



リファレンス・マニュアル：コマンド

## **Adaptive Server<sup>®</sup> Enterprise**

15.7 ESD #2

ドキュメント ID : DC36423-01-1572-01

改訂 : 2012 年 7 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、the Sybase trademarks page (<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 Oracle およびその関連会社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

第 1 章	コマンド	1
	alter database	2
	alter encryption key	16
	alter login	25
	alter login profile	31
	alter precomputed result set	35
	alter...modify owner	37
	alter role	43
	alter table	48
	alter thread pool	97
	begin...end	99
	begin transaction	100
	break	101
	checkpoint	102
	close	104
	commit	105
	compute 句	107
	connect to...disconnect	116
	continue	119
	create archive database	120
	create database	122
	create default	138
	create encryption key	142
	create existing table	146
	create function	153
	create function (SQLJ)	156
	create index	159
	create login	182
	create login profile	185
	create plan	188
	create precomputed result set	190
	create procedure	193
	create procedure (SQLJ)	210
	create proxy_table	215
	create role	219

create rule .....	222
create schema.....	226
create service.....	228
create table .....	233
create thread pool .....	288
create trigger .....	290
create view .....	305
dbcc.....	314
deallocate cursor .....	342
deallocate locator .....	343
declare .....	344
declare cursor .....	346
delete .....	354
delete statistics.....	363
disk init .....	365
disk mirror .....	373
disk refit.....	378
disk reinit .....	379
disk remirror .....	384
disk resize .....	387
disk unmirror .....	389
drop database .....	393
drop default .....	396
drop encryption key.....	398
drop function .....	400
drop function (SQLJ) .....	401
drop index .....	402
drop login .....	404
drop login profile.....	406
drop precomputed result set .....	408
drop procedure.....	409
drop role .....	411
drop rule.....	413
drop service.....	415
drop table .....	416
drop thread pool .....	419
drop trigger .....	420
drop view.....	421
dump configuration.....	422
dump database .....	424
dump transaction.....	448
execute.....	469
fetch .....	477
goto label.....	485

grant .....	486
grant role .....	524
group by 句と having 句 .....	527
if...else .....	541
insert .....	544
kill .....	554
load database .....	557
load transaction .....	578
lock table .....	593
merge .....	596
mount .....	599
online database .....	605
open .....	608
order by 句 .....	609
prepare transaction .....	616
print .....	617
quiesce database .....	620
raiserror .....	625
readtext .....	630
reconfigure .....	635
refresh .....	636
remove java .....	637
reorg .....	640
return .....	647
revoke .....	650
revoke role .....	667
rollback .....	669
rollback trigger .....	672
save transaction .....	673
select .....	675
set .....	712
setuser .....	773
shutdown .....	775
transfer table .....	780
truncate lob .....	789
truncate precomputed result set .....	790
truncate table .....	791
union operator .....	793
unmount .....	797
update .....	800
update all statistics .....	812
update index statistics .....	817
update statistics .....	823
update table statistics .....	834

	use .....	836
	waitfor.....	837
	where 句 .....	839
	while .....	848
	writetext.....	850
<b>第 2 章</b>	<b>Interactive SQL コマンド.....</b>	<b>855</b>
	clear .....	856
	configure .....	857
	connect.....	858
	disconnect .....	861
	exit.....	862
	input .....	863
	output .....	869
	parameters .....	874
	read .....	875
	set connection .....	877
	start logging.....	878
	stop logging .....	879
	system.....	880
<b>索引</b> .....		<b>881</b>

# コマンド

ここでは、Transact-SQL 文を構成するコマンド、句、およびその他の要素について説明します。

コマンドのパーミッション・チェックは、細密なパーミッションの設定によって異なる場合があります。詳細については、各コマンドの「パーミッション」セクションを確認してください。細密なパーミッションの詳細については、『セキュリティ管理ガイド』の「細密なパーミッションの使用」を参照してください。

## alter database

**説明** データベースに加え、アーカイブされたデータベースの変更済みページ・セクションに割り当てられた空き領域を増やします。DML ログ・レベル、圧縮のデフォルト、ロー内ラージ・オブジェクト (LOB) 記憶領域など、データベース全体のプロパティを 1 つ以上変更します。

**構文**

```
alter database database_name
    [on {default | database_device} [= size]
    [, database_device [= size]]...]
    [log on {default | database_device} [= size]
    [, database_device [= size]]...]
    set { [durability = { no_recovery | at_shutdown | full}]
    [[,] dml_logging = {full | minimal}]
    [[,] template = { database_name | NULL}]
    [, [no] async_init]
    [, compression = {none | row | page}]
    [, lob_compression = {compression_level | off}]
    [,] inrow_lob_length = value [log off database_device
    [= size | [from logical_page_number] [to logical_page_number]]]
    [, database_device
    [= size | [from logical_page_number] [to logical_page_number]]]
    [with override]
    [for load]
    [for proxy_update]
```

**パラメータ** *database\_name*  
データベースの名前を指定します。データベース名には、リテラル、変数、またはストアド・プロシージャのパラメータが使用できません。

**on**  
データベース拡張のサイズおよびロケーションを示します。ログとデータが別々のデバイス・フラグメントにある場合は、on をデータ・デバイスに使用して、log on をログ・デバイスに使用します。

**default**  
alter database を使用して任意のデフォルト・データベース・デバイスにデータベース拡張を行うことを示します (『リファレンス・マニュアル：プロシージャ』の「システム・プロシージャ」の sp\_helpdevice ストアド・プロシージャを参照してください)。明確にロケーションを指定せずにデータベース拡張のサイズだけを指定するには、次のコマンドを使用してください。

```
on default = size
```

データベース・デバイスのステータスをデフォルトに変更するには、sp\_diskdefault を使用します。



**database\_device**

データベースを拡張するデータベース・デバイスの名前です。1つのデータベースは、複数のデータベース・デバイスを占有できません。この場合、各デバイスに異なる量の領域を割り当てることが可能です。データベース・デバイスを Adaptive Server<sup>®</sup> に追加するには、**disk init** を使用します。

**size**

データベース拡張に割り付ける領域の量です。指定できる単位は、"k" または "K" (キロバイト)、"m" または "M" (メガバイト)、"g" または "G" (ギガバイト)、および "t" または "T" (テラバイト) です。大文字と小文字の区別はなく、一重引用符と二重引用符のいずれも使用できます。Sybase<sup>®</sup> は常に単位を指定することをおすすめします。単位を含めない場合は、引用符を省略することができます。ただし、単位を付加する場合は、必ず引用符を付けてください。**size** の単位を指定しなかった場合、値はメガバイト単位であると見なされます。

値を指定しない場合、**alter database** は 1MB または 4 アロケーション・ユニットのどちらか大きい方の分だけデータベースを拡張します。次の表は、拡張可能な最小サイズを示します。

サーバの論理ページ・サイズ	データベースの最小拡張サイズ
2K	1MB
4K	1MB
8K	2MB
16K	4MB

**log on**

データベースのトランザクション・ログ用に追加する領域を指定することを示します。**log on** 句では、**on** 句と同じデフォルトが使用されます。

**durability**

データベースの持続性レベルを決定します。次のいずれかを指定します。

- **full** — すべてのトランザクションがディスクに書き込まれます。データベースの作成時に持続性レベルを指定しなかった場合、これがデフォルトになり、サーバの障害から完全なリカバリができるようになります。**tempdb**を除くシステム・データベースはすべて、この持続性レベル(ディスク上に常駐するデータベースの従来の持続性レベル)を使用します。**tempdb**は**no\_recovery**の持続性レベルを使用します。
- **no\_recovery** — トランザクションは、サーバの稼働中のみ、持続性が維持されます。サーバに障害が発生するか、サーバを正常に停止すると、持続性が完全に失われます。持続性が**no\_recovery**に設定されたディスク上に常駐するデータベースに対して、**Adaptive Server**は実行時にディスク・デバイスに定期的に、データを書き込みますが、この方法は制御されていません。停止(正常停止、強制停止、またはサーバの障害と再起動)後も、**no\_recovery**で作成されているデータベースはリカバリされませんが、**model**から、またテンプレート・データベースが定義されている場合はテンプレート・データベースから再作成されます。
- **at\_shutdown** — トランザクションは、サーバの稼働中と正常終了後に、持続性が維持されます。サーバに障害が発生すると、持続性が完全に失われます。

**[no] async\_init**

非同期のデータベース初期化の有効と無効を切り替えます。

**compression**

このデータベース内に新しく作成されたテーブルに **alter database** によって適用される圧縮レベルを指定します。

- **none** — データは圧縮されません。
- **row** — 個々のロー内の1つ以上のデータ項目を圧縮します。**Adaptive Server**は、非圧縮形式に比べ、圧縮形式でのデータの格納により記憶領域を節約できる場合にのみ、**row**圧縮形式でデータを格納します。
- **page** — ページがいっぱいになると、ロー圧縮された既存のデータ・ローが、ページ・レベルの圧縮を使用して圧縮され、ページ・レベルの辞書、インデックス、文字コードのエントリが作成されます。

Adaptive Server では、データがロー・レベルで圧縮されてからしかページ・レベルで圧縮されないため、圧縮を `page` に設定することは、`page` と `row` の両方が圧縮されることを意味します。

`lob_compression = off | compression_level`

新しく作成したテーブルの圧縮レベルを決定します。`off` を選択すると、テーブルに LOB 圧縮は使用されません。

`compression_level`

テーブルの圧縮レベルは次のとおりです。

- 0 – LOB カラムは圧縮されません。
- 1 ~ 9 – Adaptive Server は ZLib 圧縮を使用します。通常、圧縮の数値が大きいほど、Adaptive Server での LOB データの圧縮レベルが高くなり、圧縮データと非圧縮データの比率も大きくなります (つまり、非圧縮データと比べて、圧縮データによる記憶領域の節約量 (バイト) が多くなります)。

ただし、圧縮の量は LOB の内容によって異なります。圧縮レベルが高いほど、圧縮に使用される CPU の量が多くなります。そのためレベル 9 を使用すると、圧縮率が最高になりますが、CPU の使用量も最高になります。

- 100 – Adaptive Server は FastLZ 圧縮を使用します。CPU 使用量が最も少ない圧縮率です。通常、短いデータに使用します。
- 101 – Adaptive Server は FastLZ 圧縮を使用します。値に 101 を指定すると、100 を指定した場合よりも、CPU の使用量は若干多くなりますが、圧縮率は良くなります。

`inrow_lob_length = value`

ロー内 text または image LOB カラムのバイト数、およびロー内 Unitext LOB カラムの文字数を指定します。

`log off database_device`

データベースのログの不要部分を指定されたデータベース・デバイスから削除します。`log off` を使用すると、データベースのログに割り当てた領域の量だけでなく、アーカイブ・データベースの変更済みページ・セクションの量が減ります。

`log off` は、`log on`、`for load`、または `with override` を含むその他の `alter database` パラメータとは併用できません。

= *size*

コマンドの影響を受けるデバイスの最後の空き領域の量を指定します。この場合、デバイスの最後はそのデバイスでこのデータベースが使用する番号が最大の論理ページです。このコマンドは、指定されたデバイス上のすべてのログ・ページをデータベースから削除し、物理記憶領域を指定します。

```
alter database sales_db log off mylogdev
```

アロケーション・ユニットの数に正確に一致するように切り上げたサイズ指定。16KB の物理ページ・サイズを使用するインストールでは、そのサーバ上の各アロケーション・ユニットは 4MB であるため、50MB ではなく 52MB を削除します。Adaptive Server が指定された領域よりも小さい領域を削除するのは、そのデバイス上でデータベースのログ・セグメントが使用する領域がそれよりも小さい場合のみです。

from *logical\_page\_number*

このコマンドの影響を受ける最初のページ番号を指定します。Adaptive Server では、アロケーション・ページのページ ID を示すように、この番号が自動的に調整されます。デフォルトの from ロケーションは、デバイス上の最も小さい番号が指定された論理ページです。

to *logical\_page\_number*

このコマンドの影響を受ける最後のページ番号を指定します。削除できるのは完全なアロケーション・ユニット (256 ページの倍数) のみなので、ページ番号は Adaptive Server により自動的にアロケーション・ユニットの最後のページに上向き調整されます。

*logical\_page\_number* がアロケーション・ページと同じ番号 (256 で割り切れる数値) である場合、そのアロケーション・ユニットは影響を受けません。たとえば、to 512 は 512 までのページに影響しますが、512 ページは含まれません。

デフォルトの to ロケーションは、デバイス上で番号が最大の物理ページです。

dml\_logging {minimal | default}

insert コマンド、update コマンド、delete コマンドのログ・レベルを示します。full (デフォルト) を使用すると、すべてのトランザクションの完全な記録のログに対する変更が記録されます。データベースに minimal ロギングが使用されている場合、Adaptive Server は syslogs にローまたはページの変更のログを記録しません。しかし、Adaptive Server はメモリ内ロギング作業を行い、トランザクションのロールバックなどの実行時のオペレーションをサポートすることがあります。

**template**

データベースに使用するテンプレートを以下のいずれかです。

- **database\_name** – 持続性が **full** の、ディスク上に常駐するユーザ・データベースで、利用可能な状況にあります。
- **NULL** – 現在のテンプレート・データベースに対するバインドを削除します。データベースは、次のサーバの再起動時にテンプレート・データベースとして、**model** を使用します。

**with override**

データとトランザクション・ログが同じデバイスに混在することになり、データベースを最新のものにリカバリできなくなる可能性がある場合でも、**Adaptive Server** にデバイス仕様を強制的に指定します。このパラメータを使用しないで同じデバイスにログとデータを混在させると、**alter database** コマンドは正常に動作しません。ログとデータが混在していても、**with override** を使用している場合は、警告を受けませんが、コマンドは正常に動作します。

**for load**

ダンプからロードされているデータベースの、領域の割り付けとセグメントの使用状況を再作成する必要があるときに、**create database for load** の後にかぎって使用します。

**for proxy\_update**

プロキシ・データベース内でプロキシ・テーブルの再同期を強制的に実行します。

**例**

**例 1** デフォルト・データベース・デバイスの **2K** の論理ページに設定されているユーザ・データベースに、**3MB (1,536 ページ)** を追加します。

```
alter database mydb
```

**例 2** データベース・デバイス **newdata** にある、**pubs2** データベースに割り付けられた領域に **3MB** を追加します。

```
alter database pubs2 on newdata = 3
```

**例 3** **userdata1** のデータ用の領域に **10MB**、**2K** の論理ページに設定されているサーバの **logdev** のログ用の領域に **2MB** を追加します。

```
alter database production
on userdata1 = "10M"
log on logdev = '2.5m'
```

**例 4** リラックス持続性データベース **pubs5\_rddb** の持続性レベルを変更して、持続性が **full** の通常のデータベースに変更します。

```
alter database pubs5_rddb
set durability = full
```

**例 5** pubs3 データベースのテンプレートを変更します。

```
alter database pubs3
set template = new_pubs_template_db
```

**例 6** ディスク上に常駐するリラックス持続性データベースの持続性レベルを変更します。

```
alter database pubs7 set durability=at_shutdown
```

**例 7** 持続性レベルが full に設定されている model データベースの DML ロギング・レベルを変更します。model から作成されたデータベースは、この変更の後に、最低限のロギング・レベル・プロパティを継承します。

```
alter database model set dml_logging = minimal
```

**例 8** pubs2 データベースをページ・レベルの圧縮に変更します。

```
alter database pubs2
set compression = page
```

**例 9** LOB 圧縮を使用するように pubs2 データベースを変更します。

```
alter database pubs2 set lob_compression = 100
```

**例 10** この例では、pubs データベースを変更してロー内の LOB カラムを 400 バイトに変更します。

```
alter database pubs
set inrow_lob_length = 400
```

**例 11** 50MB のデータベース sales\_db をデバイス mylogdev から削除します。

```
alter database sales_db log off mylogdev='50M'
```

この例では、mylogdev 上にある sales\_db の物理ページを番号の大きいものから、最大 50MB まで削除します。

**例 12** 削除するページを正確に指定して、データベース sales\_db の領域をデバイス mylogdev から削除します。

```
alter database sales_db log off mylogdev from 7168 to
15360
```

論理ページ 15360 はアロケーション・ページであるため、この例は mydev 上にあるすべての論理ページ (7168 から 15359 まで) に影響します。ただし、ページ 15360 は影響を受けません。さらに、指定範囲内のページでも mylogdev に物理的に存在しないものも影響を受けません。

## 使用法

## 制限事項

- 単位を含めない場合は、引用符を省略することができます。ただし、単位を付加する場合は、必ず引用符を付けてください。
- 固定長カラムのすべてとローのオーバーヘッドの合計サイズが、テーブルのロック・スキームとページ・サイズによって許容されるサイズを超えている場合は、Adaptive Server によってエラーが報告されます。
- Adaptive Server では、`create database` と `alter database` の場合は 256 の論理ページをひとまとまりとしてデータベースの領域が割り付けられます。そのため、指定したサイズはそのサイズに最も近いアロケーション・ユニットの倍数に丸められます。
- `size` は `float` データ型として指定できますが、指定したサイズは、そのサイズに最も近いアロケーション・ユニットの倍数に丸められます。
- Adaptive Server では、次の場合はテーブルは作成されますが、データ操作言語による操作を行うとサイズ制限に関するエラーが表示されます。
  - 1つの可変長カラムの長さが、カラムの最大サイズを超えている場合。
  - データオンリー・ロック・テーブルでは、開始カラムを除いた可変長カラムのオフセットが 8191 バイトの制限を超えている場合。
- Adaptive Server は、要求された空き領域を割り付けられない場合には、それにできるだけ近い大きさの領域をデバイスごとに割り付け、各データベース・デバイスに割り付けられた空き領域を通知するメッセージを表示します。
- `alter database` を使用するには、`master` データベースを使用しているか、または `master` データベースにあるストアド・プロシージャを実行していなければなりません。
- `master` データベースは、マスタ・デバイス上でのみ拡張できます。`alter database` を使用して `master` データベースをその他のデータベース・デバイス上に拡張すると、エラー・メッセージが表示されます。たとえば、次のようにします。

```
alter database master on master = 1
```
- `create database` または `alter database` を使用して領域をデータベース・デバイスに割り付けると、割り付けはデバイス・フラグメントを表し、ローとして `sysusages` に入力されます。

- ダンプ処理中のデータベースで `alter database` を使用した場合、ダンプが終了しないと `alter database` は処理を完了できません。Adaptive Server は、データベースの領域の使用状況を示すメモリ内マップをダンプ中にロックします。このメモリ内マップがロックされている間に `alter database` を発行すると、Adaptive Server は、ダンプが完了した後にディスクからマップを更新します。`alter database` に割り込みが行われると、Adaptive Server は `sp_dbremap` を実行するように指示します。`sp_dbremap` を実行しない場合、拡張した領域は、サーバを再起動するまで Adaptive Server で使用できません。
- オフライン・データベース上で `alter database on database_device` を使用できます。

#### アーカイブ・データベースに対する `alter database` の使用

`alter database` を使用すると、領域がなくなったときにかぎらず、いつでもアーカイブ・データベースの変更済みページ・セクションの領域を追加できます。変更済みページ・セクションの領域を増やすと、サスペンドされたコマンドの処理を再開できます。構文は次のとおりです。

```
alter database database_name
  [ on database_device [= size]
    [, database_device [= size]]...]
```

#### インメモリ・データベースおよびリラックス持続性データベースの変更

- `model`、`master`、または `sysystemdb` を、テンプレート・データベースとして指定する設定はできません。
- `use template` 句のデータベース名を `NULL` にすると、既存のあらゆるテンプレート・データベースに対するバインドが削除され、`model` がテンプレート・データベースとして定義されます。
- データベース・リカバリ順序のシーケンスでそのテンプレート・データベースよりも前にあるデータベースのテンプレート定義を変更すると、サーバを再起動したときに、データベース・リカバリ順序が自動的に変更され、新しいテンプレート・データベースが従属データベースの前に表示されます。
- `durability` または `dml_logging` の設定を変更すると、`alter database` により、コマンドの実行前にデータベースのシングルユーザ・モードへの設定が自動的に試みられます。`alter table` を発行する前に、手動でデータベースをシングルユーザ・モードに設定できます。
- 持続性レベルの設定を変更するには、事前にデータベースをシングルユーザ・モードにしておく必要があります。



- インメモリ・データベースのサイズは、インメモリ・データベースをすでにホストしているメモリ内記憶域キャッシュでのみ増やすことができます。
- システム・データベース、テンプレート・データベース、またはローカル・テンポラリー・データベースの持続性レベルは変更できません。
- データベースの持続性レベルを **full** に変更すると、ロード・シーケンスが中断されます。たとえば、持続性が **full** のディスク上に常駐するデータベースの場合、
  - a データベースをダンプする
  - b `dump transaction` を実行する
  - c 2 回目の `dump transaction` を実行する
  - d 持続性を `no_recovery` に変更する
  - e 持続性を **full** に変更する

これらの操作を実行すると、3 回目の `dump transaction` を実行できません。代わりに、完全な `dump database` を実行する必要があります。

#### 領域を割り付けた後の *master* のバックアップ

- `alter database` を使用した場合は、その都度 `dump database` を使用して *master* データベースのバックアップを作成します。バックアップによって、*master* が損傷した場合でも、簡単かつ安全にリカバリできます。
- `alter database` を使用したときに *master* のバックアップをとらなかった場合でも、`disk refit` を使用すると変更内容をリカバリできる場合があります。

#### ログを別のデバイスに入れる

- `create database` に対して `log on` 拡張機能を使用しているときにトランザクション・ログに割り付ける記憶領域を増やすには、`alter database` コマンドを発行するときに `log on` 句でログのデバイス名を指定します。
- `create database` で `log on` 拡張機能を使用しないでログを別のデバイスに配置すると、ハード・ディスクの障害が発生したときに完全にリカバリできないことがあります。この場合、`log on` 句付きで `alter database` を実行してから、`sp_logdevice` を使用してログを専用のデバイスに移動することによって、ログを拡張できます。

### 圧縮用のデータベースの変更

- `set compression` は、新しく作成されたテーブルのみに適用されるデータベース全体の圧縮レベルを指定します。
- `set lob_compression` は、単独で使用することも、その他の `set` サブ句とともに使用することもできます。ただし、その他の `set` サブ句では、データベースをシングルユーザ・モードにする必要があります(データベースの持続性レベルを変更した場合など)。

### ロー内の LOB カラム

- データベース全体でロー内の LOB 長を増やしたり減らしたりするには、`inrow_lob_length` を使用します。
- `inrow_lob_length` を変更すると、将来の `create table` カラム・コマンドまたは `alter table add` カラム・コマンドによる LOB カラムの作成に影響します。有効値の範囲は 0 からデータベースの論理ページ・サイズまでです。

### ログ領域の縮小

`alter database` の `log off` 変形について

- `log off` オプションでは、削除するページの範囲を論理ページとして指定しますが、実際に削除されるのは対応する物理ページです。論理ページは、穴を形成するため、使用不可のページとしてデータベースに残ります。穴とは、対応する物理記憶領域のない 1 つまたは複数のアロケーション・ユニットのことです。
- どのアロケーション・ユニット(データベースの作成時またはデータベースへの領域の追加時に 256 データ・ページに分割される空間)が `master.dbo.sysusages` テーブルに含まれるデバイス上に存在するかに関する情報。このテーブルには、ディスク区分がデータベース ID、開始論理ページ番号、サイズ(論理ページ数)、デバイス ID、デバイス内の開始オフセットによってリストされています。
- 指定された `to` ページの値が `from` ページの値より小さい場合、これらのページが交換されます。つまり `to` ページが `from` ページになり、その逆も同様です。`from` と `to` に同じページが指定されている場合、このコマンドはそのページを含むアロケーション・ユニットのみに影響します。このコマンドでは、コマンド・エラーが生じるような `to` ページの調整は行われません。
- 句を何も指定しないと、デバイス全体に影響します。これは `log off device from 0` と同等です。このコマンドは、指定されたデバイス上のすべてのログ・ページをデータベースから削除し、物理記憶領域を指定します。

```
alter database sales_db log off mylogdev
```

- `alter database` でエラーが検出されると、このコマンドは実行されず、次のような理由を示すメッセージが返されます。
  - データベース・ログが小さくなりすぎた。
  - `syslogs` に割り付けられたページが削除するフラグメントに含まれている。つまり、アクティブ・ログ削除対象のログ・フラグメント内の領域を使用している。
  - フラグメントを削除した後のログの空き領域が小さすぎるため、ラストチャンス・スレッシュホールドに対応できない。

#### 領域の使用状況をヘルプで参照する

- データベースで使用中のデバイス・フラグメントの名前、サイズ、使用状況を確認するには、`sp_helpdb dbname` を実行してください。
- 現在のデータベースで使用されている領域を確認するには、`sp_spaceused` を実行してください。

#### system セグメントと default セグメント

- `system` セグメントと `default` セグメントは、`alter database` コマンドの `on` 句で指定された新しいデータベース・デバイスのそれぞれにマップされます。これらのセグメントのマップを解除するには、`sp_dropsegment` を使用してください。
- `alter database` を `override` 句なしで使用して、すでにデータベースで使用中のデバイス上でデータベースを拡張する場合は、そのデバイスにマップされているセグメントも拡張されます。`override` 句を使用する場合は、`on` 句内で指定されているデバイス・フラグメントはすべてシステム/デフォルト・セグメントになり、`log on` 句で指定されているデバイス・フラグメントはすべてログ・セグメントになります。

#### `alter database` を使用してスリープ中のプロセスを起動する

- ユーザ・プロセスがログ・セグメントのラストチャンス・スレッシュホールドに達しているために中断されている場合、`alter database` を使用して、そのログ・セグメントの領域を拡張します。空き領域の総量がラストチャンス・スレッシュホールドを超えると、プロセスが起動します。

*for proxy\_update* の使用

- オプションを指定しないで `for proxy_update` 句を入力した場合は、データベースのサイズは拡張されません。代わりに、プロキシ・テーブルがある場合は、プロキシ・テーブルがプロキシ・データベースから削除され、`create database ... with default_location = 'pathname'` で指定したパス名から取得したメタデータに基づいてプロキシ・テーブルが再作成されます。
- データベースのサイズを拡張するために、`alter database` を他のオプションとともに使用した場合は、サイズが拡張されてからプロキシ・テーブルが同期されます。
- `for proxy_update` を使用すると、データベース管理者は、単一のリモート・サイトにあるすべてのテーブルの正確で最新のプロキシ表示を 1 つのステップで簡単に取得できます。
- 再同期は、HA クラスタ環境内のプライマリ・サーバだけではなく、すべての外部データ・ソースでサポートされています。また、データベースは、`for proxy_update` 句によって作成されている必要はありません。`create database` コマンドまたは `sp_defaultloc` によってデフォルトの記憶ロケーションが指定されている場合、データベース内に含まれるメタデータは、リモートの記憶ロケーションにあるメタデータと同期できます。
- データベースを正しく同期させて、再ロードしたプライマリ・データベースの内容に適したスキーマがすべてのプロキシ・テーブルに定義されるようにするには、プロキシ・データベースをホストするサーバで `for proxy_update` 句を実行する必要がある場合があります。

## 標準規格

ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

## パーミッション

`sp_altermessage` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	<p>細密なパーミッションが有効化されている場合、<code>sybsecurity</code> では、データベース所有者であるか、次の権限の所有者であることが要求される。</p> <p><code>own database</code> (<code>sybsecurity</code> に対する) または <code>manage auditing</code>.</p> <p>その他のすべてのデータベースについては、データベース所有者であるか、<code>own database</code> 権限 (データベースに対する) が必要。</p>
細密なパーミッションが無効	<p>細密なパーミッションが無効化されている場合、データベース所有者であるか、<code>sa_role</code> または <code>ss_role</code> が付与されたユーザ (<code>sybsecurity</code> に対する) であることが必要。</p>

## 監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
2	alter	alter database	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – alter size</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [create database](#), [disk init](#), [drop database](#), [load database](#)

システム・プロシージャ [sp\\_addsegment](#), [sp\\_dropsegment](#), [sp\\_helpdb](#), [sp\\_helpsegment](#), [sp\\_logdevice](#), [sp\\_renamedb](#), [sp\\_spaceused](#)

## alter encryption key

**説明** 暗号化キーの現在のパスワードを変更し、キー・コピーを追加または破棄し、暗号化キーを再生成します。

暗号化カラムの詳細については、『暗号化カラム・ユーザーズ・ガイド』を参照してください。

**構文** マスタ・キーを変更するには次の構文を使用します。

```
alter encryption key [dual] master
  with char_string { add encryption
    { with passwd char_string for user user_name [for recovery]
      | for automatic_startup }
  | modify encryption { with passwd char_string [for recovery]
    | for automatic_startup }
  | drop encryption
    { for user user_name | for recovery | for automatic_startup }
  | regenerate key
    [ with passwd char_string] | recovery encryption
    with passwd char_string | modify owner user_name }
```

syb\_extpasswdkey サービス・キーを変更するには次の構文を使用します。

```
alter encryption key syb_extpasswdkey
  [ with { static key | master key}]
  { regenerate key [ with { static key | master key } ]
  | modify encryption [ with { static key | master key } ] }
```

カラム暗号化キーを変更するには次の構文を使用します。

```
alter encryption key [[database.][owner.] keyname]
  { [ as | not default ]
  [dual] master
    [ with { static key | master key} ]
    regenerate key
    [ with { static key | master key [no] dual_control} ] | [with passwd
      'password' | system_encr_passwd | login_passwd |
      'base_key_password']
  modify encryption
    [ with {passwd {'password' | system_encr_passwd |
      login_passwd } | master key } ]
    [[no] dual_control] for automatic startup
  add encryption [ with passwd 'password' | 'key_copy_password' ]
    for user user_name
    [for [login_association | recovery | automatic_startup]]
  drop encryption for { user user_name | recovery
    [ for recovery ] | [ for automatic_startup ] }
    [ [ with passwd 'password' ]
  recover encryption with passwd 'password'
    | modify owner user_name }
```

## パラメータ

*keyname*

カラム暗号化キーの名前です。

**as [not] default**

このキーに割り当てる必要がある、またはこのキーから割り当てを解除する必要があるデータベース・デフォルト・プロパティを示します。

**[dual] master**

それらが定義されているデータベース内のその他のキーの暗号化に使用するデータベース・レベルのキーです。これらのキーはデータの暗号化には使用されません。

**static key | master key**

**with {static key | master key}** 句の最初のインスタンスは、**syb\_extpasswdkey** の現在の暗号化方法の指定にすぎません。**Adaptive Server** では、**syb\_extpasswdkey** の現在の暗号化方法が認識されるため、この句はオプションです。

**regenerate key** 操作の後に **with {static key | master key}** 句の 2 番目のインスタンスを配置することで、管理者は再生成されたキーの暗号化を静的から動的またはその逆に変更できます。この句を省略すると、再生成されたキーはこのコマンドの発行前と同様に暗号化されます。

**modify encryption** 操作の後に **with {static key | master key}** 句の 3 番目のインスタンスを配置することで、指定に応じて静的キーまたはマスタ・キーを使用するように既存のキーの保護が変更されます。この句を省略すると、デフォルトで静的キーが使用されます。

**[no] dual\_control**

マスタ・キーの作成にデュアル・コントロールが使用されているかどうかを示します。

**regenerate key**

キーを再生成することを示します。

**with passwd [*password* | *system\_encr\_passwd* | *login\_password* | *base\_key\_password*]**

カラム暗号化キーを復号化するために **Adaptive Server** が使用する現在のパスワードを指定し、次のいずれかの目的のために新しいパスワードを指定します。

- キーまたはキー・コピーの暗号化の変更。
- 新たに追加されたキーのコピーを暗号化する。キー所有者は、プライベート・パスワードまたはログイン・パスワードを使用してアクセス可能な各ユーザのキー・コピーを追加できます。

- パスワードを忘れた場合の暗号化キーのリカバリ。

Adaptive Server は、次のようなキー・パスワードをサポートします。

- *password* — 最大 255 バイトまでの長さの文字列。
- *login\_passwd* — Adaptive Server にセッションのログイン・パスワードを指定する。
- *system\_encr\_passwd* — 現在のデータベースのシステム暗号化パスワード。
- *'base\_key\_password'* — ベース・キーを暗号化するために使用するパスワード。このパスワードを知っているのはキー管理者だけです。パスワードの最大長は 255 バイトです。Adaptive Server では、最初のパスワードを使用してベース・カラム暗号化キーが復号化されます。

*with passwd* を指定しない場合、デフォルトは *system\_encr\_passwd* になります。

### modify encryption

暗号化キーまたは暗号化キー・コピーを変更することを示します。

### for automatic startup

自動マスタ・キー・アクセスを有効にしてサーバが再起動された後にキー・コピーを使用してマスタ・キーまたはデュアル・マスタ・キーにアクセスすることを示します。

### add encryption

指定されたユーザの暗号化されたキー・コピーを追加します。

*key\_copy\_password* — キー・コピーの暗号化に使用されるパスワードです。パスワードの長さは 255 バイト以下にする必要があります。Adaptive Server では、復号化ベース・キーのコピーが作成され、*key\_copy\_password* から導出されたキー暗号化キーで暗号化されます。次に、暗号化ベース・キーのコピーが新しいローとして *sysencryptkeys* に保存されます。

### for user *user\_name*

どのユーザのキー・コピーを追加または削除するかを指定します。

### for login\_association

追加されるキー・コピーに、割り当てられたユーザが最初にアクセスするとき、このユーザのログイン・パスワードによってこのキーが後に暗号化されることを示します。



**for recovery**

パスワードを紛失した場合に、キー・コピーを使用してマスタ・キーをリカバリすることを示します。

**drop encryption**

指定されたユーザのキー・コピーを削除することを示します。

**recover encryption**

ベース・キーに新しいパスワードからアクセスできるようにします。キー・コピーには適用されません。

**modify owner**

キーの所有者を指定されたユーザに変更します。

例

**例 1** `my_key` をデフォルトの暗号化キーに変更します。

```
alter encryption key my_key as default
```

キーのデフォルト・プロパティを変更するには、`sso_role` か `keycustodian_role` を持つ必要があります。上のコマンドは、実行する人によって次のようになります。

- システム・セキュリティ担当者 (SSO) の場合、前のデフォルト・キーにデフォルト・プロパティが存在すると、**Adaptive Server** によりデフォルト・プロパティが無条件で削除されます。
- キー管理者の場合は、`my_key` を所有している必要があります。キー管理者は、前のデフォルト・キーが存在する場合、そのデフォルト・キーを所有している必要があります。

`my_key` からデフォルト・プロパティを削除するには、SSO または キー管理者がキー所有者として次を実行します。

```
alter encryption key my_key as not default
```

`my_key` がデフォルト・キーでない場合は、このコマンドによってエラーが返されます。

**例 2** `important_key` 暗号化キーのパスワードを変更します。

```
alter encryption key important_key
with passwd 'oldpassword'
modify encryption
with passwd 'newpassword'
```

このコマンドは、実行する人によって次のようになります。

- キー所有者 – コマンドによりベース・キーが再暗号化されます。
- キー・コピーを割り当てられているユーザ – コマンドによりそのキー・コピーが再暗号化されます。

**例 3** キー・コピーのパスワードを現在のセッションのログイン・パスワードに変更します (キー・コピーを割り当てられているユーザのみが実行できます)。

```
alter encryption key important_key
  modify encryption
  with passwd login_passwd
```

キー・コピーのみを、ログイン・パスワードで暗号化できます。ベース・キーをログイン・パスワードで暗号化しようとする、Adaptive Server からエラーが返されます。

**例 4** important\_key 暗号化キーのパスワードをシステム・パスワードに変更します。

```
alter encryption key important_key
  with passwd 'ReallyBigSecret'
  modify encryption with passwd system_encr_passwd
```

キーにキー・コピーがない場合にのみ、キー所有者または sso\_role を持つユーザだけがこのコマンドを実行できます (コピーのあるベース・キーは、ユーザが指定したパスワードによって暗号化する必要があります)。この例では、ベース・キーの暗号化が変更されます。

**例 5** important\_key 暗号化キーのパスワードをシステム暗号化パスワードから新しいパスワードに変更します。システム暗号化パスワードがデフォルト・パスワードであるため、これを文に指定する必要はありません。

```
alter encryption key important_key
  modify encryption
  with passwd 'ReallyNewPassword'
```

**例 6** パスワード "just4now" の important\_key 暗号化キーを持つユーザ "ted" の暗号化を追加します。

```
alter encryption key important_key
  with passwd 'TopSecret'
  add encryption with passwd 'just4now'
  for user 'ted'
```

このコマンドを実行するには、キー所有者または sso\_role を持つユーザである必要があります。Adaptive Server は、"TopSecret" を使用してベース・キーを復号化し、ロー・キーのコピーを作成して、パスワード "just4now" により Ted 用にキーを暗号化します。

**例 7** Ted の暗号化を変更して、新しいパスワードを使用します。Ted のみがこのコマンドを実行できます。

```
alter encryption key important_key
```

```
with passwd 'just4now'
modify encryption
with passwd 'TedsOwnPassword'
```

**例 8 important\_key** 暗号化キーのユーザ "ted" の暗号化を削除します (このコマンドを実行するには、sso\_role を持っているか、キー所有者である必要があります)。

```
alter encryption key important_key
drop encryption for user 'ted'
```

**例 9 important\_key** の所有者を新しい所有者 "tinnap" に変更します (このコマンドを実行するには、sso\_role が付与されているか、キー所有者である必要があります)。

```
alter encryption key important_key modify owner tinnap
```

**例 10 マスタ・キー** を使用して既存の CEK “k2” を暗号化します。

```
alter encryption key k2
with passwd 'goodbye'
modify encryption
with master key
```

**例 11** 現在マスタ・キーによって暗号化されている既存の CEK “k3” を再暗号化し、デュアル・コントロールを使用するようにします。

```
alter encryption key k3
modify encryption
with master key
dual_control
```

## 例 12

**例 13** リカバリ・キー・コピーを設定して、パスワードが失われたときにキー・リカバリに使用します。

- 1 キー管理者が、パスワードで保護された新しい暗号化キーを作成します。

```
create encryption key key1 for AES passwd 'loseitl8ter'
```

- 2 キー管理者は、Charlie 用に key1 の特別な暗号化キー・リカバリ・コピーを追加します。

```
alter encryption key key1 with passwd 'loseitl8ter'
add encryption
with passwd 'temppasswd'
for user charlie
for recovery
```

- 3 Charlie は、リカバリ・コピーに別のパスワードを割り当て、このパスワードを安全な場所に保管します。

```
alter encryption key key1
  with passwd 'temppasswd'
  modify encryption
  with passwd 'finditl8ter'
  for recovery
```

- 4 キー管理者がベース・キーのパスワードを紛失した場合、Charlie からパスワードを取得し、次のコマンドを使用してキーのリカバリ・コピーからベース・キーをリカバリできます。

```
alter encryption key key1
  with passwd 'finditl8ter'
  recover encryption
  with passwd 'newpasswd'
```

## 使用法

- SSO が `alter encryption key` を発行して、キーをデータベースのデフォルトとして設定した場合、既存のキーは指定されたキーにデフォルトとして置き換えられます。
- キー管理者が `alter encryption key` を発行して、キーをデータベースのデフォルトとして設定した場合、指定されたキーと現在のデフォルト・キー（存在する場合）は、キー管理者が所有する必要があります。
- キーは、`keycustodian_role` または `sso_role` を持つユーザか、`create encryption key` コマンドのパーミッションが明示的に付与されているユーザにより、所有および管理されます。キーは、暗号化カラムからデータを処理したり、表示したりするパーミッションを持つすべてのユーザにより使用されます。Adaptive Server によるキーの保護方法により、キーへのアクセス方法が変化します。
  - a キー所有者は、システム暗号化パスワードにより暗号化キーを作成します。ユーザが暗号化データにアクセスすると、Adaptive Server によりシステム暗号化パスワードを使用してベース・キーが復号化されます。キー所有者は、ユーザのために個々のキー・コピーを作成するわけではありません。
  - b キー管理者は、明示的なパスワードによりベース・キーを暗号化します。キー・コピーを作成する代わりに、キー管理者は暗号化データを処理するすべてのユーザとこのパスワードを共有します。ユーザまたはアプリケーションは、`set encryption passwd` コマンドでこのパスワードを指定してデータにアクセスする必要があります。set encryption passwd を参照してください。

- c キー管理者は、エンド・ユーザのためにキー・コピーを追加するため、ユーザはパスワードを共有する必要がありません。ユーザは、`set encryption passwd` でキー・コピーのパスワードを入力して暗号化カラムにアクセスする必要があります。または、キー管理者はキーを割り当てる人のログイン・パスワードにより暗号化用のキー・コピーを設定できます。このパスワードは、`set encryption passwd` を使用して入力する必要はありません。
- 暗号化キーを使用してキーを作成するときに、Adaptive Server は暗号化された状態のキーに加え、キーのプロパティを `sysencryptkeys` のローとして保存します。このローがベース・キーを表します。キー所有者は、ベース・キーによる暗号化データへの独占アクセスを許可するか、`alter encryption key` を使用して個々のユーザのためのキー・コピーを追加するかを選択できます。
  - `alter encryption` に `with passwd` パラメータを含めない場合、Adaptive Server はシステム暗号化パスワードを使用します。
  - システム暗号化パスワードを使用してコピーを持つキーのベース・キーを変更することはできません。また、システム暗号化パスワードを使用してキーのコピーを暗号化することもできません。
  - キー・コピーを割り当てられたユーザは、所有するキー・コピーのみを変更します。
  - `login_association` を指定した場合、Adaptive Server は、システム暗号化パスワードを使用してキー・コピーを一時的に暗号化します。キー・コピーは、コピーの所有者がそのキーを使用してデータを暗号化または復号化するときに、コピーの所有者のログイン・パスワードによって再暗号化されます。
  - 同一のキー・コピーに対して `for recovery` と `login_association` を指定することはできません。

#### パーミッション

`alter encryption key` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	<p>細密なパーミッションが有効化されている場合、<code>alter encryption key</code> をデフォルトまたはデフォルト以外として実行するには、<code>manage column encryption key</code> 権限を持つユーザであることが必要。</p> <p>キー所有者であるか、キー・タイプに応じた次の権限が必要。</p> <ul style="list-style-type: none"><li>• カラム暗号化キー：<code>manage column encryption key</code></li><li>• マスタ・キー：<code>manage master key</code></li><li>• サービス・キー：<code>manage service key</code></li></ul> <p>上記を次のように変更する。</p> <ul style="list-style-type: none"><li>• <code>alter encryption key</code> を使用してキー・コピーの追加や削除、キーのリカバリ、キー所有者を変更する。</li><li>• <code>alter encryption key</code> を実行して、ベース・キーのパスワードを変更する。</li></ul> <hr/> <p><b>注意</b> キー・コピーのパスワードを変更するには、キー・コピーが割り当てられたユーザであることが必要です。ユーザは、自分のキー・コピーのパスワードを変更する暗黙的なパーミッションを持ちます。</p>
細密なパーミッションが無効	<p>細密なパーミッションが無効化されている場合、<code>alter encryption key</code> をデフォルトまたはデフォルト以外として実行するには、<code>sso_role</code> または <code>keycustodian_role</code> が付与されたユーザであることが要求される。</p> <p>次の操作を行うには、システム・セキュリティ担当者またはキー所有者であることが要求される。</p> <ul style="list-style-type: none"><li>• <code>alter encryption key</code> を使用してキー・コピーの追加や削除、キーのリカバリ、キー所有者を変更する。</li><li>• <code>alter encryption key</code> を実行して、ベース・キーのパスワードを変更する。</li></ul> <hr/> <p><b>注意</b> キー・コピーのパスワードを変更するには、キー・コピーが割り当てられたユーザであることが必要。ユーザは、自分のキー・コピーのパスワードを変更する暗黙的なパーミッションを持ちます。</p>
監査	暗号化カラムの監査の詳細については、『暗号化カラム・ユーザーズ・ガイド』の「暗号化カラムの監査」を参照してください。
参照	<a href="#">create encryption key</a> 、 <a href="#">drop encryption key</a> 、 <a href="#">sp_encryption</a>

## alter login

説明	ログイン・アカウントの属性を変更します。
構文	<pre>alter login login_name { [modify attribute_value_pair_list]   [add auto activated roles role_name [, role_name_list]]   [drop auto activated roles { ALL   role_name [, role_name_list]}]   [drop attribute_name_list]   [ with password caller_password modify password [immediately] new_loginName_password ] }</pre>
パラメータ	<p><i>login_name</i> 変更するログイン・アカウントの名前を指定します。</p> <p><i>modify</i> 属性が存在する場合、指定された新しい値に属性値を変更します。属性が存在しない場合は、指定された一連の属性、および対応する値がログイン・アカウントに追加されます。<i>attribute_value_pair_list</i> は、属性名および指定された値です。以下から 1 つまたは複数を選択します。</p>

パラメータ	パラメータ値	説明
login profile	有効な値は次のとおりです。 <ul style="list-style-type: none"> <li>• login_profile_name</li> <li>• ignore</li> <li>• default</li> </ul>	<ul style="list-style-type: none"> <li>• <b>login_profile_name</b> は、指定されたログイン・プロファイルを指定されたログイン・アカウントにバインドします。ログイン・プロファイルのバインドがすでに存在する場合、それは指定されたログイン・プロファイルで置換されます。</li> <li>• <b>ignore</b> は、ログイン・プロファイルのバインドをすべて消去します。ログイン・プロファイルのバインドが存在する場合、それは削除されます。デフォルト・ログイン・プロファイルは適用されず、属性はリリース 15.7 以前と同様に適用されます。</li> <li>• <b>default</b> は、既存のログイン・プロファイルのバインドを解除し、そのログイン・プロファイルをログイン・アカウントに関連付けます。ログイン・プロファイル <i>login_profile_name</i> を指定しない場合、デフォルト・ログイン・プロファイルが存在すると、そのデフォルト・ログイン・プロファイルがログイン・アカウントに関連付けられます。</li> </ul>

パラメータ	パラメータ値	説明
fullname	name_value	ログイン・アカウントを所有するユーザのフル・ネームです。フル・ネームを追加するか、既存の名前を変更します。 デフォルトは NULL です。
password expiration	有効値の範囲：0 ～ 32767 日	パスワード有効期間です。 デフォルトは 0 で、パスワードの期限は切れないことを意味します。
min password length	有効値の範囲：0 ～ 30。	必要な最小パスワード長です。 デフォルトは 6 です。
max failed attempts	有効値の範囲：-1 ～ 32767。	許容されるログインの試行回数。この回数を超えるとアカウントはロックされます。 -1 は、失敗回数は追跡されるがロックはされないことを示します。 デフォルトは 0 で、失敗回数は追跡されず、ログインの失敗によりアカウントがロックされることはないことを意味します。
authenticate with	有効な値：ASE、LDAP、PAM、KERBEROS、ANY	ログイン・アカウントの認証に使用するメカニズムを指定します。  ANY を使用すると、Adaptive Server によって定義済みの外部認証メカニズムがあるかどうか調べられます。定義済みの外部認証メカニズムが検出された場合は、それが使用され、それ以外の場合は、ASE メカニズムが使用されます。  authenticate with authentication mechanism を指定しない場合、ログイン・アカウントには ANY が使用されます。
default database	default_database_name	デフォルトとなるデータベースを指定します。 デフォルトは Master です。
default language	default_language	デフォルトとなる言語を指定します。 デフォルトは us_english です。
login script	login_script_name	有効なストアド・プロシージャを指定します。ログイン・スクリプトの場合、120 文字に制限されます。
exempt inactive lock	有効な値：TRUE または FALSE。	非アクティブなログイン・アカウントに対してロックの適用を除外するかどうかを指定します。 デフォルトは FALSE で、アカウントは適用除外の対象ではないことを示します。



**add auto activated roles**

ログイン時に自動的にアクティブ化する必要のある、ユーザ定義の役割を指定します。この役割は以前に付与されており、パスワード保護されていません。

**drop auto activated roles**

以前に付与されたユーザ定義の役割は、ログイン時に自動的にアクティブ化しないことを指定します。ALL は、付与されたすべてのユーザ定義役割を指定します。

**drop**

指定された属性をログイン・アカウントから削除します。削除する属性を以下から 1 つまたは複数指定します。

パラメータ	
login profile	指定されたログイン・アカウントからログイン・プロファイルのバインドを削除します。login profile ignore パラメータを指定した場合、このパラメータは削除され、既存のログイン・プロファイルが無視されることはなくなります。
fullname	ログイン・アカウントに関連付けられている名前を削除します。
password expiration	パスワード有効期限の値を削除します。
min password length	最小パスワード長の制限を削除します。
max failed attempts	許容される試行回数の制限を削除します。
authenticate with	認証メカニズムの仕様を削除します。
default database	デフォルトのデータベースの仕様を削除します。
default language	デフォルト言語の仕様を削除します。
login script	ログイン・スクリプトを適用する指定を削除します。
exempt inactive lock	非アクティブなためにログイン・アカウントをロックするかどうかの指定を削除します。デフォルト値 FALSE を設定します。アカウントは適用除外の対象になりません。

**with password caller\_password modify new\_loginName\_password**

ログイン・パスワードを新しく指定されたパスワードに変更します。

**immediately**

ログインしているユーザにパスワードの変更をすぐに適用するかどうかを、次のように指定します。

- **immediately** を指定する — `syslogins` テーブルのパスワードはすぐに変更され、ログインしているユーザのパスワードは、ログイン中でも、更新されます。
- **immediately** を指定しない — 呼び出し元を除き、ログインしたすべてのユーザの古いパスワードが、再接続時まで保持されます。

**例** **例 1** ログイン・プロファイル `emp_lp` をログイン・アカウント `ravi` にバインドします。

```
alter login ravi modify login profile emp_lp
```

**例 2** `ignore` を指定すると、すべてのログイン・プロファイルは、`users_1` ログイン・アカウントにバインドされているか、定義済みのデフォルト・ログイン・プロファイルにバインドされているかにかかわらず、無視されます。

```
alter login users_1 modify login profile ignore
```

**例 3** 2つのログイン・プロファイルを作成します。1つ目はデフォルト・プロファイルの `general_lp`、2つ目のログイン・プロファイル名は `emp_lp` で、特定のグループの従業員に対して定義されます。ログイン・プロファイルを作成すると、両方のログイン・プロファイルの属性がログイン・アカウントに適用されます。属性の適用順序の詳細については、『セキュリティ管理ガイド』の「ログイン・プロファイル属性とパスワード・ポリシー属性の適用」を参照してください。

```
create login profile general_lp as default with default database master
default language us_english
track lastlogin true authenticate with ASE
```

```
create login profile emp_lp with default database empdb authenticat with
LDAP
```

次のコマンドは、ログイン・プロファイル `emp_lp` をログイン・アカウント `users_2` にバインドします。 `default language` および `track lastlogin` は、ログイン・プロファイル `emp_lp` では定義されていませんが、デフォルト・ログイン・プロファイルでは定義されています。したがって、`default language` および `track lastlogin` の値は、`general_lp` から適用されます。

```
alter login users_22 modify login profile emp_lp
```

**例 4** 2つのログイン・プロファイルを作成します。1つ目は新しい従業員用の `newEmployee_lp`、2つ目は既存の従業員用の `default_lp` です。

```
create login profile newEmployee_lp with login script "newEmp_script"
```

```
create login profile default_lp as default with login script "def_script"
```

次のコマンドは、ログイン時にログイン・スクリプト `newEmp_script` を `employee_new` に適用します。

```
create login employee_new with password myPasswd33 login profile
"newEmployee_lp"
```

ログイン・プロファイル `default_lp` は、ログイン時に、ログイン・プロファイルによってログイン・スクリプトが指定されていない既存のアカウントに適用されます。

**例 5** 契約社員に付与され自動的にアクティブ化されるさまざまな役割の強制方法を示します。

```
create login profile contractEmp_lp
grant role access_role to contractEmp_lp
alter login profile contractEmp_lp add auto activated roles access_role
create login contractEmp_emp1 with password c_Emp43 login profile
"contract_lp"
create login contractEmp_emp2 with password c_Emp44 login profile
"contract_lp"
create login contractEmp_emp3 with password c_Emp44 login profile
"contract_lp"
```

#### 使用法

異なるログイン・プロファイルから属性が取得された場合や、`sp_passwordpolicy` を使用して値が指定されている場合、ログイン・アカウント属性の適用方法は優先度の規則によって決定されます。

優先度の規則については、『セキュリティ管理ガイド』の「ログイン・プロファイル属性とパスワード・ポリシー属性の適用」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

#### パーミッション

`alter login` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効な場合、一般にアカウントに対する <code>alter login</code> を使用するには <code>manage any login</code> 権限が必要である。ログイン・アカウントのパスワードを変更するには、 <code>change password privilege</code> を所有しているか、アカウントの所有者であることが要求される。アカウントの所有者は、アカウントのフル・ネームの変更が許可される。
細密なパーミッションが無効	細密なパーミッションが無効の場合、一般にアカウントに <code>alter login</code> を使用するには、 <code>sso_role</code> が必要である。アカウント所有者には、アカウントのパスワードおよびフル・ネームの変更が許可される。

#### 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
138	login_admin	alter login	キーワードの内容 : <ul style="list-style-type: none"> <li>• MODIFY attribute_value_pair_list</li> <li>• DROP attribute_name_list</li> <li>• MODIFY PASSWORD</li> <li>• ADD AUTO ACTIVATED ROLES role1 [role2][ ...[roleN]...]</li> <li>• DROP AUTO ACTIVATED ROLES {ALL   role1 [, role2 [... [, roleN] ...]}</li> </ul>

## 参照

コマンド create login、create login profile、alter login profile、drop login、drop login profile

マニュアル ログイン・アカウントの変更の詳細については、『セキュリティ管理ガイド』を参照してください。

関数 lprofile\_id、lprofile\_name

システム・プロシージャ sp\_passwordpolicy、sp\_displaylogin、sp\_displayroles、sp\_locklogin

## alter login profile

**説明** ログイン・プロファイルの属性を変更します。

**構文**

```
alter login profile login_profile_name
  { [as [ not ] default ]
    | [modify attribute_value_pair_list]
    | [add auto activated roles role_name [, role_name_list]]
    | [drop auto activated roles { ALL | role_name [, role_name_list]}]
    | [drop attribute_name_list] }
```

**パラメータ** `as [ not ] default`  
`as default` は、デフォルト・ログイン・プロファイルになるようにログイン・プロファイルを変更します。`as not default` は指定されたログイン・プロファイルのデフォルト・プロパティを削除します。

`login_profile_name`  
 変更するログイン・プロファイルの名前を指定します。

`modify`  
 属性が存在する場合、指定された新しい値に属性値を変更します。属性が存在しない場合は、指定された一連の属性、および対応する値が指定されたログイン・プロファイルに追加されます。  
`attribute_value_pair_list` は、属性名および指定された値です。以下の属性を 1 つまたは複数指定します。

パラメータ	パラメータ値	説明
default database	default_database_name	Adaptive Server のデータベースを指定します。 デフォルトは <code>master</code> です。
default language	default_language	言語を指定します。 デフォルトは <code>us_english</code> です。
login script	login_script_name	有効なストアド・プロシージャを指定します。 ログイン・スクリプトの場合、120 文字に制限されます。
authenticate with	有効な値 : ASE、LDAP、PAM、KERBEROS、ANY	ログイン・アカウントの認証に使用するメカニズムを指定します。  ANY を使用すると、Adaptive Server によって定義済みの外部認証メカニズムがあるかどうか調べられます。定義済みの外部認証メカニズムが検出された場合は、それが使用され、それ以外の場合は、ASE メカニズムが使用されます。  <code>authenticate with authentication mechanism</code> を指定しない場合、ログイン・アカウントには ANY が使用されます。

パラメータ	パラメータ値	説明
track lastlogin	有効な値：TRUE または FALSE。	最終ログイン更新を有効にします。デフォルトは TRUE で、更新を意味します。
stale period	1 .. 32767 日。 持続時間：D (日数)、W (週数)、M (月数)、Y (年数)。	ログイン・アカウントが非アクティブ状態を保持できる期間を示します。この期間を経過すると、非アクティブであることが理由でロックされます。デフォルトは D (日数) です。

#### add auto activated roles

ログイン時に自動的にアクティブ化する必要のある、ユーザ定義の役割を指定します。この役割は以前に付与されており、パスワードで保護されていません。指定された役割がログインを許可されていない場合、エラーが生成されます。デフォルトでは、ユーザ定義の役割はログイン時に自動的にアクティブ化されません。

#### drop auto activated roles

以前に付与されたユーザ定義の役割は、ログイン時に自動的にアクティブ化しないことを指定します。ALL は、付与されたすべてのユーザ定義役割を指定します。

#### drop attribute\_name\_list

以下を削除します。

- **default database** — デフォルトのデータベースの指定を削除します。
- **default language** — デフォルト言語の指定を削除します。
- **login script** — ログイン・スクリプト適用の指定を削除します。
- **authenticate with** — アカウントに関連付けられている認証メカニズムの指定を削除します。
- **track last login** — 最終ログイン更新の有効化の指定を削除します。
- **stale period** — ロックされるまでのログイン・アカウントの非アクティブ状態に関するすべての制限を削除します。

**例** **例 1** `eng_lp` をデフォルト・ログイン・プロファイルとして設定します。既存のデフォルト・ログイン・プロファイルがある場合、そのデフォルト・プロパティは削除されます。

```
alter login profile eng_lp as default
```

**例 2** 以前に付与された `program_role`、`product_role`、`admin_role` の役割がパスワード保護されていない場合、ログイン時にそれらを自動的にアクティブ化するようにログイン・プロファイル `mgr_lp` を変更します。

```
alter login profile mgr_lp add auto activated roles
program_role, product_role, admin_role
```

**例 3** ログイン・プロファイル `mgr_lp` を変更し、以前に付与された役割 `admin_role` のログイン時の自動アクティブ化を除去します。

```
alter login profile mgr_lp drop auto activated roles
admin_role
```

**例 4** ログイン・プロファイル `mgr_lp` を変更し、`login script` 属性を削除します。削除すると、`mgr_lp` に関連付けられていたログイン・アカウントは、デフォルト・ログイン・スクリプト（定義されている場合）を使用します。デフォルト・ログイン・スクリプトが定義されていない場合は、ログイン・スクリプト属性はデフォルト値に設定され、ログイン時にはログイン・スクリプトは起動されません。

```
alter login profile mgr_lp drop login script
```

#### 使用法

- 異なるログイン・プロファイルから属性が取得された場合や、`sp_passwordpolicy` を使用して値が指定された場合、ログイン・アカウント属性の適用方法は優先度の規則によって決定されます。優先度の規則については、『セキュリティ管理ガイド』の「ログイン・プロファイル属性とパスワード・ポリシー属性の適用」を参照してください。
- ログイン時に起動するログイン・スクリプトを指定できます。詳細については、『セキュリティ管理ガイド』の「ログイン・スクリプトの呼び出し」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

#### パーミッション

`alter login profile` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効化されている場合、 <code>alter login profile</code> の実行には <code>manage any login profile</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効化されている場合、 <code>alter login profile</code> の実行には <code>sso_role</code> が必要。

