告	357
例外	358

初級レベル ANSI SQL 準拠の場合、SQLSTATE コードが必要です。これらは、次の 2 種類の条件についての診断情報を提供します。

- 「警告」－ ユーザに通知は必要だが、SQL 文の正常な実行の妨げとなるほど重大ではない状態。
- 「例外」－ データベースに対する SQL 文の実行が無効となる状態。

個々の SQLSTATE コードは、2 文字クラスとそれに続く 3 文字サブクラスで構成されます。クラスはエラー・タイプについての一般的な情報を指定します。サブクラスは、より詳細な情報を指定します。

SQLSTATE コードは、前述の条件が検出されたときに表示されるメッセージと一緒に、**sysmessages** システム・テーブルに保管されます。Adaptive Server のエラー条件はすべての SQLSTATE コードと一致するわけではありません (ANSI SQL に準拠しているものだけです)。また、複数の Adaptive Server のエラー条件が、1 つの SQLSTATE の値と重複する場合があります。

警告

現在、Adaptive Server では次の SQLSTATE 警告条件が 1 つだけ検出されません。表 6-1 にこの警告条件を示します。

表 6-1: SQLSTATE 警告

メッセージ	値	説明
警告：null 値は set 関数では使用できません。	01003	null 値をとる式で集合関数 (avg、max、min、sum、または count) を使用した場合。
Adaptive Server から返された文字またはバイナリ・データがトランケートされました。クライアント・アプリケーションは、結果カラムまたは出力パラメータとして、255 バイトを超えるデータをサポートしていません。	01004	文字、unichar、またはバイナリ・データが 255 バイトにトランケートされる場合。データには次のものがある。 <ul style="list-style-type: none"> クライアントが WIDE TABLES プロパティをサポートしない select 文の結果。 WIDE TABLES プロパティをサポートしないリモート Adaptive Server または Open Server 上の RPC へのパラメータ。

例外

Adaptive Server は、次に示すタイプの例外を検出します。

- 基本的な違反
- データ例外
- 整合性の制約違反
- 無効なカーソル状態
- 構文エラーおよびアクセス・ルール違反
- トランザクション・ロールバック
- with check option 違反

表 6-2 から表 6-8 では、例外条件について説明します。例外の各クラスは該当する表に示します。各表では、条件はメッセージ・テキストのアルファベット順にソートされています。

基本的な違反

相互関係による違反は、1 つのローだけを返すクエリが Embedded SQL™ アプリケーションに複数のローを返す場合に発生します。

表 6-2: 基本的な違反

メッセージ	値	説明
サブクエリが 2 個以上の値を返しました。サブクエリが、=、!=、<、<=、>、>= に続く場合、またはサブクエリが式で使用される場合は、無効です。	21000	次のような条件のときに発生する。 <ul style="list-style-type: none"> • スカラ・サブクエリ、またはロー・サブクエリが複数のローを返す場合。 • Embedded SQL の select into parameter_list クエリが複数のローを返す場合。

データ例外

次のような値が入力された場合、データ例外が発生します。

- データ型に対してエントリが長すぎる
- 不正なエスケープ・シーケンスを含んでいる
- その他のフォーマット・エラーを含んでいる

表 6-3: データ例外

メッセージ	値	説明
算術オーバーフローが起きました。	22003	次のような条件のときに発生する。 <ul style="list-style-type: none"> • 算術演算子または <code>sum</code> 関数を使い、真数値型が精度または位取りを失う場合。 • トランケート、丸め、または <code>sum</code> 関数によって、概数値型が精度または位取りを失う場合。
データの例外 - 文字列データが正しくトランケートされました。	22001	挿入または更新するデータに対して、 <code>char</code> 、 <code>unichar</code> 、 <code>univarchar</code> 、または <code>varchar</code> カラムが短すぎる場合、またはブランク以外の文字がトランケートされた場合。
ゼロによる除算が行われました。	22012	数値式が評価され除数の値が 0 の場合。
無効なエスケープ文字が検出されました。有効な文字を形成できるだけのバイト数がありません。	22019	指定されたパターンに一致する文字列の検索中に、複数文字からなるエスケープ・シーケンスが検出された場合。
パターン文字列が無効です。エスケープ文字に続く文字は、パーセント記号、アンダースコア、左大カッコ、右大カッコ、またはエスケープ文字でなければなりません。	22025	次のように検索文字列が特定のパターンと一致した場合。 <ul style="list-style-type: none"> • エスケープ文字の直後にパーセント記号、アンダースコア、またはエスケープ文字自身がない場合。 • エスケープ文字が 1 または 2 文字以外の部分文字列にパターンを区切る場合。

整合性の制約違反

`insert`、`update`、`delete` 文が、主キー制約、外部キー制約、検査制約、一意性制約、ユニーク・インデックスに違反していると、整合性の制約違反が発生します。

表 6-4: 整合性の制約違反

メッセージ	値	説明
オブジェクト <code>object_name</code> (ユニーク・インデックス <code>index_name</code> 付き) に、重複キーを持つローを挿入しようとしました。	23000	重複するローが一意性制約、またはユニーク・インデックスのあるテーブルに挿入された場合。
チェック制約違反が発生しました。データベース名 = <code>database_name</code> 、テーブル名 = <code>table_name</code> 、制約名 = <code>constraint_name</code> 。	23000	<code>update</code> または <code>delete</code> がカラムの検査制約に違反した場合。

メッセージ	値	説明
参照整合性の制約で従属する外部キーの制約違反があります。 データベース名 = database_name、 テーブル名 = table_name、制約名 = constraint_name。	23000	プライマリ・キー・テーブルの update または delete が外部キー制約に違反した場合。
外部キーの制約違反が発生しました。データベース名 = database_name、テーブル名 = table_name、制約名 = constraint_name。	23000	プライマリ・キー・テーブルに一致する値がない状態で外部キー・テーブルの insert または update を実行した場合。

無効なカーソル状態

無効なカーソル条件が発生する原因は次のとおりです。

- **fetch** が現在オープンしていないカーソルを使用する。
- **update where current of** または **delete where current of** が、修正または削除されたカーソル・ローに影響する。
- **update where current of** または **delete where current of** が、フェッチされていないカーソル・ローに影響する。

表 6-5: 無効なカーソル状態

メッセージ	値	説明
オープンしていないカーソル cursor_name を使用しようとした。詳細は、システム・ストア・プロシージャ sp_cursorinfo で確認してください。	24000	オープンしていないカーソルを使ってフェッチしようとしたか、 commit 文または (明示的および暗黙的な) rollback 文によってクローズされたカーソルを使ってフェッチしようとした場合。カーソルを再度オープンして、 fetch を繰り返して実行する必要がある。
更新または削除によって現在のカーソルの位置が削除されたために、カーソル cursor_name が暗黙にクローズされました。カーソル・スキャンの位置は回復できませんでした。これは、複数のテーブルを参照するカーソルで起こります。	24000	複数テーブルのカーソルのジョイン・カラムに削除または変更が加えられた場合。別の fetch を発行して、カーソルを再度配置する必要がある。
DELETE/UPDATE WHERE CURRENT OF、または通常の探索型の DELETE/UPDATE のために、カーソル cursor_name の現在のスキャンの位置が削除されています。UPDATE または DELETE WHERE CURRENT OF を実行する前に、新しく FETCH を実行しなければなりません。	24000	削除または変更が加えられた現在のカーソル位置で update/delete where current of を発行した場合。 update/delete where current of を再実行する前に別の fetch を発行する必要がある。
カーソル cursor_name がローに位置付けされていないため、UPDATE/DELETE WHERE CURRENT OF が実行できませんでした。	24000	次のような状態のカーソル上で update/delete where current of を発行した場合。 <ul style="list-style-type: none"> • まだローをフェッチしていない場合。 • 結果セットの最後に達した後にさらにローをフェッチした場合。

構文エラーおよびアクセス・ルール違反

SQL 文内に終了していないコメント、Adaptive Server がサポートしていない暗黙的なデータ型の変換、または他の不正な構文がある場合に、構文エラーが生成されます。