#### 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
140	security_profile	alter login profile	キーワードの内容 : <ul style="list-style-type: none"> <li>• DEFAULT</li> <li>• NOT DEFAULT</li> <li>• DROP attribute_name_list</li> <li>• MODIFY attribute_value_pair_list</li> <li>• ADD AUTO ACTIVATED ROLES role1 [role2][ ...[roleN]...]</li> <li>• DROP AUTO ACTIVATED ROLES {ALL   role1 [, role2 [... [, roleN] ...]}</li> </ul>

## 参照

コマンド create login、create login profile、alter login、drop login、drop login profile

マニュアル ログイン・プロファイルの変更の詳細については、『セキュリティ管理ガイド』を参照してください。

関数 lprofile\_id、lprofile\_name

システム・プロシージャ sp\_passwordpolicy、sp\_displaylogin、sp\_displayroles、sp\_locklogin



## alter precomputed result set

説明	事前計算済み結果セットのプロパティまたはポリシーを変更します。
構文	<pre>alter {precomputed result set   materialized view}       [owner_name.]prs_name       [(immediate   manual) refresh]       [enable   disable]       [(enable   disable) use in optimization]</pre>
パラメータ	<p><code>precomputed result set   materialized view</code> マテリアライズされたビューまたは事前計算済み結果セットを変更します。</p> <p><code>prs_name</code> 事前計算済み結果セットの名前です。完全修飾された <code>prs_name</code> にサーバまたはデータベースの名前を含めることはできません。</p> <p><code>{immediate   manual} refresh</code> リフレッシュ・ポリシーを次のいずれかから決定します。</p> <ul style="list-style-type: none"> <li>• <code>Immediate</code> — (デフォルト) 事前計算済み結果セットは基本テーブルを更新するトランザクションと同じトランザクションで更新されます。</li> <li>• <code>manual</code> — 事前計算済み結果セットは明示的に更新されます。<code>manual</code> パラメータを使用すると、明示的に <code>refresh</code> を発行するまで基本テーブルの更新が事前計算済み結果セットに反映されません。<code>manual</code> に設定されている事前計算済み結果セットは維持されないため、Adaptive Server では <code>refresh</code> パラメータの発行後も古いものであるとみなされます。この結果、クエリ・プロセッサは、クエリで古いデータが受け入れられる場合のみ、このデータをクエリの再書き込み対象として選択します。</li> </ul>

**enable | disable**

事前計算済み結果セットを演算に利用できるかどうかを指定します。このオプションは、他のすべての事前計算済み結果セットのオプションより優先されます。

- **enable** — (デフォルト) 事前計算済み結果セットを演算に利用できます。**enable** の設定された事前計算済み結果セットのみが、リフレッシュ・ポリシーに従って維持されます。
- **disable** — 事前計算済み結果セットを演算に使用できません。事前計算済み結果セットが無効化されていると、維持またはクエリの再書き込みで考慮されません。事前計算済み結果セットが **disable** に設定されていると、次の処理は行われません。
  - 最適化時のクエリの再書き込みへの使用。 **use in optimization** の指定の有無は問いません。
  - データの入力。 **with populate** の指定の有無は問いません。

**{enable | disable} use in optimization**

最適化時に事前計算済み結果セットをクエリの再書き込みに含めるかどうかを指定します。**use in optimization** はデフォルトで有効化されています。事前計算済み結果セットがクエリの再書き込みで考慮されるかどうかは、**refresh** パラメータの設定によって決まります。

- **immediate** — すべてのクエリで考慮されます。
- **manual** — クエリが古いデータを受け入れる場合のみ考慮されます。

**例** `authors_prs` のリフレッシュ・ポリシーを **manual** から **immediate** に変更します。

```
alter precomputed result set authors_prs
immediate refresh
```

**使用法**

- リフレッシュ・ポリシーを **manual** から **immediate** に変更すると、事前計算済み結果セットは自動的にリフレッシュされます。
- 事前計算済み結果セットが基盤とする基本テーブルまたはビューが削除または変更された場合、その事前計算済み結果セットは自動的に **disable** に変更されます。

**パーミッション**

`alter` コマンドを実行するには、事前計算済み結果セットの所有者である必要があります。

## alter...modify owner

**説明** データベース・オブジェクトの所有権を 1 人の所有者から別の所有者に譲渡します。

**構文**

```
alter { object_type | all } [owner.]{object_name | * }
      modify owner
      { name_in_db | loginname only login_name }
      [ preserve permissions ]
```

**パラメータ** *object\_type*  
所有権を明示的に譲渡するオブジェクトの種類です。次のオブジェクト・タイプのいずれかを指定します。

- **table** — ユーザ・テーブルおよびプロキシ・テーブル
- **view** — ビュー
- **procedure** — ストアド・プロシージャ
- **function** — ユーザ定義の関数
- **default** — テーブルの作成とは別に定義されたデフォルト
- **rule** — ルール
- **type** — ユーザ定義のデータ型
- **encryption key** — 暗号化キー

### all

使用可能なすべてのオブジェクト・タイプです。all *owner.\** として指定すると、指定された所有者が所有する使用可能なすべてのオブジェクトの所有権が譲渡されます。all *owner.object\_name* として指定すると、指定された所有者が所有する *object\_name* という名前の使用可能なオブジェクトの所有権が譲渡されます。

### *owner.*

所有者のデータベース・ユーザ ID (**uid**) によって決定されるオブジェクトの現在の所有者を示します。ユーザが独自に所有しているオブジェクトの所有権を譲渡する場合、*owner* の指定は、オプションです。**sysobjects** テーブル内のオブジェクトの所有権は、所有者のログイン名と **UID** に関連付けられています。

### *object\_name*

所有権を譲渡するオブジェクトの名前を示します。*object\_name* を同じ所有者に設定してオブジェクトの所有権を譲渡しようとする、エラー・メッセージが発生します。

\*

*owner* が所有し、*object\_type* で指定されたすべてのオブジェクト。*object\_type* が *all* の場合、*owner* が所有するすべてのオブジェクトが譲渡されます。*owner* がデータベース所有者の場合、\* は使用できません。

#### *name\_in\_db*

所有権の譲渡先の新しい所有者のデータベース・ユーザ名です。*name\_in\_db* で指定するユーザは、既存のユーザである必要があり、ゲスト、役割、グループ、またはエイリアスを指定することはできません。

#### loginame only *login\_name*

*sysobjects* で関連するオブジェクトの *loginame* フィールドのみを *login\_name* に譲渡します。*login\_name* は *syslogins* テーブル内の有効なログインである必要があります。

#### preserve permissions

所有権を譲渡するオブジェクトに対して明示的に付与されたまたは取り消されたパーミッションを保持するかどうかを示します。

- 指定すると、オブジェクトに対して明示的に付与されたまたは取り消されたすべてのパーミッションが保持され、パーミッションの *grantor* は新しい所有者に変更されます。

たとえば、*bill* は、付与オプションによりテーブル *bill\_table* に対する *select* パーミッションを *mark* に付与しました。その後、*mark* はテーブル *bill\_table* に対する *select* パーミッションを *john* に付与しました。その後、*preserve permissions* を指定してテーブルの所有権が *eric* に譲渡された場合、*mark* と *john* は *bill\_table* に対するパーミッションを保持します。

- 指定しないと、オブジェクトに対して明示的に付与されたまたは取り消されたすべてのパーミッションがシステムから削除され、その結果、*sysprotects* テーブルでオブジェクトの対応するローも削除されます。

前の所有者のオブジェクトに対する暗黙のパーミッションは保持されません。新しい所有者が暗黙のすべてのパーミッションを取得します。

たとえば、*bill* は *bill\_table* の所有者であり、*bill\_table* に対する暗黙の *alter*、*delete*、*insert*、*references*、*select*、*update* パーミッション、および明示的な *decrypt* パーミッションを処理します。*preserve permission* を指定して *eric* に所有権を譲渡した後は、*bill* は *bill\_table* に対する *decrypt* パーミッションのみを保持します。

例

**例 1** bill.author というテーブルの所有権を eric に譲渡します。

```
alter table bill.author
  modify owner eric
```

**例 2** 明示的に付与されたすべての既存のパーミッションを削除することなく、bill.vw\_author\_in\_ca という名前のビューの所有権を eric に譲渡します。

```
alter view bill.vw_author_in_ca
  modify owner eric
  preserve permissions
```

**例 3** bill が所有するすべてのテーブルの所有権を eric に譲渡します。

```
alter table bill.*
  modify owner eric
```

**例 4** bill が所有するすべてのオブジェクトの所有権を eric に譲渡します。

```
alter all bill.*
  modify owner eric
```

**例 5** 新しい所有者 cindy がすでに cindy.publisher というテーブルを所有する場合、次のコマンドは失敗します。

```
alter table bill.publisher
  modify owner cindy
```

**例 6** bill.publisher はストアド・プロシージャではないため、bill.publisher の所有権を cincy に譲渡しようとする、エラーが発生します。

```
alter procedure bill.publisher
  modify owner cindy
```

使用法

**オブジェクトの暗黙の譲渡**

次の従属オブジェクトの所有権は、依存するオブジェクトの所有権が譲渡された場合、暗黙的に譲渡されます。

- **trigger** — 所有者が同じ場合、トリガの所有権は、従属テーブルとともに更新されます。トリガが DBO 以外が所有するテーブルまたはビューに対して作成されている場合、そのトリガが所有する DBO の所有権は変更できません。
- テーブルまたはビューの作成時に定義される宣言オブジェクト。
  - デフォルト
  - 復号化デフォルト
  - Check 制約

- 参照制約
- Partition 制約
- 計算カラム

#### システム・データベース内のオブジェクト

次のシステム・データベースでオブジェクトの所有権を譲渡する場合は注意が必要です。sybsecurity、sybssystemdb、model、sybssystemprocs、sybsyntax、dbccdb、および tempdb

spt\_ プレフィックスの付いたユーザ・テーブル、sp\_ プレフィックスの付いたシステム・ストアド・プロシージャ、モニタ・テーブルなど、Sybase が提供し管理するシステム・オブジェクトの所有権は譲渡できません。これらの所有権を譲渡すると、データベースが使用不能になる場合があります。

#### 暗号化キー

暗号化キーのコピーを所有する所有者にその暗号化キーを譲渡することはできないため、コマンドは失敗します。

暗号化キーの所有者を変更しても、暗号化キーのコピーを割り当てられているユーザには影響はありません。

#### 事前計算済み結果セット

事前計算済み結果セットの所有者の変更に `alter ... modify owner` を使用することはできません。この所有者を変更するには、事前計算済み結果セットを削除して、新しい所有者で作成しなおす必要があります。

#### 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

#### パーミッション

`alter... modify owner` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	<p>細密なパーミッションが有効化されている場合、暗号化キー以外のオブジェクトの <code>alter any object owner</code> 権限が必要。</p> <p>暗号化キー所有者または次の権限 (暗号化キーのタイプによって異なる) が付与されたユーザのみが暗号化キーの所有権を譲渡できる。</p> <ul style="list-style-type: none"> <li>• カラム暗号化キー : <code>manage column encryption key</code></li> <li>• マスタ・キー : <code>manage master key</code></li> <li>• サービス・キー : <code>manage service key</code></li> </ul> <p>明示的または暗黙的にデータベース所有者のエイリアスが付与されたデータベース・ユーザ (<code>alter any object owner</code> 権限を持つ) は、具体的に所有しているオブジェクトの所有権を譲渡することができない。 <code>sysobjects.uid</code> の値にデータベース所有者のユーザ ID が含まれ、 <code>sysobjects.loginame</code> の値 NULL またはデータベース所有者のログイン名が含まれる場合に、オブジェクトがデータベース所有者によって具体的に所有されていると認識される。</p>
細密なパーミッションが無効	<p>細密なパーミッションが無効化されている場合は、次のようになる。</p> <ul style="list-style-type: none"> <li>• システム・セキュリティ担当者 (SSO) は、このコマンドを使用してオブジェクトの所有権を譲渡できる。</li> <li>• 暗号化キーの所有権を変更できるのは、SSO ユーザおよびキーの所有者に限る。</li> <li>• データベース所有者 (DBO) ユーザ、およびエイリアスとして DBO が明示的または暗黙的に指定されているユーザは、暗号化キー以外の種類のオブジェクトの所有権を譲渡できる。ただし次の制約がある。 <ul style="list-style-type: none"> <li>• データベース所有者は、自分が具体的に所有しているオブジェクトの所有権は譲渡できない。 <code>sysobjects.uid</code> の値として <code>DBO_UID</code>、 <code>sysobjects.loginame</code> の値として NULL またはデータベース所有者のログイン名を含んでいるオブジェクトは、そのデータベース所有者が具体的に所有するオブジェクトとして識別される。</li> </ul> </li> <li>• 1 つのコマンドで複数のオブジェクトの所有権を譲渡する機能は、安全上の理由で無効になっている。また、DBO オブジェクトの譲渡は、 <code>dbo.object_name</code> の形式で行う必要がある。</li> </ul>

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
124	alter	alter .. modify owner	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – 次のいずれかです。 <ul style="list-style-type: none"> <li>• USER TYPE <i>owner.obj_nam</i> – ユーザ定義データ型の所有権を変更する場合</li> <li>• NEW OWNER – <i>name_in_db</i></li> <li>• PRESERVE PERMISSIONS – オプションが指定されている場合</li> <li>• NEW LOGINAME – <i>login_name</i> (LOGINAME ONLY <i>login_name</i> が指定されている場合)</li> </ul> </li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>



## alter role

説明	役割間の相互排他的な関係を定義します。役割のパスワードの追加、削除、変更を行います。個別の役割に対するパスワード有効期間、最短のパスワード長、許容されるログイン失敗の最大回数を指定します。alter role は、役割のロックおよびロック解除も行います。
構文	<pre>alter role <i>role1</i> {add   drop} exclusive     {membership   activation} <i>role2</i>  alter role <i>role_name</i> [add passwd "<i>password</i>"       drop passwd] [lock   unlock]  alter role {<i>role_name</i>   "all overrides"}     set {passwd expiration   min passwd length       max failed_logins} <i>option_value</i></pre>
パラメータ	<p><i>role1</i> 相互排他的な関係の、いずれかの役割です。</p> <p>add 相互排他的な関係に役割を追加します。または役割にパスワードを追加します。</p> <p>drop 相互排他的な関係の役割を削除します。または役割からパスワードを削除します。</p> <p>exclusive 指定した両方の役割を相互排他的な関係にします。</p> <p>membership 同時に両方の役割をユーザに付与できないようにします。</p> <p>activation 同時に両方の役割をユーザに付与することはできても、同時に両方の役割をアクティブ化できないようにします。</p> <p><i>role2</i> 相互排他的な関係の、もう一方の役割です。</p> <p><i>role_name</i> パスワードを追加、削除、または変更する役割の名前です。 <i>role_name</i> を使用して、パスワードの有効期間、パスワードの最小長、最大ログイン失敗回数を指定します。</p> <p>passwd 役割のパスワードを追加または削除します。</p>

**password**

役割に追加するパスワードです。パスワードに変数を使用できません。パスワードの規則については、『システム管理ガイド 第1巻』の「Adaptive Server のログイン、データベース・ユーザ、クライアント接続の管理」を参照してください。

**lock**

指定した役割をロックします。

**unlock**

指定した役割のロックを解除します。

**all overrides**

特定の役割ではなく、サーバ全体で従う設定を適用します。

**set**

以後のオプションをアクティブにします。

**passwd expiration**

パスワードの有効期間を日数で指定します。0 ~ 32767 の任意の値を指定できます。

**min passwd length**

指定したパスワードに対して許容される最短の長さを指定します。

**max failed\_logins**

指定したパスワードに許可される、ログイン失敗の最大回数を指定します。

**option\_value**

passwd expiration、min passwd length、または max failed\_logins の値を指定します。all overrides を設定するには、option\_value の値を -1 に設定します。

**例** **例 1** intern\_role および specialist\_role がメンバシップ・レベルで相互排他的な関係にあることを定義します。

```
alter role intern_role add exclusive membership
specialist_role
```

**例 2** 役割がメンバシップ・レベルとアクティブ化レベルで相互排他的な関係にあることを定義します。

```
alter role specialist_role add exclusive membership
intern_role
alter role intern_role add exclusive activation
surgeon_role
```

**例 3** 既存の役割にパスワードを追加します。

```
alter role doctor_role add passwd "physician"
```

**例 4** 既存の役割からパスワードを削除します。

```
alter role doctor_role drop passwd
```

**例 5** `physician_role` をロックします。

```
alter role physician_role lock
```

**例 6** `physician_role` のロックを解除します。

```
alter role physician_role unlock
```

**例 7** `physician_role` に許可されるログイン失敗の最大回数を 5 回に変更します。

```
alter role physician_role set max failed_logins 5
```

**例 8** 既存の役割である `physician_role` の最短のパスワード長を 5 文字に設定します。

```
alter role physician_role set min passwd length 5
```

**例 9** すべての役割の最短パスワード長を無効にします。

```
alter role "all overrides" set min passwd length -1
```

**例 10** すべての役割のログイン失敗の最大回数を無効にする設定を削除します。

```
alter role "all overrides" set max failed_logins -1
```

## 使用法

- `alter role` コマンドは、役割間の相互排他的な関係の定義、および役割のパスワードの追加、削除、変更を行います。
- `all overrides` パラメータでは、次のパラメータのいずれかを指定して、`sp_configure` を使用してシステムに設定された無効化を、削除します。
  - `passwd expiration`
  - `max failed_logins`
  - `min passwd length`

役割のパスワードを削除すると、パスワードの有効期間とログイン失敗の最大回数のオプションを無効化した設定が削除されます。

- `alter role` を使用して役割をロックまたはロック解除すると、`sysssrvroles` に追加された `locksuid`、`lockdate`、`lockreason` の各カラムを設定または設定解除することになります。

### 相互排他的な役割

- 相互排他的な関係または役割階層の役割を指定する場合、特別な順序を使用する必要はありません。
- 役割階層に相互排他性を使用して、ユーザ定義の役割に制約を指定できます。
- 相互排他的メンバシップは、相互排他的アクティブ化よりも強い制限です。2つの役割をメンバシップ・レベルにおいて相互排他的と定義した場合は、それら2つの役割は暗黙的にアクティブ化レベルで相互排他的と見なされます。
- 2つの役割をメンバシップ・レベルにおいて相互排他的と定義した場合は、それらをアクティブ化レベルで相互排他的と定義しても、メンバシップ・レベルにおける定義に何も影響を与えません。アクティブ化レベルにおける相互排他性の追加または削除は、メンバシップ・レベルにおける相互排他性とは独立して行われます。
- 両方の役割をユーザまたは役割に対して付与したあとでは、2つの役割が相互排他的なプロパティであると定義できません。メンバシップ・レベルでそれらの役割が相互排他的関係にあると定義する前に、現在付与されているユーザから、付与された役割のいずれかを取り消してください。
- 2つの役割が相互排他的であるとアクティブ化レベルで定義されている場合、システム・セキュリティ担当者は両方の役割を同一のユーザに割り当てることができますが、ユーザは両方の役割を同時にアクティブ化することはできません。
- システム・セキュリティ担当者が、アクティブ化レベルで2つの役割を相互排他的であると定義し、ユーザがすでに両方の役割をアクティブ化したり、デフォルトでログイン時に両方の役割をアクティブ化するように設定した場合は、**Adaptive Server** はこれらの役割を相互排他的にしますが、矛盾する役割を持った特定のユーザを指定して警告メッセージを発行します。ユーザのアクティブ化された役割は変更されません。

### 役割のパスワードの変更

役割のパスワードを変更するには、次のように、まず既存のパスワードを削除し、新しいパスワードを追加します。

```
alter role doctor_role drop passwd
alter role doctor_role add passwd "physician"
```

---

**注意** Adaptive Server バージョン 12.x より前に存在し、ユーザ定義に付加されていたパスワードは期限切れになりません。

---

標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>alter role</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage roles</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sso_role</code> が付与されたユーザであることが必要。

**監査** `sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
85	roles	create role、drop role、alter role、grant role、または revoke role	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

**参照** コマンド [create role](#)、[drop role](#)、[grant](#)、[revoke](#)、[set](#)

マニュアル 役割の変更の詳細については、『システム管理ガイド』を参照してください。

関数 [mut\\_excl\\_roles](#)、[proc\\_role](#)、[role\\_contain](#)、[role\\_id](#)、[role\\_name](#)

システム・プロシージャ [sp\\_activeroles](#)、[sp\\_displaylogin](#)、[sp\\_displayroles](#)、[sp\\_modifylogin](#)

## alter table

### 説明

- 新しいカラムをテーブルに追加します。既存のカラムを削除または変更します。制約を追加、変更、または削除します。既存のテーブルのプロパティを変更します。テーブルのトリガを有効化または無効化します。テーブルの圧縮レベルを変更します。
- 計算カラムの追加、削除、変更をサポートし、マテリアライズされたプロパティ、**null** 入力可能性、または既存の計算カラム定義の変更を可能にします。
- 指定されたパーティション方式によるテーブルの分割や再分割、既存のパーティションへのテーブルの追加、既存のパーティションの分割やマージを行います。

### 構文

```
alter table [[database.][owner].table_name
    {add column_name datatype}
    [default {constant_expression | user | null}]
    {identity | null | not null [not materialized]}
    [off row | in row]
    [[constraint constraint_name]
    {{unique | primary key}
    [clustered | nonclustered]
    [asc | desc]
    [with {fillfactor = pct,
        max_rows_per_page = num_rows,
        reservepagegap = num_pages
        immediate_allocation}
    [on segment_name]
    | references [[database.]owner.]ref_table
    [(ref_column)]
    [match full]
    | check (search_condition)]
    [encrypt [with [database.[owner].] keyname]
    [decrypt_default {constant_expression | null}]]
    [compressed = compression_level | not compressed]
    [, next_column]...]
    | add [constraint constraint_name]
    {unique | primary key}
    [clustered | nonclustered]
    (column_name [asc | desc][, column_name [asc | desc]...])
    [with {fillfactor = pct,
        max_rows_per_page = num_rows,
        reservepagegap = num_pages}]
    [on segment_name]
    | foreign key (column_name [{, column_name}...])
    references [[database.]owner.]ref_table
    [(ref_column [{, ref_column}...])]
    [match full]
    | add lob_colname { text | image | unitext }
```

```

    [null] [ in row [ (length) ] ]
| check (search_condition)
| set dml_logging = {full | minimal | default} |
    [, compression = {none | page | row}]
    [lob_compression = off | compression_level]
| drop {column_name [, column_name]...
    | constraint constraint_name}
| modify column_name
    [datatype [null | not null]]
    [[encrypt [with keyname] [decrypt_default [value]]
    | decrypt
    ]
    [[not] compressed]
    [compressed = compression_level | not compressed]
    | modify lob_column [ in row (length)]
    [, next_column]...
| replace column_name
    default {constant_expression | user | null}
    | decrypt_default {constant_expression | null}
    | drop decrypt_default|
lock {allpages | datarows | datapages} }
| with exp_row_size=num_bytes
    | transfer table [on | off]
    | no datacopy}
| partition number_of_partitions
| unpartition
| partition_clause
| add_partition_clause

```

パーティションに使用する

alter table 構文 :

```

partition_clause ::=
partition by range (column_name[, column_name]...)
    ([partition_name] values <= ({constant | MAX}
    [, {constant | MAX}] ...) [on segment_name]
    [compression_clause] [on segment_name]
    [, [partition_name] values <= ({constant | MAX}
    [, {constant | MAX}] ...) [on segment_name]]...)

| partition by hash (column_name[, column_name]...)
    { ([partition_name] [on segment_name]
    [, partition_name [on segment_name]]...)
    [compression_clause] [on segment_name]
    | number_of_partitions
    [on (segment_name[, segment_name] ...)]}

| partition by list (column_name)
    ([partition_name] values (constant[, constant] ...)
    [on segment_name]
    [compression_clause] [on segment_name]

```

```

        [, [partition_name] values (constant[, constant] ...)
           [on segment_name]] ...)

| partition by roundrobin
  { ([partition_name] [on segment_name]
    [, [partition_name] [on segment_name]]...)
    [compression_clause] [on segment_name]
  | number_of_partitions
    [on (segment_name [, segment_name]...)]}

add_partition_clause:=
add partition
  { ([partition_name] values <= ({constant | MAX}
    [, {constant | MAX}]...)
    [on segment_name]
    [compression_clause] [on segment_name]
    [, [partition_name] values <= ({constant | MAX}
      [, {constant | MAX}] ...)
      [on segment_name]]...)
  | modify partition {partition_name [, partition_name .. ]}
  set compression [= {default | none | row | page}]

  | ([partition_name] values (constant[, constant] ...)
    [on segment_name]
    [, [partition_name] values (constant[, constant] ...)
      [on segment_name]] ...)}

```

計算カラムに使用する alter table 構文 :

```

alter table
  add column_name {compute | as}
    computed_column_expression...
    [materialized | not materialized]
  drop column_name
  modify column_name {null | not null |
    {materialized | not materialized} [null | not null] |
    {compute | as} computed_column_expression
    [materialized | not materialized]
    [null | not null]}

```

パーティションの削除、分割、マージ、移動に使用する alter table 構文 :

```

alter table table_name
  drop partition partition_name [, partition_name]...
  split partition partition_name
  merge partition {partition_name [{, partition_name}...]}
  into destination_partition_name [on segment_name]
  move partition partition_name [{, partition_name}...]
  to destination_segment_name

```



## パラメータ

*table\_name*

変更するテーブルの名前を指定します。テーブルが別のデータベース内にある場合はデータベース名を指定し、データベース内にその名前のテーブルが複数ある場合は所有者名を指定します。*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。

*add*

テーブルに追加するカラム名または制約名を指定します。コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して *add* を使用できません。

*column\_name*

テーブル内のカラムの名前を指定します。Java がデータベースで使用可能な場合は、カラムを Java-SQL カラムにすることができます。

*datatype*

bit を除くシステム・データ型、または bit を基にしたデータ型を除くユーザ定義データ型です。

データベースで Java が有効化されている場合は、*datatype* は、データベースにインストールされている Java クラスの名前、システム・クラス、またはユーザ定義によるクラスのいずれかにすることができます。『Adaptive Server Enterprise における Java』を参照してください。

*default*

カラムのデフォルト値を指定します。デフォルトを指定した場合、データの挿入時にそのカラムに値を指定しないと、Adaptive Server ではこの値が挿入されます。デフォルトに使用できるのは、*constant\_expression*、*user* (データを挿入しているユーザ名の挿入)、または *null* (*null* 値の挿入) です。

Adaptive Server は *tablename\_colname\_objid* という形式のデフォルト名を生成します。*tablename* はテーブル名の最初の 10 文字、*colname* はカラムの最初の 5 文字、*objid* はデフォルトのオブジェクト ID 番号です。デフォルトを *null* に設定すると、デフォルトは削除されます。

コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して *default* を使用できません。

*constant\_expression*

カラムのデフォルト値として使用する定数式です。グローバル変数、カラム名、他のデータベース・オブジェクト名は使用できませんが、組み込み関数を使用できます。このデフォルト値はカラムのデータ型と互換性がある必要があります。

**user**

ユーザが値を入力しない場合に、Adaptive Server によってデフォルトとして入力されるユーザ名を指定します。カラムのデータ型は、char(30)、varchar(30)、または Adaptive Server によって暗黙的に char に変換されるデータ型でなければなりません。データ型が char(30) でも varchar(30) でもないときは、トランケーションが発生することがあります。

**null | not null**

デフォルトが存在しない場合の、データ挿入時の Adaptive Server の動作を指定します。

null は、null を入力可能なカラムが追加されることを指定します。ユーザが値を入力しないと、挿入時に Adaptive Server によって null 値が割り当てられます。

bit 型のカラムのプロパティは常に not null でなくてはなりません。

not null は、null を入力できないカラムが追加されることを指定します。デフォルトが存在しない場合、挿入時に null 以外の値を入力する必要があります。

null も not null も指定しなかった場合は、デフォルトで not null が使用されます。ただし、sp\_dboption を使用してこのデフォルトを変更し、SQL 標準と互換性を持つようにすることができます。新しく追加されたカラムに not null を指定 (または暗黙的に指定) する場合は、デフォルトの句が必要です。デフォルト値は新しく追加されたカラムの既存のローすべてに使用されます。また、これは以後の挿入時にも適用されます。

**materialized | not materialized**

作成するカラムがマテリアライズされているか、マテリアライズされていないかを示します。

**encrypt [with keyname]**

暗号化カラムとその暗号化に使用されたキーを指定します。

keyname は、create encryption key を使用して作成するキーを指定します。テーブルの所有者には、keyname に対する select パーミッションが必要です。keyname を指定しないと、サーバは、create encryption key または alter encryption key を使用してデフォルトとして作成されたデフォルト・キーを探します。

サポートされているデータ型のリストについては、『暗号化カラム・ユーザーズ・ガイド』の「データの暗号化」を参照してください。

**decrypt\_default constant\_expression**

このカラムでは **decrypt** パーミッションのないユーザーにデフォルト値を返すことを指定します。*constant\_expression* は **select** 文を実行したときに、復号化された値の代わりに **Adaptive Server** から返される値です。使用可能な値は **NULL** を入力可能な列の **NULL** だけです。**decrypt\_value** によってカラムのデータ型に変換できない場合、**Adaptive Server** ではクエリが実行されたときにのみ変換エラーが検出されます。

**decrypt**

暗号化カラムを復号化します。

**compressed = compression\_level | not compressed**

ロー内のデータが圧縮されるかどうか、および圧縮されている場合のレベルを示します。

**compression\_level**

圧縮のレベルです。次の圧縮レベルがあります。

- 0 – ローは圧縮されません。
- 1 ~ 9 – **Adaptive Server** は **ZLib** 圧縮を使用します。通常、圧縮の数値が大きいほど、**Adaptive Server** での **LOB** データの圧縮レベルが高くなり、圧縮データと非圧縮データの比率も大きくなります (つまり、非圧縮データと比べて、圧縮データによる記憶領域の節約量 (バイト) が多くなります)。

ただし、圧縮の量は **LOB** の内容によって異なります。圧縮レベルが高いほど、圧縮に使用される **CPU** の量が多くなります。そのためレベル 9 を使用すると、圧縮率が最高になりますが、**CPU** の使用量も最高になります。

- 100 – **Adaptive Server** は **FastLZ** 圧縮を使用します。**CPU** 使用量が最も少ない圧縮率です。通常、短いデータに使用します。
- 101 – **Adaptive Server** は **FastLZ** 圧縮を使用します。値に 101 を指定すると、100 を指定した場合よりも、**CPU** の使用量は若干多くなりますが、圧縮率は良くなります。

圧縮アルゴリズムでは、**LOB** データを使用しないローは無視されます。

**identity**

カラムに **IDENTITY** プロパティがあることを示します。データベースの各テーブルに、次のデータ型の **IDENTITY** カラムを 1 つ作成できます。

- 位取り 0 の numeric (真数値)、または
- 任意の整数値データ型 (符号付きまたは符号なしの **bigint**、**int**、**smallint**、**tinyint** など)

**IDENTITY** カラムは更新できません。また、**null** 値を使用することもできません。

**IDENTITY** カラムには、請求書番号や従業員番号など、**Adaptive Server** で自動的に生成される連続番号が格納されます。**IDENTITY** カラムの値は、テーブル内の各ローをユニークに識別します。

**off row | in row**

Java-SQL カラムが、ローから独立して格納されるか、ローに直接割り付けられた記憶領域に格納されるかを指定します。

**in row** カラムの記憶領域は、データベース・サーバのページ・サイズと、他の変数に従って、16 キロバイト以内に設定してください。デフォルト値は **off row** です。

**constraint**

整合性制約の名前を指定します。コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して **constraint** を使用できません。

**constraint\_name**

制約の名前で、識別子の規則に従い、データベース内でユニークである必要があります。テーブル・レベルの制約の名前を指定しない場合は、**Adaptive Server** によって **tablename\_colname\_objectid** という形式の名前が生成されます。**tablename** はテーブル名の最初の 10 文字、**colname** はカラム名の最初の 5 文字、**objectid** は制約のオブジェクト ID 番号です。一意性制約またはプライマリ・キー制約の名前を指定しない場合は、**Adaptive Server** によって **tablename\_colname\_tabindid** という形式の名前が生成されます。**tabindid** はテーブル ID とインデックス ID を連結した文字列です。

制約が追加されたときにすでにテーブルに存在するデータには、制約は適用されません。

**unique**

2つのローが **null** 以外の同じ値を持たないように、指定された 1 つまたは複数のカラムの値に制約を加えます。この制約によって、制約を削除した場合にのみ削除されるユニーク・インデックスが作成されます。このオプションを **null** オプションとともに使用することはできません。

**primary key**

2つのローが同じ値を持たず、その値が **null** 値にならないように、指定されたカラムの値に制約を加えます。この制約によって、制約を削除した場合にのみ削除されるユニーク・インデックスが作成されます。

**clustered | nonclustered**

**unique** 制約 (ユニーク制約) または **primary key** 制約 (プライマリ・キー制約) によって作成されたインデックスがクラスタード・インデックスであるか、ノンクラスタード・インデックスであるかを指定します。プライマリ・キー制約では、(テーブルにクラスタード・インデックスがない場合にかぎり) **clustered** がデフォルトです。ユニーク制約では、**nonclustered** がデフォルトです。クラスタード・インデックスは、テーブルごとに 1 つだけ持つことができます。詳細については、「[create index](#)」を参照してください。

**asc | desc**

作成するインデックスを昇順 (**asc**) にするか、降順 (**desc**) にするかを指定します。デフォルトは、昇順です。

**with fillfactor=pct**

Adaptive Server で既存のデータに新しいインデックスを作成する場合の各ページのデータの占有率を指定します。"pct" はパーセンテージ (**percentage**) を意味します。fillfactor の率は、インデックスの作成時にのみ使用されます。データは変更されるので、ページが特定の満杯率で維持されることはありません。

---

**警告 !** fillfactor を使用してクラスタード・インデックスを作成すると、Adaptive Server はクラスタード・インデックスの作成時にデータを再分配するため、データが占有する記憶領域の総量に影響します。

---

fillfactor のデフォルトは 0 です。これは、[create index](#) 文で with fillfactor を指定しない場合に使用されます ([sp\\_configure](#) で値を変更している場合は除きます)。fillfactor を指定する場合は、1 ~ 100 の値を使用してください。

`fillfactor` が 0 の場合、完全に満杯になるページにクラスタード・インデックスが作成され、完全に満杯になるリーフ・ページにノンクラスタード・インデックスが作成されます。これによって、クラスタード・インデックスとノンクラスタード・インデックスの両方のインデックス B ツリー内に十分な領域が確保されます。ほとんどの場合、`fillfactor` を変更する必要はありません。

`fillfactor` を 100 に設定すると、各ページが 100% 満杯の状態、クラスタード・インデックスとノンクラスタード・インデックスの両方が作成されます。`fillfactor` を 100 にするのは、データが追加されない読み込み専用テーブルに対してのみ効果があります。

`fillfactor` 値が 100 より小さい場合 (特殊な場合である 0 は除く)、満杯でないページを持つ新しいインデックスが作成されます。大量のデータが記録されることになるテーブルにインデックスを作成する場合は、`fillfactor` を 10 に設定するのが妥当です。ただし、`fillfactor` 値を小さくすると、それぞれのインデックス (またはインデックスとデータ) にさらに多くの記憶領域が必要になります。

#### transfer table [on | off]

増分転送に対するテーブルの指定を変更します。デフォルト値は、テーブルに転送するためのマークの有無にかかわらず、変更を行いません。`alter table` コマンドで `set transfer table` を指定し、`on` または `off` の選択が現在の値と異なる場合、テーブルの指定が変更されます。

#### max\_rows\_per\_page = num\_rows

インデックスのデータ・ページとリーフレベル・ページ上のロー数を制限します。`fillfactor` とは異なり、`max_rows_per_page` 値は、`sp_chgattribute` を使用して変更されるまで保持されます。

`max_rows_per_page` 値を指定しなかった場合は、インデックス作成時に値 0 が使用されます。データ・ページに対して `max_rows_per_page` を指定するときは、0 ~ 256 の値を使用してください。ノンクラスタード・インデックスの 1 ページあたりの最大ロー数は、インデックス・キーのサイズによって決まります。また、指定した値が大きすぎる場合は、エラー・メッセージが返されます。

制約付きで作成されたインデックスの場合は、`max_rows_per_page` 値を 0 に設定すると、満杯のページを持つクラスタード・インデックスと、満杯のリーフ・ページを持つノンクラスタード・インデックスが作成されます。この値を 0 に設定すると、クラスタード・インデックスとノンクラスタード・インデックスの B ツリーに適度な領域が確保されます。

`max_rows_per_page` を 1 に設定すると、リーフ・レベルで 1 ページあたりに 1 ローザつのクラスタードおよびノンクラスタードのリーフ・インデックス・ページが作成されます。これを使用して、頻繁にアクセスされるデータについてのロック競合を減らすことができます。

`max_rows_per_page` 値を小さい値に設定すると、完全に満杯ではないページで新しいインデックスが作成されます。また、使用される記憶領域が増えるため、ページ分割も多くなることがあります。

---

**警告！** `max_rows_per_page` を使用してクラスタード・インデックスを作成すると、Adaptive Server はクラスタード・インデックスの作成時にデータを再分配するため、データが占有する記憶領域の総量に影響することがあります。

---

#### `reservepagegap = num_pages`

埋められたページと、制約によって作成されたインデックスのエクステンション I/O 割り付け操作の間に残される空のページとの比率を指定します。指定した `num_pages` のそれぞれに、将来のテーブルの拡張に備えて空のページ 1 つが残されます。有効な値は 0 ~ 255 です。デフォルト値は 0 で、この場合、空のページは残されません。

#### `immediate_allocation`

`sp_dboption 'deferred table allocation'` を有効化している場合は、明示的にテーブルを作成します。

#### `on segment_name`

インデックスが存在するか、配置されるセグメントを指定します。

`on segment_name` を使用する場合は、`create database` または `alter database` を使用して論理デバイスが事前にデータベースに割り当てられている必要があります。また、`sp_addsegment` を使用してセグメントがデータベースに作成されていることも必要です。データベースで使用できるセグメント名のリストについては、システム管理者に問い合わせるか、`sp_helpsegment` を使用して参照してください。

`clustered` を指定し、`on segment_name` オプションを使用した場合は、テーブル全体が指定したセグメントに移行します。これは、インデックスのリーフ・レベルに実際のデータ・ページがあるためです。

`on segment_name` をパーティションに使用するときは、パーティションを配置するセグメントを指定します。

### references

参照整合性の制約のカラム・リストを指定します。1つのカラム制約に指定できるカラムの値は、1つだけです。別のテーブルを参照するテーブルにこの制約を追加するには、参照元テーブルに挿入されたデータがすべて、参照先テーブルに事前に存在していなければなりません。

この制約を使用するには、参照先テーブルの **references** パーミッションが必要です。参照先テーブルの指定したカラムは、**unique** 制約または **create index** 文のどちらかで作成したユニーク・インデックスによって、制約を受けなければなりません。カラムが指定されていない場合は、参照先テーブルの適切なカラムに **primary key** 制約がなければなりません。また、参照元テーブルのカラムのデータ型は、参照先テーブルのカラムのデータ型と同じでなければなりません。

コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して **references** を使用できません。

### foreign key

リストしたカラムが、そのテーブルの外部キーであることを指定します。この外部キーに対応するプライマリ・キーは、**references** 句にリストしたカラムです。

### ref\_table

参照されるカラムを含むテーブル名です。別のデータベースのテーブルを参照することもできます。制約では、192 までのユーザ・テーブルと内部生成のワークテーブルを参照できます。

**sp\_helpconstraint** を使用すると、テーブルの参照整合性制約を確認できます。

### ref\_column

参照されるテーブルのカラム名です。

### match full

参照元ローの参照元カラムのすべての値が以下の場合、次のようになることを指定します。

- **null** — 参照整合性条件は真である。
- **null** 以外の値 — 対応するそれぞれのカラムが参照先テーブルで等しい参照先ローがある場合、参照整合性条件は真である。

どちらでもなければ、以下の場合に参照整合性条件は偽になります。

- すべての値が **null** 以外で、等しくない場合、または
- 参照元ローの参照元カラムに **null** 以外の値と **null** がある場合



**check**

Adaptive Server がテーブル内のローすべてに実行する *search\_condition* 制約を指定します。コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して **check** を使用できません。

**search\_condition**

カラムの値に対する **check** 制約を定義するブール式です。これらの制約は、次のものを含むことができます。

- **in** によって導入される定数式のリスト。
- ワイルドカード文字を含む、**like** で提供される条件の集合。

式には、算術演算と Transact-SQL 関数を使用できます。

*search\_condition* には、サブクエリ、集合関数、パラメータ、またはホスト変数は使用できません。

**next\_column**

カラム定義に記述されたものと同じ構文を使用して (カンマで区切られた) カラム定義を追加します。

**set dml\_logging**

**insert**、**update**、**delete (DML)** オペレーションのロギングの量を決定します。以下のいずれかです。

- **full** — すべてのトランザクションのログが取られます。
- **minimal** — ローまたはページの変更のログは取られません。
- **default** — ロギングをテーブルのデフォルトに設定します。

**add lob-colname { text | image | unitext }**

指定されたデータ型の LOB カラムを追加します。

**[null] [ in row [ (length) ] ]**

ロー内に残る LOB カラムの最大長を指定します。長さを指定しない場合には、Adaptive Server によって有効なデータベース全体の設定がロー内の長さに適用されます。

**in row (length)** を使用しない場合、データベース全体の設定が有効でなければ、データのロー外の記憶領域が指定された LOB カラムが追加されます。

modify *lob-column* in row [(*length*)]

LOB カラムのプロパティのみが、指定の長さまでロー内に変更されます。このコマンドを実行しても、データは移動しません。

さらに、このオプションを使用してロー内の LOB カラムの長さを増やすこともできます。

---

**注意** このオプションを使用して LOB カラムの長さを減らしたり、長さとして 0 を指定したりすることはできません。ページ上の空き領域の量により、変更後に実行される更新時に、指定されたロー内の長さまでロー外の LOB データがロー内に移動されます。

---

set compression

テーブルまたはパーティションに適用する圧縮レベルを示します。新しい圧縮レベルは、新しく挿入または更新されたデータに適用されます。

- **default** — 指定されたパーティションの圧縮レベルをテーブルの圧縮レベルにリセットします。
- **none** — このテーブルまたはパーティションのデータは圧縮されません。パーティションの場合、**none** は、テーブル圧縮が **row** 圧縮または **page** 圧縮に変更されても、このパーティションのデータが圧縮されないままであることを意味します。
- **page** — ページがいっぱいになると、ロー圧縮された既存のデータ・ローが、ページ・レベルの圧縮を使用して圧縮され、ページ・レベルの辞書、インデックス、文字コードのエントリが作成されます。パーティション・レベルまたはテーブル・レベルで **page** 圧縮を設定します。
- **Adaptive Server** では、データがロー・レベルで圧縮されてからしかページ・レベルで圧縮されないため、圧縮を **page** に設定することは、**page** と **row** の両方が圧縮されることを意味します。
- **row** — 個々のロー内の 1 つ以上のデータ項目を圧縮します。**Adaptive Server** は、非圧縮形式に比べ、圧縮形式でのデータの格納により記憶領域を節約できる場合にのみ、**row** 圧縮形式でデータを格納します。パーティション・レベルまたはテーブル・レベルで **row** 圧縮を設定します。

set lob\_compression = *compression\_level*

LOB データ型を使用するテーブルの圧縮レベルを変更します。

**drop**

テーブルから削除するカラム名または制約名を指定します。コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して **drop** を使用できません。

**modify**

データ型または **null** 値の使用の可否を変更するカラムの名前を指定します。

**[not] compressed**

変更されたカラムが圧縮されるかどうかを示します。

**replace**

後に続く **default** 句で指定した新しい値を使用して、デフォルト値を変更するカラムを指定します。コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して **replace** を使用できません。

**enable | disable trigger**

トリガを有効化または無効化します。トリガの詳細については、『システム管理ガイド』を参照してください。

**lock datarows | datapages | allpages**

テーブルに使用されるロック・スキームを変更します。

**with exp\_row\_size=num\_bytes**

予期されるロー・サイズを指定します。このパラメータは次の項目のみに適用できます。

- データロー・ロック・スキームとデータページ・ロック・スキーム
- 可変長ローがあるテーブル
- **alter table** で **alter table add** または **modify** などのデータのコピーが実行される場合。**with exp\_row\_size=num\_bytes** は **alter table lock change** オペレーションに使用できません。

有効な値は 0、1、およびテーブルのローの最小長および最大長の間の任意の値です。デフォルト値は 0 で、サーバワイドな設定が適用されることを示します。

**no datacopy**

**alter table** によってデータ・コピーを行うことなくカラムが削除されることを示します。これによって、**alter table** の実行時に **alter table... drop** コマンドがテーブルに対して実行中の他のコマンドをブロックしないようにします。

**partition number\_of\_partitions**

(*number\_of\_partitions* - 1) 個の空のパーティションを、分割されていないテーブル (パーティションが 1 つある、ラウンドロビン方式で分割されたテーブル) に追加します。したがって、テーブルのパーティションの総数は、*number\_of\_partitions* 個になります。コンポーネント統合サービス (CIS) が有効な場合は、リモート・サーバに対して **partition** を使用できません。

**unpartition**

インデックスのない、ラウンドロビン方式で分割されたテーブルを、分割されていないテーブルに変更します。コンポーネント統合サービスが有効な場合は、リモート・サーバに対して **unpartition** を使用できません。

**partition by range**

分割するレコードを、分割カラムの値に基づいて指定します。分割カラムの値がユーザ提供の上限および下限と比較されて、パーティションの割り当てが決定されます。

**column\_name**

*partition\_clause* で使用する場合にパーティション・キー・カラムを指定します。パーティション・キー・カラムを暗号化カラムにすることはできません。

**partition\_name**

テーブル・レコードを保管する新しいパーティションの名前を指定します。テーブル内またはインデックス内にすでに存在するパーティションと同じ名前は指定できません。**set quoted\_identifier** オプションを設定した場合、パーティション名に区切り識別子を使用できます。それ以外の場合、パーティション名は有効な識別子でなければなりません。

*partition\_name* を省略すると、*table\_name\_partition\_id* という形式の名前が作成されます。最大長を超える長さのパーティション名は、トランケートされます。

**values <= constant | MAX**

指定したパーティションの上限値を指定します。パーティションの上限に定数値 (*constant*) を指定すると、テーブルに暗黙的な一意性制約が適用されます。キーワード **MAX** を使用すると、特定のデータ型の最大値が指定されます。

**on segment\_name**

**partition\_clause** をパーティションに使用するとき、パーティションを配置するセグメントを指定します。 **on segment\_name** を使用する場合は、 **create database** または **alter database** を使用して論理デバイスが事前にデータベースに割り当てられている必要があります。また、 **sp\_addsegment** を使用してセグメントがデータベースに作成されていることも必要です。データベースで使用できるセグメント名のリストについては、システム管理者に問い合わせるか、 **sp\_helpsegment** を使用して参照してください。

**partition by hash**

レコードの分割に、システムから提供されるハッシュ関数を使用します。ハッシュ関数は、レコードが割り当てられるパーティションを指定するパーティション・キー値を計算します。

**partition by list**

分割するレコードを、指定したカラム内のリテラル値に基づいて指定します。パーティション・キーのカラムは 1 つだけです。リスト・パーティションには、最大で 250 個の定数をパーティション値としてリストできます。

**partition by round-robin**

レコードの分割を順次処理で行うことを指定します。ラウンドロビン分割テーブルに分割キーはありません。ユーザおよびオブティマイザは、特定のレコードがどのパーティションに配置されるかを確認することはできません。

**add partition**

範囲分割テーブルまたはリスト分割テーブルのみに適用されます。

- 範囲分割テーブルの場合、範囲分割テーブルの上端に 1 つまたは複数のパーティションを追加する。
- リスト分割テーブルの場合、新しい値のセットを持つ 1 つまたは複数のパーティションを追加する。

**modify partition**

圧縮レベルを変更するパーティションを指定します。

**compute | as**

新しい計算カラムを追加または削除します。 **create table** コマンドで定義されたルールと **alter table add** のルールに従います。

**computed\_column\_expression**

他のテーブルのカラム、ローカル変数、集合関数、またはサブクエリを含まない、有効な Transact-SQL 式です。カラム名、定数、関数、グローバル変数、case 式のいずれか、またはこれらを 1 つ以上の演算子によって結合した組み合わせを指定できます。仮想計算カラムから実体化された計算カラムを参照する場合を除き、計算カラム間での相互参照は許可されません。computed\_column\_expression 内の暗号化カラムは、参照できません。

**materialized | not materialized**

計算カラムがマテリアライズされているかどうかを指定します。これらは、modify 句の予約キーワードです。計算カラムがマテリアライズされているかどうか、つまりテーブルに物理的に保管されているかどうかを示します。デフォルトで、計算カラムは not materialized です(つまり、テーブルに物理的に保管されていません)。また、このパラメータを使用して既存の仮想計算カラムの定義を変更する(仮想計算カラムをマテリアライズする)こともできます。

**table\_name drop partition partition\_name [, partition\_name]...**

1 つ以上のリスト分割または範囲分割を削除します。alter table を使用してハッシュ分割またはラウンドロビン分割を削除することはできません。

削除する各分割について、Adaptive Server は次の処理を実行します。

- 分割上のすべてのデータを削除します。
- システム・カタログから分割定義を削除します。
- このデータ・パーティションを参照しているすべての対応するローカル・インデックス・パーティションを削除します。
- ベース・テーブルおよび各ローカル・インデックスの分割条件を再作成します。
- この分割のすべての統計情報を削除します。
- すべてのグローバル・インデックスを再構築します。

---

**注意** 別のテーブルが参照している分割をテーブルから削除しようとする、削除する分割と参照しているテーブルが空でない場合は、外部キーの制約違反の可能性があるためコマンドは失敗し、Adaptive Server にエラー・メッセージ 13971 が表示されます。

---

**split partition partition\_name into partition\_condition\_clause**

2 つ以上のパーティションにデータを再分配します。

*partition\_condition\_clause*

ソース・パーティション・データの分割方法を指定する条件を示します。通常、条件は数値の範囲またはデータの範囲です。パーティション条件には、ソース・パーティション内のデータのみをすべて含める必要があります。

*partition\_condition\_clause* には、ソース・パーティションと同じセグメントまたは新しいセグメントを指定できます。分割先のパーティション・セグメントを指定しない場合は、ソース・パーティションが置かれているセグメントに新しいパーティションが作成されます。

**merge partition**

マージと互換性のある 2 つ以上のパーティションを 1 つのパーティションに結合します。

*destination\_partition\_name*

新規または既存のパーティション。*destination\_partition\_name* が既存のパーティションである場合、マージ対象のソース・パーティションを指定することはできません。マージ先のパーティション名を指定しない場合は、システムで生成された名前が選択されます。

**move partition**

パーティション (およびそのインデックス) を指定のセグメントに移動します。

*destination\_segment\_name*

パーティションの移動先となる新規または既存のセグメント。*destination\_segment\_name* に “default” を指定することはできません。

例

**例 1** カラムをテーブルに追加します。テーブルの既存の各ローには、Adaptive Server によって null のカラム値が割り当てられます。

```
alter table publishers
add manager_name varchar (40) null
```

**例 2** IDENTITY カラムをテーブルに追加します。テーブルの既存の各ローには、Adaptive Server によってユニークで連続したカラム値が割り当てられます。IDENTITY カラムは、numeric 型または integer 型で、位取りは 0 です。精度は、カラムに挿入できる最大値 ( $10^5 - 1$  または 99,999) を決定します。

```
alter table sales_daily
add ord_num numeric (5,0) identity
```

**例 3** プライマリ・キー制約を authors テーブルに追加します。既存のプライマリ・キー制約または一意性制約がテーブルにある場合は、最初にその既存の制約を削除する必要があります (次の例を参照してください)。

```
alter table authors
add constraint au_identification
primary key (au_id, au_lname, au_fname)
```

**例 4** `au_identification` 制約を削除します。

```
alter table titles
drop constraint au_identification
```

**例 5** `authors` にインデックスを作成します。このインデックスには、値が 16 の `reservepagegap` があり、割り付けられたページの 15 ページごとに空のページが 1 つ残されます。

```
alter table authors
add constraint au_identification
primary key (au_id, au_lname, au_fname)
with reservepagegap = 16
```

**例 6** `authors` テーブルの `phone` カラムにあるデフォルトの制約を削除します。カラムで `null` 値が許可されている場合は、カラム値が指定されないと `null` が挿入されます。カラムで `null` 値が許可されていない場合にカラム値を指定しないで挿入すると、挿入操作は正常に行われません。

```
alter table authors
replace phone default null
```

**例 7** `emp` テーブルを変更して、`ssn` カラムを暗号化し、復号化のデフォルト値を指定します。

```
alter table emp modify ssn encrypt with key1
decrypt_default '000-00-0000'
```

**例 8** 機密性が失われたクレジット・カード・データを復号化します。

```
alter table stolen_ccards
modify ccard decrypt
```

カードがユーザ定義のパスワードで保護されているキーにより暗号化されている場合、このコマンドの前に `set encryption key` コマンドを使用します。

**例 9** 既存のテーブルに暗号化カラムを追加します。`keyname` が省略されているため、Adaptive Server はデータベースのデフォルト暗号化キーを検索します。

```
alter table sales_mgr
add bonus money null encrypt
```

**例 10** `ssn_key` 暗号化キーのパスワードを設定し、既存の `employee` テーブルの `ssn` カラムを暗号化します。



```
set encryption passwd '4evermore' for key ssn_key
alter table employee modify ssn
    encrypt with ssn_key
```

この例の `ssn` が "key1" により暗号化された既存の暗号化カラムである場合、`alter table` により Adaptive Server が "key1" を使用して `ssn` を復号化し、"ssn\_key" を使用して `ssn` を再暗号化します。

**例 11** すでに暗号化されている `salary` カラムに復号化のデフォルト値を追加します。

```
alter table employee replace salary
    decrypt_default $0.00
```

**例 12** 暗号化プロパティを削除せずに `salary` の復号化デフォルト値を削除します。

```
alter table employee replace salary drop
    decrypt_default
```

**例 13** 分割されていないテーブルを、3つのパーティションが個別のセグメントに配置されている範囲分割テーブルに変更します。

```
alter table titles partition by range (total_sales)
    (smallsales values <= (500) on seg1,
    mediumsales values <= (5000) on seg2,
    bigsales values <= (25000) on seg3)
```

**例 14** 別の範囲パーティションを `titles` テーブルに追加します。

```
alter table titles add partition
    (vbigsales values <= (40000) on seg4)
```

**例 15** ローレベル圧縮を使用するように `pub2` データベースの `titles` テーブルを変更します。

```
alter table titles set compression = row
```

**例 16** ページレベルの圧縮を使用するように `sales` テーブルの `Y2009` パーティションを変更します。

```
alter table sales modify partition Y2009
    set compression = page
```

**例 17** `titles` テーブルのロック・スキームをデータロー・ロックに変更します。

```
alter table titles lock datarows
```

**例 18** `primary_author` をデフォルト値として、`authors` テーブルに非 `null` のカラム `author_type` を追加します。

```
alter table authors
  add author_type varchar (20)
  default "primary_author" not null
```

**例 19** advance カラム、notes カラム、contract カラムを titles テーブルから削除します。

```
alter table titles
  drop advance, notes, contract
```

**例 20** authors テーブルの city カラムを、null のデフォルト値を持つ varchar(30) に変更します。

```
alter table authors
  modify city varchar (30) null
```

**例 21** stores テーブルの stor\_name カラムを NOT NULL に変更します。データ型である varchar(40) は変更されません。

```
alter table stores
  modify stor_name not null
```

**例 22** titles テーブルの type カラムを変更し、titles テーブルのロック・スキームを全ページ・ロックからデータロー・ロックに変更します。

```
alter table titles
  modify type varchar (10)
  lock datarows
```

**例 23** titles テーブルの notes カラムを varchar(200) から varchar(150) に変更し、デフォルト値を NULL から NOT NULL に変更して、exp\_row\_size を 40 に指定します。

```
alter table titles
  modify notes varchar (150) not null
  with exp_row_size = 40
```

**例 24** 増分転送属性を mytable に追加します。

```
alter table mytable set transfer table on
```

**例 25** 増分転送属性を mytable から削除します。

```
alter table mytable set transfer table off
```

**例 26** 1 つのクエリで、カラムの追加、変更、削除を行ってから別のカラムを追加します。ロック・スキームを変更し、新しいカラムの exp\_row\_size を指定します。

```
alter table titles
  add author_type varchar (30) null
  modify city varchar (30)
```

```
drop notes
add sec_advance money default 1000 not null
lock datarows
with exp_row_size = 40
```

**例 27** 400 バイトのロー内カラム長をサポートするために `mymsgsgs` テーブルの `description` カラムを変更します。

```
alter table mymsgsgs modify description in row (400)
```

**例 28** 仮想計算カラムを追加します。

```
alter table authors
add fullname compute au_fname + ' ' + au_lname
```

**例 29** 仮想計算カラムをマテリアライズされた計算カラムに変更します。

```
alter table authors modify fullname materialized
```

**例 30** `orders` テーブルが格納されるパーティションを2つのパーティションに分割します。

```
alter table orders
split partition P2
into
( P5 values <= (25000) on seg2,
  P6 values <= (50000) on seg3)
```

**例 31** `sales` テーブルが格納される複数のパーティションを1つのパーティションにマージします。

```
alter table sales
merge partition Q1, Q2, Q3, Q4
into Y2007
```

**例 32** `orders` テーブルを `seg4` セグメントに移動します。

```
alter table orders
move partition P2 to seg4
```

**例 33** データ・コピーを指定して、`titles` テーブルから `total_sales` カラムを削除します。

```
alter table titles
drop total_sales
with no datacopy
```

#### 使用法

- 仮想ハッシュ・テーブルを含むセグメントには `alter table` を使用できません。

- VHASH テーブルを含むセグメントに対しては、**alter table** を使用することはできません。仮想ハッシュ・テーブルは排他セグメントを 1 つのみ持つ必要があり、これを他のテーブルやデータベースと共有することはできないためです。
- テーブルにカラムを追加したり、テーブルのカラムを変更または削除する前に、**sp\_depends** を実行して、変更するテーブルに依存するストアド・プロシージャがないかどうかを確認してください。このようなストアド・プロシージャが存在する場合は、それを削除して、テーブル・スキーマを変更してから必要に応じてストアド・プロシージャを再作成してください。
- **select \*** を使用しているストアド・プロシージャが、変更されたテーブルを参照する場合は、**with recompile** オプションを使用しても結果セットに新しいカラムは表示されません。新しいカラムが表示されるようにするには、ストアド・プロシージャを削除してから再作成してください。そうしないと、テーブルが変更され、新しいカラムがテーブルに追加されている場合、プロシージャ内の **insert into table1 select \* from table2** によって正しくない結果が生じることがあります。
- テーブル所有者が **alter table** を使用すると、コマンドの実行時にアクセス・ルールが無効化され、コマンドの実行が完了した時点でアクセス・ルールが有効化されます。アクセス・ルールが無効化されるのは、**alter table** の実行中にテーブルのデータがフィルタされることを防ぐためです。
- **clustered** を指定し、**on segment\_name** オプションを使用した場合は、テーブル全体が指定したセグメントに移行します。これは、インデックスのリーフ・レベルに実際のデータ・ページがあるためです。
- **alter table.. .alter table.. . transfer table** ではデータがコピーされます (カラムの追加または削除と同様)。パフォーマンスの点では、非常に負荷がかかるコマンドです。
- **alter table** はテーブルを変更する前に検査制約のためのエラー・チェックを実行します。
- **on segment\_name** をパーティションに使用する場合は、**create database** または **alter database** を使用して論理デバイスが事前にデータベースに割り当てられている必要があります。また、**sp\_addsegment** を使用してセグメントがデータベースに作成されている必要もあります。データベースで使用できるセグメント名のリストについては、システム管理者に問い合わせるか、**sp\_helpsegment** を使用して参照してください。

## 制限事項

---

**警告！** システム・テーブルは変更しないでください。

---

- デフォルト値を指定する場合、データ型が `bit` のカラムは、既存のテーブルに追加できません。デフォルト値は `0` または `1` である必要があります。
- テーブル内の最大カラム数は次のとおりです。
  - 全ページロック (APL) とデータオンリーロック (DOL) の両方のテーブルの固定長カラムでは 1024
  - APL テーブルの可変長カラムでは 254
  - DOL テーブルの可変長カラムでは 1024
- APL テーブルの可変長カラムの数が 254 を超えている場合、`alter table` を実行するとエラーが発生します。
- テーブルのロック・スキーマの変更後にコンパイル済みオブジェクトを削除し、再作成してください。
- 次のカラムには `no datacopy` パラメータを使用できません。
  - マテリアライズされたカラムまたは仮想計算カラム
  - 暗号化カラム
  - XML カラム
  - Java カラム
  - `timestamp` または `bit` データ型を使用するカラム
- ロー内の Java カラムの最大長は、テーブルのスキーム、ロック方式、ページ・サイズに対応した可変長カラムの最大サイズによって決まります。
- テーブルを異なるロック・スキームに変換するとき、ソース・テーブルのデータはターゲット・テーブルの制限に違反することはできません。たとえば、254 を超える可変長カラムのある DOL テーブルを APL テーブルに変換しようとしても、APL テーブルの可変長カラムは 254 以内に制限されているため、`alter table` は失敗します。
- 固定長データ (`char`、`binary` など) があるカラムには、表 1-1 に示されるように最大サイズの制限があります。

表 1-1: 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	2KB (2048 バイト)	1962	1960 バイト
	4KB (4096 バイト)	4010	4008 バイト
	8KB (8192 バイト)	8106	8104 バイト
	16KB (16384 バイト)	16298	16296 バイト
DOL テーブル	2KB (2048 バイト)	1964	1958 バイト
	4KB (4096 バイト)	4012	4006 バイト
	8KB (8192 バイト)	8108	8102 バイト
	16KB (16384 バイト)	16300	16294 バイト - テーブルに可変長カラムがない場合
	16KB (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 - テーブルに1つ以上の可変長カラムが存在する場合。*

\* 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。

- 1 ローあたりの可変長データの最大バイト数は、テーブルのロック・スキームによって異なります。次の表は、APL テーブルの最大カラム・サイズを示します。

ページ・サイズ	ローの最大長	カラムの最大長
2KB (2048 バイト)	1960	1960
4KB (4096 バイト)	4008	4008
8KB (8192 バイト)	8104	8157
16KB (16384 バイト)	16296	16227

次の表は、DOL テーブルの最大カラム・サイズを示します。

ページ・サイズ	ローの最大長	カラムの最大長
2KB (2048 バイト)	1960	1958
4KB (4096 バイト)	4008	4006
8KB (8192 バイト)	8157	8102
16KB (16384 バイト)	16294	16294

- `alter table` を使用して、宣言制約または検査制約を追加してから同じバッチまたはプロシージャ内でテーブルにデータを挿入することはできません。`alter` 文と `insert` 文を 2 つの異なるバッチまたはプロシージャに分けるか、`execute` を使用してアクションを別々に実行してください。
- デフォルトを含む次の変数は、`alter table` 文で使用することはできません。

```

declare @a int
select @a = 2
alter table t2 add c3 int
default @a

```

こうすると、「デフォルトの中で変数は許されません」というエラー・メッセージ 154 が表示されます。

- SQL ユーザ-定義関数は現時点では `create proxy table`、`create table at remote server`、または `alter table` ではサポートされていません。

---

**注意** SQL 関数の実行に必要な構文は次のようになります。  
`username.functionname()`

---

#### *alter table* と暗号化カラム

- `alter table` を暗号化カラムの追加または変更を使用する場合、テーブルに多くのローがあると、長い時間がかかることがあります。
- 暗号化のためにカラムを変更すると、テーブルのロー・サイズが増加する場合があります。
- 次の場合には、カラムの暗号化または復号化に `alter table` を使用することはできません。
  - カラムがクラスタード・インデックスまたは配置インデックスに属する場合。そのようなカラムを暗号化または復号化するには、インデックスを削除し、カラムを変更して、インデックスを再作成する。
  - テーブルにトリガが定義されている場合。カラムを変更する前にトリガを削除する。その後、トリガを再作成する。
- クラスタード・インデックスまたは配置インデックスに属する暗号化カラムのデータ型を変更すると、インデックスが不順になり、`alter table` でエラーが表示されます。データ型を変更する前にインデックスを削除してください。その後、インデックスを再作成します。
- 暗号化できるデータ型は次のとおりです。
  - `int`、`smallint`、`tinyint`
  - `unsigned int`、`unsigned smallint`、`unsigned tinyint`
  - `bigint`、`unsigned bigint`
  - `decimal`、`numeric`
  - `float4`、`float8`

- money、smallmoney
- date、time、smalldatetime、datetime、bigdatetime
- char、varchar
- unichar、univarchar
- binary、varbinary
- bit
- 暗号化されたデータのディスク上での基本データ型は、varbinaryです。NULL 値は暗号化されません。
- クラスタ化インデックスまたは配置インデックスに属する暗号化カラムのデータ型を変更すると、インデックスが不順になり、alter table にエラーが表示されます。データ型を変更する前にインデックスを削除してから、インデックスを再作成します。
- alter table を使用して次の操作を行うと、エラー・メッセージが表示されます。
  - 計算カラムを暗号化カラムに変更するか、暗号化カラムを計算カラムに変更する。
  - 計算カラムで使用された式で参照されているカラムを暗号化する。
  - 暗号化カラムを参照するように計算カラムを変更する。
  - ファンクション・インデックスのメンバであるカラムを暗号化する。
  - 暗号化カラムをパーティション・キーとして指定する。
  - すでにパーティション・キーとして使用されているカラムを暗号化できるようにする。

---

**注意** 暗号化カラム間の参照整合性は、カラムが同じキーで暗号化されている場合にサポートされます。詳細については、『暗号化カラム・ユーザーズ・ガイド』の「データの暗号化」を参照してください。

---

#### テーブルの圧縮の変更

- 将来のデータの挿入または更新では、set compression を使用してテーブルの圧縮レベルを変更します。set compression は、すでに圧縮されている既存のデータ・ローまたはデータ・ページには影響しませんが、テーブルへの排他的アクセスが必要です。



- テーブルの圧縮レベルを変更する同じコマンドでパーティション圧縮レベルを変更することはできません。これらのオペレーションは別々のコマンドとして実行する必要があります。
- `set compression` は、その他の `set` パラメータとともに使用できます。
- テーブルの圧縮レベルの変更は、すでに明示的に定義された圧縮レベルのないパーティションのみに影響します。明示的に定義された圧縮レベルのないすべてのパーティションは、テーブルの圧縮レベルを黙示的に継承します。たとえば、テーブルの圧縮レベルを非圧縮からロー・レベルの圧縮に変更した場合、圧縮レベルが `none` のすべてのパーティションは変更されませんが、圧縮レベルが定義されていないパーティションはロー・レベル圧縮に変更されます。
- テーブルの圧縮レベルを変更しても、既存のカラムの圧縮レベルは変更されません。たとえば、`my_table` およびそのカラムが圧縮されていない場合、`my_table` の圧縮レベルを変更しても、そのカラムは最初是非圧縮のままです。ただし、それらのカラムに十分なデータが書き込まれ、圧縮メカニズムがトリガされると、`Adaptive Server` はそれらを個別に圧縮します。
- 新しく追加したカラムのデフォルトの動作は、テーブルの圧縮設定によって異なります。圧縮テーブルの場合、カラムのデータ型によって圧縮レベルが決定されます。非圧縮のテーブルの場合、新しいカラムも非圧縮になります。
- テーブルに圧縮済みのマテリアライズされた計算カラムを追加することもできますが、後で圧縮することもできます。

#### compression パラメータとその他の alter table パラメータ間の相互作用

コマンドがデータの移動を要する場合、移動先パーティションが圧縮されていると、`Adaptive Server` では、移動元パーティションにある非圧縮のデータ・ローがすべて圧縮されます。`compression` 句を含めると、`alter table` にはパラメータ間の次のような相互作用が含まれます。

- `set` 一 次のような制限があります。
  - `set` を `add` 句、`drop` 句、または `modify` 句と組み合わせることはできない。
  - `modify partition set` 句をその他の `modify column_name` パラメータと組み合わせることはできない。
  - `all` キーワードを `modify partition` とともに使用してその句にパーティション名を含めることはできない。

- **add**
  - **NULL** が許可されている圧縮カラムと **NULL** が許可されていない圧縮カラムを既存のテーブルに追加することができる。**NULL** が許可されていないカラムでは、データの移動が必要である。
  - 圧縮テーブルに追加されたカラムが適切なデータ型用に設定されている場合、ロー圧縮が使用される。
  - カラムのデータ型を、圧縮テーブルでロー圧縮可能な型に変更しても、カラムの圧縮レベルは変更されない。
  - 新しいカラムに **not compressed** を指定しなかった場合、テーブルの圧縮レベルが継承される。たとえば、テーブルがローレベル圧縮の場合、そのテーブルに追加するカラムもすべてローレベル圧縮を使用する。
- **drop**
  - 圧縮カラムを削除するとデータの移動が生じる。
  - テーブルまたはパーティション内のその他のカラムが圧縮されていない場合や、圧縮できない場合は、最後の圧縮カラムを削除する前に、圧縮状態を **none** に変更する必要がある。
- **modify**
  - 既存のテーブル内の圧縮カラムを **not compressed** に変更することができる。また、その逆の変更も可能である。
  - カラムのデータ型は圧縮レベルと同時に変更できる。ただし、変更後のデータ型はロー圧縮可能である必要がある。変更後のデータ型が圧縮可能でない場合、**Adaptive Server** は **modify** オペレーション中に圧縮の要求を無視する。
  - 圧縮カラムとして作成できないカラムの圧縮レベルを変更しようとする、エラーが発行される。たとえば、仮想計算カラムまたはロー内の **Java** カラムを圧縮しようとする、エラー・メッセージが表示される。
- **add**、**drop**、**modify** の組み合わせ
  - 1 つの **alter table** 文に複数の圧縮カラムが含まれる複数の **add** パラメータ、**drop** パラメータ、または **modify** パラメータを発行できる。この **alter table** コマンドでのデータの移動は、パラメータのデータの移動の制限によって左右される。
  - **alter table** でデータの移動が必要な場合、**add** パラメータ、**drop** パラメータ、または **modify** パラメータの影響を受けないカラムの圧縮レベルには変更できない。

- テーブルの再分割 – 新しいパーティションに対して `compression` 句を指定しない場合、`Adaptive Server` では次のように新しいパーティションの圧縮レベルが設定されます。
  - ソース・テーブルおよびそのすべてのパーティションが非圧縮の場合は、非圧縮
  - ソース・テーブルのすべてのパーティションが同じ圧縮レベルで圧縮されている場合は、ソース・テーブルと同じ圧縮レベル
  - 次の場合は非圧縮になる。
    - テーブルまたは個々のパーティションの圧縮が異なる場合
    - ソース・テーブルは圧縮されていないが、そのパーティションの一部が圧縮されている場合

元の圧縮パーティションから新しいパーティションに圧縮属性をユニークにマップし移行することは困難な場合があるため、`Adaptive Server` では、新しいパーティションは圧縮されません。`alter table` コマンドの一部として圧縮レベルを明示的に指定し、圧縮する必要があるターゲット・パーティションを指定する必要があります。

- `add partition` – `compression` 句でパーティション・レベルを指定しない場合、新しく追加したパーティションはテーブルの圧縮レベルを継承します。
- `drop partition` – テーブルに複数のパーティションが含まれる場合、テーブルの圧縮パーティションを 1 つだけ削除しても、テーブルの圧縮レベルは変更されません。

テーブルに圧縮が定義されている場合、パーティションが削除されても圧縮はそのまま、その後のパーティションでは `Adaptive Server` により自動的に圧縮が設定されます。
- ロック・スキームの変更 – テーブルのロック・スキームを全ページロックからデータオンリーロックに変更した場合、`Adaptive Server` では、データの移動が必要になります。ロック・スキームの変更時、テーブルまたは個々のパーティションの圧縮レベルを同時に変更することはできません。代わりに、`set compression` コマンドを実行して、ロック・スキームの変更前に圧縮レベルを指定する必要があります。

- テーブルの分割解除 – テーブルの分割を解除する場合、少なくとも 1 つのソース・パーティションが圧縮されていると、**alter table** の実行によりテーブル全体が圧縮済みとマークされます。テーブルが最初に非圧縮になっている場合、Adaptive Server では警告メッセージが発行されます。
- その他のコマンド – カラムのデフォルト値を指定するパラメータ、トリガを有効化または無効化するパラメータ、カラム・レベルまたはテーブルレベルの制約を追加または削除するパラメータなどは、カラムレベルまたはパーティション・レベルの圧縮またはコピー・データを指定するコマンドと組み合わせることはできません。
- **alter table** にデータの移動が含まれない場合は、既存のデータは影響を受けず、このコマンドが挿入されると Adaptive Server によってテーブルの圧縮レベルがデータに適用されます。**alter table** にデータの移動が含まれる場合は、テーブルの圧縮レベルに従って既存のデータが圧縮または圧縮解除されます。
- 次の **alter table** イベントには、データの移動が含まれます。
  - 非 null カラムの追加
  - カラムの削除
  - 長さ拡張のためのカラムの変更 (たとえば `smallint` を `int` に、または `char(3)` を `varchar(45)` にするなど)
- 次の **alter table** イベントには、データの移動は含まれません。
  - null カラムの追加
  - null テキスト・カラムの追加 (非 null テキスト・カラムの追加には制限がある)
  - 長さ拡張のための可変長カラムの変更 (たとえば `varchar(5)` を `varchar(20)` に、または `varchar(20)` を `varchar(40)` にするなど)
- プロキシ・テーブルやパーティション、またはプロキシ・テーブルのカラムは圧縮できません。

#### ラージ・オブジェクトを使用したテーブルの圧縮レベルの変更

- テーブルのラージ・オブジェクト (LOB) 圧縮レベルの変更は、明示的に定義された圧縮レベルのない LOB カラムのみに影響します。明示的に定義された圧縮レベルのないカラムは、テーブルの圧縮レベルを暗黙的に継承します。

- 新しく追加した、LOB compression 句を指定していない LOB カラムのデフォルトの動作は、テーブルの LOB 圧縮レベルによって異なります。LOB 圧縮テーブルの場合、Adaptive Server では、カラムに対してテーブルの LOB 圧縮レベルを使用します。LOB の非圧縮のテーブルの場合、新しく追加した LOB カラムは非圧縮のままです。

LOB データを含むテーブルの場合、compression パラメータとその他の alter table パラメータ間の相互作用は次のとおりです。

- drop column – カラムの削除後、テーブルに圧縮 LOB カラムが含まれない場合、テーブルではテーブルレベルの LOB カラム圧縮レベルが使用されます。
  - add column
    - NULL が許可されている圧縮 LOB カラムは追加できるが、NULL が許可されていない圧縮 LOB カラムは追加できない。
    - LOB 圧縮が設定されていないテーブルでは、デフォルトでは、新しく追加した LOB カラムは圧縮されない。LOB 圧縮サブ句により新しく追加した LOB カラムは、指定に応じて圧縮される場合と、圧縮されない場合がある。
  - modify column
    - 既存の圧縮 LOB カラムの圧縮を解除できる。新しく挿入したデータは圧縮解除されるが、既存のデータは圧縮されたままである。
    - 既存の LOB カラムの圧縮レベルを変更できる。新しく挿入したデータは新しい圧縮レベルを想定するが、既存のデータでは元の圧縮レベルが維持される。
    - 非圧縮 LOB カラムを compressed に変更することができる。
    - 通常のカラムを LOB カラム (圧縮、非圧縮を問わず) に変更することはできない。
    - 圧縮 LOB カラムは、以下のようなカラムに変更することができる。
      - 圧縮 text カラム (nchar、nvarchar、unichar、univarchar を使用)
      - 圧縮 image カラム (varbinary と binary を使用)
      - 圧縮 unitext カラム (nchar、nvarchar、unichar、univarchar、varbinary、binary を使用)
- 圧縮ロー外 java カラムを通常のカラムに変更することはできません。

Adaptive Server は LOB データの圧縮を解除し、通常のカラム長に収まるように必要に応じてデータをトランケートし、通常のデータ型に変換します。通常カラムの最大長は、Adaptive Server のページ・サイズによって決定されます。

- add、drop、modify、set lob\_compression の組み合わせ
  - 1 つの alter table コマンドで複数の add サブコマンド、drop サブコマンド、または modify サブコマンドを発行できる。または 1 つ以上の圧縮カラムを必要とする set lob\_compression サブ句および set compression サブ句を発行できる。
  - LOB 圧縮テーブルにカラムを追加し、コマンドに set lob\_compression = 0 を含めると、新しく追加したカラムは圧縮されない。
  - 通常非圧縮テーブルにカラムを追加し、コマンドに set lob\_compression = compression\_level を含めると、新しく追加したカラムが圧縮される。

既存の LOB データは alter table コマンドの影響を受けません。LOB 圧縮属性の変更による影響を受けるのは将来の DML のみです。既存の LOB データの圧縮または圧縮解除には、update および select into を使用します。

#### テーブルについての情報の取得

- テーブルとカラムについての情報を取得するには、sp\_help を使用してください。
- テーブル名を変更するには、システム・プロシージャ sp\_rename を実行します (システム・テーブルの名前は変更しないでください)。
- 整合性制約 (unique、primary key、references、check) または default 句については、この章の「create table」を参照してください。

#### インデックスの昇順または降順の指定

- インデックスのソート順を指定するには、インデックス・カラムの名前の後に asc または desc キーワードを使用します。カラムが、クエリの order by 句で指定された順序と同じ順序になるようにインデックスを作成すると、クエリ処理中のソートする手順を省略できます。『パフォーマンス&チューニング・シリーズ：基本』の「インデックスとパフォーマンス」を参照してください。

#### データベース間の参照整合性制約の使用

- データベース間の制約を作成すると、各データベースの sysreferences テーブルに次の情報が格納されます。

表 1-2: 参照整合性の制約について保管される情報

sysreferences に格納される情報	参照先テーブルについての情報を持つカラム	参照元テーブルについての情報を持つカラム
キー・カラム ID	refkey1 ~ refkey16	fokey1 ~ fokey16
テーブル ID	reftabid	tableid
データベース ID	pmrydbid	frgnbdbid
データベース名	pmrydbname	frgndbname

- 参照元テーブルやそのデータベースを削除すると、Adaptive Server は外部キー情報を参照先データベースから削除します。
- 参照元テーブルは参照先テーブルの情報に依存しているため、Adaptive Server で次の操作は実行できません。
  - 参照先テーブルを削除する
  - 参照先テーブルがある外部データベースを削除する
  - `sp_renamedb` を使用していずれかのデータベースの名前を変更する

`alter table` を使用して、データベース間の参照整合性制約を最初に削除する必要があります。

- データベース間の制約を追加または削除した場合や、データベース間の制約を含むテーブルを削除した場合は、影響を受けたデータベースを両方ともダンプしてください。

---

**警告！** これらのデータベースの以前のダンプをロードすると、データベースの破壊の原因となります。

---

- `sysreferences` システムテーブルには、外部データベースの `name` と ID 番号が格納されます。Adaptive Server では、`load database` を使用してデータベース名を変更したり、別のサーバにデータベースをロードしたりすると、参照整合性は保証されません。

---

**警告！** データベースを別の名前ロードするか、またはデータベースを別の Adaptive Server に移動するためにデータベースをダンプするには、事前に `alter table` を使用して外部参照整合性制約をすべて削除してください。

---

### デフォルトの変更

- カラムのデフォルトを作成するには、`create table` 文または `alter table` 文の中でカラム制約としてデフォルトを宣言するか、`create default` 文を使用してデフォルトを作成し、`sp_bindefault` を使用してカラムにバインドします。
- `sp_bindefault` を使用してカラムにバインドされたユーザ定義のデフォルトは置換できません。最初に、`sp_unbindefault` を使用してデフォルトのバインドを解除してください。
- `create table` または `alter table` を使用してデフォルトのカラム値を宣言する場合は、`sp_bindefault` を使用してそのカラムにデフォルトをバインドすることはできません。そのデフォルトを `null` に変更して削除した後で、ユーザ定義のデフォルトをバインドしてください。デフォルトを `NULL` に変更すると、デフォルトはバインド解除され、`sysobjects` テーブルから削除されます。

### インデックスの記憶領域管理プロパティの設定

- `alter table` 文内の記憶領域管理プロパティ `fillfactor`、`max_rows_per_page`、`reserverpagegap` は、`primary key` 制約または `unique` 制約で作成されるインデックスに適用されます。制約によって全ページロック・テーブルにクラスタード・インデックスが作成された場合、記憶領域管理プロパティはテーブルのデータ・ページに影響します。
- テーブルやインデックスの `max_rows_per_page` または `reserverpagegap` の変更、テーブルの `exp_row_size` 値の変更、あるいは `fillfactor` 値の格納を行うには、`sp_chgattribute` を使用します。
- インデックスの記憶領域管理プロパティは、インデックスが次のような場合に適用されます。
  - テーブルのロック・スキームを全ページ・ロックからデータオンリー・ロックへ、またはデータオンリー・ロックから全ページ・ロックへ変更する `alter table` コマンドの結果としてインデックスが再作成される場合。「[ロック・スキームの変更](#)」(92 ページ) を参照してください。
  - `reorg rebuild` コマンドの一部として自動的にインデックスが再構築される場合。
- テーブルの現在有効な記憶領域管理プロパティを確認するには、`sp_help` を使用します。インデックスの現在有効な記憶領域管理プロパティを確認するには、`sp_helpindex` を使用します。



- 記憶領域管理プロパティ `fillfactor`、`max_rows_per_page`、`reservepagegap` は、テーブルとインデックスの記憶領域の使用管理に、以下のように役立ちます。
  - `fillfactor` はインデックスの作成時に余分な領域をページに残すが、`fillfactor` は永続的に保持されるわけではありません。これは、すべてのロック・スキームに該当します。
  - `max_rows_per_page` は、データまたはインデックス・ページのロー数を制限します。その主な用途は、全ページロック・テーブルでの同時実行性を向上させることです。
  - `reservepagegap` は空のページと満杯のページとの比率を指定して、エクステンツ割り付けを実行するコマンドに適用します。これは、すべてのロック・スキームに該当します。

記憶領域管理プロパティは、`alter table` コマンドや `reorg rebuild` コマンドの実行中に適用されるように、テーブルやインデックス用に格納できます。

- 次の表は、記憶領域管理プロパティとロック・スキームの有効な組み合わせを示します。`alter table` コマンドによってテーブルが変更されたために組み合わせが不適切になった場合は、システム・テーブルに格納された値は残りますが、テーブルへの操作の実行中は適用されません。プロパティが有効になるようにテーブルのロック・スキームを変更すると、値が使用されます。

パラメータ	全ページ	データページ	データロー
<code>max_rows_per_page</code>	可	不可	不可
<code>reservepagegap</code>	可	可	可
<code>fillfactor</code>	可	可	可
<code>exp_row_size</code>	不可	可	可

- 次の表は、記憶領域管理プロパティのデフォルト値と、デフォルト値を使用したときの効果を示します。

パラメータ	デフォルト	デフォルトを使用したときの効果
<code>max_rows_per_page</code>	0	ページの最大数のロー (256 まで) に適合する。
<code>reservepagegap</code>	0	ギャップを残さない
<code>fillfactor</code>	0	リーフ・ページを完全にパックする

*max\_rows\_per\_page* から *exp\_row\_size* への変換

- テーブルに *max\_rows\_per\_page* が設定されている場合、そのテーブルを全ページ・ロックからデータオンリー・ロックに変換すると、値が *exp\_row\_size* 値に変換されてから `alter table...lock` コマンドによってテーブルがその新しいロケーションにコピーされます。*exp\_row\_size* は、コピー中に適用されます。次の表は、値の変換方法を示します。

<i>max_rows_per_page</i> の設定	<i>exp_row_size</i> の設定
0	default <i>exp_row_size percent</i> で設定されたパーセント値
255	1、つまり完全にバックされたページ
1 - 254	次のいずれか小さい方 <ul style="list-style-type: none"> <li>• 最大ロー・サイズ</li> <li>• 2002/<i>max_rows_per_page</i> 値</li> </ul>

*reservepagegap* の使用

- 大量の領域を使用するコマンドでは、ページごとに割り付けるのではなく、エクステントを割り付けることによって新しい領域を割り付けます。*reservepagegap* キーワードを使用すると、これらのコマンドは、分割されるページまたはローの転送元のページの近くで以後のページ割り付けが行われるように空のページを残します。
- テーブルの *reservepagegap* 値は *sysindexes* に格納され、テーブルのロック・スキームが全ページ・ロックからデータオンリー・ロックに変更される時、またはその逆に変更される時に適用されます。格納された値を変更するには、*sp\_chgattribute* を使用してから、`alter table` を実行します。
- *clustered* キーワードで指定された全ページロック・テーブルの *reservepagegap* は、`create table` または `alter table` で以前に指定された値を上書きします。

## パフォーマンス向上のためのテーブルの分割

- 分割されていないテーブルを分割したり、すでに分割されているテーブルを再分割するには `partition by` 句を使用します。この作業にはデータ・コピーが必須であり、すべてのデータ・ローが、指定された分割条件に従って再分配されます。*Adaptive Server* の設定で並列処理が有効な場合は、この作業を並列に実行できます。`select into/bulkcopy/pllsort` データベース・オプションを `true` に設定する必要があります。テーブルにインデックスがある場合は、インデックスを削除するまで以下の変更を行うことができません。
  - 分割されていないテーブルをセマンティック分割テーブルへ

- 分割方式
- 分割キー — 分割の数、分割の境界、分割の場所などの他の分割属性を変更する場合は、インデックスを削除する必要はない。インデックスは自動的に作成される。分割キーと境界の制限の詳細については、「[create table](#)」(233 ページ) を参照。
- `add partition` 句を使用して、リスト分割テーブルまたは範囲分割テーブルに空のパーティションを追加できますが、ハッシュ分割テーブルまたはラウンドロビン分割テーブルには追加できません。

範囲分割テーブルの場合、空のパーティションはパーティション条件の最上位にしか追加できません。既存の最後のパーティションに上限値 (`values <= (MAX)`) が設定されている場合、新しいパーティションは追加できません。

- `partition number_of_partition` および `unpartition` 句は、15.0 以前のバージョンの Adaptive Server との互換性のために提供されています。`partition number_of_partition` を使用して (`number_of_partition-1`) 個のラウンドロビン方式による空のパーティションを、分割されていないテーブルにのみ追加できます。既存のデータは最初のパーティションに配置され、それ以降のデータはすべてのパーティションに分配されます。テーブルにグローバル・クラスタード・インデックスがある場合、後続のデータ・ローは最初のパーティションに配置されます。データを再配分するには、インデックスを削除して再作成します。

---

**注意** これらのコマンドは、データを移動しません。しかし、Adaptive Server よりいくつかの内部手順が実行されるため、特に大きなテーブルにこのコマンドを実行すると、即時に行われません。データの破損を避けるには、テーブルの分割または分割解除中に操作を中断しないでください。

---

`unpartition` 句は、インデックスのない、ラウンドロビン分割テーブルにのみ使用できます。

- システム・テーブルを分割することはできません。
- リモート・プロキシ・テーブルを分割することはできません。
- ユーザ定義のトランザクション内では、分割に関連する `alter table` コマンドを発行できません。
- アクティブなオープン・カーソルがあるテーブルでは、`partition by` 句を使用して分割プロパティを変更できません。

- `partition by` を使用した後は、完全なデータベース・ダンプを実行しないと、`dump transaction` を使用できません。
- 分割キーの一部であるカラムは削除できません。
- キー・カラムの変更には注意してください。キー・カラムのデータ型を変更すると、データが複数のパーティションに再配分されることがあります。『[Transact-SQL ユーザーズ・ガイド](#)』を参照してください。
- テーブルの分割プロパティを変更すると、スキームのカウントが増加します。これにより、このテーブルにアクセスする既存のストアド・プロシージャが次回実行されたときに、このストアド・プロシージャが再コンパイルされます。

#### 計算カラムの使用

- `null` 入力可能性と実体化のプロパティを指定しないで新しい計算カラムを追加すると、デフォルト・オプションは `null` 入力可能および非実体化になります。
- 新しい実体化された計算カラムを追加すると、`computed_column_expression` がテーブル内の既存のローごとに評価され、その結果がテーブルに保管されます。
- 新しい計算カラムを追加することと、計算カラムが参照するベース・カラムを追加または変更することは、同時には行えません。
- 既存の計算カラムの定義を完全に変更できます。これにより、計算カラムを削除し、同じ名前でも新しい計算カラムを容易に追加できます。このようなカラムは新しい計算カラムと同様に動作します。指定しない限り、デフォルトでカラムはマテリアライズされず、`null` 入力可能に設定されます。
- 既存の計算カラムの実体化プロパティは、このプロパティを定義する式や `null` 入力可能性などの他のプロパティを変更しないで変更できます。
- マテリアライズされた、`null` 入力不可の計算カラムを仮想カラムに変更するには、`modify` 句に "`null`" を指定する必要があります。
- 実体化されない計算カラムを実体化されるように変更すると、`computed_column_expression` がテーブル内の既存のローごとに評価され、その結果がテーブルに保管されます。
- インデックス・キーである既存のカラムを変更すると、そのインデックスは再構築されます。

- 実体化された計算カラムがインデックス・キーとして使用されている場合、このカラムを仮想カラムに変更できません。変更するには、インデックスを削除する必要があります。
- 通常のカラムを計算カラムに変更することや、計算カラムを通常のカラムに変更することはできません。
- 計算カラムによって参照されているベース・カラムは、変更または削除できません。
- 計算カラムがインデックス・キーとして使用されている場合、このカラムは削除できません。

#### IDENTITY カラムの追加

- テーブルに **numeric** 型カラムまたは整数の **IDENTITY** カラムを追加するときは、カラムの精度が既存のローを収容するのに十分なサイズかどうかを確認してください。ローの数が  $10^{\text{precision} - 1}$  を超えると、エラー・メッセージが表示され、カラムは追加されません。
- **IDENTITY** カラムをテーブルに追加するとき、**Adaptive Server** は次のように動作します。
  - **IDENTITY** のカラム値が生成されるまでテーブルをロックします。テーブルに多数のローがあると、この処理に時間がかかることがある。
  - 既存の各ローに、1 で始まる、ユニークで連続した **IDENTITY** のカラム値を割り当てる。
  - テーブルへの各挿入オペレーションのログをとる。多数のローがあるテーブルに **IDENTITY** カラムを追加する場合は、その前に **dump transaction** を使用して、データベースのトランザクション・ログをクリアする。
- テーブルにローを挿入するたびに、**Adaptive Server** によって最後の値より 1 大きい **IDENTITY** カラム値が生成されます。この値は、**alter table** 文でカラムに宣言されたデフォルトや、**sp\_bindefault** を使用してそのカラムにバインドされたデフォルトに優先します。

#### テーブル・スキーマの変更

- **add**、**drop**、**modify**、および **lock** サブ句は、既存のテーブルのスキーマを変更するのに便利です。同一のカラム名が文の中で複数回参照されないかぎり、これらのサブ句は 1 つの文の中でいくつでも、どのような順序でも使用できます。

- `select *` を使用しているストアド・プロシージャが、変更されているテーブルを参照する場合は、`with recompile` オプションを使用しても結果セットに新しいカラムは表示されません。新しいカラムが表示されるようにするには、ストアド・プロシージャを削除してから再作成してください。
- トリガが適切に起動されるようにするには、`add`、`drop`、`modify`、または `lock` オペレーションの実行後に、変更されたテーブルのすべてのトリガを削除し、再作成します。
- `alter table` を使用して `not null` カラムを追加した場合、Adaptive Server はエラー・メッセージを返します。
- テーブルのカラムをすべて削除することはできません。また、残っている最後のカラムを削除することもできません（たとえば、5つのカラムがあるテーブルから4つのカラムを削除した場合は、残りの1つのカラムは削除できません）。テーブルをデータベースから削除するには、`drop table` を使用します。
- データのコピーは、次の場合に必要です。
  - カラムを削除するとき
  - `NOT null` カラムを追加するとき
  - ほとんどの `alter table ... modify` コマンドに対して

特定の `alter table` コマンドにデータのコピーが必要かどうかを調べるには、`set noexec on` オプションと `showplan on` オプションを使用してください。

- ある `alter table` コマンドがデータのコピーを必要とするときは、他の `alter table` コマンド (`add`、`drop`、または `modify`) を使用して、修正されたテーブルのロック・スキームで変更を指定できます。
- `alter table` がデータ・コピーを実行する場合、変更しようとしているスキーマのあるテーブルを含むデータベースで `select into/bulkcopy/pilsort` をオンにします。
- 修正されたテーブルには、既存の記憶領域管理プロパティ (`max_rows_per_page`、`fillfactor` など) とテーブルのインデックスが保持されます。
- データのコピーが必要な `alter table` は、トリガを起動しません。
- `alter table` を使用すると、コンポーネント統合サービスによって作成され、保守されているリモート・プロキシ・テーブルのスキーマを変更できます。『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。

- データのコピーと、テーブル・レベルまたは参照整合性制約の追加を1つの文で実行することはできません。
- データのコピーと、クラスタード・インデックスの作成を1つの文で実行することはできません。
- not null カラムを追加する場合は、デフォルト句も指定します。この規則には、次のような例外があります。ユーザ定義データ型のカラムを追加し、そのデータ型にデフォルトがバインドされている場合、デフォルト句を指定する必要はありません。
- 全ページ・ロック・テーブルでは、常にカラムの追加、削除、変更を実行できます。ただし、データオンリー・ロック・テーブルにあるカラムの追加、削除、変更には制限があります。次の表は、この制限を示します。

インデックスのタイプ	全ページ・ロック、分割されたテーブル	全ページ・ロック、分割解除されたテーブル	データオンリー・ロック、分割されたテーブル	データオンリー・ロック、分割解除されたテーブル
クラスタード	可	可	不可	可
ノンクラスタード	可	可	可	可

クラスタード・インデックスがある分割されたデータオンリー・ロックテーブルのカラムを追加、削除、または変更する必要がある場合、次の操作を実行できます。

- クラスタード・インデックスを削除します。
  - データオンリー・ロック・テーブルを変更します。
  - クラスタード・インデックスを再作成します。
- カラムとして NOT NULL の Java オブジェクトを追加することはできません。デフォルトでは、すべての Java カラムは常に NULL のデフォルト値を持っており、varbinary 文字列または image データ型として格納されます。
  - 変更でデータ・コピーが必要とされる場合は、Java カラムを含む分割テーブルを変更できません。この場合は、まずテーブルの分割を解除して、alter table を実行してから、テーブルを再分割します。
  - キー・カラムは、インデックスまたは参照整合性制約から削除できません。キー・カラムを削除するには、インデックスまたは参照整合性制約を削除してから、キー・カラムを削除します。『Transact-SQL ユーザーズ・ガイド』を参照してください。

- デフォルトやルールがバインドされたカラムは、削除できます。カラムを削除すると、カラム固有のデフォルトもすべて削除されます。検査制約または参照制約がバインドされたカラムは削除できません。こうするには、検査制約または参照制約を削除してから、カラムを削除します。sp\_helpconstraint を使用してテーブルの制約を特定し、sp\_depends を使用してカラムレベルの依存性を特定してください。
- システム・テーブルからはカラムを削除できません。また、Sybase が提供するツールやストアド・プロシージャによって作成および使用されるユーザ・テーブルからカラムを削除することもできません。
- 一般に、テーブルが空の場合は既存のカラムのデータ型を他の任意のデータ型に変更することが可能です。テーブルが空ではない場合は、そのデータ型を、元のデータ型に明示的に変換可能な任意のデータ型に変更できます。
- 次の処理ができます。
  - 新しい IDENTITY カラムを追加する。
  - 既存の IDENTITY カラムを削除する。
  - 既存の IDENTITY のサイズを変更する。

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。

- テーブルのスキームを変更すると、スキームのカウン트가増分され、このテーブルにアクセスする既存のストアド・プロシージャが次回実行されたときに、このストアド・プロシージャが再度正規化されます。データ型に依存するストアド・プロシージャまたはビューでの変更は、データ型正規化エラーによって失敗する可能性があります。テーブルの修正されたスキームを参照するように、これらの従属オブジェクトを更新してください。

#### テーブル・スキームを変更するときの制限事項

- トランザクションの内部からは、alter table を実行できません。
- テーブルのスキームを変更すると、bcp を使用して行ったバックアップを無効にすることができます。ただし、これらのバックアップは、テーブルの現在のスキームと互換性がなくなったテーブル・スキームを使用していることがあります。
- 検査制約のある NOT null カラムを追加することは可能ですが、Adaptive Server は既存のデータに対して制約の検証を行いません。



- テーブルにクラスタード・インデックスがあり、操作にデータのコピーが必要な場合は、`alter table ... add, drop`、または `modify` コマンドを使用してテーブルのロック・スキームを変更することはできません。これを行うには、次の手順に従います。
  - a クラスタード・インデックスを削除します。
  - b テーブルのスキームを変更します。
  - c クラスタード・インデックスを再作成します。
- アクティブなオープン・カーソルがテーブルにある場合は、テーブルのスキームは変更できません。

#### *text* カラムと *image* カラムを変更するときの制限事項

- `null` 値を受け付ける *text* カラムまたは *image* カラムだけを追加できます。

*text* カラムまたは *image* カラムを追加して `null` 値以外の値を持つようにするには、`null` 値のみを受け付けるカラムを追加してから、`null` 値以外の値に更新します。
- *text* データ型のカラムは、次のデータ型にのみ変更できます。
  - `[n]char`
  - `[n]varchar`
  - `unichar`
  - `univarchar`
  - `nchar`
  - `nvarchar`
- *image* データ型のカラムは `binary` データ型または `varbinary` データ型にのみ変更できます。
- 1 つの文の中で、新しい *text* カラムまたは *image* カラムを追加してから、既存の *text* カラムまたは *image* カラムを削除することはできません。
- カラムを *text* データ型または *image* データ型に変更することはできません。

#### *unitext* カラムを含むテーブルの変更

`alter table` を使用して *unitext* カラムを変更する際は、次の制限が適用されます。

- `null` 値を受け付ける新しい *unitext* カラムを追加できます。

- `unitext` カラムは、次のデータ型にのみ変更できます。
  - `[n]char`
  - `[n]varchar`
  - `unichar`
  - `univarchar`
  - `binary`
  - `varbinary`
- カラムを `unitext` データ型に変更することはできません。
- 1つの文の中で、`unitext` カラムを追加してから、既存の `unitext` カラムを削除することはできません。

#### ロック・スキームの変更

- `alter table` は、任意のロック・スキームから他の任意のロック・スキームへの変更をサポートしています。次の変更が実行できます。
  - `allpages` から `datapages` へ、またはその逆
  - `allpages` から `datarows` へ、またはその逆
  - `datapages` から `datarows` へ、またはその逆
- 全ページ・ロック・スキームからデータオンリー・ロック・スキームへ、またはその逆に変更する前に、`sp_dboption` を使用してデータベース・オプション `select into/bulkcopy/pllsort` を `true` に設定します。次に、データベースで `checkpoint` を実行して、分割されているテーブルがあるかどうか、およびインデックスのソートで並列ソートが必要かどうかを確認します。
- ロック・スキームを全ページ・ロックからデータオンリー・ロック、またはその逆に変更した後は、`dump transaction` コマンドを使用したトランザクション・ログのバックアップができません。事前にデータベースの完全なダンプを実行しておく必要があります。
- `alter table...lock` を使用して、テーブルのロック・スキームを全ページ・ロックからデータオンリー・ロック、またはその逆に変更すると、Adaptive Server によってテーブルのデータ・ページのコピーが作成されます。データ・ページを完全にコピーするには、テーブルが存在するセグメントに十分な空間がある必要があります。インデックスを再構築するには、インデックスが存在するセグメントに十分な領域がある必要があります。

データオンリー・ロック・テーブルのクラスタード・インデックスには、データ・ページの上にリーフ・レベルのページがあります。クラスタード・インデックスのあるテーブルを全ページ・ロックからデータオンリー・ロックに変更している場合、その結果のクラスタード・インデックスはさらに多くの領域を必要とします。必要な領域は、インデックス・キーのサイズによって異なります。

テーブルが現在占有している領域がどれくらいあるかを調べるには、`sp_spaceused` を使用し、テーブルを格納できる領域を確認するには、`sp_helpsegment` を使用します。

- テーブルのロック・スキームを全ページ・ロックからデータページ・ロックに、またはその逆に変更すると、記憶領域管理プロパティは、データ・ローがコピーされるときにテーブルに適用され、データ・ローが再作成されるときにインデックスに適用されます。データオンリー・ロック・スキームを別のスキームに変更するときは、データ・ページはコピーされず、記憶領域管理プロパティも適用されません。
- テーブルが分割されている場合、ロック・スキームを変更するとローが分割から分割にコピーされます。コピー中は、分割のデータの調整は行いません。
- テーブルのロック・スキームを変更するときは、コマンドの実行が完了するまで `alter table...lock` コマンドはテーブルの排他ロックを取得します。
- `alter table...lock` を使用してデータページ・ロックからデータロー・ロックに変更するときは、データ・ページのコピーやインデックスの再構築は実行されません。システム・テーブルのみが更新されます。
- システムで他のユーザがアクティブなときにロック・スキームを変更すると、ユーザのアクティビティに次のような影響を与える可能性があります。
  - テーブルにアクセスするプロシージャ・キャッシュにあるクエリ・プランが、次回実行されるときに再コンパイルされる。
  - テーブルを使用するアクティブな複数文のプロシージャが、次の手順に進む前に再コンパイルされる。

- テーブルを使用する特定のバッチのトランザクションが終了させられる。

---

**警告！** バルク・コピー操作がアクティブであるときにテーブルのロック・スキームを変更すると、テーブルが損傷することがあります。バルク・コピーは、最初にテーブルについての情報を取得することによって作動し、テーブル情報を読み込むときとローの送信を開始するときの間にはロックを保持しないので、**alter table...lock** コマンドが起動する可能性があります。

---

#### Java-SQL カラムの追加

- データベースで Java が有効になっている場合は、Java-SQL カラムをテーブルに追加できます。『Adaptive Server Enterprise における Java』を参照してください。
- 新しい Java-SQL カラムの宣言されたクラス (*datatype*) は、**Serializable** インタフェースまたは **Externalizable** インタフェースのいずれかを実装する必要があります。
- Java-SQL カラムをテーブルに追加するときは、Java-SQL カラムを次のように指定することはできません。
  - 外部キーとして指定する
  - 参照句の中で指定する
  - **UNIQUE** プロパティを持つカラムとして指定する
  - プライマリ・キーとして指定する
- **in row** が指定されている場合、格納される値はデータ・サーバのページ・サイズに従って 16 キロバイト以内にします。
- **off row** が指定されている場合は、カラムに対して以下を行うことはできません。
  - 検査制約での参照
  - **distinct** を指定する **select** での参照
  - 比較演算子、述部、または **group by** 句の中での指定

## 共有ディスク・クラスタの制限事項

- 参照整合性の制約は、同じローカル・テンポラリ・データベースのテーブル以外のローカル・テンポラリ・データベースのカラムを参照できません。`alter table` は、別のデータベースのテーブルからローカル・テンポラリ・データベースのカラムへの参照を作成しようとする、失敗します。
- ローカル・テンポラリ・データベースに格納されている暗号化キーを持つカラムは、そのカラムのテーブルが同じローカル・テンポラリ・データベースにないと、暗号化できません。ローカル・テンポラリ・データベースにある暗号化キーを持つカラムを暗号化する場合、そのテーブルが別のデータベースにあると、`alter table` は失敗します。

## 標準規格

## ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

詳細については、『リファレンス・マニュアル：プロシージャ』の「[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。

## パーミッション

`alter table` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であるか <code>alter any table</code> 権限を持つユーザであることが必要。 <code>setuser</code> 権限が付与されたユーザは、 <code>setuser</code> コマンドを実行することによって、テーブルの所有者になり代わることができる。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者であるか <code>sa_role</code> が付与されたユーザであることが必要。データベース所有者は、 <code>setuser</code> コマンドを実行することでテーブル所有者になり代わることができる。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
3	alter	alter table	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — add column、drop column、modify column、replace column、add constraint、または drop constraint</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — set proxy が有効な場合は元のログイン名</li> <li>• set transfer table [on   off] の set オプションの設定に応じて次のようになります。 <ul style="list-style-type: none"> <li>• on — Adaptive Server は SET TRANSFER TABLE ON を監査レコードの extrainfo に出力。</li> <li>• off — Adaptive Server は SET TRANSFER TABLE OFF を出力。</li> </ul> </li> </ul>

## 参照

コマンド [create index](#), [create table](#), [dbcc](#), [drop database](#), [dump transaction](#), [insert](#), [setuser](#)

システム・プロシージャ [sp\\_chgattribute](#), [sp\\_help](#), [sp\\_helppartition](#), [sp\\_rename](#)

## alter thread pool

説明	スレッド・プールを変更します。
プロセス・モードの考慮事項	alter thread pool はプロセス・モードではサポートされていません。
構文	<pre>alter thread pool <i>pool_name</i> with { pool name = "new_name"                                 thread count = <i>thread_count</i>,                                 [pool description = "description" ]}                                 [idle timeout = <i>time_period</i>]</pre>
パラメータ	<p><i>pool_name</i> 変更するスレッド・プールの名前。</p> <p>pool name = "new_name" 変更するプールの新しい名前。</p> <p>thread count = <i>thread_count</i> 数変更後のスレッド・プール内のスレッド数。1 以上にしてください。</p> <p>pool description = "description" (任意) プールの目的についての説明。256 文字未満にしてください。</p> <p>idle timeout = <i>time_period</i> スレッドがスリープ状態になる前に作業を探すマイクロ秒単位の時間。-1 の値は、スレッドがスリープ状態にならず、実行する作業がない場合も CPU を消費し続けることを意味します。0 の値は、作業が見つからなかった場合にスレッドがただちにスリープ状態になることを意味します。</p>
例	<p><b>例 1</b> order_pool スレッド・プールの名前を sales_pool に変更します。</p> <pre>alter thread pool order_pool with pool name = 'sales_pool'</pre> <p><b>例 2</b> 7 つのスレッドを含むように sales_pool スレッド・プールを変更します。</p> <pre>alter thread pool sales_pool with thread count = 7</pre> <p><b>例 3</b> sales_pool の名前と説明を変更します。</p> <pre>alter thread pool sales_pool with pool name = "larger_sales_pool", pool description = 'thread pool exclusive to sales group'</pre> <p><b>例 4</b> 500 ミリ秒後に作業が見つからなかった場合にスレッドがスリープ状態になるように sales_pool を変更します。</p> <pre>alter thread pool order_pool with idle timeout = 500</pre>

**例 5** 作業が見つからなくてもスレッドがスリープ状態になることがないように `sales_pool` を変更します。

```
alter thread pool order_pool with idle timeout = -1
```

#### 使用法

- `thread_count` は 1 以上に設定します。
- スレッド数を減らす場合、指定したスレッド・プールは、現在実行中のタスクが停止してから、スレッド数を減らす必要があります。これにより、Adaptive Server によりプールの縮小に少々時間がかかる場合があります。
- システムで作成されたスレッド・プール (`syb_` で始まる) の名前を変更することはできません。ただし、`alter thread pool` を使用して、システムで作成されたスレッド・プール内のスレッド数やアイドル・タイムアウトを変更することはできます。
- `alter thread pool` では Transact-SQL 変数をパラメータとして使用することはできません。
- `idle timeout` はエンジン・スレッド・プールに対してのみ設定できます。
- `alter thread pool` は `execute immediate` を指定して発行できます。

#### 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

#### パーミッション

`alter thread pool` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

細密なパーミッションが有効の場合、`manage any thread pool` 権限が必要。

細密なパーミッションが無効

細密なパーミッションが無効の場合、`sa_role` が付与されていることが必要。  
`alter thread pool` パーミッションは、`grant all` コマンドには含まれていません。

#### 監査

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
143			古い名前、新しい名前、スレッド数



## begin...end

説明	<a href="#">if...else</a> などのフロー制御言語がグループ全体のパフォーマンスに作用するように、一連の SQL 文を囲みます。
構文	begin <i>statement block</i> end
パラメータ	<i>statement block</i> begin と end で囲まれた一連の文です。
例	<p><b>例 1</b> begin と end がないと、if 条件では 1 つの SQL 文しか実行されません。</p> <pre> if (select avg (price) from titles) &lt; \$15 begin     update titles     set price = price * \$2     select title, price     from titles     where price &gt; \$28 end </pre> <p><b>例 2</b> begin と end がないと、print 文は実行されません。</p> <pre> create trigger deltitle on titles for delete as if (select count (*) from deleted, salesdetail     where salesdetail.title_id = deleted.title_id) &gt; 0 begin     rollback transaction     print "You can' t delete a title with sales." end else     print "Deletion successful--no sales for this     title." </pre>
使用法	<ul style="list-style-type: none"> <li>begin...end ブロックは、他の begin...end ブロック内にネストできます。</li> </ul>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能
パーミッション	begin...end の使用にパーミッションは必要ありません。
参照	コマンド <a href="#">if...else</a>

## begin transaction

説明	ユーザ定義のトランザクションの開始点をマーク付けします。
構文	<code>begin tran[saction] [<i>transaction_name</i>]</code>
パラメータ	<i>transaction_name</i> トランザクションに割り当てられる名前です。トランザクションの名前は、識別子の規則に従っている必要があります。トランザクション名は、ネストされた <code>begin transaction/commit</code> 文または <code>begin transaction/rollback</code> 文の最も外側の組だけに使用してください。
例	明示的に <code>insert</code> 文のトランザクションを開始します。 <pre>begin transaction   insert into publishers (pub_id) values ("9999") commit transaction</pre>
使用法	<ul style="list-style-type: none"><li>• <code>begin transaction</code> 句 および <code>commit</code> 句で SQL 文とシステム・プロシージャを囲んで、トランザクションを定義します。連鎖トランザクション・モードを設定すると、Adaptive Server では <code>begin transaction</code> が暗黙的に呼び出されてから、<code>delete</code>、<code>insert</code>、<code>open</code>、<code>fetch</code>、<code>select</code>、<code>update</code> が実行されます。ただし、<code>commit</code> を使用してトランザクションを明示的にクローズする必要があります。</li><li>• トランザクション全体またはトランザクションの一部を取り消すには、<code>rollback</code> コマンドを使用します。コミットされた後ではトランザクションをロールバックできないため、<code>rollback</code> コマンドがトランザクション内になければなりません。</li></ul>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>begin transaction</code> の使用にパーミッションは必要ありません。
参照	コマンド <code>commit</code> 、 <code>rollback</code> 、 <code>save transaction</code>

## break

説明	<code>while</code> ループを終了させます。 <code>break</code> は、多くの場合 <code>if</code> 文によってアクティブ化されます。
構文	<pre>while logical_expression     statement break     statement continue</pre>
パラメータ	<p><i>logical_expression</i></p> <p>TRUE、FALSE、または <code>null</code> を返す式 (カラム名、定数、算術演算子またはビット処理演算子で連結されたカラム名と定数の任意の組み合わせ、またはサブクエリ) です。論理式に <code>select</code> 文がある場合は、<code>select</code> 文をカッコで囲みます。</p>
例	<p>平均価格が 30 ドル未満の場合は、価格を 2 倍にします。その後、最高価格を選択します。それが 50 ドル以下の場合は、<code>while</code> ループが再開され、価格をさらに 2 倍にします。最高価格が 50 ドルを超えると、<code>while</code> ループが終了してメッセージが表示されます。</p> <pre>while (select avg (price) from titles) &lt; \$30 begin     update titles     set price = price * 2     select max (price) from titles     if (select max (price) from titles) &gt; \$50         break     else         continue end begin     print "Too much for the market to bear" end</pre>
使用法	<ul style="list-style-type: none"> <li><code>break</code> は <code>while</code> ループを終了させます。次に、ループの終わりを示すキーワード <code>end</code> の後ろにある文が実行されます。</li> <li>2 つ以上の <code>while</code> ループがネストされている場合は、ネストの内側の <code>break</code> によって、1 つ外側にあるループに移行します。最初に、内側のループの終わりの後にある文がすべて実行されてから、1 つ外側にあるループが再開されます。</li> </ul>
標準規格	ANSI SQL — 準拠レベル: Transact-SQL 拡張機能
パーミッション	<code>break</code> の使用にパーミッションは必要ありません。
参照	コマンド <code>continue</code> 、 <code>while</code>

## checkpoint

**説明** ダーティ・ページ (最後に書き込みを行ってから更新されているページ) をすべて、データベース・デバイスに書き込みます。

**構文** `checkpoint [all | [dbname[, dbname, dbname, .....]]`

**例** このコマンドによって現在のデータベース内のダーティ・ページを、システムのチェックポイント・スケジュールに関係なく、すべてデータベース・デバイスに書き込みます。

`checkpoint`

**使用法**

- `checkpoint` は、アーカイブ・データベースで使用できます。ただし、チェックポイント処理でアーカイブ・データベースが自動的にチェックポイントを実行するわけではありません。
- `checkpoint` は、特別な状況での予防対策としてのみ使用してください。
- `sp_dboption` は、データベース・オプションを変更するときに、自動的に `checkpoint` をデフォルトで実行します。
- `checkpoint` を実行するデータベースを 1 つ以上指定できます。

**自動チェックポイント**

- `checkpoint` コマンドによって作成されるチェックポイントは、自動チェックポイントを補足するものです。自動チェックポイントは、最大許容リカバリ時間に設定可能な値に基づいて、Adaptive Server で計算された間隔で発生します。
- `checkpoint` は、完了したトランザクションがすべてデータベース・デバイスに確実に書き込まれたポイントを識別して、リカバリ時間を節約します。一般に `checkpoint` には約 1 秒かかりますが、Adaptive Server のアクティビティの量によってチェックポイント時間は異なります。
- Adaptive Server では、自動 `checkpoint` の間隔は、システムのアクティビティとシステム・テーブル `syscurconfigs` のリカバリ間隔の値に基づいて計算されます。システムがリカバリの実行に要する最大時間を指定すると、そのリカバリ間隔によって `checkpoint` の頻度が決まります。この値をリセットするには、`sp_configure` を実行します。
- Adaptive Server に複数のチェックポイント・プロセスを設定できます。これにより、複数のエンジンを使用してチェックポイント・タスクを頻繁に実行できるため、自動リカバリ・プロセスの実行にかかる時間が短縮されます。

- サーバのアイドル時間に、ハウスキーピング・タスクが、設定されたすべてのキャッシュ内のアクティブなバッファ・プールをすべてフラッシュできる場合は、チェックポイント・タスクが実行されます。チェックポイント・タスクは、データベースにチェックポイントを設定できるかどうかを決定します。

ハウスキーピング・タスクの結果として設定されるチェックポイントは、「**フリー・チェックポイント**」と呼ばれます。これには、データベース・デバイスへの多くのダーティ・ページの書き込みは含まれません。その作業は、ハウスキーピング・タスクによってすでに行われているためです。フリー・チェックポイントによって、データベースのリカバリ速度が向上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

## パーミッション

**checkpoint** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

細密なパーミッションが有効化されている場合は、次のようになる。

- 特定のデータベースに対して **checkpoint** を実行するには、データベース所有者であるか、(データベースに対する) **checkpoint** または **own database** 権限が必要。
- checkpoint all** を実行するには、該当するすべてのデータベースのデータベース所有者であるか、**checkpoint any database** 権限または **own and database** 権限が必要。これがない場合、**checkpoint all** は、操作者が **checkpoint** を実行するパーミッションを持つデータベースのみを対象として実行される。

細密なパーミッションが無効

細密なパーミッションが無効化されている場合は、次のようになる。

- checkpoint database** の実行には、データベース所有者であるか、次のいずれかが付与されたユーザであることが必要。
  - sa\_role**
  - replication\_role**
  - oper\_role**
- checkpoint all** の実行には、該当するすべてのデータベースの所有者であるか、次のいずれかが付与されたユーザであることが必要。
  - sa\_role**
  - replication\_role**

これらがない場合、**checkpoint all** は、所有するデータベースのみを対象として実行される。

## 参照

システム・プロシージャ [sp\\_configure](#), [sp\\_dboption](#)

## close

説明	カーソルを非アクティブにします。
構文	<code>close cursor_name</code>
パラメータ	<i>cursor_name</i> クローズするカーソル名です。
例	authors_crsr というカーソルをクローズします。 <pre>close authors_crsr</pre>
使用法	<ul style="list-style-type: none"><li>• <code>close</code> コマンドは、カーソルの結果セットを実質的に削除します。結果セット内でのカーソル位置は、クローズされたカーソルに対して不定になります。</li><li>• カーソルがすでにクローズされているか、存在しない場合は、エラー・メッセージが返されます。</li></ul>
標準規格	ANSI SQL – 準拠レベル：初級レベル
パーミッション	<code>close</code> の使用にパーミッションは必要ありません。
参照	コマンド <a href="#">deallocate cursor</a> , <a href="#">declare cursor</a> , <a href="#">fetch</a> , <a href="#">open</a>

## commit

説明	ユーザ定義のトランザクションの終了点をマーク付けします。
構文	<code>commit [tran   transaction   work] [<i>transaction_name</i>]</code>
パラメータ	<p><code>tran   transaction   work</code> トランザクションまたは処理をコミットすることを指定します。 <code>tran</code>、<code>transaction</code>、または <code>work</code> を指定する場合は、<i>transaction_name</i> も指定できます。</p> <p><i>transaction_name</i> トランザクションに割り当てられる名前です。トランザクション名は識別子の規則に従います。トランザクション名は、ネストされた <code>begin transaction/commit</code> 文または <code>begin transaction/rollback</code> 文の最も外側の組だけに使用してください。</p>
例	<p>2 人の著者の <code>royaltyper</code> エントリを更新した後で、セーブポイント <code>percentchanged</code> を挿入し、本の価格が 10% 増加すると、著者の印税収入にどの程度の影響が出るかを確認します。トランザクションは、<code>rollback transaction</code> コマンドによってセーブポイントまでロールバックされます。</p>

```
begin transaction royalty_change

update titleauthor
  set royaltyper = 65 from titleauthor, titles
  where royaltyper = 75
  and titleauthor.title_id = titles.title_id
  and title = "The Gourmet Microwave"

update titleauthor
  set royaltyper = 35 from titleauthor, titles
  where royaltyper = 25
  and titleauthor.title_id = titles.title_id
  and title = "The Gourmet Microwave"

save transaction percentchanged

update titles
  set price = price * 1.1
  where title = "The Gourmet Microwave"

select (price * total_sales) * royaltyper
from titles, titleauthor
where title = "The Gourmet Microwave"
and titles.title_id = titleauthor.title_id
```

```
rollback transaction percentchanged
```

```
commit transaction
```