存在しないオブジェクトまたは適正なパーミッションを得ていないオブジェクトにアクセスしようとすると、アクセス・ルール違反になります。

表 6-6: 構文エラーおよびアクセス・ルール違反

メッセージ	値	説明
オブジェクト <code>object_name</code> 、データベース <code>database_name</code> 、所有者 <code>owner_name</code> について <code>command</code> パーミッションが拒否されました。	42000	適正なパーミッションを持たないユーザがオブジェクトにアクセスしようとした場合。
データ型 <code>'datatype'</code> から <code>'datatype'</code> への暗黙の変換はできません。CONVERT 関数を使用して、このクエリを実行してください。	42000	あるデータ型を、Adaptive Server が暗黙に変換することができない別のデータ型にユーザが変換しようとした場合。
<code>object_name</code> の近くに、構文エラーがあります。	42000	指定したオブジェクトの近くで不正な SQL 構文が検出された場合。
<code>insert</code> エラー。カラム名または与えられた値の数がテーブルの定義と一致しません。	42000	挿入時に無効なカラム名を使用した場合、または値の数を間違えて挿入した場合。
コメント終了マーク <code>*/</code> がありません。	42000	<code>/*</code> 開始デリミタで始まるコメントに、 <code>*/</code> 終了デリミタがない場合。
<code>object_name</code> が見つかりません。 <code>owner.objectname</code> を指定するか、 <code>sp_help</code> を使用してオブジェクトの存在を確認してください (<code>sp_help</code> ではその他のオブジェクトも表示されます)。	42000	自分が所有していないオブジェクトを参照した場合。別のユーザが所有しているオブジェクトを参照する場合は、オブジェクト名をその所有者名で修飾すること。
<code>object_name</code> に与えられるサイズ (<code>size</code>) は、最大値を超えています。許される最大のサイズは <code>size</code> です。	42000	次のような条件のときに発生する。 <ul style="list-style-type: none"> テーブル定義のすべてのカラムの合計サイズがロー・サイズの許容値を超える場合。 単一のカラムまたはパラメータのサイズがそのデータ型の許容値を超える場合。

トランザクション・ロールバック

独立性レベルが3に設定されても、Adaptive Serverが同時トランザクションの直列化を保証できない場合にトランザクションのロールバックが発生します。通常、この例外は、ディスク・クラッシュやオフライン・ディスクなどのシステム障害が原因で発生します。

表 6-7: トランザクション・ロールバック

メッセージ	値	説明
サーバ・コマンド (プロセス id #process_id) がデッドロックを検出しました。コマンドを再実行してください。	40001	Adaptive Server が、複数の同時トランザクションの直列化を保証できないことを検出した場合。

with check option 違反

ビューによって挿入または更新するデータをビューで表示することができない場合に、この例外クラスが発生します。

表 6-8: with check option 違反

メッセージ	値	説明
対象のビューは、WITH CHECK OPTION によって作成されたか、WITH CHECK OPTION によって作成された他のビュー上に定義されているために、挿入または更新が失敗しました。コマンドからの 1 個以上の結果ローが、WITH CHECK OPTION 制約での条件を満たしません。	44000	ビュー、またはそのビューに依存しているビューが with check option 句で作成された場合。

索引

記号

- ::= (BNF 表記)
 - SQL 文内 xii
- * (アスタリスク)
 - 乗法演算子 327
 - 長さを超過した数字 263
- _ (アンダースコア)
 - オブジェクト識別子プレフィクス 295, 336
 - テンポラリ・テーブル名 337
 - 文字列のワイルドカード 347, 348
- & (アンパサンド) “and” ビット処理演算子 328
- <= (以下) 比較演算子 330
- >= (以上) 比較演算子 330
- ¥ (円記号)
 - money データ型 17
 - 識別子 336
- ¥ (円記号)、文字列の改行 335
- [] (角カッコ)
 - SQL 文内 xii
 - 文字セットのワイルドカード 347, 348
- [^] (角カッコと脱字記号) 文字セット・ワイルドカード 347
- () (カッコ)
 - SQL 文内 xii
 - 複数の式 334
- ,(カンマ)
 - money 値のデフォルトの出力フォーマット 16
 - SQL 文内 xii
 - 通貨値で使用できないカンマ 17
- :(コロン) ミリ秒値の前 116
- /(スラッシュ) 算術演算子 (除法) 327
- ^ (脱字記号)
 - “exclusive or” ビット処理演算子 328
 - ワイルドカード文字 347, 349
- {} (中カッコ)
 - SQL 文内 xii
- £ (通貨ポンド記号)
 - money データ型 17
 - 識別子 336
- =(等号) 比較演算子 330
- .. (ドット) のデータベース・オブジェクト名での使用 342
- \$ (ドル記号)
 - money データ型 17
 - 識別子 336
- || (二重パイプ)
 - 文字列連結演算子 329
- % (パーセント記号)
 - 算術演算子 (モジュロ) 327
 - ワイルドカード文字 347
- |(パイプ) “or” ビット処理演算子 328
- ~ (波形記号) “not” ビット処理演算子 328
- <> (等しくない) 比較演算子 330
- ¥!=(等しくない) 比較演算子 330
- . (ピリオド)
 - 修飾子名のセパレータ 341
 - ミリ秒値の前 116
- + (プラス)
 - null 値 329
 - 算術演算子 327
 - 整数データ 12
 - 文字列連結演算子 329
- (マイナス記号)
 - 算術演算子 327
 - 整数データ 12
 - 負の通貨値 17
- > (より大きい) 比較演算子 330
- !> (より大きくない) 比較演算子 330
- < (より小さい) 比較演算子 330
- !< (より小さくない) 比較演算子 330
- “ ” (引用符)
 - datetime 値を囲む 19
 - 空文字列を囲む 333, 334
 - 比較演算子 330
 - 複数の式 334
 - リテラル指定 334
- @@authmech グローバル変数 317
- @@bootcount グローバル変数 317
- @@boottime グローバル変数 317
- @@bulkarraysize グローバル変数 317
- @@bulkbatchsize グローバル変数 318
- @@char_convert グローバル変数 318