**使用法**

- **begin transaction** 句と **commit** 句で SQL 文およびシステム・プロシージャを囲んで、トランザクションを定義します。連鎖トランザクション・モードを設定すると、Adaptive Server では **begin transaction** が暗黙的に呼び出されてから、**delete**、**insert**、**open**、**fetch**、**select**、**update** が実行されます。ただし、**commit** で、トランザクションを明示的にクローズする必要があります。
- トランザクション全体または一部を取り消すには、**rollback** コマンドを使用してください。**rollback** コマンドは、トランザクション内部になければなりません。**commit** を入力した後にはトランザクションをロールバックできません。
- 現在アクティブなトランザクションがない場合、**commit** 文または **rollback** 文を実行しても Adaptive Server に影響はありません。

**標準規格**

ANSI SQL – 準拠レベル：初級レベル

**commit transaction** および **commit tran** 形式の文は、Transact-SQL 拡張機能です。

**パーミッション**

**commit** の使用にパーミッションは必要ありません。

**参照**

コマンド **begin transaction**、**rollback**、**save transaction**



## compute 句

**説明** クエリ結果の追加ローとして表示される合計値を生成します。

**構文**

```
start_of_select_statement
  compute row_aggregate (column_name)
    [, row_aggregate (column_name)]...
    [by column_name [, column_name]...]
```

**パラメータ** *row\_aggregate*

次のいずれかを指定します。

- **sum** – (数値) カラムにある値の合計
- **avg** – (数値) カラムにある値の平均
- **min** – カラムの最小値
- **max** – カラムの最大値
- **count** – カラム内の **integer** の値の数
- **count** – カラム内の **bigint** の値の数

*column\_name*

カラムの名前。カッコで囲む必要があります。sum および avg では、数値カラムだけを使用できます。integer、numeric、decimal のカラムだけが sum と avg で使用できます。

**by**

サブグループの、ローの集約値を計算します。by 項目の値が変更されると、常にローの集約値が生成されます。by を使用する場合は、order by も使用する必要があります。

by の後に複数の項目をリストすると、グループがサブグループに分割され、グループ化される各レベルに関数が適用されます。

**例**

**例 1** 価格が 12 ドルを超える料理雑誌の、タイプごとの価格の合計を計算します。

```
select type, price
from titles
where price > $12
      and type like "%cook"
      order by type, price
compute sum (price) by type

type      price
-----
mod_cook      19.99
sum
```

```

-----
                19.99
type           price
-----
trad_cook      14.99
trad_cook      20.95
sum
-----
                35.94
(5 rows affected)

```

**例 2** 価格が 12 ドルを超える料理雑誌の、タイプごとの価格と前渡し金の合計を計算します。1つの `compute` 句で、同じカラムのグループに複数の集合関数を使用できます。

```

select type, price, advance
from titles
where price > $12
      and type like "%cook"
      order by type, price
compute sum (price), sum (advance) by type

type           price           advance
-----
mod_cook        19.99                0.00
sum
-----
                19.99                0.00

type           price           advance
-----
trad_cook        14.99            8,000.00
trad_cook        20.95            7,000.00
sum
-----
                35.94            15,000.00
(5 rows affected)

```

**例 3** 価格が 12 ドルを超える料理雑誌の、タイプごとの価格と最大の前渡し金の合計を計算します。1つの `compute` 句で、同じカラムのグループに複数の集合関数を使用できます。

```

select type, price, advance
from titles
where price > $12
      and type like "%cook"
      order by type, price
compute sum (price), max (advance) by type

```

```

type      price      advance
-----
mod_cook  19.99      0.00
sum
-----
          19.99
max
-----
          0.00

type      price      advance
-----
trad_cook 14.99      8,000.00
trad_cook 20.95      7,000.00
sum
-----
          35.94
max
-----
          8,000.00

```

(5 rows affected)

**例 4** type と pub\_id で分類し、タイプと出版社 ID の組み合わせごとに心理学の本の価格の合計を計算します。

```

select type, pub_id, price
from titles
where price > $10
      and type = "psychology"
      order by type, pub_id, price
compute sum (price) by type, pub_id

type      pub_id      price
-----
psychology 0736      10.95
psychology 0736      19.99
sum
-----
          30.94

type      pub_id      price
-----
psychology 0877      21.59
sum
-----
          21.59

```

(5 rows affected)

**例 5** 価格が 10 ドルを超える心理学の本の価格の合計を計算するほか、複数の `compute` を使用して複数のグループを作成し、`type` と `pub_id` ごとの合計を計算します。

```
select type, pub_id, price
from titles
where price > $10
      and type = "psychology"
order by type, pub_id, price
compute sum (price) by type, pub_id
compute sum (price) by type
```

type	pub_id	price
psychology	0736	10.95
psychology	0736	19.99
		sum
		-----
		30.94

type	pub_id	price
psychology	0877	21.59
		sum
		-----
		21.59
		sum
		-----
		52.53

(6 rows affected)

**例 6** 価格が 10 ドルを超える料理雑誌の価格の合計と前渡し金の合計を計算します。

```
select type, price, advance
from titles
where price > $10
      and type like "%cook"
compute sum (price), sum (advance)
```

type	price	advance
mod_cook	19.99	0.00
trad_cook	20.95	8,000.00
trad_cook	11.95	4,000.00
trad_cook	14.99	7,000.00
	sum	sum
	-----	-----
	67.88	19,000.00

(5 rows affected)

**例 7** 料理雑誌の価格の合計と、式で使用された価格の合計を計算します。

```
select type, price, price*2
from titles
  where type like "%cook"
compute sum (price), sum (price*2)
```

type	price	
mod_cook	19.99	39.98
mod_cook	2.99	5.98
trad_cook	20.95	41.90
trad_cook	11.95	23.90
trad_cook	14.99	29.98
	sum	sum
	=====	=====
	70.87	141.74

#### 使用法

- `compute` 句では、1つの結果セット内でディテール・ローと合計ローを確認できます。サブグループの合計値、および同じグループの複数の集約値を計算できます。
- 合計、合計カウントなどを生成するために、`compute` に `by` を付けないで使用できます。`compute` に `by` を付けない場合、`order by` 句はオプションになります。例 6 を参照してください。
- `compute by` 句を使用する場合は、`order by` 句も使用する必要があります。`compute by` 句の後にリストする各カラムは、`order by` 句の後にリストする各カラムと一致するか、それらのサブセットである必要があります。また、左から右に同じ順序で同じ式から始まっている必要があります。式を省略することはできません。たとえば、`order by` 句が `order by a, b, c` である場合、`compute by` 句には、次のいずれか (またはすべて) を指定できます。

```
compute by a, b, c
compute by a, b
compute by a
```

#### 制限事項

- `compute` 句内では 127 を超える集合カラムを使用できません。
- カーソル宣言では、`compute` 句を使用できません。
- 合計値は式とカラムの両方で計算できます。`compute` 句で指定する式やカラムは、`select` リストにもなければなりません。

- カラム名のエイリアスは、`compute` 句のローの集合関数に対する引数としては使用できませんが、`select` リスト、`order by` 句、および `compute` の `by` 句の中では使用できます。
- `compute` 句を含む `select` 文では、`select` リストでのカラムの順序が `compute` 句の集合の順序より優先されます。OpenClient™、JDBC、DB-Library™ のプログラムを作成するときは、集合の結果が正しい場所に配置されるように、この優先度に注意する必要があります。
- `compute` 句がある文は通常のテーブルを生成しないため、`compute` 句と同じ文中では `select into` を使用できません。
- `compute` 句に `group by` 句がある場合は、以下の制限があります。
  - `compute` 句では、255 を超える集合関数を使用できない。
  - `group by` 句は、255 を超えるカラムを含めることはできない。
- `compute` 句に含まれるカラムの長さは、255 バイトを超えることはできません。

#### 新しいローとしての `compute` 結果の表示

- 集合関数は通常、テーブル内の指定されたすべてのローに対して、または各グループに対して 1 つの値を生成し、この合計値は新しいカラムとして表示されます。次に例を示します。

```
select type, sum (price), sum (advance)
from titles
where type like "%cook"
group by type
type
-----
```

mod_cook	22.98	15,000.00
trad_cook	47.89	19,000.00

(2 rows affected)

- `compute` 句では、1 つのコマンドでディテール・ローと合計ローを取得できます。次に例を示します。

```
select type, price, advance
from titles
where type like "%cook"
order by type
compute sum (price), sum (advance) by type
type      price      advance
-----
```

mod_cook	2.99	15,000.00
mod_cook	19.99	0.00

```

Compute Result:
-----
                22.98          15,000.00
type           price          advance
-----
trad_cook      11.95           4,000.00
trad_cook      14.99           8,000.00
trad_cook      20.95           7,000.00

```

```

Compute Result:
-----
                47.89          19,000.00
(7 rows affected)

```

- さまざまなタイプの `compute` 句の出力とグループ化は次のとおりです。

句とグループ化	出力	例
1つの <code>compute</code> 句、同じ関数	1つのディテール・ロー	1, 2, 4, 6, 7
1つの <code>compute</code> 句、異なる関数	関数のタイプごとに1つのディテール・ロー	3
複数の <code>compute</code> 句、同じグループのカラム	<code>compute</code> 句ごとに1つのディテール・ロー (ディテール・ローが一緒に出力される)	異なる関数を持った1つの <code>compute</code> 句を持っているのと同じ結果
複数の <code>compute</code> 句、異なるグループのカラム	<code>compute</code> 句ごとに1つのディテール・ロー (グループによって、異なる場所にディテール・ローが出力される)	5

### 大文字と小文字の区別

サーバに大文字と小文字を区別しないソート順が設定されている場合、指定したカラムのデータの大文字と小文字を `compute` 句は区別しません。次はその例です。

```

select * from groupdemo
lname      amount
-----
Smith      10.00
smith      5.00
SMITH      7.00
Levi       9.00
Levi       20.00

```

`lname` に対して `compute by` 句を使用すると、次のような結果が生成されます。

```
select lname, amount from groupdemo
order by lname
compute sum (amount) by lname
lname      amount
-----
```

lname	amount
Levi	9.00

```
Compute Result:
-----
                9.00
```

```
lname      amount
-----
```

lname	amount
Levi	20.00

```
Compute Result:
-----
                20.00
```

```
lname      amount
-----
```

lname	amount
smith	5.00
SMITH	7.00
Smith	10.00

```
Compute Result:
-----
                22.00
```

大文字／小文字およびアクセントの有無を区別しないサーバで上記と同じクエリを実行すると、次のような結果が生成されます。

```
lname      amount
-----
```

lname	amount
Levi	9.00
Levi	20.00

```
Compute Result:
-----
                29.00
```



```
      lname      amount
-----
smith          5.00
SMITH         7.00
Smith         10.00
```

```
Compute Result:
-----
                22.00
```

標準規格

ANSI SQL – 準拠レベル: Transact-SQL 拡張機能

参照

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

関数 [avg](#), [count](#), [max](#), [min](#), [sum](#)

## connect to...disconnect

**説明** (コンポーネント統合サービスのみ) 指定されたサーバに接続します。また、接続されているサーバとの接続を解除します。

**構文** この構文は Adaptive Server に一字一句そのまま送られます。この構文は、コンポーネント統合サービスを使用して別のサーバへのパススルーを作成するときに使用します。

```
connect to server_name
disconnect
    [from ASE]
    [all]
    [connection_name]
```

この構文は、コンポーネント統合サービスを使用せずに Adaptive Server に対して新しい JDBC レベルの接続を開きます。引数は、どのような順序で指定してもかまいません。引数を指定しない場合、接続パラメータの入力を求めるプロンプトが表示されます。

```
connect
    [to ASE engine_name]
    [database database_name]
    [as connection_name]
    [user user_id]
    [identified by password]]]
```

この構文は、Adaptive Server に対して新しい JDBC レベルの接続を開きます。コンポーネント統合サービスは使用しません。

```
connect using connect_string
```

### パラメータ

*server\_name*  
パススルー接続が必要なサーバです。

from ASE  
現在の Adaptive Server との接続を切断します。

all  
すべての Adaptive Server との接続を切断します。

*connection\_name*  
指定された接続を切断します。

*engine\_name*  
指定されたエンジンに接続します。

*database\_name*  
指定されたデータベースに接続します。

**connection\_name**

設定された接続に接続します。

**user\_id**

この ID を使用してユーザに接続します。

**connection\_string**

事前に決められた接続文字列を使用して接続します。

## 例

**例 1** SYBASE と名付けられたサーバに対して、パススルー接続を確立します。

```
connect to SYBASE
```

**例 2** 接続されているサーバを切断します。

```
disconnect
```

**例 3** すべてのサーバとの接続を切断します。

```
disconnect all
```

## 使用法

- **connect to** は、パススルー接続が必要なサーバを指定します。パススルー・モードによって、リモート・サーバ上でネイティブな操作が実行できます。
- **server\_name** は、**sys.servers** テーブル上のサーバ名であり、サーバ・クラスとネットワーク名が定義されている必要があります。
- ユーザのために **server\_name** 接続を確立する場合、コンポーネント統合サービスは次の識別子のいずれかを使用します。
  - **sysattributes** で記述されたリモート・ログイン・エイリアス (該当する場合)
  - ユーザ名とパスワード

いずれの場合も、指定したサーバに対して接続が不可能なときは、**Adaptive Server** によってエラー・メッセージが返されます。

- パススルー接続を行った後に、次の言語テキストが受信されると、コンポーネント統合サービスは **Transact-SQL** パーサとコンパイラをバイパスします。コンポーネント統合サービスは、文を指定のサーバに直接渡します。また、その結果を、**Open Client** インタフェースで認識でき、クライアント・プログラムに返せるフォームに変換します。
- **connect to** コマンドによって確立された接続を切断するには、**disconnect** コマンドを使用します。このコマンドが使えるのは、**connect to** を使用して接続を確立した場合のみです。

- `disconnect` コマンドは、`disc` と略して使用できます。
- `connect` が発行されておらず、サーバがリモート・サーバと接続されていない場合、`disconnect` コマンドはエラーを返します。

## 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

## パーミッション

`connect` コマンドを使用するには、`connect` 権限が必要です。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
90	security	connect to	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – connect to</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [create existing table](#), [grant](#)システム・プロシージャ [sp\\_addserver](#), [sp\\_autoconnect](#), [sp\\_helpserver](#), [sp\\_passthru](#), [sp\\_remotesql](#), [sp\\_serveroption](#)

## continue

説明	<code>while</code> ループを再開します。 <code>continue</code> は、たいていの場合、 <code>if</code> 文によってアクティブ化されます。
構文	<pre>while <i>boolean_expression</i>     <i>statement</i>     break     <i>statement</i> continue</pre>
例	<p>平均価格が 30 ドル未満の場合は、価格を 2 倍にします。次に、最高価格を選択します。最高価格が 50 ドル以下の場合は、<code>while</code> ループが再開され、価格をさらに 2 倍にします。最高価格が 50 ドルを超えると、<code>while</code> ループが終了してメッセージが表示されます。</p> <pre>while (select avg (price) from titles) &lt; \$30 begin     update titles     set price = price * 2     select max (price) from titles      if (select max (price) from titles) &gt; \$50         break     else         continue end  begin print "Too much for the market to bear" end</pre>
使用法	<code>continue</code> は <code>while</code> ループを再開し、 <code>continue</code> の後の文をすべて省略します。
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>continue</code> の使用にパーミッションは必要ありません。
参照	コマンド <code>break</code> , <code>while</code>

## create archive database

説明	アーカイブ・データベースを作成します。
構文	<pre>create archive database <i>db_name</i>     [on <i>db_device</i> [= <i>size</i>]     [, <i>db_device</i> [= <i>size</i>] ] ...]     with scratch_database = <i>db_name</i></pre>
パラメータ	<p><b>on</b> 変更済みページ・セクションを指定します。変更済みページを格納する従来のデータベース記憶領域が必要です。変更済みページ・セクションのロケーションとサイズを指定するには、<b>on</b> 句を使用します。</p> <p><b>db device</b> 変更済みページ・セクションを作成するデータベース・デバイスを指定します。</p> <p><b>size</b> 作成する変更済みページ・セクションのサイズを指定します。<b>size</b> を省略すると、5120 ページが割り付けられます。</p> <p><b>with scratch_database</b> (スクラッチ・データベースがまだ存在しない場合は必須) アーカイブ・データベースに関する情報が保持される既存のデータベースの名前を指定します。アーカイブ・データベース内の論理ページを物理ページにマップする <b>sysaltusages</b> システム・テーブルは、スクラッチ・データベースに格納される。</p>
例	<p>次に、一般的なアーカイブ・データベース・コマンド・シーケンスの例を示します。</p> <ol style="list-style-type: none"> <li>必要に応じてスクラッチ・データベースを作成します。 <pre>create database scratchdb     on datadev1 = 100     log on logdev1 = 50</pre> <p>これにより、<b>scratchdb</b> という名前の 150MB の従来のデータベースが作成されます。</p> </li> <li>上の手順で作成したデータベースをスクラッチ・データベースとして指定します。 <pre>sp_dboption "scratchdb", "scratch database", "true"</pre> </li> <li>アーカイブ・データベースを作成します。 <pre>create archive database archivedb</pre> </li> </ol>

```
on datadev2 = 20
with scratch_database = scratchdb
```

これにより、20MB の変更済みページ・セクションが格納される、**archivedb** という名前のアーカイブ・データベースが作成されます。

- 4 アーカイブ・データベースをマテリアライズします。

```
load database archivedb
from "/dev/dumps/050615/proddb_01.dmp"
stripe on "/dev/dumps/050615/proddb_02.dmp"
```

- 5 データベースをオンラインにします。

```
online database archivedb
```

- 6 **dbcc** コマンドを使用して、アーカイブ・データベースの一貫性を検査します。次に例を示します。

```
dbcc checkdb(archivedb)
```

- 7 トランザクション・ログ・ダンプをロードし、アーカイブ・データベースからオブジェクトをリストアします。

```
load tran archivedb
from "/dev/dumps/050615/proddb1_log_01.dmp"
load tran archivedb
from "/dev/dumps/050615/proddb1_log_02.dmp"
online database archivedb
select * into proddb.dbo.orders from
archivedb.dbo.orders
load tran archivedb
from "/dev/dumps/050615/proddb1_log_03.dmp"
online database archivedb
```

#### 使用法

- **master** データベースのダンプをアーカイブ・データベースにロードできます。
- インメモリ・データベースはアーカイブ・データベースとして使用できません。さらに、インメモリ・データベースはスラッチ・データベースとしても使用しないことをおすすめします。

#### パーミッション

**create archive database** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <b>create database</b> 権限が必要。
---------------	---

細密なパーミッションが無効	細密なパーミッションが無効の場合、システム管理者であるか、 <b>create database</b> 権限が必要。
---------------	---

## create database

説明 新しいデータベースを作成します。

構文 非クラスタ環境の構文：

```
create [inmemory] [temporary] database database_name
  [use database_name as template]
  [on {default | database_device} [= size]
  [, database_device [= size]]...]
  [log on database_device [= size]
  [, database_device [= size]]...]
  [with {dbid = number, default_location = "pathname", override}]
  | [.,]durability = { no_recovery
  | at_shutdown
  | full} ]
  [, [no] async_init]
  [ [.,] compression = {none | row | page}]
  [ [.,] lob_compression = {compression_level | off}]
  [ [.,] inrow_lob_length = value ] }...
  [for {load | proxy_update}]
```

クラスタ環境の構文：

```
create [ [ global | system ] temporary ] database database_name
  [ for instance instance_name ]
  [on {default | database_device} [= size]
  [, database_device [= size]]...]
  [log on database_device [= size]
  [, database_device [= size]]...]
  [with {override | default_location = "pathname"}]
  [for {load | proxy_update}]
```

パラメータ

**temporary**

テンポラリ・データベースを作成することを示します。

**inmemory**

インメモリ・データベースに必要です。

***database\_name***

新しいデータベースの名前です。名前は識別子の規則に従っている必要があります。変数を使うことはできません。

**on**

データベースのロケーションとサイズを示します。

**default**

`create database` を使用して、新しいデータベースを `sysdevices.status` に示される任意のデフォルト・データベース・デバイスに配置できることを示します。明確にロケーションを指定せずにデータベースのサイズだけを指定するには、次のコマンドを使用してください。



```
on default = size
```

データベース・デバイスのステータスをデフォルトに変更するには、`sp_diskdefault` を使用します。

#### `database_device`

データベースを配置するデバイスの論理名です。データベースは、それぞれ異なる量の領域で複数のデータベース・デバイス上に分けることができます。データベース・デバイスを `Adaptive Server` に追加するには、`disk init` を使用します。

#### `size`

データベース拡張に割り付ける領域の量です。指定できる単位は、“k”または“K”(キロバイト)、“m”または“M”(メガバイト)、“g”または“G”(ギガバイト)、および“t”または“T”(テラバイト)です。これらには大文字と小文字の区別はなく、一重引用符と二重引用符のどちらも使用できます。常に単位を指定することをおすすめします。単位を含めない場合は、引用符を省略することができます。ただし、単位を付加する場合は、必ず引用符を付けてください。`size` の単位を指定しなかった場合、値はメガバイト単位であると見なされます。

#### `log on`

データベース・ログに対するデバイスの論理名を指定します。`log on` 句には、複数のデバイスを指定できます。

#### `with`

このパラメータはどのような順序で指定しても構いません。`with` 句を使用する場合、次のオプションを 1 つ以上指定する必要があります。

- `with dbid = number` — 新しいデータベースの `dbid` を指定します。`dbid` を明示的に指定しない場合、未使用の `dbid` がサーバによって割り当てられます。
- `with default_location` — 新しいテーブルの記憶ロケーションを指定します。`for proxy_update` 句も指定している場合は、リモート・テーブルまたはビューごとに 1 つのプロキシ・テーブルが、指定したロケーションから自動的に作成されます。
- `with override` — データとトランザクション・ログが同じデバイスに混在することになり、データベースを最新のものにリカバリできなくなる可能性がある場合でも、`Adaptive Server` にデバイス仕様を強制的に指定します。この句を使用しないで同じデバイスにログとデータを混在させようとすると、`create database` コマンドは正常に動作しません。ログとデータが混在していても、`with override` を使用している場合は、警告を受けますが、コマンドは正常に動作します。

**durability =**

データベースの持続性レベルを決定します。

- **full** — すべてのトランザクションがディスクに書き込まれます。データベースの作成時に持続性レベルを指定しなかった場合、これがデフォルトになり、サーバの障害から完全なリカバリができるようになります。すべてのシステム・データベースは、この持続性レベル(ディスク上に常駐するデータベースの従来 of 持続性レベル)を使用します。
- **no\_recovery** — トランザクションがディスクに対して持続性が維持されず、サーバの障害時または停止時にすべての変更内容が失われます。ディスクベースのデータベースの場合、**Adaptive Server** は実行時にディスク・デバイスに定期的に、ただし制御されていない手段でデータを書き込みます。停止(正常停止、強制停止、またはサーバの障害と再起動)後も、**no\_recovery** で作成されているデータベースはリカバリされませんが、**model**、またはテンプレート・データベースが定義されている場合は、それらのデータベースから再作成されます。
- **at\_shutdown** — トランザクションは、サーバの稼働中と正常終了後に、持続性が維持されます。サーバに障害が発生すると、持続性が完全に失われます。

**[no] async\_init**

非同期のデータベース初期化の有効と無効を切り替えます。

**compression**

新しく作成したテーブルまたはパーティションに適用する圧縮レベルを示します。

- **none** — データは圧縮されません。
- **row** — 個々のロー内の1つ以上のデータ項目を圧縮します。**Adaptive Server** は、非圧縮形式に比べ、圧縮形式でのデータの格納により記憶領域を節約できる場合にのみ、**row** 圧縮形式でデータを格納します。
- **page** — ページがいっぱいになると、ロー圧縮された既存のデータ・ローが、ページ・レベルの圧縮を使用して圧縮され、ページ・レベルの辞書、インデックス、文字コードのエントリが作成されます。

**Adaptive Server** では、データがロー・レベルで圧縮されてからしかページ・レベルで圧縮されないため、圧縮を **page** に設定することは、**page** と **row** の両方が圧縮されることを意味します。

**lob\_compression = off | compression\_level**

新しく作成したテーブルの圧縮レベルを決定します。off を選択すると、テーブルに LOB 圧縮は使用されません。

圧縮アルゴリズムでは、LOB データを使用しないローは無視されます。

テーブルの圧縮レベル。次の圧縮レベルがあります。

- 0 – ローは圧縮されません。
- 1 ~ 9 – Adaptive Server は ZLib 圧縮を使用します。通常、圧縮の数値が大きいほど、Adaptive Server での LOB データの圧縮レベルが高くなり、圧縮データと非圧縮データの比率も大きくなります (つまり、非圧縮データと比べて、圧縮データによる記憶領域の節約量 (バイト) が多くなります)。

ただし、圧縮の量は LOB の内容によって異なります。圧縮レベルが高いほど、圧縮に使用される CPU の量が多くなります。そのためレベル 9 を使用すると、圧縮率が最高になりますが、CPU の使用量も最高になります。

- 100 – Adaptive Server は FastLZ 圧縮を使用します。CPU 使用量が最も少ない圧縮率です。通常、短いデータに使用します。
- 101 – Adaptive Server は FastLZ 圧縮を使用します。値に 101 を指定すると、100 を指定した場合よりも、CPU の使用量は若干多くなりますが、圧縮率は良くなります。

**inrow\_lob\_length = value**

バイト数を指定します。inrow\_lob\_length の有効値の範囲は 0 からデータベースの論理ページ・サイズまでです。0 を指定すると LOB 指定がデータベース全体でオフになり、個別の in row 句のない LOB カラムはすべてロー外の LOB カラムとして作成されます。

**dml\_logging**

DML オペレーションのログ・レベルを指定します。

**full | minimal**

DML オペレーションの完全なロギングまたは最低限のロギングを指定します。

**for load**

データベース・ダンプのロードだけに使用可能な、新しいバージョンの create database を呼び出します。[「for load オプションの使用」\(135 ページ\)](#) を参照してください。

**for proxy\_update**

メタデータをリモートのロケーションから自動的に取得し、プロキシ・テーブルを作成します。with default\_location も指定していないと、for proxy\_update は使用できません。

**global temporary**

グローバル・テンポラリー・データベースを作成することを示します。

**system temporary**

ローカル・システム・テンポラリー・データベースを作成することを示します。

**temporary**

テンポラリー・データベースを作成することを示します。

**for instance instance\_name**

作成するローカル・システム・テンポラリー・データベースまたはローカル・テンポラリー・データベースを所有するインスタンスを指定します。このパラメータは、グローバル・テンポラリー・データベースを作成する場合には使用されません。

---

**注意** ローカル・ユーザ・テンポラリー・データベースは、このデータベースを所有するインスタンスから作成する必要があります。ローカル・システム・テンポラリー・データベースは、どのインスタンスからも作成できます。

---

**例**

**例 1** pubs という名前のデータベースを作成します。

```
create database pubs
```

**例 2** pubs という名前の 4MB のデータベースを作成します。

```
create database pubs
on default = 4
```

size の単位を指定しなかった場合、pubs に指定する値はメガバイト単位であると見なされます。

**例 3** datadev デバイスに 3MB の領域を持ち、moredatadev デバイスに 2MB の領域を持つ pubs という名前のデータベースを作成します。

```
create database pubs
on datadev = "3M", moredatadev = '2.0m'
```

**例 4** datadev デバイスにデータ用の 3MB の領域を持ち、logdev デバイスにログ用の 0.5GB の領域を持つ pubs という名前のデータベースを作成します。

```
create database pubs
  on datadev='3m'
  log on logdev='0.5g'
```

**例 5** proxydb という名前のプロキシ・データベースを作成しますが、プロキシ・テーブルは自動的に作成しません。

```
create database proxydb
with default_location
  "UNITEST.pubs.dbo."
```

**例 6** proxydb という名前のプロキシ・データベースを作成し、自動的にプロキシ・テーブルを作成します。

```
create database proxydb
on default = "4M"
with default_location
  "UNITEST.pubs2.dbo."
for proxy_update
```

**例 7** proxydb という名前のプロキシ・データベースを作成し、リモート・データベースからすべてのリモート・テーブルのメタデータを取得します。

```
create database proxydb
on default = 4
with default_location
  "UNITEST.pubs2.."
for proxy_update
```

**例 8** with dbid が 15 の pubs という名前のデータベースを作成します。

```
create database pubs with dbid = 15
```

**例 9** datadev デバイスにデータ用の 3MB の領域を持ち、logdev デバイスにログ用の 1MB の領域を持つ mytempdb1 という名前のテンポラリ・データベースを作成します。

```
create temporary database mytempdb1
  on datadev = '3m' log on logdev = '1M'
```

**例 10** クラスタ環境で、"ase1" にローカル・ユーザ・テンポラリ・データベースを作成します。所有者インスタンス ("ase1") から次のコマンドを実行します。

```
create temporary database local_tempdb1 for instance
  ase1
```

または

```
create temporary database local_tempdb1
```

**例 11** クラスタ環境で、"asel" にローカル・システム・テンポラリ・データベースを作成します。このコマンドは、クラスタの任意のインスタンスから実行できます。

```
create system temporary database local_systempdb1 for
instance asel
```

**例 12** クラスタ環境で、グローバル・テンポラリ・データベースを作成します。

```
create global temporary database global_tempdb1
```

**例 13** imdb\_data\_dev1 (データ) および imdb\_logdev (ログ) という 2 つの異なるインメモリ記憶デバイス上にインメモリ・データベースを作成します。

```
create inmemory database imdb2
on imdb_data_dev1 = '1.0g'
log on imdb_logdev = '0.5g'
with durability = no_recovery
```

**例 14** 複数のインメモリ記憶デバイス上にインメモリ・データベースを作成します。imdb\_data\_dev1 および imdb\_data\_dev2 にはすべてのデータが格納され、inmem\_logdev にはログが格納されます。

```
create inmemory database imdb3
on imdb_data_dev1 = '100m',
   imdb_data_dev2 = '200m'
log on inmem_logdev = '50m'
with durability=no_recovery
```

**例 15** テンプレートとして pubs2 データベースを使用して、pubs5 データベースを作成します。

```
create inmemory database pubs5
use pubs2 as template
on imdb_duck1_cach = '5m'
log on imdb_duck_log = '5m'
with durability = no_recovery
```

**例 16** pubs5\_rddb という名前のリラックス持続性データベースを作成します。

```
create database pubs5_rddb on pubs5_dev = '6M'
log on pubs5_log = '2M'
with durability = at_shutdown
```

**例 17** インメモリ・テンポラリ・データベース専用のメモリ内記憶域キャッシュを作成します。

1 メモリ内記憶域キャッシュを作成します。

```
sp_cacheconfig inmem_tempdb_cache, "40m", inmemory_storage,
"none", "cache_partition=2"
```

- 2 テンポラリ・データベースを作成するメモリ内デバイスを作成します。

```
DISK INIT name = "inmem_dev"
, physname = "inmem_tempdb_cache"
, size = "40m"
, type='inmemory'
```

- 3 インメモリ・データベースを作成します。

```
create inmemory temporary database temp_imdb
on inmem_dev = "20m"
with durability = no_recovery
```

**例 18** 持続性を `no_recovery` に設定して、既存のディスク・デバイスにテンポラリ・データベースを作成します。

```
create temporary database tempdb_rddb_norec
on datadev = "5m" log on logdev = "5m"
with durability = no_recovery
```

**例 19** `emaildb` データベースを作成し、ページレベルの圧縮を設定します。

```
create database emaildb
on email_dev = '50M'
with compression = page
```

**例 20** `email_lob_db` データベースを作成し、LOB 圧縮レベル 101 を設定します。

```
Create database email_lob_db
on email_lob_dev = '50M'
with lob_compression = 101
```

**例 21** 300 バイトの長さのロー内 LOB データを許容する `pubs` というデータベースを作成します。

```
create database pubs
with inrow_lob_length = 300
```

## 使用法

- `create database` は、`master` データベースから実行します。
- `size` は `float` データ型として指定できますが、指定したサイズは、そのサイズに最も近いアロケーション・ユニットの倍数に丸められます。
- データベースのサイズを明示的に指定しない場合、サイズは `model` データベースのサイズによって決まります。作成可能なデータベースの最小サイズは 4 アロケーション・ユニットです。

- Adaptive Server では、`create database` と `alter database` の場合は 256 の論理ページを 1 まとまりとしてデータベースの領域が割り付けられます。そのため、指定したサイズはそのサイズに最も近いアロケーション・ユニットの倍数に丸められます。
- 単位を付加しないと、Adaptive Server によってディスク領域のサイズがメガバイトとして解釈されます。また、この数値はサーバが使用する論理ページ・サイズに変換されます。
- データベースのロケーションとサイズを指定しない場合は、デフォルトのロケーションは `master..sysdevices` に示された任意のデフォルト・データベース・デバイスになります。デフォルトのサイズは、`model` データベースのサイズと `sysconfigures` 内の `default database size` パラメータのうちの大きい方です。

システム管理者は、`sp_configure` を使用して `default database size` の値を変更し、Adaptive Server を再起動することによって、デフォルトのサイズを拡大することができます。`default database size` パラメータは、`model` データベースと同じ大きさまたはそれ以上とします。`model` データベースのサイズを増やした場合は、デフォルトのサイズも増やす必要があります。

必要な場所に必要なサイズの領域が Adaptive Server によって確保されない場合は、Adaptive Server はデバイスごとに必要なサイズにできるだけ近い量を確保して、どのデバイスにどれだけの領域が割り付けられたかを示すメッセージを表示します。データベースの最大サイズは、システムによって異なります。

- 次の構文でプロキシ・データベースが作成されている場合の注意を下記に示します。

```
create database mydb on my_device
with default_location = "pathname" for proxy_update
```

デバイス名の存在によってサイズの計算が無視されるほか、デフォルトのデータベース・サイズ (`model` データベースのサイズ) がすべてのプロキシ・テーブルを格納するのに十分な大きさでないと、このコマンドは正常に動作しないことがあります。

コンポーネント統合サービスでデータベース・サイズを予測できるようにする場合は、次のように、デバイス名や他のオプションをこのコマンドに指定しないでください。

```
create database mydb
with default_location = "pathname" for proxy_update
```



## 制限事項

- Adaptive Server では、最大で 32,767 個のデータベースを管理できます。
- dbid は、1 以上 32,767 (dbid の最大数) 未満である必要があります。
- Adaptive Server が一度に作成できるデータベースは 1 つだけです。データベースの作成要求が 2 つ発生すると、一方のユーザに次のメッセージが表示されます。
- model database in use:cannot create new database
- create database または alter database を使用して領域をデータベース・デバイスに割り付けると、割り付けのそれぞれはデバイス・フラグメントに相当し、ローとして sysusages に入力されます。
- どのデータベースでも、指定できるセグメントの最大数は 32 です。セグメントには、特定の Adaptive Server に使用できるデータベース・デバイスのサブセットの名前が付けられます。セグメントの詳細については、『システム管理ガイド』を参照してください。

## テンポラリ・データベース

- create temporary database コマンドとともに with default\_location パラメータまたは for proxy\_update パラメータを使用することはできません。使用すると、次の 2 つの例のようにエラーが発生します。

```
1> create temporary database tbl with default_location
    "remSERVER.mydb.."
```

```
Msg 102, Level 15, State 7:
Server 'ebi_SUS_AS125x_SUN32', Line 1:
Incorrect syntax near 'create temporary database'.
```

```
1> create temporary database tbl with default_location
    "remSERVER.mydb.." for proxy_update
```

```
Msg 102, Level 15, State 7:
Server 'ebi_SUS_AS125x_SUN32', Line 1:
Incorrect syntax near 'create temporary database'.
```

- データベースのテンポラリ・ステータスはテンポラリ・データベースの作成時に設定され、sysdatabases エントリの status3 フィールドの値 0x00000100 (10 進数 256) によって示されます。
- model から継承したすべてのオプションに加えて、システムの tempdb のようなテンポラリ・データベースには次のデータベース・オプションが設定されています。
  - select into/bulkcopy
  - trunc log on chkpt

- システムの `tempdb` と同様に、`guest` ユーザがテンポラリ・データベースに追加され、`create table` パーミッションが `PUBLIC` に付与されます。
- サーバが再起動するたびにテンポラリ・データベースが再作成されるので、テンポラリ・データベースの作成時には、使用されていないページはクリアされません。

#### 圧縮データベースの作成

- `create table . . . with compression` の圧縮設定は、`create database` 圧縮設定を無効にします。
- `select into` で作成したテンポラリ・テーブルはデータベースから圧縮レベルを継承しません。
- `model` データベースの圧縮のデフォルト設定は、`none` (`model` に基づき、すべてのデータベースのデータ圧縮はオフ) です。
- `tempdb` またはその他のテンポラリ・データベースでデータ圧縮を有効にすると、セッションまたはストアド・プロシージャで作成されたテンポラリ・テーブルは、`tempdb` の圧縮レベルを継承しません。
- テンポラリ・テーブルの圧縮レベルは、それを作成したタスクのバインド先のデータベースによって決定されます。

#### ロー内の LOB を使用したデータベースの作成

- ロー内サイズには、データベースで許容される最大ロー・サイズまでを指定できます。Adaptive Server では、1 ページで使用可能な空き領域からページ・オーバーヘッドまたはロー・オーバーヘッドを差し引いた値に基づいて、挿入および更新の実行時にロー内 LOB 記憶域の個々のカラムのサイズの制限値を引き下げます。
- データベース全体に有効なロー内 LOB の長さを指定したデータベースでテーブルを作成すると、カラムの定義の構文で `off row` を指定しない限り、そのテーブル内のすべての LOB カラムはロー内として作成されます。カラムのロー内の長さ (バイト) は、このデータベース全体の設定によって指定されます。
- `inrow_lob_length` 設定のデフォルト値は、0 バイトで、Adaptive Server バージョン 15.7 にアップグレードしても既存のデータベースの動作は変更されません。このデフォルトを変更すると、ロー内の記憶領域上にある、異なる要件を持つアプリケーションがロー内に格納される LOB データの量を制御できます。

- データベース全体の設定が適用されるのは、新しく作成されたテーブル、またはそのデータベース全体の設定が適用または変更された後で既存のテーブルに追加された **LOB** カラムのみです。テーブルの最初の作成時に各 **LOB** カラムが継承するロー内 **LOB** の長さは、データベース全体の設定が変更された場合でも、そのままです。個々の **LOB** カラムのロー内の長さ、またはロー内のプロパティを変更するには、`alter table modify column` を使用します。
- デフォルトの長さを変更するには、`alter database` の `inrow_lob_length` パラメータを使用します。

#### インメモリ・データベースおよびリラックス持続性データベースの作成

- インメモリ・データベースとしてリストする `database_device` は、メモリ内記憶デバイスである必要があります。
- `for load` パラメータは、データベースが最初は `load database` コマンドを使用したロード待ちの状態で作成されたことを示します。
- 次の処理はできません。
  - デフォルト・デバイス上のインメモリ・データベースの作成。
  - システム・データベースとしてのインメモリ・データベースの使用 (`tempdb` 以外)。
  - インメモリ・データベースのアーカイブ・データベースとしての使用。インメモリ・データベースはスクラッチ・データベースとして使用しないことをおすすめします。
  - 作成するデータベースとそのテンプレート・データベースへの同じ名前の使用。
  - テンプレート・データベースとしての `model` を含むシステム・データベースの指定。
  - ディスクベースの記憶デバイスとキャッシュベースの記憶デバイスの同時使用。**Adaptive Server** は、メモリ内記憶域キャッシュですべて作成されたデータベースをインメモリ・データベースと同じように処理します。したがって、次のデバイスは使用できません。
    - あるインメモリ・データベースについて、別のメモリ内記憶域キャッシュで作成されたメモリ内記憶デバイス。
    - 別のインメモリ・データベースについて、すべてまたは一部があるメモリ内記憶域キャッシュで作成されたメモリ内記憶デバイス。

- **use as template** パラメータと **with default\_location =** パラメータの同時使用。
- **for load** パラメータおよび **for proxy update** パラメータと **use as template** パラメータの同時使用。
- 次の処理ができます。
  - すべてのメモリ内記憶デバイスがメモリ内記憶域キャッシュによってホストされている場合、名前付きキャッシュに存在するメモリ内記憶デバイス上へのインメモリ・データベースの作成。
  - 同じメモリ内記憶デバイス上でのログとデータが混在するインメモリ・データベースの作成。ただし、ログとデータが混在するインメモリ・データベースは1つのキャッシュ上に存在する必要があります。
- インメモリ・データベースを作成する場合は、**durability = no\_recovery** パラメータが必須になります。このパラメータを指定すると、サーバの再起動時にインメモリ・データベースが必ず再作成されます。

#### **model** から作成される新しいデータベース

- **Adaptive Server** は、**model** データベースをコピーして新しいデータベースを作成します。
- テーブル、ストアド・プロシージャ、ユーザ定義データ型、他のオブジェクトの追加やデータベース・オプションの設定の変更を行うと、**model** をカスタマイズできます。新しいデータベースは、これらのオブジェクトと設定を **model** から継承します。
- リカバリ性を保証するため、**create database** は、**model** データベースのコピー時に初期化されなかったページをすべてクリアしなければなりません。データベースのサイズやシステムの処理速度によっては、この作業に数分かかることがあります。

データベース・ダンプをロードするためにデータベースを作成する場合は、**for load** オプションを使用すると、ページをクリアする作業を省略できます。こうすると、非常に速くデータベースを作成できます。

#### **データベースのリカバリ性の保証**

- 新しいデータベースを作成したら、必ず **master** データベースのバックアップをとってください。バックアップによって、**master** が損傷した場合でも、簡単かつ安全にリカバリできます。

---

**注意** データベースを作成したときに **master** のバックアップをとらなかった場合でも、**disk reinit** を実行すれば変更内容をリカバリできることがあります。

---

- **with override** 句を使用すると、ログ・セグメントとデータ・セグメントを同じデバイスに混在させることができます。ただし、完全にリカバリできるようにするため、**log on** で指定した1つまたは複数のデバイスはデータを格納する物理デバイスとは別のデバイスにしてください。ハード・ディスクがクラッシュした場合は、データベース・ダンプとトランザクション・ログからデータベースをリカバリできます。

小さいデータベースは、トランザクション・ログとデータの両方を保管する単一のデバイス上に作成できますが、**dump database** コマンドを使用してバックアップをとる必要があります。

- トランザクション・ログに必要なデバイスのサイズは、更新アクティビティの量やトランザクション・ログ・ダンプの頻度によって異なります。通常は、データベース自体に割り付ける領域の10～25%をログ・デバイスに割り付けるのが妥当です。トランザクション・ログ・デバイスに割り付けた領域は後からは変更できず、データの保存には使用できないため、小さな領域から始めることをおすすめします。

#### **for load** オプションの使用

**sp\_addsegment** を使用してデータベースに追加していない場合は、記憶媒体の障害からのリカバリや、マシン間でのデータベースの移動に **for load** オプションを使用できます。**alter database for load** を使用して、ロードするデータベース・ダンプの作成元であるデータベースのイメージに、新しいデータベースを作成してください。ダンプを新しいデータベースにロードするときに領域の割り付けを複製する方法については、『システム管理ガイド』を参照してください。

- **for load** オプションを使用してデータベースを作成すると、データベース・ダンプをロードするまでは、次のコマンドしか実行できません。
  - **alter database for load**
  - **drop database**
  - **load database**

データベース・ダンプを新しいデータベースにロードしたら、データベースで **dbcc** 診断コマンドも使用できます。 **online database** コマンドを発行した後は、使用できるコマンドに制限はありません。

- for load オプションを使用して作成されたデータベースは、sp\_helpdb の出力に "don't recover" (リカバリ禁止) のステータスが表示されます。

### データベースについての情報の取得

- データベースについてのレポートを取得するには、sp\_helpdb を実行します。
- データベース内で使用されている領域についてのレポートを取得するには、sp\_spaceused を使用します。

### with default\_location および for proxy\_update の使用

for proxy\_update 句を指定しない場合、with default\_location 句の動作は、sp\_defaultloc によって実行される動作と同じになります。したがって、新規および既存のテーブル作成ではデフォルトの記憶領域のロケーションが設定されますが、create database の処理時にプロキシ・テーブル定義の自動インポートは行われません。

- default\_location を指定しないで for proxy\_update を指定すると、エラーが表示されます。
- for proxy\_update オプションを使用してプロキシ・データベースを作成すると、コンポーネント統合サービスが呼び出され、次の処理を行います。
  - プライマリ・サーバのデータベースで検出された実際のテーブルとビューを表すプロキシ・テーブルをすべて含むのに必要なデータベース・サイズを予測する。予測は、すべてのプロキシ・テーブルとインデックスを含むのに必要なデータベース・ページの数である。サイズが指定されておらず、データベース・デバイスも指定されていない場合は、この予測が使用される。
  - コンパニオン・サーバのデータベースで検出された実際のテーブルおよびビューを表すプロキシ・テーブルすべてを作成する。
  - プロキシ・テーブルのすべてのパーミッションを public に付与する。
  - guest ユーザをプロキシ・データベースに追加する。

- このデータベースが “Is\_A\_Proxy” であることを示すように、データベースのステータスを設定する。このステータスは、`master.dbo.sysdatabases.status3` に含まれている。

## 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

## パーミッション

`create database` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>create database</code> 権限が必要。 <code>sybsecurity</code> データベースを作成する場合は、 <code>manage auditing</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、システム管理者であるか、 <code>create database</code> 権限が必要。 <code>sybsecurity</code> データベースを作成するのは、システム・セキュリティ担当者である必要がある。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
9	create	create database	<ul style="list-style-type: none"> <li><i>Roles</i> – 現在のアクティブな役割</li> <li><i>Keywords</i> または <i>options</i> – NULL</li> <li><i>Previous value</i> – NULL</li> <li><i>Current value</i> – NULL</li> <li><i>Other information</i> – NULL</li> <li><i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter database](#), [disk init](#), [drop database](#), [dump database](#), [load database](#), [online database](#)

システム・プロシージャ [sp\\_changedbowner](#), [sp\\_diskdefault](#), [sp\\_helpdb](#), [sp\\_logdevice](#), [sp\\_renamedb](#), [sp\\_spaceused](#)

## create default

**説明** 挿入時に、明示的に指定された値がない場合、1つのカラム(またはユーザ定義データ型のすべてのカラム)に挿入する値を指定します。

**構文** `create default [owner.]default_name  
as constant_expression`

**パラメータ** *default\_name*  
デフォルトの名前です。名前は識別子の規則に従っている必要があります。変数を使うことはできません。現在のデータベース内で別のユーザが所有しているデフォルトと同じ名前で別のデフォルトを作成する場合は、所有者の名前を指定します。*owner*のデフォルト値は現在のユーザです。

*constant\_expression*  
カラム名や他のデータベース・オブジェクトの名前を含まない式です。グローバル変数と、データベース・オブジェクトを参照しない組み込み関数を指定できます。文字定数および日付定数は引用符で囲み、バイナリ定数として"0x"プレフィックスを使用します。

**例** **例 1** `@@spid` グローバル変数を使用する D1 というデフォルトを作成します。

```
create default D1 as @@spid
```

**例 2** デフォルト値を定義してから、適切なカラム、またはユーザ定義のデータ型にそれをバインドします。

```
create default phonedflt as "UNKNOWN"  
sp_bindefault phonedflt, "authors.phone"
```

`authors` テーブルの `phone` カラムにエントリがない場合にかぎり、指定したデフォルトが使用されます。エントリがないことと、`null` 値が入力されていることとは、意味が異なります。デフォルトを取得するには、デフォルトの設定を持つカラムをカラム・リストに含めずに、[insert](#) コマンドを発行してください。

**例 3** バインドされるカラムに、現在の日付を挿入するデフォルト値 `today's_date` を作成します。

```
create default today's_date as getdate ()
```

**使用法**

- `sp_bindefault` を使用して、カラム、または Adaptive Server で提供されているデータ型以外のユーザ定義データ型にデフォルトをバインドします。



- 古いデフォルトのバインドを解除しなくても、新しいデフォルトをデータ型にバインドできます。新しいデフォルトが古いデフォルトを上書きし、バインドを解除します。
- `create default` はデフォルトを作成する前に検査制約のためのエラー・チェックを実行します。
- デフォルトのソース・テキストを隠すには、`sp_hidetext` を使用します。

#### 制限事項

- 現在のデータベース内でのみ、デフォルトを作成できます。
- 1 つのバッチ内で `create default` 文を他の文と結合することはできません。
- 同じ名前の新しいデフォルトを作成するには、`drop default` を使用して事前に元のデフォルトを削除しておきます。`sp_unbindefault` を使用してデフォルトのバインドを解除してから削除してください。

#### データ型の互換性

- カラムのデータ型と互換性のないデフォルト値を挿入すると、Adaptive Server によってエラー・メッセージが生成されます。たとえば、“N/A” などの文字式を `integer` カラムにバインドする場合、カラム値を指定しない `insert` はエラーになります。
- デフォルト値が文字カラムには長すぎる場合、Adaptive Server は `string_truncation` オプションの設定に応じて、文字列のトランケートまたは例外の生成を行います。`set` コマンドを参照してください。

#### デフォルトについての情報の取得

- デフォルトの定義は、`syscomments` に格納されます。
- デフォルトがカラムにバインドされると、そのオブジェクト ID は `syscolumns` に格納されます。デフォルトがユーザ定義データ型にバインドされると、そのオブジェクト ID は `systypes` に格納されます。
- デフォルトの名前を変更するには、`sp_rename` を使用します。
- デフォルトのテキストについてのレポートを取得するには、`sp_helptext` を使用します。

### デフォルトとルール

あるカラムに、デフォルトとそのデフォルトに関連したルールの両方が存在する場合は、デフォルト値がそのルールに違反しないようにしてください。ルールに違反するデフォルトは挿入できません。ルールに違反するデフォルトを挿入すると、Adaptive Server によってエラーメッセージが生成されます。

### デフォルトと null

カラムで null が許容されず、そのカラムに対しデフォルトを作成しない場合は、そのカラム用の値を含まないローを挿入しようとしても、挿入できません。この場合、Adaptive Server はエラー・メッセージを生成します。

表 1-3 は、デフォルトの有無とカラムが null であるか not null であるかの定義との関係を示します。

表 1-3: null とカラムのデフォルトとの関係

カラム null 型	エントリなし、デフォルトなし	エントリなし、デフォルトあり	エントリは null、デフォルトなし	エントリは null、デフォルトあり
NULL	Null が挿入される。	デフォルト値が挿入される。	Null が挿入される。	Null が挿入される。
NOT NULL	エラー、コマンドは失敗する。	デフォルト値が挿入される。	エラー、コマンドは失敗する。	エラー、コマンドは失敗する。

### create table でのデフォルト値の指定

create default ではなく、create table 文の default 句を使用して、カラムのデフォルトを定義することもできます。ただし、定義したカラムのデフォルトはそのテーブル固有のものであり、別のテーブルにバインドすることはできません。整合性制約の詳細については、「create table」および「alter table」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

ANSI SQL に準拠したデフォルトを作成するには、create table 文の default 句を使用します。

### パーミッション

create default のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、create default 権限が必要。別のユーザのデフォルトを作成するには、create any default 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、sa_role が付与されている、または create default 権限を持つユーザであることが必要。

### 監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
14	create	create default	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter table](#), [create rule](#), [create table](#), [drop default](#), [drop rule](#)

システム・プロシージャ [sp\\_bindefault](#), [sp\\_help](#), [sp\\_helptext](#), [sp\\_rename](#), [sp\\_unbindefault](#)

## create encryption key

**説明** 暗号化キーを作成します。キーおよび暗号化に関連する情報はすべて、`create encryption key` によってカプセル化されます。この文では、暗号化アルゴリズムとキー・サイズ、キーのデフォルト・プロパティ、キーを暗号化するためのオプションのユーザ指定のパスワード、暗号化プロセスでの初期化ベクトルまたは埋め込みの使用を指定できます。

Adaptive Server では、キーの生成と暗号化に Security Builder Crypto が使用されます。

**構文** マスタ・キーを作成するには次の構文を使用します。

```
create encryption key [dual] master
    [for AES] with passwd char_literal
```

サービス・キーを作成するには次の構文を使用します。

```
create encryption key syb_extpasswdkey
    [ with { static key | master key } ]
create encryption key syb_syscommkey
    [ with { static key | master key } ]
```

カラム暗号化キーを作成するには次の構文を使用します。

```
create encryption key [[database.][owner.]]keyname
    [as default]
    [for algorithm_name]
    [with [{{passwd {char_literal | system_encr_passwd} | master key}}]
    [key_length num_bits]
    [init_vector {null | random}]
    [pad {null | random}]
    [[no] dual_control]]
```

**パラメータ**

**keyname**

現在のデータベース内のユーザのテーブル、ビュー、プロシージャ・ネーム・スペース内でユニークな名前を使用してください。キーが別のデータベースにある場合は、**database** 名を指定します。別のユーザのキーを作成する場合は、**owner** 名を指定します。**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。他のユーザのキーを作成できるのは、システム・セキュリティ担当者だけです。

**as default**

システム・セキュリティ担当者またはキー管理者は、暗号化のためのデータベースのデフォルト・キーを作成できます。データベースのデフォルト暗号化キーが存在する場合、テーブルの作成者は、`create table`、`alter table`、`select into` で **keyname** を使用せずに暗号化を指定できます。Adaptive Server は同じデータベースのデフォルト・キーを使用します。デフォルト・キーは変更できます。[「alter encryption key」 \(16 ページ\)](#) を参照してください。

**for algorithm\_name**

使用するアルゴリズムを指定します。AES (Advanced Encryption Standard) のみがサポートされます。AES では、キー・サイズとして 128 ビット、192 ビット、256 ビット、ブロック・サイズとして 16 バイトがサポートされています。

**AES**

AES (Advanced Encryption Standard) 暗号化アルゴリズムを使用してデータを暗号化します。

**syb\_extpasswdkey | syb\_syscommkey**

- **syb\_extpasswdkey** — **sysattributes** のすべての外部パスワードは強力な暗号化アルゴリズムを使った新しいキーを使用して再暗号化されます。
- **syb\_syscommkey** — その後 **sp\_hidetext** の実行ではすべて強力な暗号化アルゴリズムを使った新しいキーが使用されます。新しいキーを使用して既存のデータベース・オブジェクトを暗号化するには、そのオブジェクト上で **sp\_hidetext** を実行する必要があります。

**static key | master key**

静的なキーまたはマスタ・キーを使用して暗号化キーを作成していることを示します。

**keylength num\_bits**

作成するキーのサイズ (ビット単位)。AES の有効なキー長は 128、192、256 ビットです。デフォルトのキー長は 128 ビットです。

**password\_phrase**

引用符で囲まれた最長 255 バイトの英数字文字列で、カラム暗号化キーの暗号化に使用されるキー (キー暗号化キー) を生成するために使用されます。

**init\_vector random**

暗号化中に初期化ベクトルを使用するように指定します。暗号化アルゴリズムで初期化ベクトルを使用すると、まったく同じ 2 つのプレーン・テキストに対して異なる暗号テキストが生成され、暗号分析でデータのパターンが検出されるのを防ぎます。初期化ベクトルを使用すると、データのセキュリティが強化されます。

初期化ベクトルによってパフォーマンスが影響を受けます。インデックス作成や、ジョインと検索の最適化は、暗号化キーに初期化ベクトルが指定されていないカラムでしか実行できません。

デフォルトは、初期化ベクトルの使用、つまり **init\_vector random** です。初期化ベクトルの使用は、暗号化の暗号ブロック連鎖 (CBC) モードの使用を意味します。**init\_vector null** の設定は、電子コード・ブック (ECB) モードを意味します。

**init\_vector null**

暗号化中に初期化ベクトルを使用しません。この指定により、カラムがインデックスに対応できるようになります。

**pad null**

デフォルト値で、データのランダム埋め込みを行いません。カラムでインデックスをサポートする必要がある場合は、埋め込みを使用できません。

**pad random**

ランダムなバイトをデータに自動的に埋め込んでから暗号化します。暗号テキストのランダム化のために初期化ベクトルではなく埋め込みを使用できます。埋め込みは、プレーン・テキストの長さがブロック長の半分より短いカラムにのみ適切です。AES アルゴリズムの場合、ブロック長は 16 バイトです。

**[no] dual\_control**

マスタ・キーの作成にデュアル・コントロールが使用されているかどうかを示します。

**例** **例 1** "safe\_key" という 256 ビットのキーをデータベースのデフォルト・キーとして指定します。システム・セキュリティ担当者は以下を入力します。

```
create encryption key safe_key as default for AES with keylength 256
```

**例 2** ランダム埋め込みを使用してカラムを暗号化する、"salary\_key" という 128 ビットのキーを作成します。

```
create encryption key salary_key for AES with init_vector null pad random
```

**例 3** 初期化ベクトルを使用してカラムを暗号化する、"mykey" という 192 ビットのキーが作成されます。

```
create encryption key mykey for AES with keylength 192 init_vector random
```

**例 4** ユーザ指定のパスワードによって保護されるキーを作成します。

```
create encryption key key1 with passwd 'Worlds1Biggest6Secret'
```

キーによって暗号化されたカラムにアクセスするには、キーを保護しているユーザ指定のパスワードを入力する必要があります。[「set」\(712 ページ\)](#) を参照してください。

**例 5** "safe\_key" という 256 ビットのキーをデータベースのデフォルト・キーとして指定します。このキーはパスワードを指定しないため、Adaptive Server では safe\_key のキー暗号化キーとしてデータベースレベルのマスタ・キーが使用されます。マスタ・キーがない場合は、システム暗号化パスワードが使用されます。

```
create encryption key safe_key as default for AES with keylength 256
```

**例 6** マスタ・キーと “Whybothor” の組み合わせを使用して CEK k3 を暗号化します。

```
create encryption key k3 with passwd 'Whybothor' dual_control
create encryption key k1 with keylength 192
```

<p><b>使用法</b></p>	<p>ユーザ指定のパスワードは Adaptive Server には保存されません。保存されるのは、sysencryptkeys.eksalt 内の「ソルト」として知られる検証バイトの文字列です。この文字列により、後で暗号化または復号化を行う際に使用されるパスワードがキーの正しいパスワードであるかどうか Adaptive Server で認識されます。keyname によって暗号化されたカラムにアクセスするには、Adaptive Server にパスワードを入力する必要があります。</p>
<p><b>パーミッション</b></p> <p>細密なパーミッションが有効</p>	<p>create encryption key のパーミッション・チェックは、細密なパーミッションの設定によって異なります。</p> <p>細密なパーミッションが有効化されている場合、次の権限、または暗号化キーのタイプに応じた権限が必要。</p> <ul style="list-style-type: none"> <li>• カラム暗号化キー : create encryption key または manage column encryption key</li> <li>• マスタ・キー : manage master key</li> <li>• サービス・キー : manage service key</li> </ul>
<p>細密なパーミッションが無効</p>	<p>細密なパーミッションが無効の場合、sso_role または keycustodian_role が付与されたユーザであるか、暗号化キー作成権限が必要。</p>
<p><b>参照</b></p>	<p><b>コマンド</b> <a href="#">alter encryption key</a>, <a href="#">drop encryption key</a>, <a href="#">grant</a>, <a href="#">revoke</a></p> <p><b>マニュアル</b> 監査の詳細については、『暗号化カラム・ユーザーズ・ガイド』の「暗号化カラムの監査」を参照してください。</p>

## create existing table

**説明** (コンポーネント統合サービスのみ) プロキシ・テーブルを作成し、次にリモート・テーブルからメタデータを検索して格納し、そのデータをプロキシ・テーブルに配置します。こうすると、リモート・ロケーションのテーブル、ビュー、またはプロシージャにプロキシ・テーブルをマップできるようになります。

プロキシ・テーブルを作成する方法としては、`create proxy_table` コマンドの使用をおすすめします。この方法を使用すると、カラム定義を定義する必要がなくなります。

**構文**

```
create existing table table_name (column_list)
    [on segment_name]
    [[external {table | procedure | file | connection_type}] at pathname
    [column delimiter "string"]]
```

**パラメータ**

*table\_name*  
作成するプロキシ・テーブルの名前を指定します。

*column\_list*  
リモート・テーブルに関する情報を格納するカラム・リストの名前を指定します。

on *segment\_name*  
リモート・テーブルがあるセグメントを指定します。

external  
オブジェクトがリモート・オブジェクトであることを指定します。

table  
リモート・オブジェクトがテーブルまたはビューであることを指定します。デフォルトは、`external table` です。

procedure  
リモート・オブジェクトがストアド・プロシージャであることを指定します。

file  
リモート・オブジェクトがファイルであることを指定します。

*connection\_type*  
リモート・プロシージャ・コールが現在の接続と別の接続のどちらを使用するかを決定します。有効な値は次のとおりです。

- `non_transactional` — RPC の実行には別の接続を使用します。
- `transactional` — RPC の実行には既存の接続を使用します。

デフォルトは `transactional` です。



**at *pathname***

リモート・オブジェクトのロケーションを指定します。*pathname* の形式は、次のとおりです。*server\_name.dbname.owner.object*。ここで、

- *server\_name* (必須) – リモート・オブジェクトがあるサーバの名前です。
- *dbname* (任意) – このオブジェクトがあるリモート・サーバが管理するデータベースの名前です。
- *owner* (任意) – リモート・オブジェクトを所有するリモート・サーバのユーザの名前です。
- *object* (必須) – リモートのテーブル、ビュー、またはプロシージャの名前です。

**column delimiter**

フラット・ファイルにアクセスするときに各レコード内のフィールドを区切るために使用します。カラム・デリミタの長さは、最大 16 バイトです。

**string**

カラム・デリミタ文字列には、あらゆる文字シーケンサを使用できますが、文字列が 16 バイトを超えた場合は、最初の 16 バイトのみが使用されます。ファイル以外にマップされたプロキシ・テーブルでカラム・デリミタを使用すると、構文エラーが発生します。

**例****例 1** プロキシ・テーブル *authors* を作成します。

```
create existing table authors
(
  au_id          id,
  au_lname      varchar (40)    NOT NULL,
  au_fname      varchar (20)    NOT NULL,
  phone         char (12),
  address       varchar (40)    NULL,
  city          varchar (20)    NULL,
  state         char (2)        NULL,
  zip           char (5)        NULL,
  contract      bit
)
at "nhserver.pubs2.dbo.authors"
```

**例 2** プロキシ・テーブル *syb\_columns* を作成します。

```
create existing table syb_columns
(
  id            int,
  number       smallint,
```

```
colid      tinyint,  
status    tinyint,  
type      tinyint,  
length    tinyint,  
offset     smallint,  
usertype  smallint,  
cdefault  int,  
domain    int,  
name      varchar (30),  
printfmt  varchar (255)      NULL,  
prec      tinyint          NULL,  
scale     tinyint          NULL  
)  
at "remotel.master.dbo.columns"
```

**例 3** リモート・サーバ **SERVER\_A** の **blurbs** テーブルに **blurbs** という名前のプロキシ・テーブルを作成します。

```
create existing table blurbs  
(  
  author_id    id      not null,  
  copy         text    not null  
)  
at "SERVER_A.db1.joe.blurbs"
```

**例 4** **p1** という名前のリモート・プロシージャに **rpc1** という名前のプロキシ・テーブルを作成します。

```
create existing table rpc1  
(  
  column_1     int,  
  column_2     int  
)  
external procedure  
at "SERVER_A.db1.joe.p1"
```

#### 使用法

- リモート・オブジェクトがファイルでないかぎり、  
**create existing table** は新しいテーブルを作成しません。代わりに、コンポーネント統合サービスがテーブル・マッピングを検査し、**column\_list** 内の情報がリモート・テーブルと一致していることを確認します。そして、基本となるオブジェクトの存在を確認し、リモート・テーブルに関するメタデータを検索して格納します。
- ホスト・データ・ファイルまたはリモート・サーバ・オブジェクトが存在しない場合は、コマンドは拒否され、エラー・メッセージが表示されます。

- オブジェクトが存在する場合は、システム・テーブル `sysobjects`、`syscolumns`、`sysindexes` が更新されます。確認には、次の手順が必要です。
  - a 既存のオブジェクトの性質が判断されます。ホスト・データ・ファイルの場合、ファイル編成やレコード・フォーマットを調べる必要があります。リモート・サーバ・オブジェクトの場合、オブジェクトがテーブル、ビュー、RPC のいずれであるかを調べる必要があります。
  - b リモート・サーバ・オブジェクトの場合 (RPC を除く)、テーブルまたはビューのために取得されたカラム属性を `column_list` 内で定義されたカラム属性と比較します。
  - c ホスト・データ・ファイルまたはリモート・サーバ・テーブルからインデックス情報を抽出し、システム・テーブル `sysindexes` のローを作成するために使用します。これによって、`Adaptive Server` 用語でインデックスとキーが定義され、クエリ・オプティマイザがテーブル上に存在する可能性があるすべてのインデックスを検証できるようになります。
- `on segment_name` 句は、ローカルに処理されるもので、リモート・サーバには渡されません。
- 既存のテーブルを正しく定義したら、そのテーブルに対して `update statistics` コマンドを発行します。これによってクエリ・オプティマイザは、インデックスの選択およびジョイン順に関して最善の選択を行うことが可能になります。
- コンポーネント統合サービスは、リモート・カラムが `NULL` と定義されている場合でも、`NOT NULL` として定義されたカラムを持つプロキシ・テーブルの作成を可能にします。また、不適当な組み合わせであることを知らせる警告を表示します。
- `at` キーワードによって得られるロケーション情報は、`sp_addobjectdef` によって得られる情報と同じです。この情報は、`sysattributes` テーブルに格納されます。
- コンポーネント統合サービスは、リモート・テーブルのインデックスごとに `systabstats` カタログのレコードを挿入または更新します。リモート・インデックスには詳細な構造情報は関係がないため、`id`、`indid`、`rowcnt` という最小数のカラムだけが `systabstats` レコードに設定されます。
- 外部ファイルのデータ型は、`text`、`image`、Java ADT 以外でなければなりません。

### データ型の変換

- `create existing table` コマンドを使用するときは、**Adaptive Server** で認識されるデータ型を使用してすべてのデータ型を指定する必要があります。リモート・サーバ・テーブルが異機種のサーバのクラスにある場合は、データが検索されるときに、リモート・テーブルのデータ型は指定された **Adaptive Server** 型に自動的に変換されます。変換が行われないときは、コンポーネント統合サービスはそのテーブルの定義付けを許可しません。
- サポートされているサーバ・クラス、およびコンポーネント統合サービスによって暗黙的に行われる可能性があるすべてのデータ型の変換については、『コンポーネント統合サービス・ユーザーズ・ガイド』の、該当する項を参照してください。

### サーバ・クラスによる変換

- すべてのサーバ・クラスで、リモート・サーバのテーブルにあるカラムよりも少ないカラムを指定できます。
- すべてのサーバ・クラスが、名前によってカラムと一致します。
- すべてのサーバ・クラスで、リモート・テーブルにあるカラムのデータ型との変換が可能なデータ型にカラムのタイプを指定できます。

### リモート・プロシージャ

- プロキシ・テーブルがプロシージャ型のテーブルであるときは、リモート・プロシージャの結果セットの記述に一致するカラム・リストを指定する必要があります。`create existing table` では、このカラム・リストの正確性が検証されません。
- プロシージャには、インデックスは作成されません。
- コンポーネント統合サービスでは、`insert` または `select` を使用してソートや他のテーブルとのジョインを行ったり、別のテーブルに挿入したりできる仮想テーブルとしてリモート・プロシージャの結果セットが扱われます。ただし、プロシージャ型のテーブルは読み込み専用と見なされるため、次のコマンドをテーブルに対して発行することはできません。
  - `alter table`
  - `create index`
  - `delete`
  - `insert`
  - `truncate table`
  - `update`

- カラムがリモート・プロシージャの結果セットの一部でないことを指定するには、カラム名をアンダースコア (\_) で開始します。このようなカラムはパラメータ・カラムと呼ばれます。次に例を示します。

```
create existing table rpcl
(
  a          int,
  b          int,
  c          int,
  _p1       int null,
  _p2       int null
)
external procedure
at "SYBASE.sybsystemprocs.dbo.myproc"
```

この例では、パラメータ・カラム `_p1` と `_p2` が入力パラメータです。これらは結果セットでは予期されていませんが、次のクエリで参照できます。

```
select a, b, c from t1
where _p1 = 10 and _p2 = 20
```

コンポーネント統合サービスは `@p1` および `@p2` という名前を使用して、探索引数をパラメータとしてリモート・プロシージャに渡します。

- `create existing table` 文のパラメータ・カラムの定義には、以下の制約があります。
  - パラメータ・カラムに `null` 値を入力可能であること。
  - パラメータ・カラムを通常の結果カラムより先に定義できない。パラメータ・カラムは、カラム・リストの最後に置く必要がある。
- パラメータ・カラムが `select` リストに含まれている場合にパラメータとしてリモート・プロシージャに渡されると、`where` 句によって戻り値が割り当てられます。
- パラメータ・カラムが `select` リストには含まれていても、`where` 句には表示されないか、またはパラメータとしてリモート・プロシージャに渡されない場合、その値は `NULL` になります。

- **Adaptive Server** のクエリ・プロセッサによって探索可能な引数と見なされると、パラメータ・カラムはパラメータとしてリモート・プロシージャに渡されます。パラメータ・カラムは、**or** 述部に含まれていなければ、探索可能な引数と見なされます。たとえば、次のクエリの 2 行目にある **or** 述部によって、パラメータ・カラムをパラメータとして使用することはできません。

```
select a, b, c from t1
where _p1 = 10 or _p2 = 20
```

### 暗号化カラム

**create existing table** を実行すると、リモート・テーブルの任意の暗号化カラム・メタデータによって **syscolumns** が自動的に更新されます。**encrypt** キーワードは **create existing table** コマンドのカラム・リストに含めることはできません。

### 標準規格

ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

### パーミッション

**create existing table** パーミッションは、デフォルトでテーブル所有者に設定され、譲渡することはできません。

### 参照

コマンド [alter table](#), [create table](#), [create proxy\\_table](#), [drop index](#), [insert](#), [order by 句](#), [set](#), [update](#)

## create function

説明	ユーザ定義関数を作成します。この関数は保存された Transact-SQL ルーチンで、指定値を返します。
構文	<pre>create function [ <i>owner_name</i>.] <i>function_name</i>     [ ( @<i>parameter_name</i> [as] <i>parameter_datatype</i> [= default ]       [,...n ] ) ]     returns <i>return_datatype</i>     [ with recompile ]     as     [begin]     <i>function_body</i>     return <i>scalar_expression</i>     [end]</pre>
パラメータ	<p><i>owner_name</i> ユーザ定義関数を所有するユーザ ID の名前です。既存のユーザ ID にする必要があります。</p> <p><i>function_name</i> ユーザ定義関数の名前です。関数名は識別子の規則に従い、データベース内と所有者に対しユニークにしてください。関数名を、他の Adaptive Server 関数と同じ名前にすることはできません。</p>

---

**注意** ユーザ定義関数を参照または呼び出すには、*owner\_name.function\_name* を指定し、次にカッコを続けます (下記の「例」の項にある **BONUS** 関数を参照してください)。カッコ内のすべてのパラメータに引数として式を指定します。関数を呼び出す際に引数リストにパラメータ名を指定できません。すべてのパラメータに引数値を指定し、その引数値は **create function** 文でパラメータが定義されたシーケンスと同じである必要があります。関数のパラメータがデフォルト値を持つ場合、関数を呼び出してデフォルト値を取得する際、キーワード「default」を指定します。

---

**@parameter\_name**

ユーザ定義関数のパラメータです。create function 文で、1 つ以上のパラメータを宣言できます。関数は、最大 2,047 のパラメータを持つことができます。宣言された各パラメータの値は、パラメータのデフォルトを定義しないかぎり、関数の実行時にユーザが指定します。

アットマーク (@) を最初の文字として使用し、パラメータ名を指定します。パラメータの名前は、識別子の規則に従う必要があります。パラメータは関数に対してローカルです。他の関数でも同じパラメータ名を使用できます。

パラメータにデフォルト値がある場合、ユーザは、関数を呼び出してデフォルト値を取得する際にキーワード「default」を指定する必要があります。

**parameter\_datatype**

パラメータのデータ型です。すべてのスカラ・データ型と Java ADT (抽象データ型) は、ユーザ定義関数のパラメータとして使用できます。ただし、ユーザ定義関数では、timestamp、text、image、unitext がサポートされません。

**with recompile**

Adaptive Server がこの関数のプランを保存しないことを意味します。代わりに、関数が SQL 文で最初に参照されるたびに新しいプランが作成されます。この関数の実行を通常どおりに行わない場合、つまり、新しいプランが必要な場合、with recompile を使用します。

**return\_datatype**

スカラの戻り値、ユーザ定義関数です。return\_datatype は、text、image、unitext、および timestamp を除くスカラ・データ型および Java ADT のいずれかです。

**scalar\_expression**

スカラ関数によって返されたスカラ値を指定します。

計算カラムと check 制約定義を含め、スカラ式が使用されるスカラ値関数を呼び出すことができます。

**function\_body**

一連の Transact-SQL 文を指定します。これには二次的な影響はありませんが、関数値を定義します。function\_body は、スカラ関数と複数文のテーブル値関数にのみ使用されます。スカラ関数では、function\_body はスカラ値を評価する一連の Transact-SQL 文です。

例

BONUS というユーザ定義関数を作成します。

```
create function BONUS(@salary int, @grade int, @dept_id int)
```



```

returns int
as
begin
declare @bonus int
declare @cat int
set @bonus = 0
select @cat = dept_cat from department
      where dept_id = @dept_id

if (@cat < 10)
begin
set @bonus = @salary *15/100

end
else
begin
set @bonus = @salary * 10/100
end
return @bonus
end

```

#### 使用法

- ユーザ定義関数の所有者が、内部参照されたすべてのデータベース・オブジェクトも所有する場合、関数の `execute` パーミッションを持つ他のユーザのすべてに、関数実行時に参照されたすべてのオブジェクトへのアクセス・パーミッションが自動的に付与されます。
- 関数が作成されると、Adaptive Server は、この関数が SQL ユーザ定義関数か SQLJ ユーザ定義関数のいずれであるかをチェックします。SQLJ ユーザ定義関数である場合、Adaptive Server は "sa" パーミッションの有無をチェックします。SQL 関数である場合、Adaptive Server は `create function` 権限の有無をチェックします。

#### パーミッション

`create function` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>create function</code> 権限が必要。他のユーザに対して <code>create function</code> を実行するには、 <code>create any function</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、 <code>create database</code> 権限が必要。

## create function (SQLJ)

**説明** Java の静的メソッドに SQL ラップを追加することにより、ユーザ定義の関数を作成します。メソッドによって定義された値を返すことができます。

**構文**

```
create function [owner.]sql_function_name
    ([ sql_parameter_name sql_datatype
      [(length)| (precision[, scale ])]
    [, sql_parameter_name sql_datatype
      [(length)| (precision[, scale])]]
    ...])
returns sql_datatype
    [(length)| (precision[, scale])]
[modifies sql data]
[returns null on null input |
  called on null input]
[deterministic | not deterministic]
[exportable]
language java
parameter style java
external name 'java_method_name
    [(java_datatype[, java_datatype
    ...])]'
```

**パラメータ**

*sql\_function\_name*

関数の Transact-SQL 名です。名前は識別子の規則に従っている必要があります。また、変数を使うことはできません。

*sql\_parameter\_name*

関数の引数の名前です。関数が実行されるたびに、それぞれの入力パラメータに値を渡します。SQLJ 関数には引数を付けなくてもかまわないため、パラメータは必要に応じて使用します。

パラメータの名前は、識別子の規則に従っている必要があります。パラメータ値に英数字以外の文字が含まれている場合には、その文字を引用符で囲んでください。データベース名または所有者名によって修飾されるオブジェクト名にはピリオドが含まれているため、そのようなオブジェクト名をパラメータ値に指定する場合は、引用符で囲まなければなりません。パラメータ値が数字で始まっている場合も、引用符で囲んでください。

*sql\_datatype [(length) | (precision [, scale])]*

パラメータの Transact-SQL データ型です。これらのパラメータの詳細については、「[create procedure](#)」(193 ページ)を参照してください。

*sql\_datatype* は、SQL プロージャのシグニチャです。

**returns sql\_datatype**

関数の結果データ型を指定します。

**modifies sql data**

Java メソッドが、SQL オペレーションを呼び出したり、データベースの SQL データの読み込みまたは変更を行ったりすることを示します。これはデフォルトであり、唯一の実装です。また、ANSI 標準と構文の互換性を持つようにするためのものです。

**deterministic | not deterministic**

ANSI 標準と構文の互換性を持つようにするためのものです。現在は実装されていません。

**exportable**

Adaptive Server の OmniConnect™ 機能を使用して、リモート・サーバでプロシージャを実行することを指定します。プロシージャと、組み込まれるメソッドのいずれも、リモート・サーバに存在する必要があります。

**language java**

外部ルーチンが Java で記述されていることを指定します。SQLJ 関数には、この句が必要です。

**parameter style java**

実行時に外部ルーチンに渡されるパラメータが Java のパラメータであることを指定します。SQLJ 関数には、この句が必要です。

**external**

SQL 以外のプログラミング言語で記述された外部ルーチンの SQL 名が `create function` によって定義されることを示します。

**name**

外部ルーチン (Java メソッド) の名前を指定します。指定される名前である、'`java_method_name [java_datatype{[, java_datatype] ...}]`' はリテラル文字列で、一重引用符で囲まれている必要があります。

**java\_method\_name**

外部 Java メソッドの名前を指定します。

**java\_datatype**

マップ可能な Java データ型またはマップ可能な結果セットを指定します。これは、Java メソッドのシングニチャです。

**例**

`java.lang.Math.sqrt()` メソッドを呼び出す関数 `square_root` を作成します。

```
create function square_root
  (input_number double precision) returns
  double precision
  language java parameter style java
  external name 'java.lang.Math.sqrt'
```

使用法

- Adaptive Server の組み込み関数と同じ名前の SQLJ 関数は作成できません。
- 同じクラス名やメソッド名のユーザ定義関数 (Java の静的メソッドに基づいたもの) と SQLJ 関数を作成できます。

**注意** Adaptive Server の探索順序では、常に SQLJ 関数が先に見つかります。

- create function 文では、パラメータを 31 個まで使用できます。
- 関数が作成されると、Adaptive Server は、この関数が SQL ユーザ定義関数か SQLJ ユーザ定義関数のいずれであるかをチェックします。SQLJ ユーザ定義関数である場合、Adaptive Server は "sa" パーミッションの有無をチェックします。SQL 関数である場合、Adaptive Server は create function 権限の有無をチェックします。

パーミッション

create function (SQLJ) のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、create function 権限が必要。他のユーザに対して create function を実行するには、create any function が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者か sa_role が付与されたユーザであることが必要。

監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
97	install	create function	<ul style="list-style-type: none"> <li>• Roles — 現在のアクティブな役割</li> <li>• Keywords または options — NULL</li> <li>• Previous value — NULL</li> <li>• Current value — NULL</li> <li>• Other information — NULL</li> <li>• Proxy information — set proxy が有効な場合は元のログイン名</li> </ul>

参照

create function の詳細については、『Adaptive Server Enterprise における Java』を参照してください。

コマンド [create function \(SQLJ\)](#), [drop function \(SQLJ\)](#)

システム・プロシージャ [sp\\_depends](#), [sp\\_help](#), [sp\\_helpjava](#), [sp\\_helprotect](#)

## create index

### 説明

テーブル内の 1 つ以上のカラム ( 計算カラムとそれ以外のカラムの両方 ) にインデックスを作成します。分割インデックスを作成します。

計算カラムを普通のカラムと同じようにインデックス・キーとして使用し、関数ベースのインデックスを作成できます。関数ベースのインデックスには、インデックス・キーとして 1 つ以上の式が含まれます。

既存の `create index` 構文では計算カラムにインデックスを作成できますが、関数ベースのインデックスには追加の構文が必要です。

### 構文

```
create [unique] [clustered | nonclustered] index index_name
  on [[database.] owner.] table_name
    (column_expression [asc | desc]
     [, column_expression [asc | desc]]...)
  [with {fillfactor = pct,
        max_rows_per_page = num_rows,
        reservepagegap = num_pages,
        consumers = x, ignore_dup_key, sorted_data,
        [ignore_dup_row | allow_dup_row],
        statistics using num_steps values}]
  [on segment_name]
  [index_partition_clause]
```

インデックス・パーティションを作成する場合 :

```
index_partition_clause::=
  [local index [partition_name [on segment_name]
  [, partition_name [on segment_name]...]]]
```

関数ベースのインデックスを作成する場合 :

```
create [unique | nonclustered] index index_name
  on [[database.] owner.] table_name
    (column_expression [asc | desc]
     [, column_expression [asc | desc]]...
```

### パラメータ

#### unique

インデックス値 ( 「キー値」 とも呼ばれます ) が重複しないようにします。インデックスの作成時 ( データがすでに存在する場合 ) や、`insert` または `update` を使用してデータを追加するたびに、システムによって重複するキー値がないかがチェックされます。重複したキー値が存在するか、複数のローに `null` 値がある場合、コマンドは正常に動作しません。また、重複するエントリを示すエラー・メッセージが表示されます。

**警告！** テーブルに `null` 以外の `text`、`unitext`、または `image` カラムがあると、Adaptive Server では重複するローが検出されません。

---

`allow_dup_row` オプションを使用してインデックスを作成すると、重複するキー値を生成する `update` コマンドおよび `insert` コマンドは正常に動作します。

複合インデックス (キー値が複数のカラムから作成されるインデックス) も、ユニークにすることができます。

デフォルトはユニークではありません。重複するローがある 1 つのテーブルに、ユニークでないクラスタード・インデックスを作成するには、`allow_dup_row` または `ignore_dup_row` を指定します。「[重複ロー](#)」(172 ページ) を参照してください。

範囲分割テーブル、リスト分割テーブル、ハッシュ分割テーブルでユニークなローカル・インデックスを作成すると、インデックス・キー・リストは、分割キー・リストのスーパーセットになります。

#### **clustered**

現在のデータベース・デバイスにあるローの物理的な順序が、インデックス付きのローの順序と同じであることを意味します。クラスタード・インデックスの下部、つまり「[リーフ・レベル](#)」には、実際のデータ・ページがあります。クラスタード・インデックスを使用すると、ほとんどの場合ノンクラスタード・インデックスよりも速くデータを検索できます。1 つのテーブルにはクラスタード・インデックスを 1 つだけ作成できます。「[クラスタード・インデックスの作成](#)」(170 ページ) を参照してください。

`clustered` を指定しないと、`nonclustered` であると見なされます。

#### **nonclustered**

ローの物理的な順序が、インデックス付きのローの順序と異なることを意味します。ノンクラスタード・インデックスのリーフ・レベルには、データ・ページのローを示すポインタが含まれています。1 つのテーブルにつき最大 249 のノンクラスタード・インデックスを設定できます。

#### **index\_name**

インデックスの名前です。インデックス名はテーブル内でユニークでなければなりません、データベース内でユニークである必要はありません。

**table\_name**

インデックスを作成する 1 つまたは複数のカラムが含まれているテーブルの名前です。テーブルが別のデータベース内にある場合はデータベース名を指定し、データベース内にその名前のテーブルが複数ある場合は所有者名を指定します。*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。

**column\_expression**

少なくとも 1 つのベース・カラムを参照し、他のテーブルのカラム、ローカル変数およびグローバル変数、集合関数、またはサブクエリを含まない、有効な Transact-SQL 式です。

---

**注意** *column\_expressions* は、15.0 より前の Adaptive Server バージョンで使用された *column\_name* 変数を置き換えるものです。

---

**asc | desc**

指定したカラムのインデックスを昇順で作成するか、降順で作成するかを指定します。デフォルトは、昇順です。

**fillfactor**

Adaptive Server で既存のデータに新しいインデックスを作成する場合の各ページのデータの占有率を指定します。*fillfactor* の率は、インデックスの作成時にのみ使用されます。データは変更されるので、ページが特定の満杯率で維持されることはありません。

指定した値は *sysindexes* に保存されません。保管された *fillfactor* 値を作成するには、*sp\_chgattribute* を使用してください。

*fillfactor* のデフォルトは 0 です。これは、*create index* 文で *with fillfactor* を指定しない場合に使用されます (*sp\_configure* で値を変更している場合は除きます)。 *fillfactor* を指定する場合は、1 ~ 100 の値を使用してください。

*fillfactor* が 0 の場合、完全に満杯になるページでクラスタード・インデックスが作成され、完全に満杯になるリーフ・ページでノンクラスタード・インデックスが作成されます。これによって、クラスタード・インデックスとノンクラスタード・インデックスの両方のインデックス B ツリー内に十分な領域が確保されます。ほとんどの場合、*fillfactor* を変更する必要はありません。

*fillfactor* を 100 に設定すると、各ページが 100% 満杯の状態で、クラスタード・インデックスとノンクラスタード・インデックスの両方が作成されます。*fillfactor* を 100 にするのは、データが追加されない読み込み専用テーブルに対してのみ効果があります。

fillfactor 値が 100 より小さい場合 ( 特殊な場合である 0 は除く )、満杯でないページを持つ新しいインデックスが作成されます。大量のデータが記録されることになるテーブルにインデックスを作成する場合は、fillfactor に 10 を選択するのが妥当です。ただし、fillfactor 値を小さくすると、それぞれのインデックス ( またはインデックスとデータ ) にさらに多くの記憶領域が必要になります。

---

**警告 !** fillfactor を使用してクラスタード・インデックスを作成すると、Adaptive Server はクラスタード・インデックスの作成時にデータを再分配するため、データが占有する記憶領域の総量に影響します。

---

#### max\_rows\_per\_page

インデックスのデータ・ページとリーフレベル・ページ上のロー数を制限します。fillfactor とは異なり、max\_rows\_per\_page 値は、sp\_chgattribute を使用して変更されるまで保持されます。

max\_rows\_per\_page の値を指定しなかった場合、テーブルの作成時に値 0 が使用されます。テーブルおよびクラスタード・インデックスの値の範囲は、2KB のページで 0 ~ 183KB、16KB のページで 0 ~ 1486KB です。

ノンクラスタード・インデックスのページあたりの最大ロー数は、インデックス・キーのサイズによって決まります。指定した値が大きすぎると、エラーメッセージが返されます。

max\_rows\_per\_page 値を 0 に設定すると、満杯のページでクラスタード・インデックスが作成され、満杯のリーフ・ページでノンクラスタード・インデックスが作成されます。これによって、クラスタード・インデックスとノンクラスタード・インデックスの両方のインデックス B ツリー内に十分な領域が確保されます。

max\_rows\_per\_page を 1 に設定すると、リーフ・レベルで 1 ページあたりに 1 ローずつのクラスタード・インデックスとノンクラスタード・インデックスの両方が作成されます。値を低くすると、頻繁にアクセスされるデータのロック競合を減らすことができます。ただし、max\_rows\_per\_page 値を小さい値に設定すると、完全に満杯ではないページを持つ新しいインデックスが作成されます。また、使用される記憶領域が増えるため、ページ分割も多くなることがあります。

コンポーネント統合サービスが有効な場合、リモート・サーバに対して max\_rows\_per\_page を使用することはできません。



---

**警告！** `max_rows_per_page` を使用してクラスタード・インデックスを作成すると、Adaptive Server はクラスタード・インデックスの作成時にデータを再分配するため、データが占有する記憶領域の総量に影響することがあります。

---

#### `with reservepagegap = num_pages`

埋められたページと、エクステント I/O 割り付けオペレーション中に残される空ページの比率を指定します。指定した各 `num_pages` につき 1 つの空のページが、今後のインデックスの拡張に備えて残されます。有効な値は 0 ～ 255 です。デフォルトは 0 です。

#### `with consumers`

インデックス作成のためのソート・オペレーションを実行するコンシューマ・プロセスの数を指定します。使用できるワーカー・プロセスの数とデータ・パーティションの数によっては、インデックスのソートに使われるコンシューマ・プロセスの実際数が、ここで指定された数とは異なる場合があります。

#### `ignore_dup_key`

ユニーク・インデックス (クラスタードまたはノンクラスタード) を持つテーブルへの重複キーの入力をキャンセルします。重複キーの `insert` または `update` を実行しても Adaptive Server によってキャンセルされ、情報メッセージが表示されます。このキャンセルの後、重複キーがあるトランザクションは最後まで実行されます。

`ignore_dup_key` が設定されているかどうかに関係なく、重複値や複数の `null` 値があるカラムでユニーク・インデックスを作成することはできません。作成しようとするとき、Adaptive Server によってエラー・メッセージが表示され、重複する最初の値が示されます。Adaptive Server によってユニーク・インデックスがカラムに作成される前に、重複する値を削除してください。

#### `ignore_dup_row`

重複するローがあるテーブル上にユニークでない新しいクラスタード・インデックスを作成できます。`ignore_dup_row` は、テーブルから重複するローを削除し、重複するローを作成する `insert` と `update` をキャンセルしますが、トランザクション全体はロールバックしません。「[重複ロー](#)」(172 ページ) を参照してください。

#### `allow_dup_row`

重複するローがあるテーブル上にユニークでないクラスタード・インデックスを作成できます。また、`update` 文や `insert` 文を使用したローの重複も許可されます。「[重複ロー](#)」(172 ページ) を参照してください。

**sorted\_data**

テーブル内のデータがすでにソート順になっている場合(たとえば、すでにソートされたデータを **bcp** を使用して空のテーブルにコピーしている場合)、クラスタード・インデックスまたはユニークなノンクラスタード・インデックスを作成する処理速度が速くなります。「ソートを高速化するための **sorted\_data** オプションの使用」(174 ページ)を参照してください。

**with statistics using num\_steps values**

クエリの最適化に使用されるヒストグラム用に生成されるステップの数を指定します。この句を省略した場合、次のようになります。

- 先行のインデックス・カラムにヒストグラムが格納されていない場合、デフォルト値は 20 です。
- インデックス・カラムの先行カラムのヒストグラムがすでに存在する場合は、現在のステップ数が使用されます。

**num\_steps** に 0 を指定すると、インデックスは再作成されますが、インデックスの統計はシステム・テーブルで上書きされません。

実際のステップ数は、指定したステップ数と異なる場合があります。**num\_steps** で指定するヒストグラム・ステップを  $M$  とし、**histogram tuning factor** パラメータを  $N$  とすると、実際のステップ数は  $M \sim M*N$  の範囲で、分散時に存在した頻度セル数によって決定します。

**on segment\_name**

指定したセグメントにインデックスを作成します。

**on segment\_name** オプションを使用するには、事前に **disk init** を使用してデバイスを初期化し、**sp\_addsegment** を使用してデータベースにセグメントを追加しておきます。データベースで使用できるセグメント名のリストを生成するには、システム管理者に問い合わせるか、または **sp\_helpsegment** を使用してください。**on segment\_name** は、次の 2 つの場所で使用可能です。

- **index\_partition\_clause** の直前 — セグメントが **index\_partition\_clause** で明示的に定義されていないすべてのパーティションに使用するグローバル・デフォルトを定義します。
- 句自体の中 — 個々のパーティションごとにセグメントを指定できるようになります

両方の場所で **on segment\_name** を使用する例については、例の項を参照してください。

**local index**

セマンティック分割テーブルの場合、ベース・テーブルと常に同等の条件で分割するインデックスを指定します。つまり、テーブルとインデックスは同じ分割キーと分割条件を共有します。ラウンドロビン分割テーブルの場合、ローカル・インデックスとは、テーブルの各インデックス・パーティション内のインデックス・キーが、1つのみのテーブル・パーティション内のデータ・ローを参照することです。

セマンティック分割テーブルとラウンドロビン分割テーブルのいずれの場合も、各テーブル・パーティションにはそれに対応するインデックス・パーティションが1つだけあります。

**partition\_name**

インデックスを保管する新しいパーティションの名前を指定します。テーブル内またはインデックス内にすでに存在するパーティションと同じ名前は指定できません。set quoted\_identifier オプションを設定した場合、パーティション名に区切り識別子を使用できません。それ以外の場合、パーティション名は有効な識別子でなければなりません。

partition\_name を省略すると、table\_name\_partition\_id という形式の名前が作成されます。最大長を超える長さのパーティション名は、トランケートされます。

例

**例 1** authors テーブルの au\_id カラムに au\_id\_ind という名前のインデックスを作成します。

```
create index au_id_ind on authors (au_id)
```

**例 2** authors テーブルの au\_id カラムに au\_id\_ind という名前のユニークなクラスタード・インデックスを作成します。

```
create unique clustered index au_id_ind
on authors (au_id)
```

**例 3** titleauthor テーブルの au\_id カラムと title\_id カラムに ind1 という名前のインデックスを作成します。

```
create index ind1 on titleauthor (au_id, title_id)
```

**例 4** authors テーブルの zip カラムに zip\_ind という名前のノンクラスタード・インデックスを作成し、それぞれのインデックス・ページを4分の1だけデータで埋め、ソートを4つのコンシューマ・プロセスに限定します。

```
create nonclustered index zip_ind
on authors (postalcode)
with fillfactor = 25, consumers = 4
```

**例 5** `pub_id` には昇順で、`pubdate` には降順でインデックスを作成します。

```
create index pub_dates_ix
on titles (pub_id asc, pubdate desc)
```

**例 6** オプティマイザの統計に 50 のヒストグラム手順を使用し、インデックスの 40 ページごとに 1 つの空のページを残して、`title_id` にインデックスを作成します。

```
create index title_id_ix
on titles (title_id)
with reservepagegap = 40,
statistics using 50 values
```

**例 7** ローカルのクラスタード・インデックスを、分割されている `salesdetail` テーブルに作成します。`clust_idx` インデックスには、`salesdetail` のパーティション方式、分割キー、パーティション境界が継承されます。

```
create clustered index clust_idx
on salesdetail (ord_num) local index
```

**例 8** 分割されない、ノンクラスタードのグローバル・インデックスを、分割されている `sales` テーブルに作成します。このテーブルは、`date` カラムの範囲によって分割されています。

```
create nonclustered index global_idx
on sales (order_num)
```

**例 9** 最初に `pback_sales` テーブルを作成し、3 つのデータ・パーティションに分割します。

```
create table pback_sales (c1 int, c2 int,
c3 varchar (20)) partition range (c1)
(p1 c1 values <= (10),
p2 c1 values <= (20),
p3 c1 values <= (MAX))
```

次に、ローカルの関数ベース・インデックスをパーティション `p1` に作成します。

```
create index fc_idx on pback_sales (c1*c2) local index
p1
```

**例 10** 関数ベース・インデックスを作成します。

```
create index sum_sales on mytitles (price * total_sales)
```

**例 11** パーティション名の前と後の両方に `on segment_name` 句を指定します。

```

use tempdb
go
if not exists(select 1 from tempdb..syssegments where name = 'seg1')
    exec sp_addsegment seg1,tempdb,master
go
if not exists(select 1 from tempdb..syssegments where name = 'seg2')
    exec sp_addsegment seg2,tempdb,master
go
if not exists(select 1 from tempdb..syssegments where name = 'seg3')
    exec sp_addsegment seg3,tempdb,master
go
if not exists(select 1 from tempdb..syssegments where name = 'seg4')
    exec sp_addsegment seg4,tempdb,master
go
if exists(select 1 from sysobjects where name = 't1')
    drop table t1
go
create table t1 (a int, b varchar(30)) partition by roundrobin (p1 on seg1, p2 on seg2)
go
create index t1_i1 on t1 (a) local index
go
create index t1_i2 on t1 (a) on seg3 local index ip1 on seg4
go
sp_help t1
go

```

次のように出力されます。

```

Name Owner Object_type Create_date
-----
t1   dbo   user table   Aug 7 2008 11:14AM

(1 row affected)
Column_name Type      Length Prec Scale Nulls Default_name Rule_name
Access_Rule_name Computed_Column_object Identity
-----
a           int        4 NULL  NULL  0 NULL      NULL
NULL                                     0
b           varchar    30 NULL  NULL  0 NULL      NULL
NULL                                     0
Object has the following indexes

index_name index_keys index_description index_max_rows_per_page
index_fillfactor index_reservepagegap index_created index_local
-----
t1_i1      a          nonclustered                0
0
0 Aug 7 2008 11:14AM Local Index
t1_i2      a          nonclustered                0
0
0 Aug 7 2008 11:14AM Local Index

```

## create index

---

```
(2 rows affected)
index_ptn_name  index_ptn_seg
-----
t1_i1_952063116 default
t1_i1_968063173 default
ipl             seg4
t1_i2_1000063287 seg3
```

```
(4 rows affected)
このオブジェクトに定義されたキーはありません。
name type          partition_type partitions partition_keys
-----
t1  base table roundrobin          2 NULL
```

```
(1 row affected)

partition_name partition_id pages row_count segment create_date
-----
p1              920063002    1         0 seg1    Aug 7 2008 11:14AM
p2              936063059    1         0 seg2    Aug 7 2008 11:14AM
```

### Partition\_Conditions

NULL

Avg_pages	Max_pages	Min_pages	Ratio(Max/Avg)	Ratio(Min/Avg)
1	1	1	1.000000	1.000000

Lock scheme Allpages

属性 'exp\_row\_size' は allpages ロック・スキームのあるテーブルには適用できません。

属性 'concurrency\_opt\_threshold' は allpages ロック・スキームのあるテーブル

allpages lock scheme.

exp_row_size	reservepagegap	fillfactor	max_rows_per_page	identity_gap	ascinserts
0	0	0	0	0	0

```
(1 row affected)
concurrency_opt_threshold optimistic_index_lock dealloc_first_txtpg
-----
0                          0                          0
```

```
(1 row affected)
(return status = 0)
```

### 使用方法

- テーブルにデータを追加してインデックスのキーの配置を変える場合は、定期的に [update statistics](#) を実行してください。クエリ・オプティマイザは、[update statistics](#) によって作成された情報を基にしてテーブル上でクエリを実行する最良のプランを選択します。

- ノンクラスタード・インデックスを作成するときにテーブルにデータがある場合、Adaptive Server は新しいインデックス上で `update statistics` を実行します。クラスタード・インデックスを作成するときにテーブルにデータがある場合は、Adaptive Server はすべてのテーブルのインデックス上で `update statistics` を実行します。
- ジョインで定期的に変更されるすべてのカラムにインデックスを作成します。
- コンポーネント統合サービスが有効な場合は、`create index` コマンドが再構築され、テーブルに対応した Adaptive Server に直接渡されます。
- 仮想ハッシュ・テーブルを含むセグメントで `create index` (クラスタードまたはノンクラスタード) を使用することはできません。仮想ハッシュ・テーブルは他のテーブルやデータベースと共有できない排他セグメント 1 つのみを必要とするためです。
- `online` パラメータを指定して、同時に `writetext` を実行できます。

#### インデックスとストアド・プロシージャの作成

`create index` 文の実行後、Adaptive Server によってストアド・プロシージャが自動的に再コンパイルされます。`create index` を実行する前に開始した特別なクエリは引き続き動作しますが、新しいインデックスを利用することはできません。

Adaptive Server バージョン 12.5 以前では、`create index` はキャッシュされたストアド・プロシージャによって無視されていました。

#### 効率的なインデックス作成

- インデックスによってデータの検索は速くなりますが、データの更新は遅くなります。セグメントが別々の物理デバイスにある場合に、あるセグメントにテーブルを 1 つ作成し、そのノンクラスタード・インデックスを別のセグメントに作成すると、パフォーマンスが向上します。
- テーブルが分割されていて、サーバが並列処理の可能な構成になっている場合、Adaptive Server は並列にインデックスを作成できます。また、Adaptive Server は、ソート・バッファを使用してソートに必要な I/O の量を減らすことができます。『パフォーマンス&チューニング・ガイド：オプティマイザと抽象プラン』の「並列ソート」を参照してください。

- クラスタード・インデックスが作成されると、ノンクラスタード・インデックスが自動的に再構築されるため、ノンクラスタード・インデックスを作成する場合は、事前にクラスタード・インデックスを作成しておいてください。
- データオンリーロック・テーブルに並列ソートを使用しているときは、空のテーブルの場合でも、ワーカー・プロセスの数をパーティションの数以上に設定してください。また、データベース・オプション `select into/bulkcopy/pilsort` も有効にする必要があります。

#### クラスタード・インデックスの作成

- テーブルは、そのテーブルのクラスタード・インデックスを「追跡」します。テーブルを作成してから `on segment_name` 拡張機能を使用して `create clustered index` を実行すると、テーブルはインデックスが作成されたセグメントに移行します。

特定のセグメントにテーブルを作成し、セグメントを指定しないでクラスタード・インデックスを作成すると、Adaptive Server はデフォルト・セグメントにテーブルを移動して、そこでクラスタード・インデックスを作成します。

`text`、`unitext`、`image` の各データはそれぞれ別のページ・チェーンに保管されているため、`on segment_name` を使用してクラスタード・インデックスを作成しても、`text` カラムと `image` カラムが移動されることはありません。

- クラスタード・インデックスを作成するために、Adaptive Server は既存のデータを複製します。インデックスが完了すると、サーバが元のデータを削除します。クラスタード・インデックスを作成する前に、`sp_spaceused` を使用して、データベースに使用できる空き領域が少なくともテーブルのサイズの 120 パーセントはあることを確認してください。
- クラスタード・インデックスは、通常、テーブルのプライマリ・キー（ローをユニークに識別する 1 つまたは複数のカラム）に作成します。プライマリ・キーは、`sp_primarykey` を使用して、データベースに記録できます（フロントエンド・プログラムや `sp_depends` で使用するため）。
- クラスタード・インデックスで重複するローを許可するには、`allow_dup_row` を指定してください。

#### 圧縮テーブルのインデックスの作成

- Adaptive Server では、テーブルにソートが必要な場合、データ・コピー・オペレーション中に圧縮可能なローをすべて圧縮します。



- `sorted_data` を使用してクラスタード・インデックスが作成されている場合は、データ・コピーは実行されず、クラスタード・インデックスの構築中にデータ・ローが圧縮されることはありません。
- `Adaptive Server` では、インデックス・キー値の圧縮は行われません。データ・ローの値のみが圧縮されます。
- キー・カラムが圧縮されている場合でも、インデックス・キー・カラムを選択し、一意のインデックスを作成できます。ノンクラスタードのユニーク性チェックを実行するには、インデックス・ページで非圧縮インデックス・キーを確認します。
- `Adaptive Server` では非圧縮インデックス・キーおよびローのフォーマットを使用して、`ignore_dup_key`、`ignore_dup_row`、`allow_dup_row` に対するサポートを確認します。
- `Adaptive Server` ではローレベルの圧縮に対してのみ `fillfactor` パラメータを適用します。
- `Adaptive Server` では、`fillfactor` パラメータを全ページロック・クラスタード・インデックスのデータ・ページに適用するときに、以下を考慮します。
  - 最終的な圧縮ロー・フォーマット
  - 圧縮に必要な領域

`Adaptive Server` では、後続のページ圧縮オペレーションに使用されるページ領域をさらに圧縮することがあり、それによって `fillfactor` 値が低下します。

- `Adaptive Server` ではページ・レベルで `respagegap` パラメータを適用するので、圧縮による影響を受けることはありません。
- `max_rows_per_page` にはページのデータ・ローのみが含まれており、隠しページ辞書、インデックス、文字コード化のエントリは含まれていません。

#### 暗号化カラムのインデックスの作成

暗号化キーで初期化ベクトルまたはランダム埋め込みが指定されていない場合、暗号化カラムにインデックスを作成できます。暗号化カラムのインデックスは、等しいか等しくないかを照合するときは役立ちますが、大文字小文字を区別しないデータの一致やデータの範囲検索には役立ちません。

等価および不等価探索やジョインでのパフォーマンスを向上させるには、暗号化カラムでインデックスを作成します。

`create index` で、以下を作成すると、エラー・メッセージが表示されます。

- 暗号化カラムを参照している式を使用した関数インデックス
- 初期化ベクトルまたはランダム埋め込みによって暗号化されたカラムでのインデックス

---

**注意** 関数インデックスの式には、暗号化カラムを使用できません。

---

### インデックスの昇順または降順の指定

インデックス・カラムの名前の後に `asc` または `desc` キーワードを使用して、インデックス・キーのソート順を指定します。カラムが、クエリの `order by` 句で指定された順序と同じ順序になるようにインデックスを作成すると、クエリ処理中のソートする手順を省略できます。『パフォーマンス&チューニング・シリーズ：ロック』の「インデックスとパフォーマンス」を参照してください。

### インデックスに必要な空き領域

- 領域は、一度に 1 エクステント、すなわち 8 ページ単位でテーブルおよびインデックスに割り付けられます。エクステントが満杯になるたびに、別のエクステントが割り付けられます。割り付けられた領域とインデックスに使用されている領域の量を確認するには、`sp_spaceused` を使用してください。
- 場合によっては、`sorted_data` オプションを使用することで、[表 1-6 \(175 ページ\)](#) に示されているようにデータ・ローのコピーを省略できます。その場合、インデックス構造のための追加領域のみが確保されれば十分です。キーのサイズによって異なりますが、この領域は通常、テーブルのサイズの約 20 パーセントです。

### 重複ロー

- `ignore_dup_row` オプションおよび `allow_dup_row` オプションは、ユニークでないノンクラスタード・インデックスを作成する場合は関係しません。Adaptive Server は内部でそれぞれのノンクラスタード・インデックスにユニークなロー ID 番号を付けるため、データ値が同じ場合でも、重複するローについては問題ありません。
- `ignore_dup_row` と `allow_dup_row` は同時には使用できません。
- 全ページロック・テーブルで、ユニークでないクラスタード・インデックスを作成した場合、重複キーは作成できますが、重複ローは `allow_dup_row` オプションを指定しないかぎり作成されません。この動作は、データオンリーロック・テーブルでは異なります。詳細については、[表 1-4](#) を参照してください。

- `allow_dup_row` を使用して、ユニークでないクラスタード・インデックスを重複するローがあるテーブルに作成できます。`allow_dup_row` オプションを使用せずに作成したユニークでないクラスタード・インデックスがテーブルにある場合は、`insert` コマンドまたは `update` コマンドを使用して新しい重複するローを作成することはできません。

テーブルのインデックスにユニークなものがある場合は、そのユニーク性の稼働条件の方が `allow_dup_row` オプションよりも優先されます。テーブルのいずれかのカラムにユニーク・インデックスがある場合は、`allow_dup_row` オプションを使用してもインデックスを作成することはできません。

- また、`ignore_dup_row` オプションも、ユニークでないクラスタード・インデックスに使用できます。`ignore_dup_row` オプションは、データのバッチから重複するローを削除します。`ignore_dup_row` は、重複するローを作成する `insert` または `update` をすべてキャンセルしますが、トランザクション全体はロールバックしません。
- 表 1-4 は、`allow_dup_row` と `ignore_dup_row` が、重複するローがあるテーブルにユニークでないクラスタード・インデックスを作成しようとするときや、テーブルに重複するローを入力しようとするときにどのような影響を与えるかを示します。

表 1-4: ユニークでないクラスタード・インデックスにおける重複ローのオプション

オプションの設定	重複するローがあるテーブルへのインデックスの作成	インデックス付きのテーブルへの重複するローの挿入
どのオプションも指定しない場合	<code>create index</code> は失敗する。	<code>insert</code> は失敗する。
<code>allow_dup_row</code> を設定	<code>create index</code> は完了する。	<code>insert</code> は完了する。
<code>ignore_dup_row</code> を設定	インデックスは作成されるが、重複ローは削除され、エラー・メッセージが表示される。	重複ロー以外のローがすべて挿入され、エラー・メッセージが表示される。

表 1-5 は、さまざまなインデックスで使用できるインデックス・オプションを示します。

表 1-5: インデックス・オプション

インデックス・タイプ	オプション
クラスタード	ignore_dup_row   allow_dup_row
ユニークなクラスタード	ignore_dup_key
ノンクラスタード	なし
ユニークなノンクラスタード	ignore_dup_key

### インデックスの代わりとしての一意性制約の使用

- `create index` の代わりに、`create table` 文または `alter table` 文を使用して一意性制約を指定すると、暗黙的にユニーク・インデックスを作成できます。一意性制約によって、テーブルのカラムにクラスタード・インデックスまたはノンクラスタード・ユニーク・インデックスが作成されます。このように暗黙的に作成されたインデックスは制約の名前に従って命名され、`create index` を使用して作成されたインデックスと同じ規則に従います。
- 一意性制約が指定されたインデックスの削除に `drop index` 文を使用することはできません。`alter table` 文を使用してその制約を削除するか、またはそのテーブルを削除すると、インデックスは削除されます。ユニーク制約の詳細については、「`create table`」を参照してください。

### ソートを高速化するための `sorted_data` オプションの使用

- `sorted_data` オプションは、ソート手順を省略したり、場合によってはデータ・ローを新しいページにコピーする手順を省略することによって、インデックスの作成に必要な時間を短縮できます。その処理速度は大きなテーブルほど速くなり、1GB を超えるテーブルでは数倍も速くなります。

`sorted_data` を指定しても、データがソート順になっていない場合は、Adaptive Server によってエラー・メッセージが表示され、コマンドは正常に動作しません。

重複キーを持つローがなければ、ユニークでないノンクラスタード・インデックスの作成は完全に実行されます。重複キーを持つローがあるときは、Adaptive Server によってエラー・メッセージが表示され、コマンドは正常に動作しません。

- クラスタード・インデックスを作成するときの `sorted_data` の影響は、テーブルが分割されているかどうか、`create index` コマンドの中で特定の他のオプションが使用されているかどうかによって異なります。オプションの中には、分割されていないテーブルのデータのコピーと、分割されているテーブルのソートおよびデータのコピーを常に必要とするものもあります。またその他では、以下を使用する場合にかぎって、データのコピーを必要とします。
  - `ignore_dup_row` オプション
  - `fillfactor` オプション
  - テーブル・データが配置されているセグメントとは別のセグメントを指定するための `on segmentname` 句
  - テーブルと対応している値とは別の値を指定するための `max_rows_per_page` 句
- 表 1-6 は、分割されたテーブルと分割されていないテーブルに対して、いつソートが必要となり、いつテーブルのコピーが行われるかを示します。

表 1-6: `sorted_data` オプションを使用してクラスタード・インデックスを作成する

オプション	分割されたテーブル	分割されていないテーブル
どのオプションも指定しない場合	並列ソートは、ラウンドロビン分割テーブルにクラスタード・インデックスを作成し、データをコピーし、分割に均等に分配し、インデックス・ツリーを作成する場合にのみ必要。	並列ソートまたは非並列ソートがデータをコピーし、インデックス・ツリーを作成する。
<code>with sorted_data</code> のみまたは <code>with sorted_data on same_segment</code>	インデックス・ツリーのみ作成する。データのソートまたはコピーは実行しない。また、並列実行は行わない。	インデックス・ツリーのみ作成する。データのソートまたはコピーは実行しない。また、並列実行は行わない。
<code>with sorted_data</code> および <code>ignore_dup_row</code> または <code>fillfactor</code> または <code>on other_segment</code> または <code>max_rows_per_page</code>	並列ソートは、データをコピーし、分割に均等に分配し、インデックス・ツリーを作成する。	データをコピーし、インデックス・ツリーを作成する。データのソートを実行しない。並列実行は行わない。

#### ヒストグラムのステップ数の指定

- インデックスの先行カラムのヒストグラムにステップ数を指定するには、`with statistics` 句を使用します。ヒストグラムは、クエリを最適化するときを使用され、カラムの探索指数に一致するローの数を特定します。

- カラムの `sysstatistics` の値を更新しないでインデックスを再作成するには、ステップ数に `0` を使用します。こうすると、`optdiag` で変更された統計値が上書きされなくなります。
- `histogram_tuning_factor` パラメータに値を指定した場合、`create index` では分離された頻度セルの数に応じて `20` ~ `M*20` ステップの間の数を使用されます。デフォルトは `20` ですが、`using step values` オプションを使用して別の数を指定できます。

#### 記憶領域管理プロパティ

- `fillfactor`、`max_rows_per_page`、`reservepagegap` は、次のようなさまざまな面でインデックス・ページの記憶領域の管理に役立ちます。
  - `fillfactor` は、すべてのロック・スキームのインデックスに適用される。全ページロック・テーブルのクラスタード・インデックスの場合は、テーブルのデータ・ページに影響する。その他のインデックスでは、インデックスのリーフ・レベルに影響する。
  - `max_rows_per_page` は、全ページロック・テーブルのインデックス・ページにのみ適用される。
  - `reservepagegap` は、すべてのロック・スキームのテーブルとインデックスに適用される。
- `reservepagegap` は、次の場合にインデックスの領域使用量に影響します。
  - インデックスの作成時
  - インデックスでの `reorg` コマンドの実行時
  - クラスタード・インデックス作成後の、ノンクラスタード・インデックスの再構築時
- `create clustered index` コマンドで `reservepagegap` の値が指定された場合、この値は次の対象に適用されます。
  - 全ページロック・テーブルのデータとインデックス・ページ
  - データオンリーロック・テーブルのインデックス・ページのみ
- `num_pages` 値は、新しい領域の必要性に応じて既存のページに近い領域がインデックスによって割り付けられるように、インデックスのリーフ・レベルでの充填ページと空のページの比率を指定します。たとえば、`reservepagegap` を `10` に設定すると、使用ページ `9` ページごとに `1` つの空のページが残されます。

- 全ページロック・テーブルに対して `create clustered index` を `reservepagegap` とともに指定すると、以前 `create table` または `alter table` によって指定された値がすべて上書きされます。
- インデックスの記憶領域管理プロパティは、`sp_chgattribute` を使用して変更できます。`sp_chgattribute` を使用してプロパティを変更しても、テーブルのインデックスの記憶領域にはすぐに影響しません。`reorg rebuild` など、以後の大規模な割り付けを実行するときは、`sp_chgattribute` 値を使用します。
- `sp_chgattribute` によって設定された `fillfactor` 値は、`sysindexes` の `fill_factor` カラムに格納されます。`alter table...lock` コマンドまたは `reorg rebuild` コマンドの結果としてインデックスが再作成されるときは、`fillfactor` が適用されます。

### インデックスのオプションとロック・モード

表 1-7 は、全ページロック・テーブルとデータオンリーロック・テーブルでサポートされているインデックス・オプションを示します。データオンリーロック・テーブルでは、`create index` の実行中に `ignore_dup_row` と `allow_dup_row` オプションが実行されますが、`insert` 操作と `update` 操作の際は実行されません。データオンリーロック・テーブルでは、常に重複するローの挿入が可能です。

表 1-7: ロック・スキームでサポートされている `create index` オプション

インデックス・タイプ	全ページロック・テーブル	データオンリーロック・テーブル	
		インデックスの作成中	挿入の実行中
クラスタード	<code>allow_dup_row</code> 、 <code>ignore_dup_row</code>	<code>allow_dup_row</code> 、 <code>ignore_dup_row</code>	<code>allow_dup_row</code>
ユニーク・クラスタード	<code>ignore_dup_key</code>	<code>ignore_dup_key</code>	<code>ignore_dup_key</code>
ノンクラスタード	なし	なし	なし
ユニーク・ノンクラスタード	<code>ignore_dup_key</code>	<code>ignore_dup_key</code>	<code>ignore_dup_key</code>

表 1-8 は、クラスタード・インデックスが指定されたテーブルに重複するローを挿入しようとするコマンドの動作と、クラスタード・インデックスがいつ削除され、再作成されるかを示します。

表 1-8: 重複するローのオプションの実行とエラー

オプション	全ページロック・テーブル	データオンリーロック・テーブル
どのオプションも指定しない場合	挿入は失敗し、エラーメッセージ 2615 が表示される。インデックスの再作成は正しく行われる。-	挿入は正常に行われる。インデックスの再作成は失敗し、エラーメッセージ 1508 が返される。-

オプション	全ページロック・テーブル	データオンリーロック・テーブル
allow_dup_row	挿入とインデックスの再作成が正常に行われる。	挿入とインデックスの再作成が正常に行われる。
ignore_dup_row	挿入は失敗し、「重複ローは無視されました。」というメッセージが表示される。インデックスの再作成は正常に行われる。	挿入は正常に行われる。インデックスの再作成によって、重複するローが削除される。

#### データオンリーロック・テーブルでの `sorted_data` オプションの使用

- `create index` の `sorted_data` オプションは、空のテーブルへのバルク・コピー操作の直後にのみ使用できます。テーブルのデータ変更によって追加のページ割り付けが発生すると、`sorted_data` オプションは使用できなくなります。
- 異なる値を記憶領域管理プロパティに指定すると、`sorted_data` のソート抑制機能が無効になることがあります。

#### テーブルとインデックスについての情報の取得

- 複合インデックスのあるインデックスは、それぞれ `sysindexes` の 1 つのローで表されます。
- インデックスによって検索されるデータの順序と、Adaptive Server にインストールされているソート順の作用については、「`order by` 句」を参照してください。
- テーブルのインデックスに関する情報を取得するには、`sp_helpindex` を実行してください。インデックス・パーティションに関する情報は、`sp_helppartitions` を実行して取得することもできます。
- 各インデックス・パーティションとデータ・パーティションは、`syspartitions` 内の 1 つのローで表されます。

#### 計算カラムへのインデックスの作成

- 実体化された計算カラムは、通常のカラムですが、インデックス・キーとして使用できます。
- 仮想カラムを実体化されたカラムに変換してインデックスを作成するには、`alter table modify` に `materialized` オプションを指定して実行してから、`create index` を実行します。
- インデックス・キーとして使用する計算カラムは `deterministic` である必要はありませんが、`nondeterministic` なカラムは参照元のクエリに影響を与える可能性があることに注意してください。