索引

- @@cis_rpc_handling グローバル変数 318
- @@cis_version グローバル変数 318
- @@client_csexpansion グローバル変数 318
- @@client_csid グローバル変数 318
- @@client_csname グローバル変数 318
- @@cmpstate グローバル変数 318
- @@connections グローバル変数 318
- @@cpu_busy グローバル変数 318
- @@curluid グローバル変数 318
- @@cursor_rows グローバル変数 318
- @@datefirst グローバル変数 319
- @@dbts グローバル変数 319
- @@error グローバル変数 319
- @@errorlog グローバル変数 319
- @@failedoverconn グローバル変数 319
- @@fetch_status グローバル変数 319
- @@guestuserid グローバル変数 319
- @@hacmpservername グローバル変数 319
- @@haconnection グローバル変数 319
- @@heapmssize グローバル変数 319
- @@identity グローバル変数 319
- @@idle グローバル変数 319
- @@invaliduserid グローバル変数 319
- @@io_busy グローバル変数 319
- @@isolation グローバル変数 319
- @@kernel_addr グローバル変数 319
- @@kernel_size グローバル変数 319
- @@kernelmode グローバル変数 319
- @@langid グローバル変数 319
- @@language グローバル変数 319
- @@lastkpgendate グローバル変数 319, 320
- @@lastlogindate グローバル変数 320
- @@lock_timeout グローバル変数 320
- @@max_connections グローバル変数 320
- @@max_precision グローバル変数 320
- @@maxcharlen グローバル変数 320
- @@maxgroupid グローバル変数 320
- @@maxpagesize グローバル変数 320
- @@maxspid グローバル変数 320
- @@maxsuid グローバル変数 320
- @@maxuserid グローバル変数 320
- @@mempool_addr グローバル変数 320
- @@min_poolsize グローバル変数 320
- @@mingroupid グローバル変数 320
- @@minspid グローバル変数 320
- @@minsuid グローバル変数 320
- @@minuserid グローバル変数 320
- @@monitors_active グローバル変数 320
- @@ncharsize グローバル変数 320
- @@nestlevel グローバル変数 320
- @@nodeid グローバル変数 320
- @@optgoal グローバル変数 320
- @@options グローバル変数 320
- @@optlevel グローバル変数 320
- @@opttimeout グローバル変数 320
- @@pack_received グローバル変数 320
- @@pack_sent グローバル変数 321
- @@packet_errors グローバル変数 321
- @@pagesize グローバル変数 321
- @@parallel_degree グローバル変数 321
- @@probesuid グローバル変数 321
- @@procid グローバル変数 321
- @@recovery_state グローバル変数 321
- @@remotestate グローバル変数 321
- @@repartition_degree グローバル変数 321
- @@resource_granularity グローバル変数 321
- @@rowcount グローバル変数 321
- @@scan_parallel_degree グローバル変数 321
- @@servername グローバル変数 321
- @@setrowcount グローバル変数 321
- @@shmem_flags グローバル変数 322
- @@spid グローバル変数 322
- @@sqlstatus グローバル変数 322
- @@ssl_ciphersuite グローバル変数 322
- @@stringsize グローバル変数 322
- @@tempdbid グローバル変数 322
- @@textcolid グローバル変数 37, 322
- @@textdataptid グローバル変数 322
- @@textdbid グローバル変数 37, 322
- @@textobjid グローバル変数 37, 322
- @@textptid グローバル変数 322
- @@textptr グローバル変数 37, 322
- @@textptr_parameters グローバル変数 322
- @@textsize グローバル変数 37, 322
- @@textts グローバル変数 37, 322
- @@thresh_hysteresis グローバル変数 322
- @@timeticks グローバル変数 322
- @@total_errors グローバル変数 322
- @@total_read グローバル変数 322
- @@total_write グローバル変数 322
- @@tranchained グローバル変数 323
- @@trancount グローバル変数 323
- @@transactional_rpc グローバル変数 323
- @@transtate グローバル変数 323
- @@unicharsize グローバル変数 323
- @@version グローバル変数 323
- @@version_as_integer グローバル変数 323
- @@version_number グローバル変数 323

数字

- “0x”プレフィクス 29, 30
- 2つの一重引用符の使用
 - 複数の式 334
 - 文字列 26
- 21世紀番号 19

A

- abort オプション、`lct_admin` 関数 167
- abs 算術関数 46
- ACF。「Application Context Facility」参照
- acos 算術関数 47
- alter table コマンド、`timestamp` カラムの追加 285
- and キーワード
 - 範囲終了 331
 - 複数の式 333
- and (&) ビット処理演算子 328
- ANSI SQL のデータ型 10
- Application Context Facility (ACF) 236
- arithabort オプション、`set arith_overflow` 9
- ASCII 文字 48
- ascii 文字列関数 48
- asehostname 関数 49
- asin 算術関数 50
- atan 算術関数 51
- atn2 算術関数 52
- audit_event_name 関数 55
- authmech システム関数 57
- avg 集合関数 53

B

- Backus Naur Form (BNF) 表記 xi, xii
- between キーワード 331
- bigint データ型 12
- biginttohex データ型変換関数 58
- binary
 - sort 250
 - 式 325
 - 式、連結 329
 - ソート 82
 - データ型 29–31
 - データ型、“0x”プレフィクス 29
 - データ型、“x”プレフィクス 30
 - データ型、後続ゼロ 29
 - ビット処理演算子のデータのバイナリ表現 328

- binary データ型 29–31
- bintostr 関数 59
- bit データ型 31
- BNF 表記、SQL 文内 xi, xii

C

- cache_usage 関数 60
- caldayofweek 日付要素 116
- calweekofyear 日付要素 116
- calyearofweek 日付要素 116
- case 式 61–63, 190–191
 - null 値 62, 75, 190
- cast 関数 64–66
- cdw。「caldayofweek 日付要素」参照
- ceiling 算術関数 67
- char データ型 24–26
 - 複数の式 334
- char 文字列関数 69
- char_length 文字列関数 71
- character データ型 24–28
- charindex 文字列関数 73
- coalesce 関数 75–76
- coalesce キーワード、`case` 75
- col_length システム関数 77
- col_name システム関数 78
- compare システム関数 79
- convert データ型変換関数 84
 - 日付スタイル 85
 - 連結 329
- cos 算術関数 89
- cot 算術関数 90
- count 集合関数 91
- count_big 集合関数 93–94
- CP 850 スカンジナビア語
 - 辞書 82, 250
- CP 850 代替言語
 - アクセント記号なし 82, 250
 - 大文字小文字の優先設定なし 82, 250
 - 小文字優先 82, 250
- create table コマンドと null 値 333
- create_locator システム関数 95
- current_date 日付関数 96, 97, 98
- current_time 日付関数 99
- curunreservedpgs システム関数 100
- cwk。「calweekofyear 日付要素」参照
- cyr。「calyearofweek 日付要素」参照

索引

D

data_pages システム関数 102–103
datachange システム関数 104–105
datalength システム関数 106
 col_length との比較 77
date and time データ型 19–24
date データ型 18
dateadd 日付関数 107
datediff 関数 111–112
datediff 日付関数 110
datefirst オプション、*set* 114, 118
dateformat オプション、*set* 20
datetime 日付関数 114
datetime 日付関数 116
datetime データ型 19–24
 値と比較 23
 比較 330
 日付関数 117
 変換 23
day 日付関数 120
day 日付要素 116
dayofyear 日付要素の省略形と値 116
db_id システム関数 123, 125
db_instanceid システム関数 124
db_name システム関数 125
db_recovery_status 関数 126
DB-Library でのオーバフロー・エラー 54, 270
DB-Library プログラム、オーバフロー・
 エラー 54, 270
decimal データ型 12–14
degrees 算術関数 127
delete コマンドおよび *text* ロー 36
derived_stat システム関数 128
difference 文字列関数 133
double precision データ型 15

E

e または E 指数表記
 float データ型 5
 money データ型 17
 概数値データ型 15
escape キーワード 350–351
exp 算術関数 135

F

float データ型 15
floor 算術関数 136

G

GB ピンイン 82, 250
get_appcontext セキュリティ関数 138
getdate 日付関数 139
getutcdate による GMT の取得 140
guest ユーザ 293

H

has_role システム関数 141
hash システム関数 143
hashbytes システム関数 145
hexbintoint データ型変換関数 147
hexint 関数 147, 148
hexint データ型変換関数 148
host_id システム関数 149
host_name システム関数 150
hour 日付要素 116

I

ID
 sa_role およびデータベース所有者 293
 サーバ・ユーザ (*suser_id*) 273
 ユーザ (*user_id*) 293
identity_burn_max 関数 152
ID、サーバの役割と *role_id* 229
ID、ユーザ
 user_id 関数 272
 サーバ・ユーザ 273
 データベース (*db_id*) 123
image データ型 32–41
 null 値 35
 禁止されている動作 37
 初期化 34
index_col システム関数 153
index_colorder システム関数 154
index_name システム関数 155
instance_id システム関数 151
instance_name 関数 156
int データ型 12
 集合関数 54, 270

intohex データ型変換関数 157
 is_quiesced 関数 158–159
 is_sec_service_on セキュリティ関数 160
 is_singleusermode システム関数 161
 isdate システム関数 162
 isnull システム関数 163
 isnumeric 関数 164
 ISO 8859-5 キリル語辞書 82, 250
 ISO 8859-5 ロシア語辞書 82, 250
 ISO 8859-9 トルコ語辞書 82, 251
 iso_1 文字セット 343
 isql ユーティリティ・コマンド
 『ASE ユーティリティ・ガイド』参照
 概数値データ型 15

L

latin-1 英語、フランス語、ドイツ語
 アクセント記号なし 82, 250
 辞書 82, 250
 latin-1 スペイン語
 アクセント記号なし 82, 250
 大文字小文字の区別なし 82, 250
 lc_id 関数 165
 lc_name 関数 166
 lct_admin システム関数 167, 168
 left システム関数 170
 len 文字列関数 171
 license_enabled システム関数 172
 like キーワード
 日付の検索 23
 ワイルドカード文字 347
 list_appcontext セキュリティ関数 173
 locator_literal システム関数 174
 locator_valid システム関数 175
 lockscheme システム関数 176
 log 算術関数 176, 177
 log10 算術関数 178
 longsysname データ型 32
 lower 文字列関数 179
 lprofile_id 文字列関数 180
 lprofile_name 文字列関数 181
 ltrim 文字列関数 182

M

Macintosh 文字セット 343
 max 集合関数 183
 millisecond 日付要素 116
 min 集合関数 184
 minute 日付要素 116
 mi. 「minute 日付要素」参照
 mm. 「month 日付要素」参照
 model データベース、ユーザ定義データ型 43
 money データ型 16
 算術演算子 16
 money データ型の負の記号 (-) 17
 month 日付関数 185
 month 日付要素 116
 mut_excl_roles システム関数 186

N

“N/A”、“N/A”の使用 333
 nchar データ型 25–26
 newid システム関数 187
 next_identity システム関数 189
 “none”、“NULL”の使用 333
 not like キーワード 346
 not null 値
 スペース 28
 null カラムの内部データ型 9
 「データ型」参照
 null 値
 text カラムと image カラム 35
 カラムのデータ型の変換 28
 デフォルト・パラメータ 332
 複数の式 331
 null 値、where 句 332
 null 文字列、文字カラム 268, 333
 nullif キーワード 190
 nullif 式 190–191
 numeric データ型 12
 nvarchar データ型 25–26
 スペース 25

O

object_attr システム関数 192
 object_id システム関数 196
 object_name システム関数 197
 object_owner_id 関数 198
 order by 句 247

P

pagesize システム関数 199
 partition_id 関数 201
 partition_name 関数 202
 partition_object_id システム関数 203
 patindex 文字列関数 204
 text/image 関数 39
 pi 算術関数 207
 power 算術関数 208
 proc_role システム関数 209
 pssinfo 関数 211

Q

quarter 日付要素 116

R

radians 算術関数 212
 rand 算術関数 213, 214
 rand2、算術関数 214
 readtext コマンドおよび *text* データの初期化要件 36
real データ型 15
 replicate 文字列関数 215
 reserve オプション、*lct_admin* 関数 167
 reserve_identity 関数 216
 reserved_pages システム関数 219
 return_lob システム関数 223
 reverse 文字列関数 224
 right 文字列関数 225, 226
 rm_appcontext セキュリティ関数 227
 role_contain システム関数 228
 role_id システム関数 229
 role_name システム関数 230
 roles
 has_role を使用したチェック 141
 proc_role を使用したチェック 209
 show_role を使用したシステム定義役割の表示
 243
 round 算術関数 231
 row_count システム関数 233
 rtrim 文字列関数 234

S

sdci_intempdbconfig 関数 235
 second 日付要素 116
 select コマンド 247
 for browse 284
 set_appcontext セキュリティ関数 236
 setdata システム関数 238
 Shift-JIS バイナリ順 83, 251
 show_cached_plan_in_xml システム関数 239
 show_dynamic_params_in_xml システム関数 242
 show_role システム関数 243
 show_sec_services セキュリティ関数 244
 sign 算術関数 245
 sin 算術関数 246
 smalldatetime データ型 19
 日付関数 117
 smallint データ型 12
 smallmoney データ型 16
 sortkey 関数 247
 sortkey システム関数 247
 soundex 文字列関数 252
 sp_bindefault システム・プロシージャおよびユーザ定義
 データ型 43
 sp_bindrule システム・プロシージャおよびユーザ定義
 データ型 43
 sp_help システム・プロシージャ 43
 space 文字列関数 253
 spid_instance_id システム関数 254
 SQL (Sybase データベースでの使用)。
 「Transact-SQL」参照
 SQL 規格
 連結 329
 SQL 内の整数データ 325
 SQLSTATE コード 357-362
 例外 358-362
 sqrt 算術関数 256
 square 算術関数 255
 stddev 統計集合関数。「stddev_samp」参照
 stddev_pop 統計集合関数 260
 stddev_samp 統計集合関数 261
 stdev 統計集合関数。「stddev_samp」参照
 stdevp 統計集合関数。「stddev_pop」参照
 str 文字列関数 262, 263
 str_replace 文字列関数 264
 strtobin システム関数 266
 stuff 文字列関数 267
 substring 文字列関数 269
 sum 集合関数 270
 suser_id システム関数 272
 suser_name システム関数 273

syb_quit システム関数 274
 syb_sendmsg 関数 275
 sys_tempdbid システム関数 276
 syscolumns テーブル 31
 sysindexes テーブルおよび *name* カラム 35
 sysname データ型 32
 sysserverroles テーブルと *role_id* システム関数 229

T

tan 算術関数 277
 tempdb データベース、ユーザ定義データ型 43
 tempdb と tempdb_id システム関数 278
 tempdb_id システム関数 278
 text 関数と image 関数
 textptr 279
 textvalid 280
 text データおよび *image* データの記憶領域の管理 35
 text データ型 32-41
 convert コマンド 38
 null 値 35
 null 値での初期化 34
 禁止されている動作 37
 text データ型と *ascii* 文字列関数 48
 text または *image* カラムの初期化 36
 textptr 関数 279
 textptr テキストとイメージ関数 279
 textvalid テキストとイメージ関数 280
 then キーワード。「when...then 条件」参照
 timestamp データ型 17
 tsequal 関数による比較 284
 自動更新 17
 ブラウザ・モード 17, 284
 tinyint データ型 12
 to_unichar 文字列関数 281
 tran_dumpstable_status 文字列関数 282
 Transact-SQL
 予約語 353-354
 Transact-SQL 拡張機能 10
 true/false (真/偽) データ、*bit* カラム 31
 tsequal システム関数 284

U

UDP のメッセージ機能 275
 uhighsurr 文字列関数 286
 ulowsurr 文字列関数 287
 unitext データ型 32-41

unsigned bigint データ型 12
 unsigned int データ型 12
 unsigned smallint データ型 12
 upper 文字列関数 288, 289
 us_english 言語、曜日設定 118
 uscalars 文字列関数 289
 used_pages システム関数 290
 user システム関数 292
 user_id システム関数 293
 user_name システム関数 294
 using bytes オプション、
 patindex 文字列関数 199, 204, 205

V

valid_name システム関数 295
 文字セットの変更後の使用 343
 valid_user システム関数 296
 var 統計集合関数。「var_samp」参照
 var_pop 統計集合関数 298
 var_samp 統計集合関数 299
 varbinary データ型 29-30, 247
 varchar データ型 25-26
 datetime 値の変換 23
 スペース 25
 複数の式 334
 variance 統計集合関数。「var_samp」参照
 varp 統計集合関数。「var_pop」参照

W

week 日付要素 116
 weekday 日付要素 116
 when キーワード。「when...then 条件」参照
 when...then 条件 61
 where 句、null 値 332
 wk。「week 日付要素」参照
 workload_metric システム関数 302
 writetext コマンドおよび *text* データの初期化要件 36

X

xa_bqual システム関数 303
 xa_gtrid システム関数 305
 xact_connmigrate_check システム関数 307
 xact_owner_instance システム関数 308
 xmlextract システム関数 309

索引

xmlparse システム関数 310
xmlpresentation システム関数 311
xmltable システム関数 312
xmltest システム関数 313
xmlvalidate システム関数 314

Y

year 日付関数 315
year 日付要素 116
yes/no データ、*bit* カラム 31
yy。「year 日付要素」参照

あ

空きページ、*curunreservedpgs* システム関数 101
アクセントの区別、ワイルドカード文字 347
アスタリスク (*)
乗法演算子 327
長さを超過した数字 263
値の比較
difference 文字列関数 133
timestamp 284
複数の式 330
アプリケーション・コンテキスト
削除 227
取得 138
設定 236
リスト作成 173
アプリケーション・コンテキストの削除 227
アプリケーション・コンテキストの設定 236
アプリケーションの属性 236
アンダースコア ()
オブジェクト識別子プレフィクス 295, 336
テンポラリ・テーブル名 337
文字列のワイルドカード 347, 348
暗黙の変換、データ型 8, 334