### 分割されたインデックスの作成

- ローカル・インデックスは、ベース・テーブルのパーティション方式、パーティション・カラム、およびパーティション境界（範囲パーティションおよびリスト・パーティションの場合）を継承します。
- Adaptive Server はローカル・インデックスを保持し、ベース・テーブルが別の分割キーで再分割された場合にローカル・インデックスを再作成します。
- Adaptive Server は以下をサポートします。

インデックス・タイプ	テーブルのタイプ
ローカルのクラスタードおよびノンクラスタードの分割インデックス	分割されたテーブル
グローバルの分割されていないクラスタード・インデックス	ラウンドロビン分割テーブル
グローバルの分割されていないノンクラスタード・インデックス	すべての分割されたテーブル

- 範囲分割テーブル、ハッシュ分割テーブル、およびリスト分割テーブルの場合、クラスタード・インデックスは常にローカルです。"local index" が構文に含まれているかどうかに関係なく、ローカルのクラスタード・インデックスが作成されます。

### 関数ベース・インデックスの作成

- インデックスは、式で直接作成できます。
- 式は **deterministic** でなければなりません。
- Adaptive Server では、式インデックス・キーの **deterministic** プロパティが検証されないため、ユーザはこのプロパティを手動で管理する必要があります。このプロパティが変更されると、予想しない結果になります。
- 関数ベース・インデックスは常に **deterministic** でなければならぬため、結果は事前に評価され、再評価なしで再利用されます。すべての関数ベースのインデックス・キーは **deterministic** であることが前提とされ、クエリで参照されたときに事前に評価された値が使用されます。この値が再評価されるのは、そのベース・カラムの値が変更されたときだけです。
- 1つのインデックスには、複数の関数ベース・インデックス・キー、または関数ベース・インデックス・キーと通常のカラムの組み合わせが存在できます。

- インデックス・キーとして使用する式は **deterministic** でなければなりません。式キーは、計算カラム・インデックス・キーとは異なります。計算カラム・インデックス・キーは一度だけ評価すればよく、**deterministic** プロパティは不要です。しかし、式は、クエリ内で指定されるたびに再評価される必要があります、常に同じ結果を返す必要があります。
- 関数ベースのインデックスによって参照されるユーザ定義関数が削除されたり、無効になったりすると、その関数を呼び出すオペレーションはすべて失敗します。
- **Adaptive Server** はクラスタード関数ベース・インデックスをサポートしません。
- **sorted\_data** オプションを指定して関数ベースのインデックスを作成することはできません。
- インデックス・キーを式に作成すると、後続のクエリでは、その式がインデックス・キーの作成に使用した式と正確に同じである場合にのみ、式がインデックス・キーとして認識されます。
- ベース・カラムに対して **insert**、**delete**、**update** オペレーションを行うと、関数ベースのインデックス・キーは常に自動的に更新されます。

## 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

## パーミッション

**create index** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であるか <b>create any index</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者であるか <b>sa_role</b> が付与されたユーザであることが必要。 <b>create index</b> パーミッションは、デフォルトではテーブル所有者にあり、譲渡することはできない。

## 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
104	create	create index	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – インデックスの名前</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter table](#), [create table](#), [drop index](#), [insert](#), [order by 句](#), [set](#), [update](#)  
システム・プロシージャ [sp\\_addsegment](#), [sp\\_chgattribute](#),  
[sp\\_helpcomputedcolumn](#), [sp\\_helpindex](#), [sp\\_helpsegment](#), [sp\\_spaceused](#)  
ユーティリティ [optdiag](#)

## create login

**説明** ログイン・アカウントを作成して、アカウントに割り当てるパスワード、アカウントのログイン・プロファイル、ユーザ提供のパラメータを指定します。

**構文** create login *login\_name* with [encrypted]  
password *password*  
[*attribute\_value\_pair\_list*]

**パラメータ** *login\_name*  
作成するログイン・アカウントの名前を指定します。先頭の文字は英字とし、長さは 30 文字以内とします。

**with encrypted**  
新しいログイン・アカウントの暗号化パスワードを指定します。

**password *passwordValue***  
新しいログイン・アカウントのパスワードを指定します。

***attribute\_value\_pair\_list***  
ログイン・アカウントに追加する属性および対応する値のリストです。*attribute\_value\_pair\_list* は、属性名と値です。以下のパラメータを 1 つ以上指定する必要があります。

パラメータ	パラメータ値	説明
login profile	有効な値 : <ul style="list-style-type: none"> <li>login_profile_name</li> <li>ignore</li> </ul>	<ul style="list-style-type: none"> <li>login_profile_name は、指定されたログイン・プロファイルを指定されたログイン・アカウントにバインドします。</li> <li>ignore は、ログイン・プロファイルのバインドをすべて消去する。デフォルト・ログイン・プロファイルは適用されず、属性はリリース 15.7 以前と同様に適用されます。</li> </ul> <p>ログイン・プロファイルが指定されない場合は、デフォルト・ログイン・プロファイルが適用されます。『セキュリティ管理ガイド』の「ログイン・プロファイル属性とパスワード・ポリシー属性の適用」を参照。</p>
suid	有効な値 : [-2, -1, 0, 1, 2] を除く [-32768, 2147483647] の範囲のユニーク値。	デフォルトでは suid が生成され、作成時にログイン・アカウントに自動的に割り当てられます。
fullname	<i>name_value</i>	ログイン・アカウントを所有するユーザのフル・ネームです。 デフォルトは NULL です。

パラメータ	パラメータ値	説明
login script	<i>login_script_name</i>	有効なストアド・プロシージャを指定します。ログイン・スクリプトの場合、120 文字に制限されます。
password expiration	有効値の範囲：0 ～ 32767 日。	パスワード有効期間です。デフォルトは 0 で、パスワードの期限は切れないことを意味します。
min password length	有効値の範囲：0 ～ 30。	必要な最小パスワード長です。デフォルトは 6 です。
max failed attempts	有効値の範囲：-1 ～ 32767。	許容されるログインの試行回数。この回数を超えるとアカウントはロックされます。 -1 は、失敗回数は追跡されるがロックはされないことを示します。 デフォルトは 0 で、失敗回数は追跡されず、ログインの失敗によりアカウントがロックされることはないことを意味します。
default database	default_database_name	デフォルトとなるデータベースを指定します。デフォルトは Master です。
default language	default_language	デフォルトとなる言語を指定します。デフォルトは us_english です。
authenticate with	有効な値：ASE、LDAP、PAM、KERBEROS、ANY	ログイン・アカウントの認証に使用するメカニズムを指定します。 ANY を使用すると、Adaptive Server によって定義済みの外部認証メカニズムがあるかどうか調べられます。定義済みの外部認証メカニズムが検出された場合は、それが使用され、それ以外の場合は、ASE メカニズムが使用されます。 authenticate with authentication mechanism を指定しない場合、ログイン・アカウントには ANY が使用されます。
exempt inactive lock	有効な値：TRUE または FALSE。	非アクティブなログイン・アカウントに対してロックの適用を除外するかどうかを指定します。デフォルトは FALSE で、アカウントは適用除外の対象ではないことを示します。

## 例

**例 1** パスワード `itsA8secret` を使用してログイン・アカウントを作成し、ログイン・プロファイル `emp_lp` とサーバ・ユーザ ID 7 を適用します。また、非アクティブのためにアカウントがロックされないことを指定します。

```
create login ravi with password itsA8secret login profile
emp_lp suid 7 exempt inactive lock true
```

## 使用法

- 異なるログイン・プロファイルから属性が取得された場合や、`sp_passwordpolicy` を使用して値が指定された場合、ログイン・アカウント属性の適用方法は優先度の規則によって決定されます。
- 管理を簡単にするために、すべてのユーザの **Adaptive Server** のログイン名をオペレーティング・システムのログイン名と同じにすることを強くおすすめします。こうすることによって、オペレーティング・システムと **Adaptive Server** 間で、監査データを簡単に関連付けることができます。この推奨方法を実行しない場合は、オペレーティング・システムとサーバ・ログイン名の間に対応関係を示す記録を残しておいてください。

## 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

## パーミッション

`create login` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage any login</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sso_role</code> が付与されたユーザであることが必要。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
103	<code>login_admin</code>	<code>create login</code>	キーワードの内容は次のとおりです。 <code>MODIFY attribute_value_pair_list</code>

## 参照

**コマンド** `create login profile`、`alter login`、`alter login profile`、`drop login`、`drop login profile`

**マニュアル** ログイン・アカウントの作成の詳細については、『セキュリティ管理ガイド』を参照してください。優先度の規則については、『セキュリティ管理ガイド』の「ログイン・プロファイル属性とパスワード・ポリシー属性の適用」を参照してください。

**関数** `lprofile_id`、`lprofile_name`

**システム・プロシージャ** `sp_passwordpolicy`、`sp_displaylogin`、`sp_displayroles`、`sp_locklogin`

## create login profile

説明	指定した属性でログイン・プロファイルを作成します。
構文	<pre>create login profile <i>login_profile_name</i> [ as default ]     [ with { attributes from <i>login_name</i>   <i>attribute_value_pair_list</i> } ]</pre>
パラメータ	<p><i>login_profile_name</i> 作成するログイン・プロファイルの名前を指定します。</p> <p><b>as default</b> 作成されたログイン・プロファイルが sa および probe 以外のすべてのログイン・アカウントのデフォルトとして設定します。</p> <p><b>with attributes from <i>login_name</i>   <i>attribute_value_pair_list</i></b> <i>login_name</i> が指定されると、指定されたログイン・アカウントからの属性値を使用してログイン・プロファイルを作成します。 <i>attribute_value_pair_list</i> は、属性名および対応する値です。1 つまたは複数の以下の属性と値を指定します。</p> <ul style="list-style-type: none"> <li>• <b>default database <i>default_database_name</i></b> — デフォルト・データベースを指定します。デフォルトは Master です。</li> <li>• <b>default language <i>default_language</i></b> — デフォルト言語を指定します。デフォルトは us_english です。</li> <li>• <b>login script <i>login_script_name</i></b> — 有効なストアド・プロシージャを指定します。ログイン・スクリプトの場合、120 文字に制限されます。</li> <li>• <b>authenticate with</b> — ログイン・アカウントの認証に使用するメカニズムを指定します。有効な値：ASE、LDAP、PAM、KERBEROS、ANY</li> </ul> <p>ANY を使用すると、Adaptive Server によって定義済みの外部認証メカニズムがあるかどうか調べられます。定義済みの外部認証メカニズムが検出された場合は、それが使用され、それ以外の場合は、ASE メカニズムが使用されます。</p> <p><b>authenticate with <i>authentication mechanism</i></b> を指定しない場合、ログイン・アカウントには ANY が使用されます。</p> <ul style="list-style-type: none"> <li>• <b>track lastlogin</b> — 最終ログイン更新を有効にします。有効な値：TRUE または FALSE。デフォルトは TRUE で、更新を意味します。</li> <li>• <b>stale period</b> — ログインアカウントがロックされるまでに非アクティブであることができる期間を示します。有効な値は 1 ～ 32767 日。持続時間：D (日数)、W (週数)、M (月数)、Y (年数)。デフォルトは D (日数) です。</li> </ul>

- **profile id** — ID 領域をログイン・アカウントのサーバ・ユーザ ID (suid) と共有します。デフォルトでは **login profile ID** が生成され、作成時にログイン・プロファイルに自動的に割り当てられます。

有効値は、ログイン・アカウントとログイン・プロファイル間で一意の値です。範囲：[-32768, 2147483647]。除外値：-2、-1、0、1、2

**例** **例 1** ログイン・プロファイルを作成します。設定されない属性値は、優先度ルールに従います。

```
create login profile eng_lp
```

詳細については、『セキュリティ管理ガイド』の「ログイン・プロファイル属性とパスワード・ポリシー属性の適用」を参照してください。

**例 2** ログイン・プロファイルを作成し、ログイン属性値をログイン・アカウント ravi から新しいログイン・プロファイル ravi\_lp に転送します。設定されない属性値は、優先度ルールに従います。

```
create login profile ravi_lp with attributes from ravi
```

**例 3** 認証方法 ASE を含むログイン・プロファイル sa\_login\_profile を作成します。

```
create login profile sa_login_profile with authenticate with ASE
```

**使用法** 異なるログイン・プロファイルから属性が取得された場合や、sp\_passwordpolicy を使用して値が指定された場合、ログイン・アカウント属性の適用方法は優先度の規則によって決定されます。

**標準規格** ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

**パーミッション** create login profile のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

細密なパーミッションが有効の場合、manage any login profile 権限を持つユーザであることが必要。

細密なパーミッションが無効

細密なパーミッションが無効の場合、sso\_role が付与されたユーザであることが必要。

**監査** sysaudits の event カラムと extrainfo カラムの値は次のとおりです。



イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
137	security_profile	create login profile	キーワードは DEFAULT を含む ログイン・プロファイルがデフォルトになる場合： {attributes from login_name   attribute_value_pair_list}

## 参照

**コマンド** create login、alter login、alter login profile、drop login、drop login profile

**マニュアル** ログイン・プロファイルの作成、ログイン時のログイン・スクリプトの呼び出し、優先度ルールの詳細については、『セキュリティ管理ガイド』を参照してください。

**関数** lprofile\_id、lprofile\_name

**システム・プロシージャ** sp\_passwordpolicy、sp\_displaylogin、sp\_displayroles、sp\_locklogin

## create plan

**説明** 抽象プランを作成します。

**構文**

```
create plan query plan
    [into group_name]
    [and set @new_id]
```

**パラメータ** *query*

クエリの **SQL** テキストを含む文字列リテラル、パラメータ、またはローカル変数です。

*plan*

抽象プランの式を含む文字列リテラル、パラメータ、またはローカル変数です。

into *group\_name*

抽象プラン・グループの名前を指定します。

and set @*new\_id*

変数内の抽象プランの ID 番号を返します。

**例** **例 1** 指定したクエリの抽象プランを作成します。

```
create plan "select * from titles where price > $20" " (t_scan titles)"
```

**例 2** *dev\_plans* グループのクエリの抽象プランを作成し、変数 @*id* のプラン ID を返します。

```
declare @id int
create plan "select au_fname, au_lname from authors
where au_id = '724-08-9931' "
" (i_scan au_id_ix authors)"
into dev_plans
and set @id
select @id
```

**使用法**

- `create plan` は、`into` で指定されたグループに抽象プランを保存します。グループ名が指定されていないと、プランは現在アクティブなプラン・グループに保存されます。
- `create plan` で指定されたクエリと抽象プランは、**SQL** 構文としての有効性が検査されず、プランは抽象プランの構文としての有効性が検査されません。また、プランが **SQL** テキストと互換性を持つかどうかも確認されません。`create plan` を使用してプランを作成した場合は必ず、`create plan` 文で指定したクエリを実行して正当性をすぐに検査してください。

- グループの別のクエリ・プランに同じ SQL テキストがある場合は、`set plan replace on` を使用して `replace` モードを有効にしてください。そうしないと、`create plan` コマンドは正常に動作しません。
- `@new_id` は、宣言してから `and set` 句で使用してください。
- `into` で指定する抽象プラン・グループは、既存のグループである必要があります。
- `sp_configure "enable literal autoparam", 1` システム・プロシージャを使用してサーバ全体でリテラルのパラメータ化を有効化している場合、`Adaptive Server` では抽象プラン作成時に指定したリテラルのパラメータ化がすべて無視されます。

## 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

## パーミッション

`create plan` の使用にパーミッションは必要ありません。

## 参照

コマンド `set plan`

マニュアル 『パフォーマンス&amp;チューニング・ガイド：オプティマイザと抽象プラン』

システム・プロシージャ `sp_add_qpgroup`、`sp_configure "enable literal autoparam"`、`sp_find_qplan`、`sp_help_qplan`、`sp_set_qplan`

## create precomputed result set

**説明** 事前計算済み結果セットとその管理に必要なポリシーを作成します。

**構文**

```
create {precomputed result set | materialized view}
    [owner_name.]prs_name [(alternative_column_name
    [[constraint constraint_name]
        unique (column_name,...)]
    [(immediate | manual) refresh]
    [{populate | nopopulate}]
    [enable | disable]
    [{enable | disable } use in optimization]
    [lock { datarows | datapages | allpages}]
    [on segment_name]
    [partition_clause]
```

as query\_expression

### パラメータ

precomputed result set | materialized view

マテリアライズされたビューまたは事前計算済み結果セットを作成します。

*prs\_name*

事前計算済み結果セットの名前。完全修飾された *prs\_name* にサーバまたはデータベースの名前を含めることはできません。

*alternative\_column\_name*

事前計算済み結果セット・カラムの名前を示します。

constraint *constraint\_name*

整合性制約の名前を導入します。この名前は識別子の規則に従い、データベース内でユニークである必要があります。

一意性制約またはプライマリ・キー制約の名前を指定しないと、Adaptive Server は *tablename\_colname\_tabindid* という形式の名前を生成します。*tabindid* はテーブル ID とインデックス ID を連結した文字列です。

unique (*column\_name*,...)

指定されたカラムの値を制限し、2つのローが同じ値を持たないようにします。

**{immediate | manual} refresh**

リフレッシュ・ポリシーを次のいずれかから決定します。

- **Immediate** — (デフォルト) 基本テーブルを更新するトランザクションと同じトランザクションで事前計算済み結果セットを更新します。
- **manual** — 事前計算済み結果セットを明示的に更新します。**manual** パラメータを使用すると、明示的に **refresh** を発行するまで基本テーブルの更新が事前計算済み結果セットに反映されません。**manual** の事前計算済み結果セットは維持されないため、**Adaptive Server** では **refresh** パラメータの発行後も古いものであるとみなされます。この結果、クエリ・プロセッサはクエリで古いデータが受け入れられる場合のみ、このデータをクエリの再書き込みの対象として選択します。

**populate | nonpopulate**

事前計算済み結果セットの作成後にデータを入力するか、メタデータ情報のみで作成し、後からデータを入力するかを指定します。

- **populate** — (デフォルト) **create** コマンドの一環として結果セットにデータが入力されます。
- **nonpopulate** — **create** コマンドの実行時に結果セットにデータは入力されません。**nonpopulate** を指定した場合、事前計算済み結果セットで **enable** パラメータは実行できません。**refresh** コマンドの次回実行時に、事前計算済み結果セットが有効化されます。

**enable | disable**

事前計算済み結果セットを演算に利用できるかどうかを指定します。このオプションは、他のすべての事前計算結果セットのオプションより優先されます。

- **enable** — (デフォルト) 演算に利用できます。**enable** の設定された事前計算済み結果セットのみが、リフレッシュ・ポリシーに従って維持されます。
- **disable** — (デフォルト) 演算に利用できません。事前計算済み結果セットが無効化されていると、維持またはクエリの再書き込みで考慮されません。事前計算済み結果セットが **disable** に設定されていると、次の処理は行われません。
  - 最適化時のクエリの再書き込みへの使用。**use in optimization** の指定の有無は問いません。
  - データの入力。**with populate** の指定の有無は問いません。

{enable | disable} use in optimization

最適化時に事前計算済み結果セットをクエリの再書き込みに含めるかどうかを指定します。**use in optimization** はデフォルトで有効化されています。事前計算済み結果セットがクエリの再書き込みで考慮されるかどうかは、**refresh** パラメータの設定によって決まります。

- **immediate** — すべてのクエリで考慮されます。
- **manual** — クエリが古いデータを受け入れる場合のみ考慮されます。

lock {datarows | datapages | allpages}

事前計算済み結果セットが使用するロックのレベルを示します。

例 事前計算済み結果セット **prs\_1** を作成します。

```
create precomputed result set prs_1
as select col1, col2 from test_table
```

使用法 次の **set** パラメータは、事前計算済み結果セットの作成または変更前に設定する必要があります。

- **set ansinull on**
- **set arithabort on**
- **set arithignore off**
- **set string\_rtruncation on**

標準規格 **create precomputed result set** コマンドは、Transact-SQL 拡張機能で、SQL 規格には含まれません。

パーミッション 事前計算済み結果セットの作成には、**create table** 権限および **create view** 権限が必要です。

監査 事前計算済み結果セットの作成は監査対象ではありません。

## create procedure

### 説明

ユーザが指定する 1 つ以上のパラメータを受け入れるストアド・プロシージャまたは拡張ストアド・プロシージャ (ESP) を作成します。

---

**注意** プロシージャを作成するための SQLJ コマンドの構文と使用方法については、「[create function \(SQLJ\)](#)」(156 ページ) を参照してください。

---

### 構文

```
create procedure [owner.]procedure_name[;number]
  [(@parameter_name datatype [(length) | (precision [, scale])]
    [= default][output]
  [, @parameter_name datatype [(length) | (precision [, scale])]
    [= default][output]]...)]
  [with {recompile | execute as {owner | caller}} ]
  as {SQL_statements | external name dll_name}
```

### パラメータ

#### *procedure\_name*

プロシージャの名前です。名前は識別子の規則に従っている必要があります。また、変数を使うことはできません。現在のデータベース内で別のユーザが所有しているプロシージャと同じ名前での別のプロシージャを作成するには、所有者の名前を指定します。*owner* のデフォルト値は現在のユーザです。

#### *;number*

同じ名前を共有するプロシージャをグループ化するための任意の整数です。この整数を使用すると、1 つの [drop procedure](#) 文によってこれらのプロシージャをまとめて削除できます。同じアプリケーションで使用するプロシージャは、この方法でグループ化されます。たとえば、*orders* アプリケーションで使用するプロシージャが *orderproc;1*、*orderproc;2* などと指定されている場合に、この文を使用して次のようにグループ全体を削除できます。

```
drop proc orderproc
```

プロシージャをグループ化したら、グループ内のプロシージャを個別に削除することはできません。たとえば、次の文は使用できません。

```
drop procedure orderproc;2
```

「評価済み設定」で Adaptive Server を実行している場合は、プロシージャをグループ化することはできません。評価済み設定ではプロシージャのグループ化を拒否するように要求するため、どのストア・プロシージャにもユニークなオブジェクト識別子があり、個別に削除できます。プロシージャのグループ化を禁止するには、システム・セキュリティ担当者が `sp_configure` を使用して `allow procedure grouping` をリセットします。評価済み設定の詳細については、『システム管理ガイド』を参照してください。

#### *parameter\_name*

プロシージャの引数の名前です。プロシージャが実行されるときに、それぞれのパラメータに値を渡します。`create procedure` 文では、パラメータ名は任意です。これは、プロシージャには引数を付けなくてもかまわないためです。

パラメータ名の前には `@` 記号を付けてください。また、識別子の規則に従っていなければなりません。パラメータ名には、`@` 記号を含めて 30 文字まで使用できます。識別子の場合にはこれより長い名前が使用できます。パラメータはプロシージャに対してローカルです。同じパラメータ名を他のプロシージャでも使用できます。

パラメータ値に英数字以外の文字が含まれている場合には、その文字を引用符で囲んでください。データベース名または所有者名によって修飾されるオブジェクト名にはピリオドが含まれているため、そのようなオブジェクト名をパラメータ値に指定する場合も、引用符で囲まなければなりません。文字パラメータの値が数字で始まる場合も、引用符で囲んでください。

#### *datatype[(length) | (precision [, scale])]*

パラメータのデータ型です。『リファレンス・マニュアル：ビルディング・ブロック』の「第 1 章 システム・データ型とユーザ定義データ型」にある「ユーザ定義データ型」(49 ページ)を参照してください。ストア・プロシージャのパラメータには、`text`、`unitext`、または `image` データ型、また `text`、`unitext`、または `image` を基本とするユーザ定義データ型を使用することはできません。

`char`、`varchar`、`unichar`、`univarchar`、`nchar`、`nvarchar`、`binary` および `varbinary` のデータ型は、カッコで囲み *length* を指定する必要があります。長さを指定しなかった場合、Adaptive Server はパラメータ値を 1 文字の長さにトランケートします。

`float` データ型では、バイナリの *precision* をカッコで囲んで指定します。精度を指定しなかった場合には、Adaptive Server は使用しているプラットフォームのデフォルトの精度を使用します。



numeric データ型と decimal データ型では、*precision* と *scale* をカッコで囲み、カンマで区切って指定します。精度と位取りを指定しなかった場合、Adaptive Server はデフォルトの精度 18 と位取り 0 を使用します。

#### default

プロシージャのパラメータに対するデフォルト値を定義します。デフォルトが定義されている場合は、パラメータ値を指定しなくてもプロシージャを実行できます。デフォルトは定数でなければなりません。プロシージャが *like* キーワードの付いたパラメータ名を使用している場合は、デフォルトにワイルドカード文字 (%、\_、[ ], [^] ) を使用することもできます (例 2 を参照してください)。

デフォルトに *null* を指定することもできます。プロシージャ定義には、パラメータ値が *NULL* の場合に何らかの動作をするように指定できます (例 3 を参照してください)。

#### output

パラメータがリターン・パラメータであることを示します。その値は、このプロシージャを呼び出した *execute* コマンドに返すことができます。呼び出したプロシージャに情報を返すには、リターン・パラメータを使用します。

ネストされたプロシージャのいくつかのレベルを介してパラメータ値を返すには、最上位のプロシージャを呼び出す *execute* コマンドなどの、パラメータ名付きの *output* オプションが各プロシージャに含まれていなければなりません。

*output* キーワードは、*out* と省略できます。

#### with recompile

プロシージャが実行されるたびに Adaptive Server が新しいプランを作成することを意味します。このオプションは、プロシージャの実行を通常どおりに行わない場合、つまり、新しいプランが必要な場合に指定してください。with recompile 句は、拡張ストアド・プロシージャの実行には影響しません。

#### with execute as

所有者と呼び出し元のいずれとしてプロシージャを実行するかを指定します。所有者として実行した場合、プロシージャのすべてのアクションがプロシージャ所有者の権限と照らしてチェックされます。呼び出し元として実行した場合、プロシージャのすべてのアクションがプロシージャの呼び出し元の権限と照らしてチェックされます。

**owner**

プロシージャ所有者に代わってランタイム・パーミッションのチェック、DDLの実行、オブジェクト名の解決を行います。代替の構文として、`execute as definer` も使用できます。

**caller**

ランタイム・パーミッションのチェック、DDLの実行、プロシージャの呼び出し元の代理としてのオブジェクト名の変換を行います。代替の構文として、`execute as invoker` も使用できます。

**SQL\_statements**

プロシージャが実行する動作を指定します。`create view`、`create default`、`create rule`、`create procedure`、`create trigger`、`use` を除き、あらゆる SQL 文をいくつでも指定できます。

多くの場合、SQL 文 `create procedure` には、`declare`、`if...else`、`while`、`break`、`continue`、`begin...end`、`goto label`、`return`、`waitfor`、`/* comment */` などのフロー制御言語が 1 つ以上含まれています。これらの制御言語は、プロシージャに定義されているパラメータを参照することもできます。

SQL 文は、別のデータベースにあるオブジェクトが適切に修飾されていれば、そのオブジェクトを参照できます。

**external name**

拡張ストアードプロシージャを作成します。`as external name` コマンドには `number` パラメータを指定できません。

**dll\_name**

拡張ストアード・プロシージャを実行する関数を持つダイナミック・リンク・ライブラリ (DLL) またはシェアド・ライブラリの名前を指定します。`dll_name` は拡張子を使用せずに指定できるだけでなく、Windows NT の `.dll`、Sun Solaris の `.so` などのプラットフォーム指定の拡張子を使用しても指定できます。拡張子を指定するときは、`dll_name` 全体を引用符で囲んでください。

**例**

**例 1** テーブル名を指定した場合、`showind` プロシージャは、その名前、およびカラムにあるインデックスの名前と ID 番号を表示します。

```
create procedure showind @tablename varchar (30)
as
select sysobjects.name, sysindexes.name, indid
from sysindexes, sysobjects
where sysobjects.name = @tablename
and sysobjects.id = sysindexes.id
```

`showind` を実行する場合、次の構文を使用します。

```
execute showind titles
execute showind @tablename = "titles"
```

これがファイルまたはバッチの最初の文である場合は、次の形式を使用することもできます。

```
showind titles
```

**例 2** ユーザがパラメータを指定しなかった場合、このプロシージャはシステム・テーブルに関する情報を表示します。

```
create procedure
showsysind @table varchar (30) = "sys%"
as
select sysobjects.name, sysindexes.name, indid
from sysindexes, sysobjects
where sysobjects.name like @table
and sysobjects.id = sysindexes.id
```

**例 3** このプロシージャは、パラメータが **null** (ユーザがパラメータを指定していない場合) のときに行われる処理を指定します。

```
create procedure
showindnew @table varchar (30) = null
as
if @table is null
print "Please give a table name"
else
select sysobjects.name, sysindexes.name, indid
from sysindexes, sysobjects
where sysobjects.name = @table
and sysobjects.id = sysindexes.id
```

**例 4** このプロシージャは、2つの整数パラメータを乗算し、その積をoutputパラメータの *@result* に返します。

```
create procedure mathtutor @mult1 int, @mult2 int,
@result int output
as
select @result = @mult1 * @mult2
```

次のように、プロシージャに3つの整数を渡して実行した場合、**select**文は乗算を行って値を割り当てますが、リターン・パラメータは出力しません。

```
mathtutor 5, 6, 32
(return status 0)
```

**例 5** 次の例では、プロシージャと **execute** 文の両方に、パラメータ名付きで **output** が含まれているため、プロシージャは呼び出し側に値を返すことができます。

```
declare @guess int
```

```

select @guess = 32
exec mathtutor 5, 6, @result = @guess output

(1 row affected)
(return status = 0)

Return parameters:

@result
-----
          30

```

**execute** 文の出力パラメータと後続のパラメータ **@result** は、次のようにして渡さなければなりません。

```
@parameter = value
```

- リターン・パラメータの値は、その値の変更の有無に関係なく、常にレポートされます。
- **@result** は、**mathtutor** に渡されるパラメータ名であるため、呼び出し側のバッチで宣言する必要はありません。
- **@result** の変更された値は、**execute** 文の中で割り当てられた変数で呼び出し元に返されますが (この例では **@guess**)、独自の見出し (**@result**) で表示されます。

**例 6** リターン・パラメータは、バッチまたは呼び出し側のプロシージャ内の追加の **SQL** 文で使用することができます。この例は、**@guess** の値をプロシージャの呼び出し時に別の変数名 **@store** に保管することによって **execute** 文の後の条件句で使用する方法を示しています。リターン・パラメータを **SQL** バッチの一部である **execute** 文の中で使用すると、バッチ内の後続の文が実行される前に、戻り値が見出しと一緒に出力されます。

```

declare @guess int
declare @store int
select @guess = 32
select @store = @guess
execute mathtutor 5, 6, @result = @guess output
select Your_answer = @store, Right_answer = @guess
if @guess = @store
    print "Right-o"
else
    print "Wrong, wrong, wrong!"

(1 row affected)
(1 row affected)
(return status = 0)

```

```
Return parameters:

@result
-----
          30
Your_answer Right_answer
-----
          32          30
```

```
(1 row affected)
Wrong, wrong, wrong!
```

**例 7** `xp_echo` の拡張ストアド・プロシージャを作成します。このプロシージャは入力パラメータ `@in` を付け、それを出力パラメータ `@out` にエコーします。プロシージャのコードは `xp_echo` の関数内にあります。この関数は、コンパイルされた後、`sqlsrv.dll` という DLL にリンクされます。

```
create procedure xp_echo @in varchar (255),
                        @out varchar (255) output
as external name "sqlsrv.dll"
```

**例 8** `execute as owner` 句を使用してプロシージャを作成します。Jane はプロシージャを作成し、Bill がプロシージャを実行する際は、`execute` パーミッションのみを必要とします。テーブル `emp_interim` は、Jane が作成し、所有しています。Jane に `create table` パーミッションがない場合は、プロシージャが失敗します。

```
create procedure p_emp
with execute as owner as
select * into emp_interim
from jane.employee
grant execute on p_emp to bill
```

**例 9** `execute as caller` 句を使用してプロシージャを作成します。Jane はプロシージャを作成し、Bill がプロシージャを実行するには、`execute` パーミッションが必要です。Jane は、`p_emp` と `jane.employee` の両方を所有しています。Bill には、`jane.employee` に対する `select` パーミッションが必要になります。テーブル `emp_interim` は Bill が作成し、所有しています。Bill には、`create table` パーミッションが必要です。

```
create procedure p_emp
with execute as caller as
select * into emp_interim
from jane.employee
grant execute on p_emp to bill
```

**例 10** `execute as owner` 句を使用して、修飾されていない名前が付けられたオブジェクトを参照するプロシージャを作成します。Jane はプロシージャを作成し、Bill がプロシージャを実行します。Adaptive Server は、Jane が所有する `t1` という名前のテーブルを探します。jane.t1 がなければ、dbo.t1 を検索します。t1 が dbo.t1 に変換された場合、t1 に対する `insert` パーミッションが Jane に付与されている必要があります。

```
create procedure insert p
  with execute as owner as
  insert t1 (c1) values (100)
grant execute on insert p to bill
```

**例 11** `execute as caller` 句を使用して、修飾されていない名前が付けられたオブジェクトを参照するプロシージャを作成します。Jane はプロシージャを作成し、Bill がプロシージャを実行します。Adaptive Server は、Bill が所有する `t1` という名前のテーブルを探します。bill.t1 がなければ、dbo.t1 を検索します。t1 が dbo.t1 に変換された場合、t1 に対する `insert` パーミッションが Bill に付与されている必要があります。

```
create procedure insert p
  with execute as caller as
  insert t1 (c1) values (100)
grant execute on insert p to bill
```

**例 12** `execute as owner` 句を使用して、完全修飾名が付けられた別のデータベースにネストされたプロシージャを呼び出すプロシージャを作成します。Jane はプロシージャを作成し、Bill がプロシージャを実行します。Jane に関連付けられたログインは、otherdb 内のユーザ Jane に変換されます。Adaptive Server は、otherdb のユーザ Jane に jim.p\_child に対する `execute` パーミッションがあるかどうかを調べます。`execute as owner` を使用して jim.p\_child が作成されている場合、p\_child は Jim に代わって実行されます。jim.p\_child が `execute as caller` を使用して作成されている場合、または `execute as clause` なしで作成されている場合、p\_child は Jane に代わって実行されます。

```
create procedure p master
  with execute as owner
  as exec otherdb.jim.p_child
grant execute p master to bill
```

**例 13** `execute as caller` 句を使用して、完全修飾名が付けられた別のデータベースにネストされたプロシージャを呼び出すプロシージャを作成します。Jane はプロシージャを作成し、Bill がプロシージャを実行します。Bill に関連付けられたログインは、`otherdb` 内のユーザ Bill に変換されます。Adaptive Server は、`otherdb` のユーザ Bill に `jim.p_child` に対する `execute` パーミッションがあるかどうかを調べます。`execute as owner` を使用して `jim.p_child` が作成されている場合、`p_child` は Jim に代わって実行されます。`jim.p_child` が `execute as caller` を使用して作成されている場合、または `execute as clause` なしで作成されている場合、`p_child` は Bill に代わって実行されます。

```
create procedure p master
with execute as caller
as exec otherdb.jim.p_child
grant execute on p master to bill
```

#### 使用法

- 設定の変更によって、予期しない結果が表示されないようにするには、`create procedure` を実行する前に `set rowcount 0` を最初の文として実行します。`set` のスコープは、`create procedure` コマンドにのみ制限されているため、プロシージャが終了すると以前の設定にリセットされます。
- プロシージャの作成が完了したら、プロシージャの名前とパラメータとともに `execute` コマンドを発行することによってそのプロシージャを実行できます。プロシージャがバッチの最初の文である場合は、`execute` キーワードを使用せずに名前を指定できます。
- `sp_hidetext` を使用すると、`syscomments` に格納されたプロシージャのソース・テキストを隠すことができます。
- ストアド・プロシージャ・バッチが正常に実行されると、Adaptive Server は `@@error` グローバル変数を 0 に設定します。

#### 制限事項

- ストアド・プロシージャには、最大 2048 のパラメータを指定できます。
- プロシージャ内のローカルおよびグローバル変数の最大数は、使用できるメモリによってのみ制限されます。
- ストアド・プロシージャには、最大 16 MB のテキストを保管できます。
- 単一のバッチ内で `create procedure` 文を他の文と結合することはできません。

- ストアド・プロシージャは、現在のデータベース内でしか作成できませんが、他のデータベースからオブジェクトを参照することはできます。プロシージャの中で参照するオブジェクトの大半は、プロシージャの作成時にすでに存在しているものでなければなりません。ただし、**drop table**、**create index**、**truncate table**などの文については含めることができます。これらの文は、その基本となるオブジェクトがプロシージャの作成時に存在していなくても、**create procedure** 文の中で使用できます。

1つのプロシージャ内で、あるオブジェクトを作成してから、そのオブジェクトを参照できます。ただし、オブジェクトが作成後に参照される場合にかぎります。

プロシージャ内で **alter table** を使用してカラムを追加してから、そのプロシージャ内でそのカラムを参照することはできません。

- **create procedure** 文で **select \*** を使用すると、プロシージャでは (**with recompile** オプションを使用して **execute** を実行した場合も) テーブルに追加した新しいカラムが選択されません。プロシージャを **drop** して再作成してください。そうしないと、新しいカラムが両方のテーブルに追加されている場合、プロシージャ内の **insert into table1 select \* from table2** の **insert...select** 文によって正しくない結果が生じることがあります。
- ストアド・プロシージャ内では、オブジェクト (テンポラリ・テーブルを含む) を作成し、削除すると、同じ名前を使用して新しいオブジェクトを作成できません。ストアド・プロシージャ内で定義されたオブジェクトは、プロシージャがコンパイルされるときではなく、実行されるときに **Adaptive Server** によって実際に作成されます。

---

**警告！** インデックスの削除や再作成などのデータベースのある種の変更によって、オブジェクト ID が変更されることがあります。その場合は、ストアド・プロシージャが自動的に再コンパイルされて、サイズがいく分増加することがあります。サイズの増加を考慮して、常に空き領域を少し残しておいてください。

---

#### 拡張ストアド・プロシージャ

- **as external name** 構文を使用すると、**create procedure** は拡張ストアド・プロシージャ (ESP) を登録します。拡張ストアド・プロシージャは、**Transact-SQL** コマンドではなく、手続き型言語関数を実行します。



- (Windows) ESP 関数は C ランタイム・シグナル・ルーチン呼び出しません。Open Server™ は、Windows NT 上でシグナル処理をサポートしないので、このような呼び出しを行うと XP Server が失敗する可能性があります。
- マルチスレッドをサポートするには、ESP 関数は Open Server の `srv_yield` 関数を使用します。この関数は、XP Server スレッドを中断し、スケジュールの再編成を行って、同じかそれ以上の優先度を持つ他のスレッドの実行を可能にします。
- DLL 検索はプラットフォームに依存したメカニズムです。Windows NT では、DLL ファイル名を検索するときのシーケンスは次のようになっています。
  - a アプリケーションのロード元のディレクトリ
  - b 現在のディレクトリ
  - c システム・ディレクトリ (System32)
  - d PATH 環境変数に指定されているディレクトリ

DLL が最初の 3 つのディレクトリにない場合は、DLL のあるディレクトリを含むように PATH を設定してください。

UNIX プラットフォームでは、個々のプラットフォームによって検索方法が異なります。DLL または共有ライブラリが見つからない場合には、`$SYBASE/lib` を検索します。

絶対パス名はサポートされていません。

#### **execute as** ストアド・プロシージャ

- **execute as** 句ありまたはなしで定義されたネスト・プロシージャに文が埋め込まれている場合でも、**set session authorization** 文を **execute as owner** ストアド・プロシージャ内で使用することはできません。
- **execute as** 句は SQLJ プロシージャには対応していません。
- **execute as caller** を使用して作成されている場合、同じプロシージャのプロシージャ・キャッシュのプランをユーザ間で共有することはできません。これは、プロシージャ内のオブジェクトをプロシージャを実行するユーザに変換する必要があるためです。この理由により、多数のユーザがプロシージャを実行する場合、プロシージャ・キャッシュの使用量が増加することがあります。特定のユーザがプロシージャを再度実行した場合は、そのユーザのプランが再利用されます。

`execute as` ストアド・プロシージャの詳細については『セキュリティ管理ガイド』の「ユーザ・パーミッションの管理」を参照してください。

#### システム・プロシージャ

- システム管理者は、`sybsystemprocs` データベースに新しいシステム・プロシージャを作成できます。システム・プロシージャ名は、「`sp_`」という文字で始まります。これらのプロシージャは、プロシージャ名を指定することで、どのデータベースからも実行できます。データベース名 `sybsystemprocs` でプロシージャ名を修飾する必要はありません。システム・プロシージャ作成についての詳細は、『システム管理ガイド』を参照してください。
- システム・プロシージャの結果は、プロシージャが実行されるコンテキストによって異なることがあります。たとえば、`sp_foo` はシステム関数 `db_name()` を実行しますが、このプロシージャが実行されるデータベースの名前を返します。`pubs2` データベースから実行すると、値 "`pubs2`" が返されます。

```
use pubs2
sp_foo
-----
pubs2
```

`sybsystemprocs` から実行すると、値 "`sybsystemprocs`" が返されます。

```
use sybsystemprocs
sp_foo
-----
sybsystemprocs
```

#### ネストされたプロシージャ

- あるストアド・プロシージャから別のストアド・プロシージャを呼び出すと、プロシージャがネストされます。
- 別のプロシージャを呼び出すプロシージャを実行すると、呼び出されたプロシージャは、呼び出し側のプロシージャによって作成されたオブジェクトにアクセスできます。
- 呼び出されたプロシージャが実行されるとネスト・レベルは増加し、実行が終了するとネスト・レベルは減少します。ネスト・レベルの最大値である 16 レベルを超えると、トランザクションが失敗します。
- プロシージャ名か、または実際のプロシージャ名の代わりに変数名によって、別のプロシージャを呼び出すことができます。
- 現在のネスト・レベルは、`@@nestlevel` グローバル変数に格納されています。

- `execute as` プロシージャは、`execute as` 句ありまたはなしで作成されたネスト・プロシージャでネストできます。

#### プロシージャのリターン・ステータス

- ストアド・プロシージャは、「リターン・ステータス」と呼ばれる整数値を返す場合があります。リターン・ステータスは、プロシージャが正常に実行されたこと、または発生したエラーのタイプを示します。
- ストアド・プロシージャを実行すると、適切なステータス・コードが自動的に返されます。現在の Adaptive Server は、次のステータス・コードを返します。

コード	意味
0	プロシージャが正常に実行された
-1	オブジェクトがない
-2	データ型のエラー
-3	プロセスがデッドロックの被害対象として選択された
-4	パーミッション・エラー
-5	構文エラー
-6	その他のさまざまなユーザ・エラー
-7	領域不足などのリソース・エラー
-8	致命的ではない内部の問題
-9	システムの限界に達した
-10	致命的な内部不整合
-11	致命的な内部不整合
-12	テーブルまたはインデックスが破壊されている
-13	データベースが破壊されている
-14	ハードウェア・エラー

-15 ~ -99 のコードは、将来の使用に備えた未使用のコードです。

- ユーザは、`return` 文を使用してユーザ定義のリターン・ステータスを生成できます。ステータスは、0 ~ -99 の値を除く任意の整数値です。次の例では、本に有効な契約がある場合は値 “1” が返され、それ以外の場合は値 “2” が返されます。

```
create proc checkcontract @titleid tid
as
if (select contract from titles where
    title_id = @titleid) = 1
    return 1
else
```

```
        return 2
    checkcontract @titleid = "BU1111"
        (return status = 1)
    checkcontract @titleid = "MC3026"
        (return status = 2)
```

- プロシージャ実行中に複数のエラーが発生した場合は、絶対値が最大のコードが返されます。ユーザ定義の戻り値は、システム定義の戻り値に優先します。

#### オブジェクト識別子

- ストアド・プロシージャの名前を変更するには、`sp_rename` を使用します。
- 拡張ストアド・プロシージャの名前を変更するには、プロシージャを削除し、名前を付け直します。続いて、対応する関数を再コンパイルした後、プロシージャを再作成します。
- プロシージャが無効なテーブル名、カラム名、またはビュー名を参照する場合、`create procedure` コマンドを実行する前に `set quoted_identifier on` を設定して、それぞれの名前を二重引用符で囲んでください。プロシージャを実行するときは、`quoted_identifier` オプションは必要ありません。
- プロシージャが参照するオブジェクトのいずれかの名前が変更されている場合は、そのプロシージャを削除して再作成してください。
- ストアド・プロシージャの中では、他のユーザがそのストアド・プロシージャを使用する場合、`create table` コマンドおよび `dbcc` コマンドと使用するオブジェクト名をオブジェクト所有者の名前で修飾してください。たとえば、テーブル `marytab` を所有している "mary" というユーザが、他のユーザもそのストアド・プロシージャを実行できるようにするには、(上のコマンドと一緒にテーブルを使用するときに)ストアド・プロシージャ内で自分のテーブルの名前を修飾する必要があります。これは、プロシージャを実行するときにオブジェクト名が解析されるためです。他のユーザがこのプロシージャを実行しようとする、`Adaptive Server` はストアド・プロシージャを実行しているユーザが所有する `marytab` というテーブルではなく、ユーザ "mary" が所有する `marytab` というテーブルを探します。

したがって、`marytab` が修飾されず、ユーザ “john” がプロシージャを実行しようとした場合、Adaptive Server は、プロシージャの所有者 (この例では “mary”) が所有する、またはユーザ・テーブルが存在しない場合はデータベース所有者が所有する `marytab` というテーブルを探します。たとえば、`mary.marytab` テーブルが削除された場合、プロシージャは `dbo.marytab` を参照します。

プロシージャがコンパイルされるときにオブジェクト名が解析されるため、ストアド・プロシージャ内の他の文 (たとえば、`select` や `insert`) で使用されるオブジェクト名を修飾する必要はありません。

### テンポラリ・テーブルおよびプロシージャ

- 現在のセッションでテンポラリ・テーブルが作成される場合、テンポラリ・テーブルを参照するプロシージャを作成できます。プロシージャが終了すると、プロシージャ内で作成したテンポラリ・テーブルは消去されます。『Transact-SQL ユーザーズ・ガイド』を参照してください。
- `sp_help` などのシステム・プロシージャは、`tempdb` から使用する場合にのみテンポラリ・テーブルで動作します。

### プロシージャのオプション設定

ストアド・プロシージャ内で `set` コマンドを使用できます。ほとんどの `set` オプションは、プロシージャの実行中は有効な状態が維持され、終了後は以前の設定に戻されます。

ただし、オブジェクト所有者だけが使用できる `set` オプション (`identity_insert` など) を、オブジェクト所有者以外のユーザが使用しようとしても、そのストアド・プロシージャを実行できません。

### プロシージャ情報の取得

- プロシージャによって参照されるオブジェクトについてのレポートを取得するには、`sp_depends` を使用します。
- `syscomments` に格納されている `create procedure` 文のテキストを表示するには、プロシージャ名をパラメータとして指定して `sp_helptext` を使用します。`sp_helptext` を使用するときには、プロシージャが存在するデータベースを使用していなければなりません。システム・プロシージャのテキストを表示するには、`sybsystemprocs` データベースから `sp_helptext` を実行してください。
- システム拡張ストアド・プロシージャと対応する DLL のリストを参照するには、`sybsystemprocs` データベースから `sp_helpextendedproc` を実行してください。

## 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

## パーミッション

プロシージャの作成時は、プロシージャで参照されるテーブルやビューなどのオブジェクトに対するパーミッション・チェックは実行されません。したがって、そのオブジェクトにアクセスしなくても、プロシージャを正常に作成できます。すべてのパーミッション検査は、ユーザがプロシージャを実行するときに行われます。

プロシージャの実行時は、オブジェクトについてのパーミッション・チェックは、プロシージャとそれが参照するすべてのオブジェクトを同じユーザが所有しているかどうかによって異なります。

- プロシージャのオブジェクトを同じユーザが所有していない場合は、そのプロシージャを起動するユーザは、オブジェクトに直接アクセスする必要があります。たとえば、ユーザがアクセスできないテーブルに対してプロシージャが選択を行う場合、プロシージャは正常に動作しません。
- プロシージャとそのオブジェクトを同じユーザが所有している場合は、特別な規則が適用されます。ユーザはプロシージャのオブジェクトに直接アクセスすることはできませんが、そのオブジェクトにアクセスするための「暗黙のパーミッション」が自動的に付与されます。テーブルとビューへの直接アクセスをユーザに認めなくても、ストアド・プロシージャによって制限付きのアクセスをユーザに与えることができます。この方法では、ストアド・プロシージャはセキュリティ機能を果たします。たとえば、プロシージャを起動するユーザは、テーブルの特定のローとカラムにしかアクセスできません。『セキュリティ管理ガイド』の「セキュリティ・メカニズムとしてのストアド・プロシージャの使用」を参照してください。

細密なパーミッション設定によって異なる `create procedure` (および拡張プロシージャ作成時) のパーミッション・チェックについて、次に説明します。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>create procedure</code> 権限が必要。他のユーザに対して <code>create procedure</code> を実行するには、 <code>create any procedure</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、 <code>create database procedure</code> 権限が必要。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
11	create	create procedure	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – <code>execute as owner</code> の場合は、プロシージャの所有者名とキーワード <code>execute as owner</code> が表示される。<code>execute as caller</code> の場合は、プロシージャの呼び出し元の名前とキーワード <code>execute as caller</code> が表示される。</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [begin...end](#), [break](#), [continue](#), [declare](#), [drop procedure](#), [execute](#), [goto label](#), [grant](#), [if...else](#), [return](#), [select](#), [waitfor](#), [while](#)

システム・プロシージャ [sp\\_addextendedproc](#)、[sp\\_helpextendedproc](#)、[sp\\_helptext](#)、[sp\\_hidetext](#)、[sp\\_rename](#)、[sp\\_help](#)

## create procedure (SQLJ)

**説明** SQL ラッパを Java の静的メソッドに追加して、SQLJ ストアド・プロシージャを作成します。ユーザが提供するパラメータを受け入れ、結果セットと出力パラメータを返します。

---

**注意** プロシージャを作成するための Transact-SQL コマンドの構文と使用方法については、「[create procedure](#)」(193 ページ)を参照してください。

---

**構文**

```
create procedure [owner.]sql_procedure_name
  ((([in | out | inout] sql_parameter_name
    sql_datatype [(length) |
    (precision[, scale])]
    [=default]
    ...))
  [, [in | out | inout] sql_parameter_name
    sql_datatype [(length) |
    (precision[, scale])]
    [=default]
    ...))
  [modifies sql data]
  [dynamic result sets integer]
  [deterministic | not deterministic]
  language java
  parameter style java
  external name 'java_method_name
  [(java_datatype[, java_datatype
  ...])]
```

**パラメータ** *sql\_procedure\_name*  
プロシージャの Transact-SQL 名です。名前は識別子の規則に従っている必要があります。また、変数を使うことはできません。現在のデータベース内で別のユーザが所有しているプロシージャと同じ名前前で別のプロシージャを作成するには、所有者の名前を指定します。*owner* のデフォルト値は現在のユーザです。

**in | out | inout**  
リストされたパラメータのモードを指定します。*in* は入力パラメータ、*out* は出力パラメータ、*inout* は入力パラメータと出力パラメータの両方であるパラメータを示します。デフォルト・モードは *in* です。



**sql\_parameter\_name**

プロシージャの引数の名前です。プロシージャが実行されるときに、それぞれの入力パラメータに値を渡します。SQLJ スタアド・プロシージャには引数を付けなくてもかまわないため、パラメータは必要に応じて使用します。

パラメータの名前は、識別子の規則に従っている必要があります。パラメータ値に英数字以外の文字が含まれている場合には、その文字を引用符で囲んでください。データベース名または所有者名によって修飾されるオブジェクト名にはピリオドが含まれているため、そのようなオブジェクト名をパラメータ値に指定する場合も、引用符で囲まなければなりません。パラメータ値が数字で始まっている場合も、引用符で囲んでください。

**sql\_datatype [(length) | (precision [, scale])]**

パラメータの Transact-SQL データ型です。

**sql\_datatype** は、SQL プロシージャのシグニチャです。

**default**

プロシージャのパラメータに対するデフォルト値を定義します。デフォルトが定義されている場合は、パラメータを指定しなくてもプロシージャを実行できます。デフォルトは定数でなければなりません。プロシージャが **like** キーワードの付いたパラメータ名を使用している場合は、デフォルトにワイルドカード文字 (%、\_、[], ^) を使用することもできます。

デフォルトに **null** を指定することもできます。プロシージャ定義には、パラメータ値が **NULL** の場合に何らかの動作をするように指定できます。

**modifies sql data**

Java メソッドが、SQL オペレーションを呼び出したり、データベースの SQL データの読み込みまたは変更を行ったりすることを示します。これはデフォルトであり、唯一の実装です。また、ANSI 標準と構文の互換性を持つようにするためのものです。

**dynamic result sets integer**

Java メソッドが SQL 結果セットを返すことを指定します。integer では、Java メソッドが返すことができる結果セットの最大数を指定します。この値は実装により定義されています。

**deterministic | not deterministic**

この構文は、SQLJ に準拠した他のベンダとの互換性を保つためにサポートされています。

**language java**

外部ルーチンが **Java** で記述されていることを指定します。SQLJ ストアド・プロシージャには、この句が必要です。

**parameter style java**

実行時に外部ルーチンに渡されるパラメータが **Java** のパラメータであることを指定します。SQLJ ストアド・プロシージャには、この句が必要です。

**external**

SQL 以外のプログラミング言語で記述された外部ルーチンの SQL 名が **create procedure** によって定義されることを示します。

**name**

外部ルーチン (**Java** メソッド) の名前を指定します。指定される名前はリテラル文字列で、次のように一重引用符で囲まれている必要があります。

```
'java_method_name [ java_datatype  
                    [{, java_datatype} ...]'
```

**java\_method\_name**

外部 **Java** メソッドの名前を指定します。

**java\_datatype**

マップ可能な **Java** データ型またはマップ可能な結果セットを指定します。これは、**Java** メソッドのシグニチャです。

例

**例 1** 2つの整数を乗算し、整数を返す SQLJ プロシージャ `java_multiply` を作成します。

```
create procedure java_multiply (param1 integer,  
                               param2 integer, out result integer)  
language java parameter style java  
external name 'MathProc.multiply'
```

**例 2** 常に 10 よりも大きい値を返します。

```
create procedure my_max (a int = 10, b int = 10)  
language java parameter style java  
external name 'java.lang.Math.max'
```

```
exec my_max  
(return status = 10)
```

```
exec my_max 8  
(return status = 10)
```

Transact-SQL の `create procedure` の例も参照してください。

## 使用法

- SQLJ の create procedure 構文は、Transact-SQL の create procedure 構文とは異なります。これは、SQLJ ANSI 標準との互換性を保つためです。Adaptive Server は、各種のストアド・プロシージャを同じ方法で実行します。
- 設定の変更によって、予期しない結果が表示されないようにするには、create procedure の実行前に set rowcount 0 を最初の文として実行します。set のスコープは、create procedure コマンドにのみ制限されているため、プロシージャが終了すると以前の設定にリセットされます。
- create procedure 文では、最大 31 の in、inout、out パラメータを使用できます。
- ANSI 標準に準拠させる場合は、パラメータ名の前に @ 記号を付けないでください。ただし、SQLJ ストアド・プロシージャを isql または Java 以外の他のクライアントから実行するときに、パラメータ名の前に @ 記号を付ける必要がある場合は、命名の順序が保持されます。

## パーミッション

プロシージャの作成時は、プロシージャで参照されるテーブルやビューなどのオブジェクトに対するパーミッション・チェックは実行されません。したがって、そのオブジェクトにアクセスしなくても、プロシージャを正常に作成できます。すべてのパーミッション検査は、ユーザがプロシージャを実行するときに行われます。

プロシージャの実行時は、オブジェクトについてのパーミッション・チェックは、プロシージャとそれが参照するすべてのオブジェクトを同じユーザが所有しているかどうかによって異なります。

- プロシージャのオブジェクトを同じユーザが所有していない場合は、そのプロシージャを起動するユーザは、オブジェクトに直接アクセスできる必要があります。たとえば、ユーザがアクセスできないテーブルに対してプロシージャが選択を行う場合、プロシージャは正常に動作しません。
- プロシージャとそのオブジェクトを同じユーザが所有している場合は、特別な規則が適用されます。ユーザはプロシージャのオブジェクトに直接アクセスすることはできませんが、そのオブジェクトにアクセスするための「暗黙のパーミッション」が自動的に付与されます。テーブルとビューへの直接アクセスをユーザに認めなくても、ストアド・プロシージャによって制限付きのアクセスをユーザに与えることができます。この方法では、ストアド・プロシージャはセキュリティ機能を果たします。たとえば、プロシージャを起動するユーザが、テーブルの特定のローとカラムにしかアクセスできない場合があります。

## create procedure (SQLJ)

---

細密なパーミッション設定によって異なる `create procedure` (および拡張プロシージャ作成時) のパーミッション・チェックについて、次に説明します。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>create procedure</code> 権限が必要。他のユーザに対して <code>create procedure</code> を実行するには、 <code>create any procedure</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、 <code>create database procedure</code> 権限が必要。

### 参照

コマンド [create function \(SQLJ\)](#), [drop procedure](#)

システム・プロシージャ [sp\\_depends](#), [sp\\_help](#), [sp\\_helpjava](#), [sp\\_helprotect](#)

## create proxy\_table

**説明** (コンポーネント統合サービスのみ) カラム・リストを指定しないでプロキシ・テーブルを作成します。コンポーネント統合サービスは、リモート・テーブルから取得するメタデータからカラム・リストを生成します。

**構文**

```
create proxy_table table_name
    [external [table | directory | file]]
    at pathname
    [column delimiter "<string>"]
```

**パラメータ** *table\_name*

後続の文によって使用されるローカルのプロキシ・テーブル名を指定します。*table\_name* の形式は、*dbname.owner.object* です。*dbname* と *owner* は任意であり、ローカル・データベースと所有者を表します。*dbname* を指定しないと、テーブルは現在のデータベースに作成されます。*owner* を指定しないと、テーブルの所有者は現在のユーザになります。*dbname* または *owner* を指定しない場合は、*table\_name* 全体を引用符で囲んでください。*dbname* だけを指定する場合は、*owner* のプレースホルダが必要です。

### external table

オブジェクトがリモートのテーブルまたはビューであることを指定します。*external table* がデフォルトであるため、この句は任意です。

### external directory

オブジェクトが、次の形式のパスを持つディレクトリであることを示します。“/tmp/directory\_name [;R]”、“R” は再帰であることを示します。

### external file

オブジェクトが、次の形式のパスを持つファイルであることを示します。“/tmp/filename”。

### at pathname

リモート・オブジェクトのロケーションを指定します。*pathname* の形式は、*server\_name.dbname.owner.object* です。

- *server\_name* — リモート・オブジェクトがあるサーバの名前です。
- *dbname* — (オプション) このオブジェクトが格納されるリモート・サーバで管理されるデータベースの名前です。
- *owner* — (オプション) リモート・オブジェクトを所有するリモート・サーバ・ユーザの名前です。
- *object* — リモートのテーブルまたはビューの名前です。

*string*

カラム・デリミタ文字列には、あらゆる文字シーケンサを使用できますが、文字列が 16 バイトを超えた場合は、最初の 16 バイトのみが使用されます。ファイル以外にマップされたプロキシ・テーブルでカラム・デリミタを使用すると、構文エラーが発生します。