い

一重引用符。「引用符」参照
インデックス
sysindexes テーブル 35
「クラスタード・インデックス」「データベース・オブジェクト」「ノンクラスタード・インデックス」参照

引用符 (“ ”)
空文字列 333, 334
比較演算子 330
複数の式 334
リテラル指定 334

う

埋め込みスペース。「スペース」「文字」参照
埋め込み、データ
ゼロ 29
テンポラリ・テーブル名のアンダースコア 337
ブランク 25

え

エスケープ文字 350
エラー
cast 関数 65
convert 関数 87
ドメイン 65, 87
円記号 (¥)
money データ型 17
識別子 336
円記号 (¥)、文字列の改行 335
演算子
算術 327
比較 330
ビット処理 328–329
優先度 327

お

大文字と小文字の区別
SQL xiii
識別子 337
比較演算子 330, 347
大文字の優先度を付けた順位 337
「大文字と小文字を区別」「order by 句」参照
オブジェクト識別子の欧州文字 343
オブジェクト名、データベース
「識別子」参照
ユーザ定義データ型名 43
オブジェクト。「データベース・オブジェクト」「データベース」参照

か

- 改行、文字列 335
- 概数値データ型 14
- 階層
 - 演算子 327
 - 「優先度」参照
- 角カッコ []
 - SQL 文内 xii
 - 脱字記号ワイルドカード文字 [^] 347, 349
 - ワイルドカード指定子 347
- 角カッコ。「角カッコ []」参照
- 各月の第 1 日目、数 112
- 角度、算術関数 47
- カッコ ()
 - SQL 文内 xii
 - この索引の「記号」の項も参照
 - 式 334
- 各国文字。「*nchar* データ型」参照
- 可変長文字。「*varchar* データ型」参照
- 加法演算子 (+) 327
- 空の文字列 (“”) または (‘ ’)
 - シングル・スペース 28
 - null として評価されない 333
 - シングル・スペース 334
- カラム
 - サイズ (リスト) 2
 - 識別 341
 - 長さ 77
 - 長さの定義 77
- カラム識別子。「識別子」参照
- カラム名
 - 修飾子 341
 - 返却 78
- 関係式 326
 - 「比較演算子」参照
- 監査
 - audit_event_name* 関数 55
- 関数
 - abs* 算術関数 46
 - acos* 算術関数 47
 - ascii* 文字列関数 48
 - asehostname* 関数 49
 - asin* 算術関数 50
 - atan* 算術関数 51
 - atn2* 算術関数 52
 - authmech* システム関数 57
 - avg* 集合関数 53
 - biginttohex* データ型変換関数 58
 - bintostr* 59
 - cache_usage* 60
 - cast* 関数 64–66
 - ceiling* 算術関数 67
 - char* 文字列関数 69
 - char_length* 文字列関数 71
 - charindex* 文字列関数 73
 - coalesce* 関数 75–76
 - col_length* システム関数 77
 - col_name* システム関数 78
 - compare* システム関数 79
 - convert* データ型変換関数 84
 - cos* 算術関数 89
 - cot* 算術関数 90
 - count* 集合関数 91
 - count_big* 集合関数 93–94
 - create_locator* システム関数 95
 - current_date* 日付関数 96, 97, 98
 - current_time* 日付関数 99
 - curunreservedpgs* システム関数 100
 - data_pages* システム関数 102–103
 - datachange* システム関数 104–105
 - datalength* システム関数 106
 - dateadd* 日付関数 107
 - datediff* 日付関数 110
 - datetime* 日付関数 114
 - datepart* 日付関数 116
 - day* 日付関数 120
 - db_id* システム関数 123, 125
 - db_instanceid* システム関数 124
 - db_recovery_status* 126
 - degrees* 算術関数 127
 - derived_stat* システム関数 128
 - difference* 文字列関数 133
 - exp* 算術関数 135
 - floor* 算術関数 136
 - get_appcontext* セキュリティ関数 138
 - getdate* 日付関数 139
 - has_role* システム関数 141
 - hash* システム関数 143
 - hashbytes* システム関数 145
 - hextobigint* データ型変換関数 147
 - hextoint* データ型変換関数 148
 - host_id* システム関数 149
 - host_name* システム関数 150
 - index_col* システム関数 153

- index_colorder システム関数 154
- index_name システム関数 155
- instance_id システム関数 151
- instance_name 156
- inttohex データ型変換関数 157
- is_quiesced 関数 158–159
- is_sec_service_on セキュリティ関数 160
- is_singleusermode システム関数 161
- isdate システム関数 162
- isnull システム関数 163
- isnumeric 164
- lc_id 165
- lc_name 166
- lct_admin システム関数 167
- left システム関数 170
- len 文字列関数 171
- license_enabled システム関数 172
- list_appcontext セキュリティ関数 173
- locator_literal システム関数 174
- locator_valid システム関数 175
- lockscheme システム関数 176
- log 算術関数 177
- log10 算術関数 178
- lower 文字列関数 179
- lprofile_id 文字列関数 180
- lprofile_name 文字列関数 181
- ltrim 文字列関数 182
- max 集合関数 183
- min 集合関数 184
- month 日付関数 185
- mut_excl_roles システム関数 186
- newid システム関数 187
- next_identity システム関数 189
- object_attr システム関数 192
- object_id システム関数 196
- object_name システム関数 197
- object_owner_id 198
- pagesize システム関数 199
- partition_id 201
- partition_id システム関数 201
- partition_name 202
- partition_name システム関数 202
- partition_object_id 203
- partition_object_id システム関数 203
- patindex 文字列関数 204
- pi 算術関数 207
- power 算術関数 208
- proc_role システム関数 209
- pssinfo 211
- pssinfo システム関数 211
- radians 算術関数 212
- rand 算術関数 213, 214
- replicate 文字列関数 215
- reserve_identity 関数 216
- reserved_pages システム関数 219
- return_job システム関数 223
- reverse 文字列関数 224
- right 文字列関数 225
- rm_appcontext セキュリティ関数 227
- role_contain システム関数 228
- role_id システム関数 229
- role_name システム関数 230
- round 算術関数 231
- row_count システム関数 233
- rtrim 文字列関数 234
- set_appcontext セキュリティ関数 236
- setdata システム関数 238
- show_cached_plan_in_xml システム関数 239
- show_dynamic_params_in_xml システム関数 242
- show_role システム関数 243
- show_sec_services セキュリティ関数 244
- sign 算術関数 245
- sin 算術関数 246
- sortkey 247
- sortkey システム関数 247
- soundex 文字列関数 252
- space 文字列関数 253
- spid_instance_id システム関数 254
- sqrt 算術関数 256
- square 算術関数 255
- stddev 統計集合関数。「stddev_samp」参照
- stddev_pop 統計集合関数 260
- stddev_samp 統計集合関数 261
- stdev 統計集合関数。「stddev_samp」参照
- stdevp 統計集合関数。「stddev_pop」参照
- str 文字列関数 262
- str_replace 文字列関数 264
- strtobin システム関数 266
- stuff 文字列関数 267
- substring 文字列関数 269
- sum 集合関数 270
- suser_id システム関数 272
- suser_name システム関数 273

syb_quit システム関数 274
 syb_sendmsg 275
 systempbid システム関数 276
 tan 算術関数 277
 tempdb_id システム関数 278
 textptr テキストとイメージ関数 279
 textvalid テキストとイメージ関数 280
 to_unichar 文字列関数 281
 tran_dumpable_status 文字列関数 282
 tsequal システム関数 284
 uhighsurr 文字列関数 286
 ulowsurr 文字列関数 287
 upper 文字列関数 288
 uscalar 文字列関数 289
 used_pages システム関数 290
 user システム関数 292
 user_id システム関数 293
 user_name システム関数 294
 valid_name システム関数 295
 valid_user システム関数 296
 var 統計集合関数。「var_samp」参照
 var_pop 統計集合関数 298
 var_samp 統計集合関数 299
 variance 統計集合関数。「var_samp」参照
 varp 統計集合関数。「var_pop」参照
 workload_metric システム関数 302
 xa_bqual システム関数 303
 xa_grid システム関数 305
 xact_connmigrate_check システム関数 307
 xact_owner_instance システム関数 308
 xmlextract システム関数 309
 xmlparse システム関数 310
 xmlpresentation システム関数 311
 xmltable システム関数 312
 xmltest システム関数 313
 xmlvalidate システム関数 314
 year 日付関数 315
 関数、組み込み、型変換 84-88
 カンマ (,)
 money 値のデフォルトの出力フォーマット 16
 SQL 文内 xii
 通貨値で使用できないカンマ 17

き

キーワード 353-356
 Transact-SQL 336, 353-354
 記号
 SQL 文内 xii
 算術演算子 327
 識別子名 336
 照合文字列 347
 通貨 336
 比較演算子 330
 ワイルドカード 347
 「ワイルドカード文字」「記号」参照
 規則
 Transact-SQL 構文 xii
 「構文」参照
 識別子名 341
 リファレンス・マニュアル xi
 ギリシャ文字 343
 キリル文字 343

<

句読表記、識別子で使用できる文字 336
 組み込み関数 46-315
 データ型変換 84-88
 組み込み関数、ACF 236
 クライアント、ホスト・コンピュータ名 150
 位取り、データ型 13
 decimal 8
 IDENTITY カラム 12
 数値 8
 データ型変換時のロス 10
 グローバル変数 319
 @@authmech 317
 @@bootcount 317
 @@boottime 317
 @@bulkarraysize 317
 @@bulkbatchsize 318
 @@char_convert 318
 @@cis_rpc_handling 318
 @@cis_version 318
 @@client_cexpansion 318
 @@client_csid 318
 @@client_csname 318
 @@cmpstate 318
 @@connections 318
 @@cpu_busy 318
 @@curloid 318

索引

`@@cursor_rows` 318
`@@datefirst` 319
`@@dbts` 319
`@@error` 319
`@@errorlog` 319
`@@failedoverconn` 319
`@@fetch_status` 319
`@@guestuserid` 319
`@@hacmpservername` 319
`@@hacnconnection` 319
`@@heapmemsize` 319
`@@identity` 319
`@@idle` 319
`@@invaliduserid` 319
`@@io_busy` 319
`@@isolation` 319
`@@kernel_addr` 319
`@@kernel_size` 319
`@@kernelmode` 319
`@@langid` 319
`@@language` 319
`@@lastkpgendate` 319, 320
`@@lastlogindate` 320
`@@lock_timeout` 320
`@@max_connections` 320
`@@max_precision` 320
`@@maxcharlen` 320
`@@maxgroupid` 320
`@@maxpagesize` 320
`@@maxspid` 320
`@@maxsuid` 320
`@@maxuserid` 320
`@@mempool_addr` 320
`@@min_poolsize` 320
`@@mingroupid` 320
`@@minspid` 320
`@@minsuid` 320
`@@minuserid` 320
`@@monitors_active` 320
`@@ncharsize` 320
`@@nestlevel` 320
`@@nodeid` 320
`@@optgoal` 320
`@@options` 320
`@@optlevel` 320
`@@opttimeout` 320
`@@pack_received` 320
`@@pack_sent` 321
`@@packet_errors` 321
`@@pagesize` 321
`@@parallel_degree` 321
`@@probesuid` 321
`@@procid` 321
`@@recovery_state` 321
`@@remotestate` 321
`@@repartition_degree` 321
`@@resource_granularity` 321
`@@rowcount` 321
`@@scan_parallel_degree` 321
`@@servername` 321
`@@setrowcount` 321
`@@shmem_flags` 322
`@@spid` 322
`@@sqlstatus` 322
`@@ssl_ciphersuite` 322
`@@stringsize` 322
`@@tempbid` 322
`@@textcolid` 322
`@@textdataptid` 322
`@@textdbid` 322
`@@textobjid` 322
`@@textptid` 322
`@@textptr` 322
`@@textptr_parameters` 322
`@@textsize` 322
`@@textts` 322
`@@thresh_hysteresis` 322
`@@timeticks` 322
`@@total_errors` 322
`@@total_read` 322
`@@total_write` 322
`@@tranchained` 323
`@@trancount` 323
`@@transactional_rpc` 323
`@@transtate` 323
`@@unicharsize` 323
`@@version` 323
`@@version_as_integer` 323
`@@version_number` 323

け

言語、代替

日付要素への影響 119

曜日順 118

現在のユーザ

`suser_id` システム関数 272

`suser_name` システム関数 273

`user_id` システム関数 293

`user_name` システム関数 294

役割 243

検索

- サーバ・ユーザ ID 272
- サーバ・ユーザ名 273, 274, 284, 290
- 式の開始位置 73
- データベース ID 123
- データベース名 125
- 有効な識別子 295
- ユーザ ID 293
- ユーザ・エイリアス 296
- ユーザ名 292, 294

減法演算子 (-) 327

J

更新

- ブラウザ・モード中の防止 284
- ブラウザ・モード内 284
- 「変更」参照 17

後続ブランク。「ブランク」参照

構文規則、Transact-SQL xii

コード、soundex 252

午前 0 時、数 112

固定長カラム

- binary データ型 29
- null 値 9
- 文字データ型 25

小文字、ソート順 337

「大文字と小文字を区別」参照

コロン (:), ミリ秒値の前 116

混合データ型、算術演算 327

K

サーバのユーザ名と ID

- suser_id 関数 272
- suser_name 関数 273

サイズ

- floor 算術関数 136
- image データ型 32
- pi 207
- text データ型 32
- カラム 77
- 識別子 (長さ) 336
- 「長さ」「数」「範囲」「サイズの制限」「領域の割り付け」参照

サイズの制限

- binary データ型 29
- char カラム 25
- double precision データ型 15
- float データ型 15
- image データ型 29
- money データ型 16
- nchar カラム 26
- nvarchar カラム 26
- real データ型 15
- varbinary データ型 29
- varchar カラム 25
- 概数値データ型 15
- 固定長カラム 25
- 真数値データ型 11
- 整数の最小値と最大値 136
- データ型 2

削除

- stuff 関数を使用した文字の削除 268
- 先行または後続ブランク 182

サブクエリ

- any キーワード 331
- 複数の式 331

サブクエリを含む all キーワード 331

三角関数 277

算術

- 演算子、式 327
- 演算、概数値データ型 14
- 演算、真数値データ型 11
- 演算、通貨データ型 16
- 式 326

算術関数

- abs 46
- acos 47
- asin 50
- atan 51
- atn2 52
- ceiling 67
- cos 89
- cot 90
- degrees 127
- exp 135
- floor 136
- log 177
- log10 178
- pi 207
- power 208
- radians 212

索引

- rand 213, 214
- round 231
- sign 245
- sin 246
- sqrt 256
- square 255
- tan 277

参照されるオブジェクトの所有権 343

し

シード値と rand 関数 213

式

- null 値を含む 331
- 引用符で囲む 334
- タイプ 325
- 定義 325
- 名前とテーブル名の修飾 342
- 式での any キーワード 331
- 式での exists キーワード 331
- 式での in キーワード 331
- 式での is not null キーワード 331
- 式での not キーワード 331
- 式での null キーワード 331
- 式での or キーワード 333
- 式内で引用符で囲む 334