例 この例では、リモート・テーブル `t1` にマップされる、`t1` という名前のプロキシ・テーブルを作成します。コンポーネント統合サービスによって、カラム・リストがリモート・テーブルから導出されます。

```
create proxy_table t1
at "SERVER_A.db1.joe.t1"
```

## 使用法

- `create proxy_table` は、`create existing table` コマンドの変形です。`create proxy_table` を使用してプロキシ・テーブルを作成する場合は、`create existing table` とは異なり、カラム・リストは指定しません。コンポーネント統合サービスは、リモート・テーブルから取得するメタデータからカラム・リストを生成します。
- `at` キーワードによって得られるロケーション情報は、`sp_addobjectdef` によって得られる情報と同じです。この情報は、`sysattributes` テーブルに格納されます。
- リモート・サーバ・オブジェクトが存在しない場合は、このコマンドは拒否されてエラー・メッセージが表示されます。
- オブジェクトが存在する場合は、ローカルのシステム・テーブルが更新されます。各カラムが使用されます。カラムとその属性が、テーブルまたはビューのために取得されます。
- コンポーネント統合サービスは、自動的にカラムのデータ型を Adaptive Server のデータ型に変換します。変換が行われないときは、`create proxy_table` コマンドはそのテーブルの定義付けを許可しません。
- リモート・サーバ・テーブル内のインデックス情報を抽出し、システム・テーブル `sysindexes` のローを作成するために使用します。これによって、Adaptive Server 用語でインデックスとキーが定義され、クエリ・オプティマイザがテーブル上に存在する可能性があるインデックスをすべて検証できるようになります。
- プロキシ・テーブルを定義したら、テーブルに対して `update statistics` コマンドを発行してください。これによって、クエリ・オプティマイザは、ジョイン順に関して最善の選択を行うことができるようになります。

- `create proxy_table table_name at pathname` を実行するときには、`pathname` によって識別されるサーバが大文字と小文字を区別しない場合 (DB2 や Oracle など)、テーブル名とカラム名は `table_name` と同じ文字であると想定されます。

`table_name` が小文字の場合、大文字と小文字を区別しないサーバから返されるカラム (通常は大文字) は、小文字で Adaptive Server に格納されます。`table_name` が大文字の場合は、カラム名は大文字の値として格納されます。`table_name` に大文字と小文字が混在している場合は、カラム名はすべて、リモート・サイトから受け取ったとおりの文字で格納されます。

- `create proxy_table` は、テンポラリ・テーブルではサポートされません。
- 単一のバッチ内で `create proxy_table` 文を他の文と結合することはできません。
- プロキシ・テーブルにはメタデータのみが保管されます。したがって、使用される領域はシステム・カタログに作成されるエントリだけです。100 のプロキシ・テーブルによって約 1MB の領域が消費されると見積もられます (テーブルごとに平均 2 つのインデックスがあると想定します)。
- SQL ユーザ-定義関数は現時点では `create proxy table`、`create table at remote server`、または `alter table` ではサポートされていません。

---

**注意** SQL 関数の実行に必要な構文は `username.functionname()` です。

---

- リモートの Adaptive Server テーブルに 1 つ以上の暗号化カラムがある場合、コンポーネント統合サービスは `syscolumns` にあるプロキシ・テーブルのメタデータを更新し、カラムの暗号化のプロパティとそのキー ID を反映します。

#### 標準規格

ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

#### パーミッション

`create proxy_table` パーミッションは、デフォルトではテーブル所有者にあり、譲渡することはできません。

#### 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

---

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
11	create	create procedure	<ul style="list-style-type: none"><li>• <i>Roles</i> – 現在のアクティブな役割</li><li>• <i>Keywords</i> または <i>options</i> – NULL</li><li>• <i>Previous value</i> – NULL</li><li>• <i>Current value</i> – NULL</li><li>• <i>Other information</i> – NULL</li><li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li></ul>

参照

コマンド [create existing table](#), [create table](#)



## create role

**説明** ユーザ定義の役割を作成し、作成時に、特定の役割に許可されるパスワード有効期間、最短のパスワード長、ログイン失敗の最大許容回数を指定します。役割の作成時に、役割にパスワードを関連付けることもできます。

**構文**

```
create role role_name [with passwd "password"  
                        [, {passwd expiration | min passwd length |  
                           max failed_logins} option_value]
```

**パラメータ**

*role\_name*

新しい役割の名前です。サーバに対してユニークで、識別子の規則に従います。*role\_name* に変数を使用することはできません。

with passwd

役割をアクティブ化するときにユーザが入力するパスワードを設定します。

*password*

役割に付加するパスワードです。パスワードは最低 6 文字からなり、識別子規則に従っていなければなりません。パスワードに変数は使用できません。

passwd expiration

password expiration interval は、パスワード有効期限の間隔を日数で指定します。0 ~ 32767 の任意の値を指定できます。たとえば、パスワードの有効期限の間隔が 30 日である新しいログインを 2007 年 8 月 1 日の午前 10 時半に作成したとすると、2007 年 8 月 31 日の午前 10 時半にパスワードの有効期限が切れます。

min passwd length

指定した役割に必要な最短のパスワード長を指定します。

max failed\_logins

指定したログインに許可される、ログイン失敗の回数を指定します。

*option\_value*

passwd expiration、min passwd length、または max failed\_logins の値を指定します。

**例**

**例 1** doctor\_role という名前の役割を作成します。

```
create role doctor_role
```

**例 2** "physician" というパスワードを持つ doctor\_role という名前の役割を作成します。

```
create role doctor_role with passwd "physician"
```

**例 3** passwd expiration を 7 日間に設定します。役割のパスワードは、指定した期間が過ぎた後（この例では 7 日間）、パスワードを最後に変更した日に有効期限が切れます。

```
create role intern_role with passwd "temp244",  
passwd expiration 7
```

**例 4** intern\_role に許可されるログイン失敗の最大回数を設定します。

```
create role intern_role with passwd "temp244"  
max failed_logins 20
```

**例 5** intern\_role の最短パスワード長を設定します。

```
create role intern_role with passwd "temp244",  
min passwd length 0
```

## 使用法

- create role は、master データベースから実行します。
- 役割にパスワードを付加すると、この役割を付与されたユーザは役割をアクティブ化するときパスワードを指定しなければなりません。

作成後に役割にパスワードを追加する方法については、「[alter role](#)」を参照してください。

---

**注意** バージョン 12.x より前に存在し、ユーザ定義に付加されていたパスワードは期限切れになりません。

---

- 役割名は、サーバ上でユニークなものでなければなりません。
- 役割の名前をユーザ名と同じにすることはできません。役割はユーザと同じ名前で作成できますが、権限を付与するときに、Adaptive Server は役割ではなくユーザに付与を行って名前の競合を解析します。

名前の競合の詳細については、「[grant role](#)」を参照してください。

## 制限事項

- 作成可能な役割の最大数は、1 つのサーバ・セッションにつき 1024 です。ただし、32 の役割は、sa\_role や sso\_role などの Sybase システムの役割のために予約されています。また、特殊なユーザ定義の役割である sa\_serverprivs\_role は、Adaptive Server によって作成されています。このため、ユーザ定義の役割の最大数は、1 つのサーバ・セッションにつき 991 になります。-

- 付加されたパスワードで役割を作成すると、デフォルトではユーザはログイン時にその役割をアクティブ化できません。役割を付与したユーザがデフォルトでログイン時に役割をアクティブ化する必要がある場合は、付加されたパスワードで役割を作成しないでください。

## 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

## パーミッション

`create role` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage roles</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sso_role</code> が付与されたユーザであることが必要。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
85	roles	<code>create role</code> 、 <code>drop role</code> 、 <code>alter role</code> 、 <code>grant role</code> 、または <code>revoke role</code>	<ul style="list-style-type: none"> <li><i>Roles</i> – 現在のアクティブな役割</li> <li><i>Keywords</i> または <i>options</i> – NULL</li> <li><i>Previous value</i> – NULL</li> <li><i>Current value</i> – NULL</li> <li><i>Other information</i> – NULL</li> <li><i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter role](#)、[drop role](#)、[grant](#)、[revoke](#)、[set](#)

システム・プロシージャ [sp\\_activeroles](#)、[sp\\_displaylogin](#)、[sp\\_displayroles](#)、[sp\\_helprotect](#)、[sp\\_modifylogin](#)

## create rule

**説明** 特定のカラムまたはユーザ定義データ型の任意のカラムに対し、受け入れ可能な値のドメインを指定し、アクセス・ルールを作成します。

**構文**

```
create [[and | or] access]] rule
      [owner.]rule_name
      as condition_expression
```

### パラメータ

#### access

アクセス・ルールを作成することを指定します。『システム管理ガイド』の「ユーザ・パーミッションの管理」を参照してください。

#### rule\_name

新しいルールの名前です。名前は識別子の規則に従っている必要があります。変数を使うことはできません。現在のデータベース内で別のユーザが所有しているルールと同じ名前で別のルールを作成するには、所有者の名前を指定します。*owner* のデフォルト値は現在のユーザです。

#### condition\_expression

ルールを定義する条件を指定します。*where* 句で有効な式はすべて使用できます。式には、算術演算子、関係演算子、*in*、*like*、*between* などを使用できます。ただし、*condition\_expression* がカラムや別のデータベース・オブジェクトを参照することはできません。データベース・オブジェクトを参照しない組み込み関数は使用できます。

*condition\_expression* は、引数を 1 つだけ使用します。引数の前には *@* 記号が付き、*update* コマンドまたは *insert* コマンドで入力された値を参照します。ルールを作成するときには、その値を表す任意の名前または記号を使用できます。文字定数と日付定数は引用符で囲んでください。バイナリ定数は、前に "0x" を付けてください。

**例** **例 1** *advance* の値を 1000 ドル未満に制限する、*limit* という名前のルールを作成します。

```
create rule limit
as @advance < $1000
```

**例 2** *pub\_id* の値を 1389、0736、または 0877 に制限する、*pubid\_rule* という名前のルールを作成します。

```
create rule pubid_rule
as @pub_id in ('1389', '0736', '0877')
```

**例 3** *value* の値を常に指示された文字で始まるように制限する、*picture* という名前のルールを作成します。

```
create rule picture
as @value like '[_-%[0-9]'
```

## 使用法

- ルールのテキストを隠すには、`sp_hidetext` を使用します。
- ルールの名前を変更するには、`sp_rename` を使用します。

## 制限事項

- 現在のデータベース内でのみ、ルールを作成できます。
- ルールが作成されたときにすでにデータベース内に存在していたデータに対しては、新しいルールは適用されません。
- 1 つのバッチ内で `create rule` 文を他の文と組み合わせることはできません。
- `Adaptive Server` が提供するデータ型や、`text`、`unitext`、`image`、または `timestamp` 型のカラムにルールをバインドすることはできません。
- 同じ名前の新しいルールを作成するには、事前に元のルールを削除しておきます。ルールのバインドを解除してから削除してください。次のコマンドを使用します。

```
sp_unbindrule objname [, futureonly]
```

## ルールのバインド

- カラムまたはユーザ定義データ型にルールをバインドするには、`sp_bindrule` を使用します。

```
sp_bindrule rulename, objname [, futureonly]
```

- ユーザ定義データ型にバインドされたルールは、そのデータ型のカラムに対して値の挿入または更新を実行するときにアクティブ化されます。ルールは、ユーザ定義データ型の変数に挿入される値のテストは行いません。
- 作成したルールは、カラムのデータ型と互換性がなければなりません。たとえば、以下を真数値カラムまたは概数値カラムのルールとして使用することはできません。

```
@value like A%
```

作成したルールに、バインドしたカラムとの互換性がない場合、バインドするときではなく、値を挿入しようとするときにエラー・メッセージが表示されます。

- 既存のルールのバインドを解除しないで、作成したルールをカラムやデータ型にバインドすることもできます。

- カラムにバインドされたルールは、ユーザ定義データ型にバインドされたルールよりも常に優先されます。どちらのルールが後でバインドされたかは関係ありません。表 1-9 は、ルールがすでに存在するカラムやユーザ定義データ型にルールをバインドした場合の優先度を示します。

表 1-9: ルールのバインドの優先度

新しいルールのバインド先	ユーザ定義のデータ型にバインドされている既存のルール	カラムにバインドされている既存のルール
ユーザ定義データ型	新しいルールに置き換わる	変更なし
カラム	新しいルールに置き換わる	新しいルールに置き換わる

- ルールは、カラム定義を上書きしません。null 値を使用できるカラムにルールをバインドする場合は、そのルールのテキストに null が含まれていない場合でも、暗黙にまたは明示的にそのカラムに NULL を挿入できます。たとえば、"@val in (1,2,3)" または "@amount > 10000" を指定するルールを作成し、このルールを null 値が使用できるテーブルのカラムにバインドしても、そのカラムには NULL を挿入できます。カラム定義は、ルールを上書きします。
- デフォルトとルールの両方がカラムに関連付けられている場合、デフォルトはそのルールによって定義された領域内になければなりません。ルールに適合しないデフォルト値は、挿入されません。ルールに違反するデフォルトを挿入すると、Adaptive Server によってエラーメッセージが生成されます。
- `create table` 文で `check` を使用してルールを定義することで、整合性制約を作成できます。ただし、その制約はテーブル固有のものであり、別のテーブルにバインドすることはできません。整合性制約については、「`create table`」および「`alter table`」を参照してください。
- ルールについてのレポートを取得するには、`sp_help` を使用します。
- システム・テーブル `syscomments` に格納されているルールのテキストを表示するには、ルール名をパラメータとして指定して `sp_helptext` を実行します。
- ルールが特定のカラムまたはユーザ定義データ型にバインドされると、そのルールの ID は、システム・テーブル `syscolumns` または `systypes` に格納されます。

標準規格	ANSI SQL — 準拠レベル：初級レベル  ANSI SQL に準拠した構文を使用してルールを作成するには、 <a href="#">create table</a> 文の <code>check</code> 句を使用します。
パーミッション	<code>create rule</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>create rule</code> 権限が必要。他のユーザーに対して <code>create function</code> を使用するには、 <code>create any rule</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>create rule</code> 権限を保持しているか、データベース所有者であるか、 <code>sa_role</code> が付与されたユーザーであることが必要。  他のユーザーに対して <code>create rule</code> を使用するには、 <code>sa_role</code> が付与されたユーザーであることが必要。

監査 `sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
13	create	create rule	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照 コマンド [alter table](#), [create default](#), [create table](#), [drop rule](#), [drop table](#)  
システム・プロシージャ [sp\\_bindrule](#), [sp\\_help](#), [sp\\_helptext](#), [sp\\_hidetext](#), [sp\\_rename](#), [sp\\_unbindrule](#)

## create schema

**説明** データベース・ユーザ用の新しいテーブル、ビュー、パーミッションを作成します。

**構文**

```
create schema authorization authorization_name
    create_object_statement
    [create_object_statement ...]
    [permission_statement ...]
```

**パラメータ**

*authorization\_name*  
データベース内の現在のユーザ名です。

*create\_object\_statement*  
[create table](#) 文または [create view](#) 文です。

*permission\_statement*  
[grant](#) コマンドまたは [revoke](#) コマンドです。

**例** `newtitles`、`newauthors`、`newtitleauthors` の各テーブル、`tit_auth_view` ビュー、および対応するパーミッションを作成します。

```
create schema authorization pogo
create table newtitles (
    title_id tid not null,
    title varchar (30) not null)

create table newauthors (
    au_id id not null,
    au_lname varchar (40) not null,
    au_fname varchar (20) not null)

create table newtitleauthors (
    au_id id not null,
    title_id tid not null)

create view tit_auth_view
as
select au_lname, au_fname
    from newtitles, newauthors,
        newtitleauthors
where
newtitleauthors.au_id = newauthors.au_id
and
newtitleauthors.title_id =
    newtitles.title_id

grant select on tit_auth_view to public
revoke select on tit_auth_view from churchy
```



使用法	<ul style="list-style-type: none"><li>スキーマは、現在のデータベース内でのみ作成できます。</li><li>「スキーマ権限識別子」とも呼ばれる <code>authorization_name</code> は、現在のユーザ名にしてください。</li><li>ユーザには、適切なコマンド・パーミッション (<code>create table</code> および <code>create view</code>) が必要です。他のデータベース・ユーザが所有するテーブルにビューを作成する場合は、そのビューが作成されるのではなく、そのビューを通してデータへアクセスするときに、ビューに対するパーミッションが検査されます。</li><li><code>create schema</code> コマンドは、以下によって終了します。<ul style="list-style-type: none"><li>通常のコマンド・ターミネータ (<code>isql</code> 内ではデフォルトは "go")。</li><li><code>create table</code>、<code>create view</code>、<code>grant</code>、<code>revoke</code> 以外の文。</li></ul></li><li><code>create schema</code> 文内のいずれかの文が失敗すると、コマンド全体が 1 つの単位としてロールバックされ、コマンドによる影響はありません。</li><li><code>create schema</code> は、テーブル、ビュー、パーミッションについての情報をシステム・テーブルに追加します。<code>create schema</code> で作成したオブジェクトを削除するには、適切な削除コマンド (<code>drop table</code> または <code>drop view</code>) を使用してください。スキーマで付与されたパーミッションまたは取り消されたパーミッションは、スキーマ作成文の外で通常の <code>grant</code> コマンドと <code>revoke</code> コマンドを使用して変更できません。</li><li>クラスタのみ - ローカル・テンポラリ・データベースのカラムを参照する参照整合性の制約は、同じローカル・テンポラリ・データベースのテーブルのものでない場合、含むことはできません。<code>create schema</code> は、別のデータベースのテーブルからローカル・テンポラリ・データベースのカラムへの参照を作成しようとすると失敗します。</li></ul>
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>create schema</code> は、データベースのユーザであれば誰でも実行できます。ただし、指定したオブジェクトをスキーマに作成するには、 <code>create table</code> および <code>create view</code> のパーミッションが必要です。
参照	コマンド <code>create table</code> 、 <code>create view</code> 、 <code>grant</code> 、 <code>revoke</code> ユーティリティ <code>isql</code>

## create service

**説明** 提供された SQL 文を指定された名前とパラメータを持つストアド・プロシージャでラップします。

**構文**

```
create service service-name [secure security_options] [, userpath path]
    [, alias alias-name]
    type { xml | raw | soap }
    [[(@parameter_name datatype [(length) | (precision [, scale ])]
    [= default][output]
    [, @parameter_name datatype [(length) | (precision [, scale ])]
    [= default][output]]...)]
as SQL_statements

security_options ::= (security_option_item [security_option_item])
```

### パラメータ

#### *service-name*

ユーザ定義 Web サービスの名前です。サービス名には、ストアド・プロシージャとして有効な任意の名前を指定できます。drop procedure コマンドがこのサービス名を使用して呼び出されると、対応するストアド・プロシージャが削除されます。既存のサービスの名前を指定すると、例外が発生します。

#### *security\_option\_item*

- clear - Web サービスへのアクセスに HTTP が使用されることを示します。
- ssl - Web サービスへのアクセスに HTTPS が使用されることを示します。

#### *path*

Web サービスにアクセスする URL に追加されるユーザ定義パスを指定するリテラル文字列です。デフォルトでは *path* は null です。

#### *alias-name*

ユーザ定義 Web サービスのエイリアスをリテラル文字列で指定します。

#### *parameter\_name*

ユーザ定義 Web サービスの引数の名前です。このパラメータの値は Web サービスの実行時に渡されます。パラメータ名の前には @ 記号を付けてください。また、識別子の規則に従っていなければなりません。これらの条件は、create procedure コマンドの *parameter\_name* パラメータの場合と同じです。

**SQL\_statements**

ユーザ定義 Web サービスが実行するアクションです。create view、create default、create rule、create procedure、create trigger、use を除き、あらゆる SQL 文をいくつでも指定できます。

**type**

- soap - HTTP の POST 要求を意味し、SOAP のすべてのルールに準拠する必要があります。返されるデータは、SQL/XML フォーマットです。
- raw - 出力が何らかの変更や再フォーマットをまったく受けずに送信されることを示します。これは HTTP の GET 要求を意味します。呼び出されたストアド・プロシージャは、出力を厳密に指定できます。
- xml - 結果セットの出力が SQL/XML フォーマットで返されることを示します。これは HTTP の GET 要求を意味します。

---

**注意** Adaptive Server ストアド・プロシージャと SOAP ユーザ定義 Web サービス間のデータ型のマッピングについては、『Web Services ユーザーズ・ガイド』を参照してください。

---

**例**

**例 1** raw 型のユーザ定義 Web サービス rawservice が作成され、現在のデータベースのバージョンが返されます。pubs2 データベースに対し isql コマンド・ラインから create service コマンドを次のように入力します。

```
1> use pubs2
2> go
1> create service rawservice type raw as select
'<html><h1>' + @@version + '</h1></html>'
2> go
```

次に、新しく作成したユーザ定義 Web サービスを展開します。

```
1> sp_webservices 'deploy', 'all'
2> go
```

新しく作成したユーザ定義 Web サービスの Web Service Definition Language は <http://myhost:8181/services/pubs2?wsdl> にあります。

新しく作成したユーザ定義 Web サービスは次の URL から利用可能です。ここで、bob と bob123 は、ユーザ定義 Web サービスの作成者のユーザ ID とパスワードです。

<http://myhost:8181/services/pubs2?method=rawservice&username=bob&password=bob123>

出力である Adaptive Server Enterprise のバージョン文字列は、ブラウザ・ウィンドウの HTML <h1> タグ内に表示されます。

**例 2** xml 型のユーザ定義 Web サービス xmlservice を作成し、現在のデータベースのバージョンを返します。pubs2 データベースに対し isql コマンド・ラインから create service コマンドを次のように入力します。

```
1> use pubs2
2> go
1> create service xmlservice userpath "testing" type xml
as select @@version
2> go
```

次に、新しく作成したユーザ定義 Web サービスを展開します。

```
1> sp_webservices 'deploy', 'xmlservice'
2> go
```

---

**注意** deploy オプションの詳細については、『リファレンス・マニュアル：プロシージャ』の「sp\_webservices」を参照してください。

---

ユーザ定義 Web サービスの WSDL は以下にあります。

```
http://myhost:8181/services/pubs2/testing?wsdl
```

ユーザ定義 Web サービスはブラウザで次の URL にアクセスして呼び出します。ここで、bob と bob123 は、ユーザ定義 Web サービスの作成者のユーザ ID とパスワードです。

```
http://myhost:8181/services/pubs2/testing?method=xmlservice&
username=bob&password=bob123
```

出力はブラウザ・ウィンドウに XML として表示されます。

**例 3** ユーザ定義 Web サービスを SOAP クライアントから使用して、ストアド・プロシージャ sp\_who を実行できるようにします。このストアド・プロシージャは引数を 1 つ取り、オプションで userpath トークンを指定します。

```
create service sp_who_service userpath
'myservices/args' type soap @loginname varchar(30) as
exec sp_who @loginname
```

Web サービスは、pubs2 データベース内に sp\_who\_service という名前で作成され、展開された後に、次の URL からアクセス可能になります。

```
http://localhost:8181/pubs2/myservices/args/sp_who_service
```

サービスの WSDL は以下にあります。

```
http://localhost:8181/pubs2/myservices/args?wsdl
```

WSDL ファイルに記述されているこの Web メソッドのシグニチャは、次のようになります。

```
DataReturn[] sp_who_service (xsd:string username,
                             xsd:string password, xsd:string loginname)
```

新しいサービスは SOAP クライアントから、varchar(30) 型の loginname というパラメータを 1 つ使用して呼び出されます。

#### 使用法

次の点以外は、結果として得られるストアド・プロシージャは `create procedure` コマンドで作成されたストアド・プロシージャと同じように動作し、既存のストアド・プロシージャに関する実行、複写、`sp_helptext`、再コンパイルのルールに従います。また `isql` から実行できます。

- 結果として得られるストアド・プロシージャは、`drop procedure` コマンドではなく、`drop service` コマンドでのみ削除できる。
- `syscomments` テーブルには、`create service` コマンドの再作成に必要な DDL が格納されます。
- 指定されたサービス名ではストアド・プロシージャ・グループを作成できない。

---

**注意** Adaptive Server Web Services Engine を介してユーザ定義 Web サービスを使用できるようにするには、`sp_webservices` の `deploy` オプションを使用する必要があります。ただし、ユーザ定義 Web サービスのストアド・プロシージャは、展開されていなくても `isql` からアクセスできます。

---

#### 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

#### パーミッション

Web サービスを使用するには、`execute` パーミッションが明示的に付与されている必要があります。

**サービス作成時のオブジェクトのパーミッション** Web サービスの作成時は、Adaptive Server は、サービスによって参照されるテーブルやビューなどのオブジェクトについてのパーミッション検査を行いません。したがって、そのオブジェクトにアクセスしなくても、Web サービスを正常に作成できます。すべてのパーミッション検査は、ユーザが Web サービスを実行するときに行われます。

**Web サービス実行時のオブジェクトのパーミッション** Web サービスの実行時は、オブジェクトについてのパーミッション検査は、Web サービスとそれが参照するすべてのオブジェクトを同じユーザが所有しているかどうかによって異なります。

- Web サービスのオブジェクトを同じユーザが所有していない場合は、そのプロシージャを起動するユーザは、オブジェクトに直接アクセスできなければなりません。たとえば、ユーザがアクセスできないテーブルに対して Web サービスが選択を行う場合、Web サービスは正常に動作しません。
- Web サービスとそれが参照するすべてのオブジェクトを同じユーザが所有している場合は、特別な規則が適用されます。Web サービスを呼び出したユーザは、オブジェクトに直接アクセスできない場合でも、そのオブジェクトにアクセスできる「暗黙的なパーミッション」が自動的に付与されます。テーブルとビューへの直接アクセスをユーザに認めなくても、ストアド・プロシージャによって制限付きのアクセスをユーザに与えることができます。この方法では、ストアド・プロシージャはセキュリティ機能を果たします。たとえば、Web サービスを起動するユーザは、テーブルの特定のローとカラムにしかアクセスできません。

暗黙のパーミッションの規則の詳細については、『システム管理ガイド』を参照してください。

細密なパーミッションの設定によって異なる `create service` のパーミッション・チェックについて、次に説明します。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>create procedure</code> 権限が必要。他のユーザに対して <code>create service</code> を使用するには、 <code>create any procedure</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>create procedure</code> 権限を保持しているか、データベース所有者であるか、 <code>sa_role</code> が付与されたユーザであることが必要。 他のユーザに対して <code>create rule</code> を使用するには、 <code>sa_role</code> が付与されたユーザであることが必要。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
11	create	create services	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## create table

### 説明

- 新しいテーブルを作成し、必要に応じて整合性制約を作成します。
- 計算カラムを定義します。
- テーブル、ロー、パーティションの圧縮レベルを定義します。
- 暗号化カラムと暗号化カラムの復号化デフォルトを定義します。
- テーブルのパーティション・プロパティを定義します。テーブル・パーティション作成の各構文を次に示します。パーティションの構文を参照してください。

### 構文

```
create table [[database.[owner].]table_name (column_name datatype
    [default {constant_expression | user | null}]
    [{identity | null | not null}]
    [ in row [(length)] | off row ]
    [[constraint constraint_name]
    {{unique | primary key}
    [clustered | nonclustered] [asc | desc]
    [with {fillfactor = pct,
        max_rows_per_page = num_rows,
        reservepagegap = num_pages}
    dml_logging = {full | minimal}
    [deferred_allocation | immediate_allocation]}
    [on segment_name]
    | references [[database.]owner.]ref_table
    [(ref_column)]
    [match full]
    | check (search_condition)}}]
[[encrypt [with [database.[owner].]key_name]
[decrypt_default constant_expression | null]]
[not compressed]
[compressed = {compression_level | not compressed}]
[[constraint [[database.[owner].]key_name]
{unique | primary key}
[clustered | nonclustered]
(column_name [asc | desc]
[{, column_name [asc | desc]}...])
[with {fillfactor = pct
max_rows_per_page = num_rows,
reservepagegap = num_pages}]
[on segment_name]
| foreign key (column_name [{, column_name}...])
references [[database.]owner.]ref_table
[(ref_column [{, ref_column}...])]
[match full]
| check (search_condition) ...}
[{{, {next_column | next_constraint}}...}]
[lock {datarows | datapages | allpages}]
```

```

[with { max_rows_per_page = num_rows,
        exp_row_size = num_bytes,
        reservepagegap = num_pages,
        identity_gap = value
        transfer table [on | off]
        dml_logging = {full | minimal}
        compression = {none | page | row}}]
[on segment_name]
[partition_clause]
[[ [ external table ] at pathname ]
[for load]
compression_clause::=
    with compression = {none | page | row}

```

パーティションに次の構文を使用します。

```

partition_clause::=
    partition by range (column_name[, column_name]...)
        ([partition_name] values <= ({constant | MAX}
            [, {constant | MAX}] ...))
            [compression_clause] [on segment_name]
        [, [partition_name] values <= ({constant | MAX}
            [, {constant | MAX}] ...))
            [compression_clause] [on segment_name]...)

    | partition by hash (column_name[, column_name]...)
        { (partition_name
            [compression_clause] [on segment_name]
            [, partition_name
            [compression_clause] [on segment_name]...)
        | number_of_partitions
            [on (segment_name[, segment_name] ...)]}

    | partition by list (column_name)
        ([partition_name] values (constant[, constant] ...)
            [compression_clause] [on segment_name]
        [, [partition_name] values (constant[, constant] ...)
            [compression_clause] [on segment_name] ...)

    | partition by roundrobin
        { (partition_name [on segment_name]
            [, partition_name
            [compression_clause] [on segment_name]...)
        | number_of_partitions
            [on (segment_name [, segment_name]...)]}

```

計算カラムに次の構文を使用します。

```

create table [[database.[owner].] table_name
    (column_name {compute | as}

```



```
computed_column_expression
[[materialized] [not compressed]] | [not materialized]]
```

仮想ハッシュ・テーブルを作成するには、次の構文を使用します。

```
create table [database.[owner].]table_name
...
| {unique | primary key}
using clustered
(column_name [asc | desc] [{, column_name [asc | desc]}...] =
(hash_factor [{, hash_factor}...]
with max num_hash_values key
```

## パラメータ

### *table\_name*

新しいテーブルの明示的な名前です。テーブルが別のデータベース内にある場合はデータベース名を指定し、データベース内にその名前のテーブルが複数ある場合は所有者名を指定します。*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。

テーブル名に変数を使用することはできません。テーブル名は、データベース内で、および所有者に対してユニークでなければなりません。`set quoted_identifier on` を設定している場合は、テーブル名に区切り識別子を使用できます。指定していない場合は、その名前は識別子の規則に従ってください。有効なテーブル名の詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の「第 4 章 式、識別子、およびワイルドカード文字」にある「識別子」(374 ページ) を参照してください。

テーブル名をシャープ記号 (#) または "tempdb.." で始めることによって、テンポラリ・テーブルを作成できます。『リファレンス・マニュアル：ビルディング・ブロック』の「第 4 章 式、識別子、およびワイルドカード文字」にある「# で始まるテーブル (テンポラリ・テーブル)」(376 ページ) を参照してください。

ユーザが `sysusers` テーブルにリストされていて、別のデータベースの `create table` パーミッションを所有している場合は、そのデータベース内にテーブルを作成できます。たとえば、次のいずれかを使用して、`otherdb` データベース内に `newtable` という名前のテーブルを作成できます。

```
create table otherdb..newtable

create table otherdb.yourname.newtable
```

**column\_name**

テーブル内のカラム名です。カラム名は、テーブル内でユニークでなければなりません。`set quoted_identifier on`を設定している場合は、カラムに区切り識別子を使用できます。指定していない場合は、その名前は識別子の規則に従ってください。有効なカラム名の詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の「[第4章 式、識別子、およびワイルドカード文字](#)」を参照してください。

**datatype**

カラムのデータ型です。システム定義またはユーザ定義のデータ型を使用できます。特定のデータ型では、次のように、カッコで囲んで長さ (*n*) を指定しなければなりません。

`datatype (n)`

この他にも、次のように精度 (*p*) と位取り (*s*) をカッコで囲んで指定しなければならないデータ型もあります。

`datatype (p,s)`

詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の「[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。

Java がデータベースで使用可能な場合は、`datatype` を、データベースにインストールされている Java クラスの名前にすることができます。これは、システム・クラスでもユーザ定義のクラスでもかまいません。詳細については、『Adaptive Server Enterprise における Java』を参照してください。

**default**

カラムのデフォルト値を指定します。デフォルトを指定した場合、データの挿入時にそのカラムに値を指定しないと、Adaptive Server によってデフォルト値が挿入されます。デフォルトには、挿入を実行しているユーザ名を挿入する定数または組み込み、または null 値を挿入する null を指定できます。Adaptive Server は `tablename_colname_objid` という形式のデフォルト名を生成します。`tablename` はテーブル名の最初の 10 文字、`colname` はカラム名の最初の 5 文字、`objid` はデフォルトのオブジェクト ID 番号です。IDENTITY プロパティを持つカラムにデフォルトを宣言しても、カラムの値には影響しません。

データベース・オブジェクトを参照しない `create table` 文の `default` セクションでは、グローバル変数を参照できます。ただし、`create table` 文の `check` セクションではグローバル変数を使用できません。

**constant\_expression**

カラムのデフォルト値として使用する定数式です。グローバル変数、カラム名、他のデータベース・オブジェクト名は使用できませんが、データベース・オブジェクトを参照しない組み込み関数を使用できます。このデフォルト値は、カラムのデータ型と互換性がなければなりません。それ以外の場合は、そのデフォルトを挿入すると、Adaptive Server によってデータ型変換エラーが生成されます。

**user | null**

ユーザが値を指定しなかった場合に、Adaptive Server によってデフォルトとして挿入されるユーザ名または null 値を指定します。user を指定する場合、カラムのデータ型は char(30) または varchar(30) でなければなりません。null を指定する場合、null 値を使用できるカラムでなければなりません。

**identity**

カラムに IDENTITY プロパティがあることを示します。データベースの各テーブルに、次のデータ型の IDENTITY カラムを 1 つ作成できます。

- 位取り 0 の numeric (真数値)
- 任意の整数値データ型 (符号付きまたは符号なしの bigint、int、smallint、tinyint など)

IDENTITY カラムは更新できません。また、null 値を使用することもできません。

IDENTITY カラムは、請求書番号や従業員番号などの、Adaptive Server によって自動的に生成される連続番号を格納するために使用します。IDENTITY カラムの値は、テーブル内の各ローをユニークに識別します。

**null | not null**

デフォルトが存在しない場合の、データ挿入時の Adaptive Server の動作を指定します。

null は、ユーザが値を指定しなかった場合に Adaptive Server が null 値を割り当てるように指定します。

not null は、デフォルト値が存在しない場合にユーザが null 以外の値を指定するように指定します。

bit 型のカラムのプロパティは常に not null でなくてはなりません。

null も not null も指定しなかった場合は、デフォルトで not null が使用されます。ただし、sp\_dboption を使用してこのデフォルトを変更し、SQL 標準と互換性を持つようにすることができます。

**in row**

データ・ページに十分な領域がある場合は常にロー内としてデータを LOB カラムに保存するように Adaptive Server に指示します。LOB カラムのデータは完全なロー内または完全なロー外として保存されます。

**length**

(任意) LOB カラム・データをロー内として保存できる最大サイズを指定します。ページに十分な領域がある限り、この値より大きな値はロー外として保存され、*length* 以下の値はロー内として保存されます。

*length* を指定しない場合には、Adaptive Server によってデータベース全体の設定がロー内の長さに使用されます。

**off row**

(任意) LOB カラムをロー外として保存するためのデフォルトの動作を提供します。in row を指定しない限り、Adaptive Server では新しいテーブルに対してこの動作をデフォルトとして使用します。off row 句を指定せずにデータベース全体のロー内の長さを設定した場合は、create table によって LOB カラムがロー内の LOB カラムとして作成されます。

**off row | in row**

Java-SQL カラムが、ローから独立して格納されるか (off row)、ローに直接割り付けられた記憶領域に格納されるか (in row) を指定します。

デフォルト値は off row です。『Adaptive Server Enterprise における Java』を参照してください。

**size\_in\_bytes**

ロー内のカラムの最大サイズを指定します。ロー内に格納されているオブジェクトは、データベース・サーバのページ・サイズと他の変数に応じて、約 16KB までを使用できます。デフォルト値は 255 バイトです。

**constraint constraint\_name**

整合性制約の名前を指定します。

*constraint\_name* は検査制約の名前です。識別子の規則に従っていると同時に、データベース内でユニークでなければなりません。参照制約または検査制約の名前を指定しないと、Adaptive Server は *tablename\_colname\_objectid* という形式の名前を生成します。

- *tablename* - テーブル名の最初の 10 文字
- *colname* - カラム名の最初の 5 文字

- *objectid* - 制約のオブジェクト ID 番号

一意性制約またはプライマリ・キー制約の名前を指定しない場合は、Adaptive Server によって *tablename\_colname\_tabindid* という形式の名前が生成されます。*tabindid* はテーブル ID とインデックス ID を連結した文字列です。

#### unique

指定されたカラムの値を制限し、2つのローが同じ値を持たないようにします。この制約によって、[alter table](#) を使用して制約を削除した場合にのみ削除されるユニーク・インデックスが作成されます。

#### primary key

2つのローが同じ値を持たず、値が null 値にならないように、指定されたカラムの値に制約を加えます。この制約によって、[alter table](#) を使用して制約を削除した場合にのみ削除されるユニーク・インデックスが作成されます。

#### clustered | nonclustered

[unique](#) または [primary key](#) の各制約によって作成されたインデックスがクラスタード・インデックスまたはノンクラスタード・インデックスであることを指定します。[clustered](#) はプライマリ・キー制約のデフォルトで、[nonclustered](#) はユニーク制約のデフォルトです。クラスタード・インデックスは、テーブルごとに1つだけ持つことができます。詳細については、「[create index](#)」を参照してください。

#### asc | desc

制約に対して作成されたインデックスが、各カラムで昇順で作成されるか、降順で作成されるかを指定します。デフォルトは、昇順です。

#### fillfactor

Adaptive Server で既存のデータに新しいインデックスを作成する場合の各ページのデータの占有率を指定します。[fillfactor](#) の率は、インデックスの作成時にのみ使用されます。データは変更されるので、ページが特定の満杯率で維持されることはありません。

[fillfactor](#) のデフォルトは 0 です。これは、[create index](#) 文で [with fillfactor](#) を指定しない場合に使用されます ([sp\\_configure](#) で値を変更している場合は除きます)。[fillfactor](#) を指定する場合は、1 ~ 100 の値を使用してください。

`fillfactor` が 0 の場合、完全に満杯になるページにクラスタード・インデックスが作成され、完全に満杯になるリーフ・ページにノンクラスタード・インデックスが作成されます。これによって、クラスタード・インデックスとノンクラスタード・インデックスの両方のインデックス B ツリー内に十分な領域が確保されます。ほとんどの場合、`fillfactor` を変更する必要はありません。

`fillfactor` を 100 に設定すると、各ページが 100% 満杯の状態、クラスタード・インデックスとノンクラスタード・インデックスの両方が作成されます。`fillfactor` を 100 にするのは、データが追加されない読み込み専用テーブルに対してのみ効果があります。

`fillfactor` 値が 100 より小さい場合 (特殊な場合である 0 は除く)、満杯でないページを持つ新しいインデックスが作成されます。大量のデータが記録されることになるテーブルにインデックスを作成する場合は、`fillfactor` を 10 に設定するのが妥当です。ただし、`fillfactor` 値を小さくすると、それぞれのインデックス (またはインデックスとデータ) にさらに多くの記憶領域が必要になります。

コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して `fillfactor` を使用できません。

---

**警告!** `fillfactor` を使用してクラスタード・インデックスを作成すると、Adaptive Server はクラスタード・インデックスの作成時にデータを再分配するため、データが占有する記憶領域の総量に影響します。

---

`decrypt_default` は、暗号化カラムに `decrypt` パーミッションを持たないユーザに返す値を `sso` が指定できるようにします。`select` 文で取得される `text`、`image`、または `unitext` のカラムは復号化デフォルト値に置き換えられます。

#### `max_rows_per_page`

インデックスのデータ・ページとリーフレベル・ページ上のロー数を制限します。`fillfactor` とは異なり、`max_rows_per_page` 値はデータが挿入または削除されても保持します。

`max_rows_per_page` の値を指定しなかった場合、テーブルの作成時に値 0 が使用されます。テーブルとクラスタード・インデックスの値は 0 ~ 256 の値を使用してください。ノンクラスタード・インデックスの 1 ページ当たりの最大ロー数は、インデックス・キーのサイズで決まります。また、指定した値が大きすぎる場合は、エラー・メッセージが返されます。

`max_rows_per_page` 値を 0 に設定すると、満杯のデータ・ページを持つクラスタード・インデックスと、満杯のリーフ・ページを持つノンクラスタード・インデックスが作成されます。これによって、クラスタード・インデックスとノンクラスタード・インデックスの両方のインデックス B ツリー内に十分な領域が確保されます。

`max_rows_per_page` の値を小さい値に設定すると、頻繁にアクセスされるデータのロック競合を減らすことができます。ただし、小さい値に設定すると、完全にいっぱいではないページを持つ新しいインデックスも作成されます。また、使用される記憶領域が増えるため、ページ分割も多くなることがあります。

コンポーネント統合サービスが有効な場合にプロキシ・テーブルを作成すると、`max_rows_per_page` は無視されます。プロキシ・テーブルにはデータは格納されません。`max_rows_per_page` を使用してテーブルを作成し、そのテーブルを参照するプロキシ・テーブルを後で作成した場合、プロキシ・テーブルから `insert` または `delete` を実行すると、`max_rows_per_page` の制限が適用されます。

#### `reservepagegap = num_pages`

埋められたページと、エクステンツ I/O 割り付けオペレーション中に残される空ページの比率を指定します。指定した `num_pages` のそれぞれに、将来のテーブルの拡張に備えて空のページ 1 つが残されます。有効な値は 0 ~ 255 です。デフォルト値は 0 です。

#### `dml_logging = {full | minimal}`

`insert`、`update`、`delete` オペレーションといくつかの形式のバルク挿入に対するロギングの量を決定します。次のいずれかになります。

- `full` — すべてのトランザクションのログが取られます。
- `minimal` — ローまたはページの変更のログは取られません。

#### `deferred_allocation`

テーブルが必要とされるまで、テーブルまたはインデックスを作成を遅らせます。遅延テーブルは `insert` データベース内に作成されます。

#### `immediate_allocation`

`sp_dboption 'deferred table allocation'` を有効化している場合は、明示的にテーブルを作成します。

**on segment\_name**

**constraint** オプションを指定して使用する場合は、指定のセグメントに作成するインデックスを指定します。**on segment\_name** オプションを使用するには、事前に **disk init** を使用してデバイスを初期化する必要があります。また、**sp\_addsegment** を使用して、セグメントをデータベースに追加しておくことも必要です。データベースで使用できるセグメント名のリストについては、システム管理者に問い合わせるか、**sp\_helpsegment** を使用して参照してください。

**clustered** を指定し、**on segment\_name** オプションを使用した場合は、テーブル全体が指定したセグメントに移行します。これは、インデックスのリーフ・レベルに実際のデータ・ページがあるためです。

**references**

参照整合性の制約のカラム・リストを指定します。1つのカラム制約に指定できるカラムの値は、1つだけです。別のテーブルを参照するテーブルにこの制約を追加するには、参照元テーブルに挿入されたデータがすべて、参照先テーブルに事前に存在していなければなりません。

この制約を使用するには、参照先テーブルの **references** パーミッションが必要です。参照先テーブルの指定したカラムは、**unique** 制約または **create index** 文のどちらかで作成したユニーク・インデックスによって、制約を受けなければなりません。カラムが指定されていない場合は、参照先テーブルの適切なカラムに **primary key** 制約がなければなりません。また、参照元テーブルのカラムのデータ型は、参照先テーブルのカラムのデータ型に一致していなければなりません。

**ref\_table**

参照されるカラムを含むテーブル名です。別のデータベースのテーブルを参照することもできます。制約では、192 までのユーザ・テーブルと内部生成のワークテーブルを参照できます。

**ref\_column**

参照されるテーブルのカラム名です。

**match full**

参照元ローの参照元カラムのすべての値が以下の場合、次のようになることを指定します。

- **null** — 参照整合性条件は真である。
- **null** 以外の値 — 対応するそれぞれのカラムが参照先テーブルで等しい参照先ローがある場合、参照整合性条件は真である。

どちらでもなければ、以下の場合に参照整合性条件は偽になります。



- すべての値が null 以外で、等しくない場合、または
- 参照元ローの参照元カラムに null 以外の値と null がある場合

#### check (search\_condition)

カラム値の check 制約と、Adaptive Server がテーブル内のローすべてに実行する *search\_condition* 制約を指定します。check 制約 ( 検査制約 ) は、テーブルまたはカラムの制約として指定できます。*create table* を使用すると、カラム定義の中で複数の check 制約を指定できます。

*create table* 文の default セクションではグローバル変数を参照できませんが、check セクションでは使用できません。

制約は、次のものを含むことができます。

- in によって導入される定数式のリスト
- ワイルドカード文字を含む、like で提供される条件の集合

カラムの検査制約およびテーブルの検査制約は、テーブル内のすべてのカラムを参照できます。

式には、算術演算子と関数を指定できます。*search\_condition* には、サブクエリ、集合関数、ホスト変数、またはパラメータは指定できません。

#### encrypt [with key\_name]

暗号化カラムを作成します。キーが別のデータベースにある場合に指定します。*key\_name* がデータベースに対してユニークでない場合は、所有者名を指定します。*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。

テーブルの作成者は、キーの select パーミッションが必要です。*key\_name* を指定しない場合は、Adaptive Server はデータベース内でデフォルトキーを捜します。

*keyname* は、*create encryption key* を使用して作成するキーを指定します。テーブルの作成者には、*keyname* に対する select パーミッションが必要です。*keyname* を指定しないと、Adaptive Server は、*create encryption key* または *alter encryption key* の *as default* 句を使用して作成されたデフォルト・キーを探します。

サポートされているデータ型のリストについては、『暗号化カラム・ユーザズ・ガイド』の「データの暗号化」を参照してください。

**decrypt\_default constant\_expression**

このカラムでは **decrypt** パーミッションのないユーザにデフォルト値を返すことを指定します。**constant\_expression** は **select** 文を実行したときに、復号化された値の代わりに **Adaptive Server** から返される固定値です。使用可能な値は **NULL** を入力可能な列の **NULL** だけです。復号化されたデフォルト値をカラムのデータ型に変換できない場合は、**Adaptive Server** でクエリが実行されたときに変換エラーが検出されます。

**compression = compression\_level | not compressed**

ロー内のデータが圧縮されるかどうか、および圧縮されている場合のレベルを示します。

**foreign key**

リストしたカラムが、そのテーブルの外部キーであることを指定します。対象となるキーは、後に続く **references** 句にリストしたカラムです。外部キーの構文は、カラムレベルでの制約ではなく、テーブルレベルでの制約だけで使用できます。

**next\_column | next\_constraint**

カラム定義またはテーブル制約の定義に記述されたものと同じ構文を使用して、カラム定義やテーブル制約を (カンマで区切って) 追加できることを示します。

**lock datarows | datapages | allpages**

テーブルに使用されるロック・スキームを指定します。デフォルトは、サーバ全体に対する設定パラメータ **lock scheme** の設定です。

**exp\_row\_size = num\_bytes**

予想されるロー・サイズを指定します。**datarows** と **datapages** のロック・スキーム、および可変長ローを持つテーブルにのみ適用されます。有効な値は 0、1、およびテーブルのローの最小長および最大長の間の任意の値です。デフォルト値は 0 で、サーバワイドな設定が適用されることを示します。

**identity\_gap value**

テーブルの **identity** ギャップを指定します。この値は、このテーブルのシステムの **identity** ギャップ設定だけを上書きします。

**value identity** ギャップの値です。**identity** ギャップの設定の詳細については、「[IDENTITY カラム](#)」(276 ページ) を参照してください。

**transfer table [on | off]**

増分転送に対するテーブルにマーク付けします。このパラメータのデフォルト値は **off** です。

### dml\_logging

insert、update、delete オペレーションといくつかの形式のバルク挿入に対するロギングの量を決定します。次のいずれかになります。

- full — すべてのトランザクションのログが取られます。
- minimal — ローまたはページの変更のログは取られません。

### compression

テーブル・レベルまたはパーティション・レベルの圧縮レベルを示します。パーティションの圧縮レベルを指定すると、テーブルの圧縮レベルが上書きされます。Adaptive Server では圧縮が設定されているパーティションのみで個々のカラムが圧縮されます。

- none — このテーブルまたはパーティションのデータは圧縮されません。パーティションの場合、none は、テーブル圧縮が row 圧縮または page 圧縮に変更されても、このパーティションのデータが圧縮されないままであることを意味します。
- row — 個々のロー内の 1 つ以上のデータ項目を圧縮します。Adaptive Server は、非圧縮形式に比べ、圧縮形式でのデータの格納により記憶領域を節約できる場合にのみ row 圧縮形式でデータを格納します。パーティション・レベルまたはテーブル・レベルで row 圧縮を設定します。
- page — ページがいっぱいになると、ロー圧縮された既存のデータ・ローが、ページ・レベルの圧縮を使用して圧縮され、ページ・レベルの辞書、インデックス、文字コードのエントリが作成されます。パーティション・レベルまたはテーブル・レベルで page 圧縮を設定します。

Adaptive Server では、データがロー・レベルで圧縮されてからしかページ・レベルで圧縮されないため、圧縮レベルを page に設定することは、page および row の両方の圧縮を意味します。

### lob\_compression = off | compression\_level

テーブルの圧縮レベルを決定します。off を選択した場合、テーブルには LOB 圧縮は使用されません。

**compression\_level**

テーブルの圧縮レベル。次の圧縮レベルがあります。

- 0 – ローは圧縮されません。
- 1～9 – Adaptive Server は ZLib 圧縮を使用します。通常、圧縮の数値が大きいほど、Adaptive Server での LOB データの圧縮レベルが高くなり、圧縮データと非圧縮データの比率も大きくなります (つまり、非圧縮データと比べて、圧縮データによる記憶領域の節約量 (バイト) が多くなります)。

ただし、圧縮の量は LOB の内容によって異なります。圧縮レベルが高いほど、圧縮に使用される CPU の量が多くなります。そのためレベル 9 を使用すると、圧縮率が最高になりますが、CPU の使用量も最高になります。

- 100 – Adaptive Server は FastLZ 圧縮を使用します。CPU 使用量が最も少ない圧縮率です。通常、短いデータに使用します。
- 101 – Adaptive Server は FastLZ 圧縮を使用します。値に 101 を指定すると、100 を指定した場合よりも、CPU の使用量は若干多くなりますが、圧縮率は良くなります。

圧縮アルゴリズムでは、LOB データを使用しないローは無視されます。

**on segment\_name**

テーブルを配置するセグメント名を指定します。on segment\_name を使用する場合は、事前に [create database](#) または [alter database](#) を使用して論理デバイスがデータベースに割り当てられている必要があります。また、[sp\\_addsegment](#) を使用してセグメントがデータベースに作成されていることも必要です。データベースで使用できるセグメント名のリストについては、システム管理者に問い合わせるか、[sp\\_helpsegment](#) を使用して参照してください。

パーティションに使用するとき、パーティションを配置するセグメントを指定します。

**external table**

オブジェクトがリモートのテーブルまたはビューであることを指定します。external table がデフォルトであるため、これを指定するかどうかは任意です。

**for load**

bcp in にのみ使用可能なテーブルと [alter table unpartition](#) オペレーションを作成します。for load を使用して作成したテーブルでは、[row\\_count\(\)](#) を使用できます。

**partition by range**

分割するレコードを、分割カラム内の指定された値の範囲に基づいて指定します。

**column\_name**

*partition\_clause* で使用する場合にパーティション・キー・カラムを指定します。

**partition\_name**

テーブル・レコードを保管する新しいパーティションの名前を指定します。テーブル内またはインデックス内にすでに存在するパーティションと同じ名前は指定できません。**set quoted\_identifier** オプションを設定した場合、パーティション名に区切り識別子を使用できます。それ以外の場合、パーティション名は有効な識別子でなければなりません。

*partition\_name* を省略すると、*table\_name\_partition\_id* という形式の名前が作成されます。最大長を超える長さのパーティション名は、トラケートされます。

**on segment\_name**

*partition\_clause* をパーティションに使用するときは、パーティションを配置するセグメントを指定します。**on segment\_name** オプションを使用するには、**disk init** を使用して事前にデバイスを初期化してください。また、システム・プロシージャ **sp\_addsegment** を使用して、セグメントをデータベースに追加しておきます。データベースで使用できるセグメント名のリストについては、システム管理者に問い合わせるか、**sp\_helpsegment** を使用して参照してください。

**values <= constant | MAX**

指定したパーティションの上限値を指定します。パーティションの上限に定数値 (**constant**) を指定すると、テーブルに暗黙的な一意性制約が適用されます。キーワード **MAX** を使用すると、特定のデータ型の最大値が指定されます。

**partition by hash**

-レコードの分割に、システムから提供されるハッシュ関数を使用します。ハッシュ関数は、レコードが割り当てられるパーティションを指定するパーティション・キー値を計算します。

**partition by list**

分割するレコードを、指定したカラム内のリテラル値に基づいて指定します。リスト分割テーブルを分割できるのは、1つのカラムだけです。各パーティションには、最大で 250 個の異なるリスト値を指定できます。

**partition by round-robin**

レコードの分割を順次処理で行うことを指定します。ラウンドロビン分割テーブルに分割キーはありません。ユーザおよびオブティマイザは、特定のレコードがどのパーティションに配置されるかを確認することはできません。

**at pathname**

リモート・オブジェクトのロケーションを指定します。at *pathname* 句を使用すると、プロキシ・テーブルが作成されます。

*pathname* は、*server\_name.dbname.owner.object;aux1.aux2* という形式をとります。ここで、

- *server\_name* (必須) — リモート・オブジェクトがあるサーバの名前です。
- *dbname* (任意) — このオブジェクトがあるリモート・サーバが管理するデータベースの名前です。
- *owner* (任意) — リモート・オブジェクトを所有するリモート・サーバのユーザの名前です。
- *object* (必須) — リモートのテーブルまたはビューの名前です。
- *aux1.aux2* (任意) — create table コマンドまたは create index コマンドの実行中にリモート・サーバに渡される文字列です。この文字列は、サーバがクラス *db2* の場合にのみ使用されます。*aux1* はテーブルを配置する DB2 データベース、*aux2* はテーブルを配置する DB2 テーブル領域です。

**{compute | as}**

カラムが計算カラムであることを示すために使用できる予約キーワードで、どちらも意味は同じです。

**computed\_column\_expression**

他のテーブルのカラム、ローカル変数、集合関数、またはサブクエリを含まない、有効な T-SQL 式です。カラム名、定数、関数、グローバル変数、case 式のいずれか、またはこれらを 1 つ以上の演算子によって結合した組み合わせを指定できます。仮想計算カラムから実体化された計算カラムを参照する場合を除き、計算カラム間での相互参照は許可されません。

**materialized | not materialized**

計算カラムを実体化してテーブルに物理的に保管するかどうかを指定します。どちらのキーワードも指定しない場合は、デフォルトで計算カラムは not materialized であり、テーブルに物理的に保管されません。

**using clustered**

仮想ハッシュ・テーブルを作成することを示します。カラムのリストは、このテーブルのキー・カラムとして扱われます。

**column\_name [asc | desc]**

ローはそのハッシュ関数に基づいて配置されるため、ハッシュ領域に [asc | desc] を使用することはできません。仮想ハッシュ・テーブルのキー・カラムに順序を指定した場合、オーバフロー・クラスタード領域でのみ使用されます。

**hash\_factor**

仮想ハッシュ・テーブルのハッシュ関数に対して必要です。ハッシュ関数の場合、キー・カラムごとにハッシュ係数が必要です。これらの係数はキー値とともに使用され、特定のローにハッシュ値を生成します。

**with max\_num\_hash\_values key**

使用できるハッシュ値の最大数。このハッシュ関数の出力における上限を定義します。

例

**例 1** @@spid グローバル変数をデフォルトのパラメータで使用して foo テーブルを作成します。

```
create table foo (
    a      int
    , b      int      default @@spid
)
```

**例 2** titles テーブルを作成します。

```
create table titles (
    title_id      tid      not null
    , title       varchar (80)  not null
    , type        char (12)    not null
    , pub_id      char (4)     null
    , price       money       null
    , advance     money       null
    , total_sales int         null
    , notes       varchar (200) null
    , pubdate     datetime    not null
    , contract    bit         not null
)
```

**例 3** for load を使用して、テーブル mytable を作成します。

```
create table mytable (
    col1      int
    , col2      int
    , col3      (char 50)
```

```
)
partitioned by roundrobin 3 for load
```

新しいテーブルは、分割解除されないかぎり、どのようなユーザ・アクティビティにも使用できません。

- 1 `bcp in` を使用して、データを `mytable` にロードします。
- 2 `mytable` を分割解除します。

これで、テーブルはどのユーザ・アクティビティにも使用できます。

**例 4** `compute` テーブルを作成します。テーブル名とカラム名である `max` と `min` は予約語であるため、二重引用符で囲まれています。`total score` のカラム名は、埋め込みブランクを含んでいるため、二重引用符で囲まれています。このテーブルを作成する前に、`set quoted_identifier on` を設定してください。

```
create table "compute" (
    "max"          int
    , "min"        int
    , "total score" int
)
```

**例 5**一意性制約を使用して `sales` テーブルとクラスタード・インデックスを一度に作成します (`pubs2` データベースのインストール・スクリプトには、別々の `create table` 文と `create index` 文があります)。

```
create table sales (
    stor_id      char (4)      not null
    , ord_num    varchar (20)  not null
    , date       datetime     not null
    , unique clustered (stor_id, ord_num)
)
```

**例 6** 2つの参照整合性の制約と1つのデフォルト値を持つ `salesdetail` テーブルを作成します。`salesdet_constr` という名前前のテーブルレベルの参照整合性の制約があり、`title_id` カラムには、名前が指定されていないカラムレベルの参照整合性の制約があります。どちらの制約も、参照先テーブル (`titles` と `sales`) にユニーク・インデックスがあるカラムを指定します。`qty` カラムに付けた `default` 句は、デフォルト値として 0 を指定します。

```
create table salesdetail (
    stor_id      char (4)      not null
    , ord_num    varchar (20)  not null
    , title_id   tid           not null
    , references titles (title_id)
    , qty       smallint default 0      not null
)
```



```

, discount float not null,

constraint salesdet_constr
foreign key (stor_id, ord_num)
references sales (stor_id, ord_num)
)

```

**例 7** 検査制約がある publishers テーブルを pub\_id カラムに作成します。このカラムレベルの制約は、pubs2 データベースに含まれる pub\_idrule の代わりに使用できます。

```

create rule pub_idrule
as @pub_id in ("1389", "0736", "0877", "1622", "1756")
or @pub_id like "99[0-9][0-9]"

create table publishers (
pub_id char (4) not null
check (pub_id in ("1389", "0736", "0877", "1622",
"1756")
or pub_id like "99[0-9][0-9]")
, pub_name varchar (40) null
, city varchar (20) null
, state char (2) null
)