識別子 335–343

short 336

大文字と小文字の区別 337

システム関数 295

長さ 335

名前の変更 343

識別子としてのユニーク名 338

時刻値

データ型 18–23

指数値 135

指数、データ型 (e または E)

float データ型 5

概数値データ型 15

通貨型 17

システム関数

authmech 57

col_length 77

col_name 78

compare 79

create_locator 95

curunreservedpgs 100

data_pages 102–103

datachange 104–105

datalength 106

db_id 123, 125

db_instanceid 124

derived_stat 128

has_role システム関数 141

hash システム関数 143

hashbytes 145

host_id 149

host_name 150

index_col 153

index_colorder 154

index_name 155

instance_id 151

is_singleusermode 161

isdate 162

isnull 163

lct_admin 167

left 170

license_enabled 172

locator_literal 174

locator_valid 175

lockscheme 176

mut_excl_roles 186

newid システム関数 187

next_identity 189

object_attr 192

object_id 196

object_name 197

pagesize 199

proc_role システム関数 209

reserved_pages 219

return_lob 223

role_contain 228

role_id 229

role_name 230

row_count 233

setdata 238

show_cached_plan_in_xml 239

show_dynamic_params_in_xml 242

show_role 243

sortkey 247

spid_instance_id 254

strtobin 266

suser_id 272

suser_name 273

syb_quit 274

sys_tempdbid 276

tempdb_id 278

tsequal 284

used_pages 290

user 292
 user_id 293
 user_name 294
 valid_name 295
 valid_user 296
 workload_metric 302
 xa_bqual 303
 xa_grid 305
 xact_connmigrate_check 307
 xact_owner_instance 308
 xmlextract 309
 xmlparse 310
 xmlpresentation 311
 xmltable 312
 xmltest 313
 xmlvalidate 314
 システム・データ型。「データ型」参照
 システム・テーブルおよび *sysname* データ型 32
 システムの役割と *show_role* 243
 自然対数 176, 177
 自動オペレーション、*timestamp* でのカラムの更新 17
 集合関数
 avg 53
 count 91
 count_big 93–94
 max 183
 min 184
 sum 270
 修飾されたオブジェクト名の中のビュー名 341
 修飾子名 341, 343
 順序
 「インデックス」「優先度」「ソート順」参照
 式中の演算子の実行 327
 式の文字順を逆にする 224
 日付要素 20
 曜日番号 118
 ジョイン
 count または count(*) 91, 93
 null 値 332
 照合
 名前とテーブル名 342
 「パターン一致」参照
 小数点
 使用できるデータ型 12
 整数データ 12
 小数点数
 round 関数 231
 str 関数、表現 263
 乗法演算子 (*) 327

省略形
 chars (characters の省略形)、patindex 199, 204
 日付要素 116
 省略した名前要素のドット (.) 342
 除法演算子 (/) 327
 シングルバイト文字セット、*char* データ型 24
 真数値データ型 11–14
 算術演算子 11

す

数 (量)
 count(*) 内のロー 91, 93
 各月の第 1 日目 112
 午前 0 時 112
 日曜日 112
 数値
 オブジェクト ID 196
 データベース ID 123
 浮動小数の乱数 213, 214
 文字列の変換 28
 曜日名 118
 数値式 325
 round 関数 231
 スクロール可能カーソル
 @@rowcount 318
 スタイル値、日付表示 85
 ストアド・プロシージャ、LOB の使用 39
 スペース内の Not null 値 28
 スペース、文字
 like *datetime* 値 23
 空の文字列 (“”) または (‘’) 333, 334
 識別子で使用できる文字 336
 テキスト文字列内への挿入 253
 「ブランク」参照
 文字データ型 25–28
 スラッシュ (/) 除法演算子 327
 スレッシュホールド、ラストチャンス 168

せ

整数の剰余。「モジュロ演算子 (%)」参照
 整数の引数からバイナリ数への変換 328
 正接、算術関数 277
 精度、データ型
 概数値データ型 15
 真数値型 13
 通貨型 16

索引

セキュリティ関数

- get_appcontext 138
- is_sec_service_on 160
- list_appcontext 173
- rm_appcontext 227
- set_appcontext 236
- show_sec_services 244

ゼロ x (0x) 29, 30

ゼロ、後続、バイナリ・データ型 29–30

先行ゼロ、自動挿入 30

先行ブランク、ltrim 関数を使用した削除 182

先行ブランク。「ブランク」「スペース」「文字」参照

そ

挿入

- 自動先行ゼロ 30

- テキスト文字列内のスペース 253

ソート順

- 比較演算子 330

- 文字列照合動作 247

属性、アプリケーションでの設定 236

速度 (サーバ)

- binary と varbinary データ型アクセス 29

た

タイ語辞書 82, 250

対数、底が 10 178

ダブルバイト文字。「マルチバイト文字セット」参照

単語、発音の似ている単語の検索 252

探索条件と datetime データ 23

ち

中カッコ {}, SQL 文内 xii

重複ロー、text または image 39

つ

追加

- タイムスタンプ・カラム 285

- 日付への間隔の追加 108

- ユーザ定義データ型 43

通貨

- 記号 336

- デフォルトのカンマの位置 16

通貨記号 17, 336

通貨ポンド記号 (£)

- money データ型 17

- 識別子 336

月の値と日付要素の省略形 116

て

底 10 の対数関数 178

定数

- 式 325

- 式での比較 334

データ型 1–43

- ANSI SQL 10

- binary 29–31

- binary での後続のゼロ 29

- bit 31

- datetime 値の比較 330

- decimal 12–14

- Transact-SQL 拡張機能 10

- varbinary 247

- 概数値 14

- 階層 6

- 混合、算術演算 327

- 真数値 11–14

- 整数 11–12

- 同義語 2

- 日付と時刻 18–23

- まとめ 2–4

- ユーザ定義 10

- 「ユーザ定義型」「各データ型名」参照

- ユーザ定義の削除 43

データ型の同義語 2

データ型の優先度。「優先度」参照

データ型のリスト 6

データ型変換

- biginttohex 58

- convert 関数 84, 87

- hextobigint 147

- hextoint 148

- hextoint 関数 147, 148

- image 65, 87

- inttohex 157

- ドメイン・エラー 65, 87

データベース
 ID 番号、`db_id` 関数 123
 「データベース・オブジェクト」参照
 名前の取得 125
 データベース・オブジェクト
 ID 番号 196
 個々のオブジェクト名参照
 識別子名 335
 ユーザ定義データ型 43
 データベース・オブジェクトの所有者と識別子 342
 データベース所有者
 オブジェクトと識別子 342
 修飾子としての名前 341, 342
 テーブル
 識別 341
 修飾子としての名前 341
 テーブルのページ
 「ページ」「データ」参照
 テキストの重複。「`replicate` 文字列」参照
 テキスト・ページ・ポインタ 77
 テキスト・ポインタ値 279
 デバイス。「`sysdevices` テーブル」参照
 デフォルト設定
 日付表示フォーマット 18, 21
 曜日順 118
 デフォルト値
 データ型長 84
 データ型の位取り 84
 データ型の精度 84
 テンポラリー・テーブル、命名 337
`sysobjects` 337
 埋め込み 337
 バイト数 337

と

同義語、`chars` と `characters`、`patindex` 199, 204
 統計集合関数
`stddev_pop` 260
`stddev_samp` 261
`stddev`。「`stddev_samp`」参照
`stdevp`。「`stddev_pop`」参照
`stdev`。「`stddev_samp`」参照
`var_pop` 298
`var_samp` 299
`variance`。「`var_samp`」参照
`varp`。「`var_pop`」参照
`var`。「`var_samp`」参照

トランケーション
`arithabort numeric_truncation` 9
`binary` データ型 29
`datediff` の結果 111
 テンポラリー・テーブル名 337
 文字列 25
 トリガ。「データベース・オブジェクト」「ストアド・プロシージャ」参照
 ドル記号 (\$)

- `money` データ型 17
- 識別子 336

 度、ラジアンへの変換 212

な

内部構造、使用されるページ 219
 長さ

- カラム 77
- 「サイズ」参照
- 式(バイト単位) 106
- 識別子 335

 名前

- `db_name` 関数 125
- `index_col` とインデックス 153
- `object_name` 関数 197
- `suser_name` 関数 273
- `user_name` 関数 294
- `valid_name` を使用したチェック 343
- 「識別子」参照
- 省略した要素 (..) 342
- データベース・オブジェクトの修飾 341, 343
- 発音の似ている名前の検索 252
- 日付要素 116
- ホスト・コンピュータ 150
- 曜日番号 118

に

二重パイプ (||)

- 文字列連結演算子 329

 日曜日、数値 112
 日本語文字セットとオブジェクト識別子 343

索引

は

- パーセント記号 (%)
 - モジュロ演算子 327
 - ワイルドカード文字 347
- 倍精度浮動小数点値 15
- バイト
 - text* と *image* データ 37
- パターン一致 344
 - charindex* 文字列関数 73
 - difference* 文字列関数 133
 - patindex* 文字列関数 205
- 発音が似ている単語。「*soundex* 文字列関数」参照
- 発音の似ている単語と名前の検索 252
- 範囲
 - datediff* の結果 111
 - 「数」「サイズ」参照
 - 認識された日付 19
 - 日付要素値 116
 - 有効な通貨値 16
 - ワイルドカード文字の指定 348, 349
- 範囲クエリ
 - and* 終了キーワード 331
 - between* 開始キーワード 331
- 番号
 - 奇数または偶数のバイナリ 30
 - 長さが超過した場合のアスタリスク (**) 263

ひ

- 比較演算子
 - 「関係式」参照
 - 記号 330
 - 複数の式 330
- 低いデータ型と高いデータ型。「優先度」参照
- 日付
 - 1753 年より前のデータ型 109
 - 使用できる最も古い日付 19, 109
 - データ型 18-23
 - デフォルト表示設定 21
 - 入力フォーマット 20
 - 比較 330
 - 表示フォーマット 18
- 日付関数
 - current_date* 96, 97, 98
 - current_time* 99
 - dateadd* 107
 - datediff* 110

- datetime* 114
- datepart* 116
- day* 120
- getdate* 139
- month* 185
- year* 315
- 日付の計算 111
- 日付要素
 - caldayofweek* 116
 - calweekofyear* 116
 - calyearofweek* 116
- 順序 20
- 省略形と値 116
- 入力 19
- ビット処理演算子 328-329
- 等しい。「比較演算子」参照
- 秒、*datediff* の結果 111
- ピリオド (.)
 - 修飾子名のセパレータ 341
 - ミリ秒値の前 116

ふ

- ブール (論理) 式 325
- フォーマット、日付。「日付」参照
- 浮動小数点データ 325
 - str* 文字表現 263
- ブラウズ・モードと *timestamp* データ型 17, 284
- プラス (+)
 - null* 値 329
 - 算術演算子 327
 - 整数データ 12
 - 文字列連結演算子 329
- プラットフォームに依存しない変換
 - 16 進文字列から整数値への変換 147, 148
 - 整数値から 16 進文字列への変換 157
- ブランク
 - like* 348
 - 後続ブランクの削除、*rtrim* 関数 234
 - 「スペース」「文字」参照
 - 先行ブランクの削除、*ltrim* 関数 182
 - 比較 330
 - 評価される空文字列 334
 - 文字データ型 25-28
- フロントエンド・アプリケーション、ブラウズ・モード 284

へ

- 平方根算術関数 256
- ページ・チェーン、*text* または *image* データ 33
- ページ、データ
 - チェーン 33
 - 内部構造で使用 219
- 変換
 - like* キーワードを使用した日付 23
 - null* 値と自動的な変換 9
 - 暗黙 8, 334
 - 大文字から小文字 179
 - 小文字から大文字 286, 287, 288, 289
 - 自動変換値 8
 - 整数値から文字値へ 69, 281
 - 度からラジアンへ 212
 - 低いデータ型から高いデータ型へ 334
 - 日付のスタイル 85
 - 文字セット間 343
 - 文字値から ASCII コードへの変換 48
 - 文字列連結 329
 - ラジアンから度へ 127

ほ

- ポインタ
 - text* および *image* ページ 279
 - text* または *image* カラム 34
 - 初期化されていない *text* または *image* カラムの *null* 279
- 他のユーザ、オブジェクトの修飾 343
- ホスト・コンピュータ名 150
- ホスト・プロセス ID、クライアント・プロセス 149

ま

- マイナス記号 (-)
 - 減法演算子 327
 - 整数データ 12
- マルチバイト文字セット
 - nchar* データ型 24
 - 識別子名 343
 - ワイルドカード文字 349

丸め 231

- datetime* 値 18
- money* 値 16
- str* 文字列関数 263
- 概数値データ型 15

み

- 短い識別子 336
- ミリ秒値、*datediff* の結果 111

め

- 明示的な *null* 値 333
- 命名
 - 規則 335-343
 - 識別子 335-343
 - データベース・オブジェクト 335-343
 - ユーザ定義データ型 43

も

- 文字
 - 数 71
 - 削除、*stuff* 関数 268
 - 「スペース」「文字」参照
 - “0x” 29, 30
 - ワイルドカード 344-351
- 文字 0x 29, 30
- 文字式
 - 構文 326
 - 定義 325
 - ブランクまたはスペース 25-28
- 文字数と日付の解釈 23
- 文字セット
 - iso_1* 343
 - オブジェクト識別子 343
 - 変換エラー 343
 - マルチバイト 343
- 文字データ、“NULL” を使用しない 333
- モジュロ演算子 (%) 327
- 文字列
 - 引用符の指定 334
 - 円記号での改行 (¥) 335
 - 空 334
 - ワイルドカード 344

索引

文字列関数

- ascii 48
- char 69
- char_length 71
- charindex 73
- difference 133
- len 171
- lower 179
- lprofile_id 180
- lprofile_name 181
- ltrim 182
- patindex 204
- replicate 215
- reverse 224
- right 225
- rtrim 234
- soundex 252
- space 253
- str 262
- str_replace 264
- stuff 267
- substring 269
- 「*text* データ型」参照
- to_unichar 281
- tran_dumpable_status 282
- uhighsurr 286
- ulowsurr 287
- upper 288
- uscalar 289

文字列、連結 329

モニタリング

システム・アクティビティ 317

や

役割階層と `role_contain` 228

役割の相互排他性と `mut_excl_roles` 186

役割、ユーザ定義と相互排他性 186

ゆ

ユーザ ID

`user_id` 関数 293

`valid_user` 関数 296

ユーザ・オブジェクト。

「データベース・オブジェクト」参照

ユーザが作成したオブジェクト。

「データベース・オブジェクト」参照

ユーザ定義データ型 10

`longsysname` 32

`sysname` 32

削除 43

作成 43

「データ型」参照

ユーザ定義の役割と相互排他性 186

ユーザ・データグラム・プロトコルの

メッセージ機能 275

ユーザ名 294

ユーザ名、検索 273, 294

優先度

式中の演算子 327

低いデータ型と高いデータ型 334

よ

曜日値、名前と番号 118

予約語 353–356

SQL92 354

Transact-SQL 353–354

「キーワード」参照

データベース・オブジェクト識別子 335, 336

より大きい。「比較演算子」参照

より小さい。「比較演算子」参照

ら

ラージ・オブジェクト (LOB)

作成 40

ストアド・プロシージャ内 39

宣言 40

ラジアン、度への変換 127

ラストチャンス・スレッシュホールド 168

ラストチャンス・スレッシュホールドと `lct_admin` 関数 168

り

リテラル値

`null` 333

データ型 6

リテラル文字の指定

`like` 一致文字列 349

引用符 (“”) 334

リファレンス情報

Transact-SQL 関数 45

データ型 1

予約語 353

リンケージ、ページ。「ページ」「データ」参照

る

ルール。「データベース・オブジェクト」参照

れ

歴史上の日付、1753 年以前 109

連結

+ 演算子を使用 329

|| 演算子を使用 329

null 値 329

ろ

論理式 325

when...then 61, 75, 190

構文 326

真理値表 333

論理式の真理値表 333

わ

ワイルドカード文字 344-351

like 一致文字列 347

「patindex 文字列関数」参照

リテラル文字 349

リテラル文字としての使用 349