```

**例 8** sales\_daily テーブルの IDENTITY カラムとして、ord\_num カラムを指定します。テーブルに最初にローを挿入するときに、Adaptive Server は IDENTITY カラムに 1 の値を割り当てます。これ以降ローを挿入するたびに、カラムの値は 1 ずつ増えていきます。

```

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

```

**例 9** new\_titles テーブルの datapages ロック・スキームと予期される 200 のロー・サイズを指定します。

```

create table new_titles (
title_id tid
, title varchar (80) not null
, type char (12)
, pub_id char (4) null
, price money null
, advance money null
, total_sales int null
, notes varchar (200) null
, pubdate datetime
)

```

```
        , contract          bit
    )
    lock datapages
    with exp_row_size = 200
```

**例 10 datarows** ロック・スキームを指定し、`reservepagegap` の値を 16 に設定して、エクステント I/O 操作によって 15 の埋められたページにつき 1 つの空のページが残されるようにします。

```
create table new_publishers (
    pub_id      char (4)      not null
    , pub_name  varchar (40)  null
    , city      varchar (20)  null
    , state     char (2)      null
)
lock datarows
with reservepagegap = 16
```

**例 11** 最低限のロギングが設定された `big_sales` というテーブルを作成します。

```
create table big_sales (
    storid      char(4)      not null
    , ord_num   varchar(20)  not null
    , order_date datetime    not null
)
with dml_logging = minimal
```

**例 12** `im_not_here_yet` という名前の遅延テーブルを作成します。

```
create table im_not_here_yet (
    col_1 int,
    col_2 varchar(20)
)
with deferred_allocation
```

**例 13** ロック・スキームにデータローを使用し、増分転送を使用できる `mytable` というテーブルを作成します。

```
create table mytable (
    f1 int
    , f2 bigint not null
    , f3 varchar (255) null
)
lock datarows
with transfer table on
```

**例 14** ローレベル圧縮を含んだ `genre` という名前のテーブルを作成します。

```

create table genre (
    mystery      varchar(50)    not null
  , novel       varchar(50)    not null
  , psych       varchar(50)    not null
  , history     varchar(50)    not null
  , art         varchar(50)    not null
  , science     varchar(50)    not null
  , children    varchar(50)    not null
  , cooking     varchar(50)    not null
  , gardening   varchar(50)    not null
  , poetry      varchar(50)    not null
)
with compression = row

```

**例 15** seg1、seg2、seg3 のセグメントに sales という名前のテーブルを作成し、seg1 を圧縮します。

```

create table sales (
    store_id     int      not null
  , order_num   int      not null
  , date        datetime not null
)
partition by range (date)
( Y2008 values <= ('12/31/2008')
  with compression = page on seg1,
  Y2009 values <= ('12/31/2009') on seg2,
  Y2010 values <= ('12/31/2010') on seg3)

```

**例 16** 5 の LOB 圧縮レベルを使用する email テーブルを作成します。

```

create table email (
    user_name    char (10)
  , mailtxt     text
  , photo       image
  , reply_mails text)
with lob_compression = 5

```

**例 17** ユニークなクラスタード・インデックスによってサポートされる制約を作成します。インデックスの順序は、stor\_id では昇順、ord\_num では降順です。

```

create table sales_south (
    stor_id      char (4)      not null
  , ord_num     varchar (20)  not null
  , date        datetime     not null
  , unique clustered (stor_id asc, ord_num desc)
)

```

**例 18** リモート・サーバ `SERVER_A` で `t1` という名前のテーブルを作成し、リモート・テーブルにマップされる `t1` という名前のプロキシ・テーブルを作成します。

```
create table t1 (  
    a    int  
    , b  char (10)  
)  
at "SERVER_A.db1.joe.t1"
```

**例 19** `employees` という名前のテーブルを作成します。 `name` は `varchar` 型、 `home_addr` は `Address` 型の `Java-SQL` カラム、 `mailing_addr` は `Address2Line` 型の `Java-SQL` カラムです。 `Address` と `Address2Line` は、いずれもデータベースにインストールされている `Java` クラスです。

```
create table employees (  
    name          varchar (30)  
    , home_addr   Address  
    , mailing_addr Address2Line  
)
```

**例 20** `identity` カラムがある `mytable` という名前のテーブルを作成します。 `identity` ギャップは `10` に設定されています。これは、メモリ内に `10` ブロック単位で `ID` 番号が割り付けられていることを示します。サーバに障害が発生したり、ウェイトなしでシャットダウンしたりすると、ローに割り当てられた最後の `ID` 番号と次の `ID` 番号との最大のギャップは `10` になります。

```
create table mytable (  
    IdNum    numeric (12,0)    identity  
)  
with identity_gap = 10
```

**例 21** テーブル `my_publishers` を作成します。このテーブルは、`state` カラム内の値に基づくリストによって分割されます。テーブル・パーティション作成の詳細については、『`Transact-SQL ユーザーズ・ガイド`』を参照してください。

```
create table my_publishers (  
    pub_id      char (4)        not null  
    , pub_name  varchar (40)    null  
    , city      varchar (20)    null  
    , state     char (2)        null  
)  
partition by list (state) (  
    west values ('CA', 'OR', 'WA') on seg1  
    , east values ('NY', 'MA') on seg2  
)
```

**例 22** `fictionsales` テーブルを作成します。このテーブルは、`date` カラムの値を基準にした範囲によって分割されます。

```
create table fictionsales (
    store_id      int          not null
  , order_num    int          not null
  , date         datetime     not null
)
partition by range (date) (
    q1 values <= ("3/31/2005") on seg1
  , q2 values <= ("6/30/2005") on seg2
  , q3 values <= ("9/30/2005") on seg3
  , q4 values <= ("12/31/2005") on seg4
)
```

**例 23** テーブル `currentpublishers` を作成します。このテーブルは、ラウンドロビンによって分割されます。

```
create table currentpublishers (
    pub_id       char (4)      not null
  , pub_name     varchar (40)  null
  , city        varchar (20)  null
  , state       char (2)      null
)
partition by roundrobin 3 on (seg1)
```

**例 24** `mysalesdetail` テーブルを作成します。このテーブルは、`ord_num` カラムの値に基づくハッシュによって分割されます。

```
create table mysalesdetail (
    store_id     char (4)      not null
  , ord_num     varchar (20)   not null
  , title_id    tid           not null
  , qty        smallint      not null
  , discount   float         not null
)
partition by hash (ord_num) (
    p1 on seg1
  , p2 on seg2
  , p3 on seg3
)
```

**例 25** 実体化された計算カラムが 1 つある `mytitles` というテーブルを作成します。

```
create table mytitles (
    title_id     tid          not null
  , title       varchar (80)  not null
  , type       char (12)     not null
)
```

```

, pub_id      char (4)          null
, price       money             null
, advance     money             null
, total_sales int              null
, notes       varchar (200)     null
, pubdate     datetime          not null
, sum_sales   compute price * total_sales
              materialized
)

```

**例 26** NULL 入力可能な暗号化カラムがある `employee` テーブルを作成します。Adaptive Server では、データベースのデフォルト暗号化キーを使用して `ssn` データを暗号化します。

```

create table employee_table (
    ssn          char(15)      null
    encrypt name char(50)
    , deptid     int
)

```

**例 27** クレジット・カード・データの暗号化カラムがある `customer` テーブルを作成します。

```

create table customer (
    ccard char(16) unique
    encrypt with cc_key
    decrypt_default 'XXXXXXXXXXXXXXXXXX', name char(30)
)

```

`ccard` カラムには一意性制約があり、暗号化に `cc_key` を使用します。`decrypt_default` 指定子が使用されているので、`decrypt` パーミッションを持たないユーザが `ccard` カラムを選択すると、Adaptive Server は実際の値ではなく値 `'XXXXXXXXXXXXXXXXXX'` を返します。

**例 28** 長さが 300 バイトのロー内の LOB カラムとして `description` を、長さの指定がない (ロー外の記憶領域のサイズを継承) ロー内の LOB カラムとして `notes` を、条件に関係なくロー外に格納されているカラムとして `reviews` を指定するテーブルを作成します。

```

create table new_titles (
    title_id      tid          not null
    , title       varchar (80) not null
    , type        char (12)    null
    , price       money        null
    , pubdate     datetime     not null
    , description text         in row (300)
    , notes       text         in row
    , reviews    text         off row
)

```

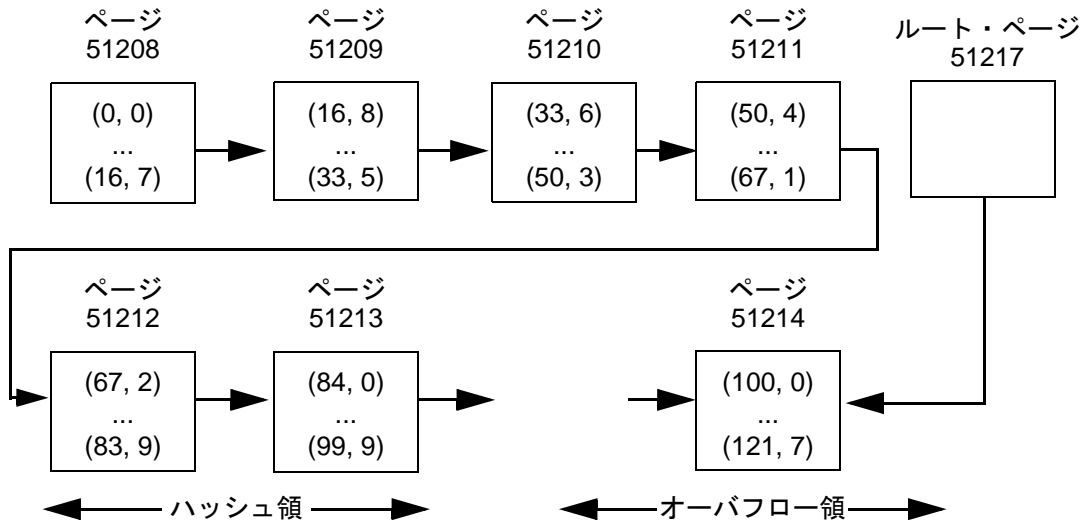
**例 29** order\_seg セグメントの pubs2 データベースに orders という仮想ハッシュ・テーブルを作成します。

```
create table orders(
    id      int
  , age    int
  , primary key using clustered (id,age) = (10,1) with
max 1000 key
)
on order_seg
```

データのレイアウトは次のとおりです。

- order\_seg セグメントはページ ID 51200 から始まります。
- 最初のデータのオブジェクトアロケーション・マップ (OAM) ページの ID は 51201 です。
- 1 ページあたりの最大ロー数は 168 です。
- ローのサイズは 10 です。
- オーバフロー・クラスタド領域のルート・インデックス・ページは 51217 です。

図 1-1: 例のデータ・レイアウト



**例 30** order\_seg セグメントの pubs2 データベースに orders という仮想ハッシュ・テーブルを作成します。

```

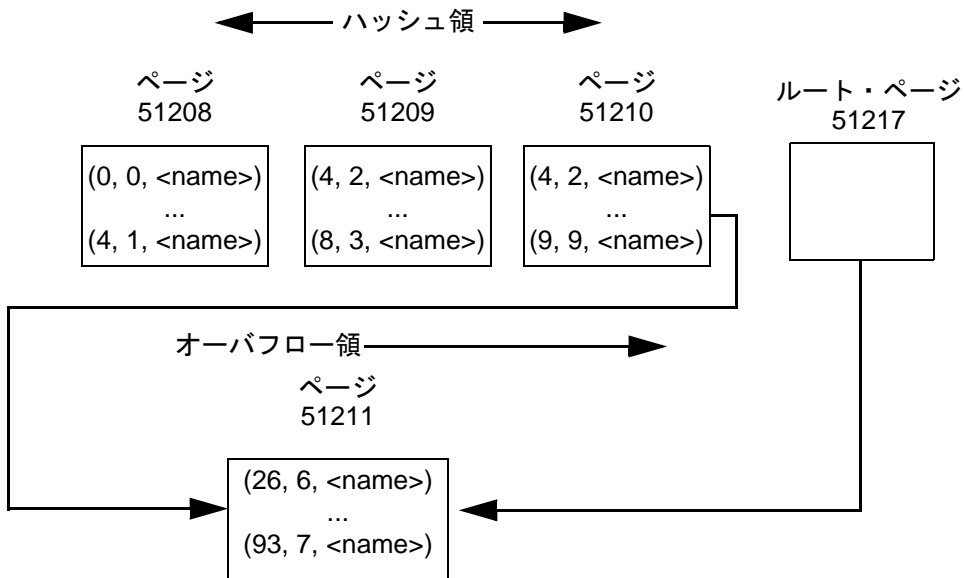
create table orders(
    id    int default NULL
    , age  int
    , primary key using
        clustered (id,age) = (10,1) with max 100 key
    , name  varchar(30)
)
on order_seg

```

データのレイアウトは次のとおりです。

- order\_seg セグメントはページ ID 51200 から始まります。
- 最初のデータ OAM ページの ID は 51201 です。
- 1 ページあたりの最大ロー数は 42 です。
- ローのサイズは 45 です。
- オーバフロー・クラスタド領域のルート・インデックス・ページは 51217 です。

図 1-2: 例のデータ・レイアウト





## 使用法

- `create table` は、テーブルを作成するほか、必要に応じて整合性制約を作成します。`create table` 文内で別のデータベースを指定しないかぎり、テーブルは現在オープンしているデータベース内に作成されます。ユーザが `sysusers` テーブルにリストされていて、別のデータベースの `create table` パーミッションを所有している場合は、そのデータベース内にテーブルまたはインデックスを作成できます。
- 領域は、一度に 1 エクステント、すなわち 8 ページ単位でテーブルおよびインデックスに割り付けられます。エクステントが満杯になるたびに、別のエクステントが割り付けられます。割り付けられた領域とテーブルに使用されている領域の量を確認するには、`sp_spaceused` を使用してください。
- ロー内の Java カラムの最大長は、テーブルのスキーム、ロック方式、ページ・サイズに対応した可変長カラムの最大サイズによって決まります。
- `create table` はテーブルを作成する前に検査制約のエラー・チェックを実行します。
- カラムの定義が `char (n) NULL` のときにコンポーネント統合サービスから `create table` を実行すると、コンポーネント統合サービスはリモート・サーバ上に `varchar (n)` のカラムを作成します。
- インデックスのソート順を指定するには、インデックス・カラムの名前の後に `asc` または `desc` キーワードを使用します。カラムが、クエリの `order by` 句で指定された順序と同じ順序になるようにインデックスを作成すると、クエリ処理中のソートする手順を省略できます。
- アプリケーションによって短いローがデータオンリーロック・テーブルに挿入され、後の更新でローの長さが増大する場合は、`exp_row_size` を使用してデータオンリーロック・テーブルのローが新しいロケーションに転送される回数を少なくします。
- `at` キーワードによって得られるロケーション情報は、`sp_addobjectdef` によって得られる情報と同じです。この情報は、`sysattributes` テーブルに格納されます。

## 制限事項

- テーブル内のカラムの最大数は、カラムの幅とサーバの論理ページ・サイズによって異なります。
  - カラムのサイズの合計は、サーバの論理ページ・サイズ以下である。

- テーブルごとのカラムの最大数は 1024 である。
- 全ページロック・テーブルの可変長カラムの最大数は 254 である。

たとえば、サーバの論理ページ・サイズが 2K で、整数カラムのテーブルがある場合、テーブル内のカラムの最大数は 1024 よりもはるかに少なくなります (1024 \* 4 バイトは、2K という論理ページ・サイズを超えます)。

カラムの最大数が 1024 を超えないかぎり、可変長カラムと固定長カラムを 1 つのテーブルに混在させることができます。たとえば、サーバの論理ページ・サイズが 8K の場合、APL 用に設定されたテーブルには、null を入力できる 254 の整数カラム (可変長カラム) と null を入力できない 770 の整数カラム、合計 1024 カラムを作成できます。

- 1 つのデータベースには最大 20 億のテーブルを、1 つのテーブルには最大 1024 のユーザ定義のカラムを作成できます。1 テーブルあたりのロー数は、使用できる記憶領域のみによって制限されます。
- Adaptive Server では、次の場合、テーブルは作成されますが、データ操作言語による操作を行うとサイズ制限に関するエラーが表示されます。
  - 可変長カラムがあるローの合計ロー・サイズが、カラムの最大サイズを超えている場合。
  - 1 つの可変長カラムの長さが、カラムの最大サイズを超えている場合。
  - データオンリーロック・テーブルでは、開始カラムを除いた可変長カラムのオフセットが 8191 バイトの制限を超えている場合。
- 固定長カラムのすべてとローのオーバーヘッドの合計サイズが、テーブルのロック・スキームとページ・サイズによって許容されるサイズを超えている場合は、Adaptive Server によってエラーが報告されます。表 1-10 は、これらの制限を示します。

表 1-10: 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 バイト	1960 バイト
	4K (4096 バイト)	4010 バイト	4008 バイト
	8K (8192 バイト)	8106 バイト	8104 バイト
	16K (16384 バイト)	16298 バイト	16296 バイト
DOL テーブル	2K (2048 バイト)	1964 バイト	1958 バイト
	4K (4096 バイト)	4012 バイト	4006 バイト
	8K (8192 バイト)	8108 バイト	8102 バイト
	16K (16384 バイト)	16300 バイト	16294 バイト テーブルに可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、ローのオーバーヘッドの 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。

- ローごとの可変長データの最大バイト数は、テーブルのロック・スキームによって異なります。

ページ・サイズ	ローの最大長	カラムの最大長
2K (2048 バイト)	1962	1960
4K (4096 バイト)	4010	4008
8K (8192 バイト)	8096	8104
16K (16384 バイト)	16298	16296

## DOL テーブルの最大カラム・サイズ

ページ・サイズ	ローの最大長	カラムの最大長
2K (2048 バイト)	1964	1958
4K (4096 バイト)	4012	4006
8K (8192 バイト)	8108	8102
16K (16384 バイト)	16300	16294

- オフセットが 8191 バイトを超える可変長カラムを使用して DOL テーブルを作成した場合は、ローをカラムに追加できません。

- `varchar`、`nvarchar`、`univarchar`、または `varbinary` の各カラムを使用して、定義された幅の合計が許可されている最大ロー・サイズより大きいテーブルを作成すると、警告メッセージが表示されますが、テーブルは作成されます。このようなローに最大バイト数を超えて挿入を行うか、ローに `update` を実行してロー・サイズの合計が最大長を超えるようにしたりすると、**Adaptive Server** によってエラー・メッセージが表示され、コマンドは正常に動作しません。
- `if..else` ブロックまたは `while` ループ内で `create table` コマンドを実行すると、**Adaptive Server** では、その条件が真かどうかを判断する前に、そのテーブルのスキーマが作成されます。このため、そのテーブルがすでに存在する場合は、エラーが発生することがあります。このようなエラーが発生しないようにするには、データベースに同じ名前のビューがないことを確認するか、または `execute` 文を次のように使用してください。

```
if not exists
    (select * from sysobjects where name="my table")
begin
    execute "create table mytable (x int)"
end
```

- デフォルト宣言または検査制約を付けて `create table` を発行してから、同じバッチまたはプロシージャ内でテーブルにデータを挿入することはできません。`create` 文と `insert` 文を2つの異なるバッチまたはプロシージャに分けるか、`execute` を使用してアクションを別々に実行してください。
- デフォルトを含む次の変数を `create table` 文で使用することはできません。

```
declare @p int
select @p = 2
create table t1 (c1 int default @p, c2 int)
```

このようにすると、「デフォルトの中で変数は使用できません」というエラー・メッセージ 154 が表示されます。

- 仮想ハッシュ・テーブルには次の制限があります。

SQL ユーザ-定義関数は現時点では `create proxy table`、`create table at remote server`、または `alter table` ではサポートされていません。

---

**注意** SQL 関数の実行に必要な構文は `username.functionname()` です。

---

- 仮想ハッシュ・テーブルにはユニークなローが必要である。仮想ハッシュ・テーブルで複数のローが同じキー・カラム値を持つことはできない。Adaptive Server では、あるローをハッシュ領域に保持し、同じキー・カラム値を持つ別のローをオーバフロー・クラスタド領域に保持することができないためである。
- 各仮想ハッシュ・テーブルは、排他セグメント上で作成する必要がある。

#### 圧縮テーブルの作成

- 特に指定をしない限り、Adaptive Server の動作は以下のようになります。
  - テーブルの作成時にデータ圧縮を null に設定する。
  - テーブルの変更時に既存の圧縮レベルを保持する。
  - すべてのパーティションを、create table 句で指定された圧縮レベルに設定する。
- テーブルをテーブルレベル圧縮で作成することはできますが、一部のパーティションは非圧縮のままにしてください。それによって、アクティブなパーティション・フォーマットの非圧縮データを保持し、必要に応じてデータを定期的に圧縮できるようになります。
- Adaptive Server では、ラウンドロビン・パーティション以外のあらゆる形式の分割に対し、パーティションレベルの圧縮をサポートしています。
- not compressed のマーク付けがあるカラムは、ローまたはページの圧縮の対象として選択されません。ただし、ロー内のカラム (マテリアライズされた計算カラムを含む) は圧縮できます。
  - 4 バイト未満のすべての固定長データをロー・レベルで圧縮することはできない。ただし、Adaptive Server ではページ・インデックス圧縮中にこれらのデータ型を圧縮する可能性がある。
  - 4 バイト以上の固定長または可変長のすべてのデータは、ロー・レベルで圧縮できる。
- デフォルトでは、Adaptive Server は非圧縮のマテリアライズされていない計算カラムを作成します。

- まず **Adaptive Server** では、ロー・レベルで圧縮可能なカラムを圧縮します。圧縮ローの長さが非圧縮ローよりも長い場合、**Adaptive Server** では圧縮ローを破棄し、非圧縮ローをディスクに格納して、圧縮によって領域を無駄にしないようにします。
- データ・ページには、圧縮データ・ローと非圧縮データ・ローが同時に含まれている場合があります。
- 固定長カラムは圧縮できます。
- **with exp\_row\_size** 句を使用すると、固定長のローに対してのみ圧縮データオンリー・ロック (DOL) テーブルを作成できます。**with exp\_row\_size** 句を使用して、全ページロック (APL) テーブルを作成することはできません。
- 予測ロー・サイズを指定した場合に、非圧縮ローの長さが予測ロー・サイズよりも小さいときは、そのローは圧縮されません。
- テーブルの圧縮を有効にすると、テーブルで実行されるすべての **bcp** オペレーションと **DML** オペレーションによって、データが圧縮されます。
- 圧縮を使用すると、ページにさらに多くのローを格納できますが、テーブルの最大ロー・サイズは変わりません。ただし、テーブルの有効最小ロー・サイズは変わる場合があります。
- ロー圧縮またはページインデックス圧縮が可能であっても、その性質により圧縮を適用できないか、圧縮する意味がないカラム (bit データ型、暗号化を使用するカラムやタイムスタンプ・カラムなど) には、**not compressed** を使用します。
- テーブルを圧縮してもそのインデックスは圧縮されません。

#### 圧縮の制限事項

- 以下の項目は圧縮できません。
  - システム・テーブル
  - ワーク・テーブル
  - ロー内の **Java** カラム
  - マテリアライズされていない圧縮カラム
  - **IDENTITY** カラム
  - データ転送用に追加されたタイムスタンプ
  - すべてのデータ型。サポートされていないデータ型のリストについては、『圧縮ユーザーズ・ガイド』を参照してください。
  - 暗号化カラム

- 設定されているロック・スキームとページ・サイズの組み合わせに対し、最小ロー・サイズが最大ユーザ・データ・ローより大きい場合、圧縮するテーブルは作成できません。たとえば、char(2007) データ型のカラム c1 を含む、2 キロバイトのページ・サイズのデータオンリーロック・テーブルを作成することはできません。このテーブルのサイズは最大ユーザー・データ・ロー・サイズを超過しているためです。ロー圧縮およびページ圧縮に対し、Adaptive Server では新しいテーブル用のチェックと同じロー・サイズ・チェックを実行します。
- 4 バイト未満の短い固定長カラムのみを含む、row 圧縮または page 圧縮の対象となるテーブルを作成することはできません。

#### ラージ・オブジェクト (LOB) データを使用するテーブルの圧縮

LOB テーブルのデータの圧縮には以下の制限が適用されます。次の処理はできません。

- 計算テキスト・カラムの圧縮
- 通常カラム、XML データに対する LOB 圧縮句 (lob\_compression = など) の発行
- システム・テーブルやワーク・テーブルに対する LOB 圧縮の使用

#### カラム定義

- ユーザ定義データ型を使用してカラムを作成する場合について説明します。
  - 長さ、精度、または位取りは変更できない。
  - null 型を使用して NOT null カラムを作成できるが、IDENTITY カラムを作成することはできない。
  - NOT NULL 型を使用して NULL カラムと IDENTITY カラムを作成できる。
  - IDENTITY 型を使用して NOT NULL カラムを作成できるが、そのカラムは IDENTITY プロパティを継承する。IDENTITY 型を使用して null カラムを作成することはできない。
- 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。データ型が変更されたことは、ユーザに通知されません。

表 1-11 は、固定長データ型とその変換後の可変長データ型を示しています。moneyn などの特定の可変長データ型は、カラム、変数、またはパラメータの作成には使用できない予約されたデータ型です。

表 1-11: null の格納に使用される可変長データ型

変換前の固定長データ型	変換後のデータ型
char	varchar
nchar	nvarchar
binary	varbinary
datetime	datetime
float	floatn
bigint、int、smallint、および tinyint	intn
unsigned bigint、unsigned int、unsigned smallint	uintn
decimal	decimaln
numeric	numericn
money、smallmoney	moneyn

- カラムのデフォルトを設定するには2つの方法があります。カラム制約としてデフォルトを `create table` または `alter table` 文内で宣言する方法と、`create default` 文を使用してデフォルトを作成してから `sp_bindefault` を使用してカラムにバインドする方法です。
- テーブルとそのカラムについてのレポートを取得するには、システム・プロシージャ `sp_help` を実行します。

#### テンポラリ・テーブル

- テンポラリ・テーブルは、テンポラリ・データベース `tempdb` に格納されます。
- テンポラリ・テーブル名の最初の13文字は、セッション内でユニークである必要があります。これらのテーブルには、現在の `Adaptive Server` セッションによってのみアクセスできます。これらのテーブルは、それぞれの名前とシステム提供の数値サフィックスに従って、`tempdb.objects` に格納され、現在のセッションの終了時、または明示的に削除されたときに消去されます。
- "`tempdb..`" プレフィックス付きで作成されたテンポラリ・テーブルは、`Adaptive Server` ユーザ・セッション間で共有できます。ユーザ間およびセッション間でテーブルを共有する場合にのみ、ストアド・プロシージャ内から "`tempdb..`" プレフィックス付きのテンポラリ・テーブルを作成します。ストアド・プロシージャ内でテンポラリ・テーブルを作成したり削除するときに、"`#`" プレフィックスを使用して、テンポラリ・テーブルが誤って共有されないようにします。



- **Adaptive Server** セッション中に複数のユーザがテンポラリ・テーブルを使用できます。ただし、数値 2 の "guest" ユーザ ID でテンポラリ・テーブルが作成されるため、通常は特定のユーザ・セッションを識別できません。複数のユーザがテンポラリ・テーブルを作成するプロセスを実行する場合、各ユーザが "guest" ユーザであり、このため uid 値がすべて同じになります。このため、特定のユーザに対応するユーザ・セッションが、テンポラリ・テーブルのどのユーザ・セッションであるかを知る方法はありません。システム管理者は `create login` を使用してテンポラリ・テーブルにユーザを追加できます。この場合、テンポラリ・テーブルのユーザ・セッションに対して個別の uid を使用できます。
- テンポラリ・テーブルに対して、ルール、デフォルト、およびインデックスを関連付けることはできますが、テンポラリ・テーブルにビューを作成したり、トリガを関連付けることはできません。
- テンポラリ・テーブルを作成するときにユーザ定義のデータ型を使用できるのは、そのデータ型が `tempdb.systypes` にある場合に限られます。現在のセッションでのみユーザ定義のデータ型を `tempdb` に追加するには、`tempdb` の使用中に `sp_addtype` を実行します。永続的にデータ型を追加するには、`model` の使用中に `sp_addtype` を実行してから、**Adaptive Server** を再起動して、`model` が `tempdb` にコピーされるようにします。

#### インデックスの使用

- テーブルは、そのテーブルのクラスタード・インデックスを「追跡」します。あるセグメントにテーブルを作成して、次に別のセグメントにそのクラスタード・インデックスを作成すると、インデックスが作成されたセグメントにテーブルが移行します。
- セグメントが別々の物理デバイスにある場合、一方のセグメントにテーブルを作成し、もう一方のセグメントにそのノンクラスタード・インデックスを作成すると、挿入、更新、選択の速度が上がります。『**Transact-SQL ユーザーズ・ガイド**』の「クラスタード・インデックスとノンクラスタード・インデックスの使用」を参照してください。

#### テーブルまたはそのカラムの名前の変更

- `sp_rename` は、テーブルまたはカラムの名前の変更で使用します。

- テーブルまたはそのカラムの名前を変更してから、`sp_depends` を使用して、そのテーブルに依存するプロシージャ、トリガ、およびビューを決定し、これらのオブジェクトを再定義します。

---

**警告！** これらの依存オブジェクトを再定義しない場合は、オブジェクトは **Adaptive Server** によって再コンパイルされると動作しなくなります。

---

### 整合性の制約の定義

- `create table` 文は、**SQL** 標準で定義された一連の整合性の制約があるため、データベースの整合性の制御に役立ちます。これらの整合性の制約句によって、ユーザがテーブルに挿入できるデータが制限されます。また、デフォルト、ルール、インデックス、およびトリガを使用して、データベースの整合性を実行することもできます。

整合性の制約を使用すると、テーブルの作成中に整合性の制御を一度に定義でき、また、これらの整合性の制御の作成を簡単にすることができます。しかし、整合性制約はスコープがさらに限定されており、デフォルト、ルール、インデックス、およびトリガほど包括的ではありません。

- 複数のカラムに作用する制約はテーブル・レベルの制約として宣言し、1つのカラムだけに作用する制約はカラム・レベルの制約として宣言します。その違いはユーザにはほとんどわかりませんが、カラムレベルの制約がチェックされるのは、カラム内の値が変更される場合のみであるのに対して、テーブルレベルの制約は、ローに対して何らかの変更が行われる場合にチェックされず。該当するカラムが変更されるかどうかは関係ありません。

カラム・レベルの制約は、カラム名とデータ型の後、区切りカンマの前に置きます (例 5 を参照)。テーブル・レベルの制約は、独立したカンマ区切りの句として入力します (例 4 参照)。**Adaptive Server** は、テーブル・レベルとカラム・レベルの制約を同じ方法で処理するため、どちらの効果も変わりありません。

- テーブル・レベルまたはカラム・レベルでは、次のタイプの制約を作成できます。
  - **unique** 制約 (一意性制約) は、テーブル内の指定したカラムに同じ値を持つローが 2 つ存在することを許可しない。また、**primary key** 制約 (プライマリー・キー制約) は、カラムに **null** 値が存在しないように要求する。

- 「参照整合性」(references) 制約は、特定のカラムに挿入されるデータまたはそのカラムで更新されるデータに対応するデータが、指定したテーブルとカラムに存在していることを要求する。
- check 制約は、カラムに挿入されるデータの値を制限する。

カラム内での null 値の使用を制限したり (null または not null キーワード)、カラムにデフォルト値を指定しても (default 句を使用)、データの整合性を実現できます。

- sp\_primarykey、sp\_foreignkey、および sp\_commonkey を使用すると、データベースのテーブル間の関係の明確化に役立つ情報を、システム・テーブルに保存できます。ただし、このプロセスは、create table 文での primary key および foreign key キーワードの機能のキー関係の実現や、機能の複製はしません。定義されたキーのレポートを取得するには、sp\_helpkey を使用します。頻繁に使用するジョインのレポートを取得するには、sp\_helpjoins を実行します。
- Transact-SQL には、整合性を確実にするメカニズムがいくつか用意されています。create table の部分として宣言できる制約の他に、ルール、デフォルト、インデックス、およびトリガも作成できます。表 1-12 は、整合性の制約をまとめ、その他の整合性の実現方法も示しています。

表 1-12: 整合性を確保する方法

create table での指定	他の方法
unique 制約	create unique index (null 値が使用できるカラムで使用)
primary key 制約	create unique index (null 値が使用できないカラムで使用)
references 制約	create trigger
check 制約 (テーブル・レベル)	create trigger
check 制約 (カラム・レベル)	create trigger または create rule、および sp_bindrule
default 句	create default および sp_bindefault

どの方法を選択するかはそれぞれの要件によって異なります。たとえば、トリガは、(他のカラムやオブジェクトを参照する場合などの)参照整合性の処理が、create table で宣言されたものより複雑になります。また、create table 文で定義された制約はそのテーブル固有のもので、ルールやデフォルトとは異なり、別のテーブルにその制約をバインドすることはできません。さらに、alter table を使用しないと、制約の削除または変更もできません。制約には、同じテーブル上でも、サブクエリや集合関数を入れることはできません。

- `create table` には、多数の制約を組み込むことができますが、次の制限事項があります。
  - `unique` 制約の数は、テーブルに設定可能なインデックスの数によって制限される。
  - 1つのテーブルに指定できる `primary key` 制約は1つだけである。
  - テーブルのカラム1つに対して組み込むことができる `default` 句は1つに限られるが、同一カラムに異なる制約を定義できる。

次に例を示します。

```
create table discount_titles
(title_id varchar (6) default "PS7777" not null
 unique clustered
 references titles (title_id)
 check (title_id like "PS%"),
 new_price money)
```

新しい `discount_titles` テーブルの `title_id` カラムは、それぞれの整合性の制約で定義されています。

- エラー・メッセージを作成して、それらを参照整合性制約と `check` 制約にバインドできます。`sp_addmessage` でメッセージを作成し、`sp_bindmsg` でメッセージを制約にバインドします。「`sp_addmessage`」と「`sp_bindmsg`」を参照してください。
- **Adaptive Server** は、参照制約を実行する前に検査制約を評価します。また、すべての整合性の制約を実行した後にトリガが評価されます。制約のいずれかが守られていない場合、**Adaptive Server** はデータ変更の文を取り消し、関連のあるトリガはどれも実行しません。ただし、制約の違反によって現在のトランザクションがロールバックされることはありません。
- 参照先テーブル内では、参照元テーブル内の値と一致するカラム値を更新したりローを削除することはできません。参照元テーブルで更新または削除を行ってから、参照先テーブルでの更新または削除を行ってください。
- 参照元テーブルを削除から、参照先テーブルを削除してください。そうしないと、制約違反が発生します。
- テーブルに定義されている制約の情報を取得するには、`sp_helpconstraint` を使用します。

## 一意性制約および主キー制約

- **unique** 制約 (一意性制約) は、カラム・レベルまたはテーブル・レベルで宣言できます。**unique** 制約では、指定したカラムにある値すべてがユニークであることが要求されます。テーブル内の指定したカラムで、2つのローが同じ値を持つことはできません。
- **primary key** 制約 (プライマリー・キー制約) は、**unique** 制約にさらに制限を加えるものです。**primary key** 制約が指定されたカラムには、**null** 値を指定できません。

---

**注意** `create table` 文の **unique** と **primary key** の制約によって、カラムのユニーク属性とプライマリ・キー属性を定義するインデックスが作成されます。`sp_primarykey`、`sp_foreignkey`、および `sp_commonkey` では、カラム間の論理関係が定義されます。これらの論理的關係には、必ずインデックスおよびトリガを使用しなければなりません。

---

- テーブルレベルの **unique** 制約または **primary key** 制約は、`create table` 文に別々の項目として組み込まれ、作成するテーブルの 1 つ以上のカラムの名前が含まれている必要があります。
- **unique** または **primary key** 制約は、指定したカラムにユニーク・インデックスを作成します。例 3 の **unique** 制約は、次の文の場合と同様にユニークなクラスタード・インデックスを作成します。

```
create unique clustered index salesind
on sales (stor_id, ord_num)
```

唯一の相違はインデックス名で、制約に名前を付けることで、`salesind` に設定できます。

- **SQL** 標準における **unique** 制約の定義では、カラム定義で **null** 値を許可しないように指定されています。**Adaptive Server** のデフォルトでは、カラム定義で **null** または **not null** を省略した場合、そのカラムは **null** 値を認めないものとして定義します (`sp_dboption` を使用して変更していない場合)。**Transact-SQL** では、制約を実行するために使用するユニーク・インデックスによって、**null** 値の入力が許可されているため、**unique** 制約付きで **null** 値の入力を許可するようカラムを定義できます。

- **unique** 制約は、デフォルトでユニーク・ノンクラスタード・インデックスを作成します。**primary key** 制約は、デフォルトでユニーク・クラスタード・インデックスを作成します。1つのテーブルに存在が可能なクラスタード・インデックスは1つだけであるため、指定できるのは **unique clustered** 制約または **primary key clustered** 制約のいずれか1つだけです。
- **create table** の **unique** および **primary key** 制約は、**create index** 文の簡略な代替として使用できます。ただし、次の点に注意してください。
  - ユニークでないインデックスは作成できない。
  - **create index** のオプションのすべてを使用できるわけではない。
  - これらのインデックスの削除には **alter table drop constraint** を使用する必要がある。

#### 参照整合性制約

- 参照整合性の制約では、制約が定義された参照元テーブルに挿入されるデータが、参照先テーブルの値と一致している必要があります。参照整合性の制約は、次の条件のいずれかかを満たします。
  - 参照元テーブルの制約の付いたカラムのデータが、**null** 値を含む。
  - 参照元テーブルの制約の付いたカラムのデータが、参照先テーブルの対応するカラムのデータ値と一致する。

たとえば **pubs2** データベースの場合、本の売上を記録する **salesdetail** テーブルに挿入するローに対して、**titles** テーブルに有効な **title\_id** が存在している必要があります。この場合、**salesdetail** は参照元テーブルで、**titles** テーブルは参照先テーブルです。現在、**pubs2** はトリガを使用して、この参照整合性を実現しています。ただし、**salesdetail** テーブルでは、次のカラム定義と参照整合性の制約を指定して、同じタスクを実行することができます。

```
title_id tid
references titles (title_id)
```

- 1つのクエリで許可されるテーブル参照の最大数は 192 です。テーブルの参照制約を確認するには、**sp\_helpconstraint** を使用します。
- テーブルに、そのテーブル自体の参照整合性の制約を組み込むことができます。たとえば、従業員とマネージャのリストを含む **pubs3** の **store\_employees** テーブルには、**emp\_id** カラムと **mgr\_id** カラムの間に次の自己参照があります。

```
emp_id id primary key,
mgr_id id null
references store_employees (emp_id),
```

この制約では、すべてのマネージャは従業員でもあり、すべての従業員は有効なマネージャに割り当てられていることが保証されます。

- 参照元テーブルが削除されるか、参照整合性の制約が削除されるまで、参照先テーブルは削除できません (テーブルに整合性の制約だけが含まれている場合を除きます)。
- **Adaptive Server** は、テンポラリ・テーブルには参照整合性の制約を適用しません。
- 別のユーザのテーブルを参照するテーブルを作成するには、参照先テーブルに **references** パーミッションが必要です。 **references** パーミッションの割り当ての詳細については、**grant** コマンドを参照してください。
- テーブルレベルの参照整合性の制約は、**create table** 文に別々の項目として含まれます。その中には、**foreign key** 句および 1 つ以上のカラム名のリストが必要です。

**primary key** 制約を使用して参照先テーブルのカラムがプライマリ・キーとして指定されている場合のみ、**references** 句のカラム名がオプションになります。

参照先カラムは、その参照先テーブルのユニーク・インデックスによって制約される必要があります。ユニーク・インデックスは、**unique** 制約か **create index** 文のいずれかを使用して作成できます。

- 参照元テーブルのカラムのデータ型は、参照先テーブルのカラムのデータ型と一致している必要があります。たとえば次のように、参照元テーブル (**test\_type**) の **col1** のデータ型が、参照先テーブル (**publishers**) の **pub\_id** のデータ型と一致している必要があります。

```
create table test_type
  (col1 char (4) not null
   references publishers (pub_id),
   col2 varchar (20) not null)
```

- 参照整合性の制約を定義するときには、参照先テーブルが存在している必要があります。相互に参照を行うテーブルには、**create schema** 文を使用して、両方のテーブルを同時に定義します。また別の方法として、制約のないテーブルを作成し、**alter table** を使用して後から制約を追加する方法もあります。詳細については、「**create schema**」または「**alter table**」を参照してください。
- **create table** の参照整合性の制約によって、データ整合性の実現が容易になります。トリガとは異なり、制約に次の機能はありません。

- データベース内の関連テーブルを使用した変更のカスケード
- 他のカラムやデータベース・オブジェクトの参照による複雑な制約の実行
- "what-if" 分析の実行

参照整合性の制約は、データ修正が制約に違反する場合でも、トランザクションをロールバックしません。トリガを使用すると、参照整合性の処理の方法に従って、トランザクションをロールバックするか継続するかを選択することができます。

**注意** Adaptive Server では、参照整合性の制約を検査してからトリガを検査するため、データ修正文がその制約に違反していると、トリガも起動されなくなります。

#### データベース間の参照整合性制約の使用

- データベース間の制約を作成すると、各データベースの `sysreferences` システム・テーブルに次の情報が格納されます。

表 1-13: 参照整合性の制約について格納される情報

sysreferences に格納される情報	参照先テーブルについての情報を持つカラム	参照元テーブルについての情報を持つカラム
キー・カラム ID	refkey1 ~ refkey16	fokey1 ~ fokey16
テーブル ID	reftabid	tableid
データベース ID	pmrydbid	frgnbdbid
データベース名	pmrydbname	frgndbname

- 参照元テーブルまたはそのデータベースを削除しても問題はありません。Adaptive Server は、外部キー情報を参照先データベースから自動的に削除します。
- 参照元テーブルは参照先テーブルの情報に依存しているため、Adaptive Server で次の操作は実行できません。
  - 参照先テーブルを削除する
  - 参照先テーブルがある外部データベースを削除する
  - `sp_renamedb` の使用によるいずれかのデータベースの名前変更
 これらの処理を行う前に、必ず `alter table` を使用してデータベース間の制約を取り除きます。



- データベース間の制約を追加または削除した場合や、データベース間の制約を含むテーブルを削除した場合は、影響を受けたデータベースを両方ともダンプしてください。

---

**警告！** データベース間の制約を含んだデータベースの以前のダンプをロードすると、データベースの破壊の原因となります。

---

- `sysreferences` システム・テーブルには、外部データベースの名前と ID 番号が格納されます。`load database` を使用して、データベース名を変更したり、別のサーバにロードしたりすると、Adaptive Server では参照整合性を保証できなくなります。

---

**警告！** データベースを別の名前ロードするか、別の Adaptive Server に移動するためにデータベースをダンプする場合は、事前に `alter table` を使用して外部参照整合性制約をすべて削除してください。

---

#### check 制約

- check 制約は、ユーザがテーブル内のカラムに挿入できる値を制限します。check 制約は、テーブルに挿入される前に null 以外のすべての値に対して一致が求められる検索条件 `search_condition` を指定します。`search_condition` には、次のものを含めることができます。
  - `in` によって指定される定数式のリスト
  - `between` で指定される定数式の範囲。
  - `like` で指定され、ワイルドカード文字を使用できる条件のセット式には、算術演算子および Transact-SQL の組み込み関数を使用できます。`search_condition` には、サブクエリ、集合関数、ホスト変数、またはパラメータは指定できません。Adaptive Server は、テンポラリー・テーブルに check 制約を適用しません。
- check 制約がカラムレベルの制約の場合は、その制約が定義されたカラムだけを参照し、テーブル内の他のカラムは参照できません。テーブルレベルの check 制約は、テーブル内のすべてのカラムを参照できます。
- `create table` を使用すると、1つのカラム定義で複数の check 制約を指定できます。
- check 整合性制約は、ルールおよびトリガの代わりに使用できません。これは、作成されるテーブルに固有のもので、別のテーブルのカラムやユーザ定義のデータ型へのバインドはできません。

- `check` 制約はカラム定義を上書きしません。 `null` 値が許可されるカラムで `check` 制約を宣言すると、 `search_condition` に `NULL` が含まれていない場合でも、暗黙的にまたは明示的に、そのカラムに `NULL` を挿入できます。たとえば、 `null` 値が許可されるテーブルカラムで “`pub_id in ( “1389”, “0736”, “0877”, “1622”, “1756” )`” または “`@amount > 10000`” を指定する `check` 制約を作成した場合でも、そのカラムに `null` を挿入できます。カラム定義は `check` 制約を上書きします。

#### IDENTITY カラム

- テーブルに初めてローを挿入するときに、 `Adaptive Server` は `IDENTITY` カラムに 1 の値を割り当てます。ローが新しく追加されるたびに、カラムの値は 1 ずつ増加します。この値は、 `create table` 文でカラムに対して宣言されたすべてのデフォルト、または `sp_bindefault` でカラムにバインドされたデフォルトよりも優先されます。

`IDENTITY` カラムに挿入できる最大値は、数値の場合は `10precision - 1` です。整数の識別子の場合、この値はデータ型で使用できる最大値（たとえば `tinyint` では 255、`smallint` では 32767）です。

識別子の詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の「[第 1 章 システム・データ型とユーザー定義データ型](#)」を参照してください。

- `IDENTITY` カラムに値を挿入すると、カラムへの初期値の指定や、エラー時に削除されたローのリストアが実行できます。テーブルの所有者、データベースの所有者、システム管理者は、ベース・テーブルに `set identity_insert table_name on` を使用した後に、1 つの値を明示的に `IDENTITY` カラムに挿入できます。 `IDENTITY` カラムでユニーク・インデックスを作成していないかぎり、 `Adaptive Server` は値の一意性を確認することができません。値は、どのような正の整数でも挿入できます。
- 必要に応じて実際のカラム名を使用する代わりに、テーブル名で修飾した `syb_identity` キーワードを使用して、 `IDENTITY` カラムを参照できます。
- システム管理者は、 `auto identity` データベース・オプションを使用すると、新しいテーブルに 10 桁の `IDENTITY` カラムを自動的に組み込むことができます。データベースでこの機能をオンにするには、次を使用してください。

```
sp_dboption database_name, "auto identity", "true"
```

ユーザが、データベースで **primary** キー、**unique** 制約、または **IDENTITY** カラムのいずれも指定しないでテーブルを作成するたびに、**Adaptive Server** では **IDENTITY** カラムが自動的に定義されます。このカラム **SYB\_IDENTITY\_COL** は、**select \*** 文を使用してカラムを検索するときには表示されません。**select** リストにカラム名を明示的に指定してください。

- サーバの障害によって、**IDENTITY** カラムの値にギャップが生じることがあります。ギャップは、トランザクションのロールバック、ローの削除、または手動による **IDENTITY** カラムへのデータの挿入によっても発生します。ギャップの最大サイズは、**identity burning set factor** と **identity grab size** 設定パラメータの設定、あるいは **create table** または **select into** 文の **identity\_gap** 値によって異なります。『**Transact-SQL ユーザーズ・ガイド**』の「データベースおよびテーブルの作成」の「テーブルの **identity** ギャップの管理」を参照してください。

#### ロック・スキームの指定

テーブルのロック・スキームを指定するには、**lock** キーワードと次のいずれかのロック・スキームを使用します。

- **allpages** ロック。クエリの影響を受けるデータ・ページとインデックスをロックします。
- **datapages** ロック。データ・ページのみをロックします。
- **datarows** ロック。データ・ローのみをロックします。

ロック・スキームを指定しないと、サーバのロック・スキームが使用されます。サーバ全体で使用されるデフォルトは、設定パラメータ **lock scheme** で設定されます。

テーブルのロック・スキームは、**alter table** コマンドを使用して変更できます。

#### 記憶領域管理プロパティ

- 記憶領域管理プロパティの **fillfactor**、**max\_rows\_per\_page**、**exp\_row\_size**、**reservepagegap** は、次のようにテーブルの記憶領域の使用管理に使用できます。
  - **fillfactor** はインデックスの作成時に余分な領域をページに残しますが、**fillfactor** は永続的に保持されるわけではありません。

- `max_rows_per_page` は、データまたはインデックス・ページのロー数を制限します。ローの数が減るとロック競合が少なくなるため、主として全ページロック・テーブルの同時実行性の改善に使用される。`max_rows_per_page` 値と、`datapages` ロックまたは `datarows` ロックを指定すると、警告メッセージが出力される。テーブルが作成され、値が `sysindexes` に格納されるが、この値は後でロック・スキームが `allpages` に変更された場合にのみ適用される。
- `exp_row_size` は、データ・ローの予期されるサイズを指定する。これは、インデックスではなく、データ・ローだけに適用され、また、可変長カラムのあるデータオンリーロック・テーブルのみに適用される。これは、データオンリーロック・テーブル内のローの転送数の軽減に使用される。また、主として、最初の挿入時はローに `null` または短いカラムがあり、その後の更新の結果、サイズが増大したテーブルで必要になる。`exp_row_size` は、ローが指定のサイズまで増大できるように、データ・ページの領域を確保する。全ページロック・テーブルの作成時に `exp_row_size` を指定すると、警告メッセージが表示される。テーブルが作成され、値が `sysindexes` に格納されるが、ロック・スキームが後で `datapages` または `datarows` に変更された場合にのみ、値が適用される。
- `reservepagegap` は空のページと満杯のページとの比率を指定して、エクステンツ割り付けを実行するコマンドに適用します。これは、すべてのロック・スキームのデータ・ページとインデックス・ページの両方に適用される。
- 表 1-14 は、記憶領域管理プロパティとロック・スキームの有効な組み合わせを示しています。`create table` コマンドに互換性のない組み合わせがある場合、警告メッセージは表示されますが、テーブルは作成されます。値は、システム・テーブルに格納されますが、適用されません。プロパティが有効になるようにテーブルのロック・スキームを変更すると、値が使用されます。

表 1-14: 記憶領域管理プロパティとロックスキーム

プロパティ	allpages	データページ	データロー
<code>max_rows_per_page</code>	可	不可	不可
<code>exp_row_size</code>	不可	可	可
<code>reservepagegap</code>	可	可	可
<code>fillfactor</code>	可	可	可

- 表 1-15 は、記憶領域管理プロパティのデフォルト値と、デフォルト値を使用したときの効果を示しています。

表 1-15: 記憶領域管理プロパティのデフォルトと効果

プロパティ	デフォルト	デフォルトを使用したときの効果
max_rows_per_page	0	ページの最大数のロー (255 まで) に適合する。
exp_row_size	0	設定パラメータ <code>default exp_row_size percent</code> によって設定され、サーバ全体に適用されるデフォルト値が使用される。
reservepagegap	0	エクステント割り付けの間、空ページを残さない。
fillfactor	0	インデックス・ページに残された領域で、リーフ・ページを完全にバックする。

### reservepagegap の使用

- 大量の領域を使用するコマンドでは、個別のページを割り付けるのではなく、エクステントを割り付けることによって新しい領域を割り付けます。reservepagegap キーワードを使用すると、これらのコマンドは、後続のページ割り付けが分割されるページまたはローの転送元のページの近くで行われるように空ページを残します。表 1-16 に、reservepagegap がいつ適用されるかを示します。

表 1-16: reservepagegap が適用されるとき

コマンド	データ・ページへの適用	インデックス・ページへの適用
高速 bcp	あり	インデックスが存在する場合は高速 bcp は使用されない。
低速 bcp	クラスタード・インデックスが指定されたテーブルではなく、ヒープ・テーブルのみ。	エクステント割り付けは実行されない。
select into	あり	ターゲット・テーブルにインデックスは存在しない。
create index または alter table...constraint	クラスタード・インデックスには可能。	あり
reorg rebuild	あり	あり
alter table...lock (全ページロックからデータオンリーロックへ、またはその反対)	あり	あり

- テーブルの reservepagegap 値は sysindexes に格納され、上記のオペレーションのいずれかがテーブルで実行されたときに適用されます。格納された値を変更するには、sp\_chgattribute を使用します。
- reservepagegap は、ワークテーブルまたはワークテーブルでのソートには適用されません。

### テーブルについての情報の取得

- `sp_help` は、指定されたテーブルとそのインデックスに割り当てられた属性 (キャッシュのバインドなど) をリストし、属性のクラス、名前、整数値、文字値、コメントなど、テーブルに関する情報を表示します。
- `sp_depends` は、テーブルに依存するデータベースのビュー、トリガ、プロシージャに関する情報を表示します。
- `sp_helpindex` は、テーブルに作成されたインデックスに関する情報をレポートします。
- `sp_helppartition` は、テーブルの分割プロパティについての情報をレポートします。

### パーティションのあるテーブルの作成

- パーティションに分かれたテーブルを作成するには、パーティションとして使用するディスク・デバイスとセグメントを準備しておく必要があります。
- 範囲分割は、ソート順に依存します。ソート順が変更された場合は、新しいソート順に合わせてテーブルを再分割する必要があります。
- 範囲分割の境界は、パーティション作成時の順序に基づいて昇順に位置する必要があります。
- `text`、`unitext`、`image`、または `bit` カラム、Java データ型、または計算カラムは分割キーの一部にできませんが、分割されたテーブルにこれらのデータ型のカラムを含めることはできます。分割の複合キーのカラム数は 31 個までです。
- 範囲分割とハッシュ分割の場合、分割キーとして最大 31 個のカラムがある複合キーを指定できます。ただし、一般には 4 個を超える分割のカラムがあるテーブルは扱いが難しく、不便です。
- 範囲分割とリスト分割の境界値は、対応する分割キーのデータ型と互換性がある必要があります。指定された境界値が互換性があってもデータ型が異なる場合、境界値は分割キーのデータ型に変換されます。Adaptive Server は以下をサポートしません。
  - 明示的な変換
  - データの消失につながる暗黙的変換
  - 範囲分割テーブルの境界として `null` を指定する
  - `binary` 以外のデータ型から `binary` または `varbinary` への変換

- リスト分割テーブルの値リストで `null` を使用できます。
- `text` カラムと `image` カラムがあるテーブルは分割できます。ただし、テーブルを分割しても `text` カラムと `image` カラムが格納される方法には影響しません。それぞれ別のパーティションに存在するからです。
- リモート・テーブルを分割することはできません。
- `Adaptive Server` は、`null` を特定の分割キー・カラムの他のどの分割キー値よりも小さい値として扱います。

#### 計算カラムのあるテーブルの作成

- `computed_column_expression` で参照できるのは、同じテーブル内のカラムだけです。
- `computed_column_expression` の `deterministic` プロパティは、データ操作に大きな影響を与えます。詳細については、『`Transact-SQL ユーザーズ・ガイド`』の「`deterministic` プロパティ」を参照してください。
- 計算カラムにデフォルト値を指定することはできません。計算カラムを `identity` カラムまたは `timestamp` カラムにすることもできません。
- `null` 入力可能性は、実体化された計算カラムにのみ指定できます。`null` 入力可能性を指定しない場合、すべての計算カラムはデフォルトで `null` 入力可能です。仮想計算カラムは常に `null` 入力可能です。
- トリガ、および `check`、`rule`、`unique`、`primary key`、`foreign key` などの制約は、実体化された計算カラムだけをサポートします。仮想計算カラムにこれらを使用することはできません。
- 計算カラムの定義で指定されたユーザ定義関数が削除されたり、無効になったりすると、その関数を呼び出す計算カラムのオペレーションはすべて失敗します。

#### 暗号化カラムのあるテーブルの作成

- 暗号化できるデータ型は次のとおりです。
  - `int`、`smallint`、`tinyint`
  - `unsigned int`、`unsigned smallint`、`unsigned tinyint`
  - `bigint`、`unsigned bigint`
  - `decimal`、`numeric`
  - `float4`、`float8`

- money、smallmoney
- date、time、smalldatetime、datetime、bigdatetime
- char、varchar
- unichar、univarchar
- binary、varbinary
- bit
- 暗号化されたデータのディスク上での基本データ型は、varbinary です。NULL 値は暗号化されません。
- create table で次の操作を行うと、エラー・メッセージが表示されます。
  - 1つ以上の暗号化カラムを参照している式に基づいて計算カラムを指定する。
  - encrypt パラメータと compute パラメータを同じカラムで使用する。
  - partition 句内で暗号化カラムをリスト。
- create table、alter table、select into オペレーションの間に、Adaptive Server によって暗号化カラム内の最大長が計算されます。データベース所有者は、スキーマ配置やページ・サイズを決定する前に、暗号化カラムの最大長を把握しておく必要があります。
- 暗号化キーに初期化ベクトルまたはランダム埋め込みを指定していない場合、暗号化カラムにインデックスを作成できます。初期化ベクトルまたはランダム埋め込みが使用されている暗号化カラムに対して create index を実行すると、エラーが発生します。
- 次の場合は、暗号化カラムに参照整合性制約を定義できます。
  - 参照元カラムと参照先カラムの両方が暗号化されている。
  - カラムの暗号化に使用するキーで init\_vector null を指定し、pad random を指定していない。
- 計算カラムは暗号化できません。また、計算カラムを定義している式に暗号化カラムを含めることはできません。create table の partition\_clause には、暗号化カラムを指定できません。

『暗号化カラム・ユーザーズ・ガイド』の「データの暗号化」を参照してください。



## 仮想ハッシュ・テーブル作成時の制限

- 仮想ハッシュ・テーブルを含むセグメントに対しては、**create table** を使用することはできません。仮想ハッシュ・テーブルは排他セグメントを 1 つのみ持つ必要があり、これを他のテーブルやデータベースと共有することはできないためです。
- 仮想ハッシュ・テーブルにはユニークなローが必要です。仮想ハッシュ・テーブルで複数のローが同じキー・カラム値を持つことはできません。**Adaptive Server** では、あるローをハッシュ領域に保持し、同じキー・カラム値を持つ別のローをオーバフロー・クラスタード領域に保持することができないためです。
- **truncate table** はサポートされていません。代わりに **delete from table\_name** を使用してください。
- **SQL92** では、関連する 2 つの一意性制約が同じキー・カラムを持つことはできません。ただし、仮想ハッシュ・テーブルの **primary key** 句は標準の一意性制約ではないので、仮想ハッシュ・キーとして同じキー・カラムを持つ別の一意性制約を宣言できます。
- テーブルを作成した後で仮想ハッシュ・クラスタード・インデックスを作成することはできないので、仮想ハッシュ・クラスタード・インデックスを削除することもできません。
- 仮想ハッシュ・テーブルは、排他セグメント上で作成する必要があります。仮想ハッシュ・テーブルを作成するためにセグメントに割り当てるディスク・デバイスは、他のセグメントと共有できません。
- 同じ排他セグメント上に 2 つの仮想ハッシュ・テーブルを作成することはできません。**Adaptive Server** では、1 データベースあたり 32 個のセグメントをサポートします。3 個のセグメントがデフォルト、システム、およびログの各セグメント用に予約されるので、1 データベースあたりの仮想ハッシュ・テーブルの最大数は 29 です。
- **alter table** コマンドと **drop clustered index** コマンドは仮想ハッシュ・テーブルに対して使用できません。
- 仮想ハッシュ・テーブルでは全ページ・ロックを使用する必要があります。
- 仮想ハッシュ・テーブルのキー・カラムとハッシュ係数は **int** データ型を使用する必要があります。

- `text` や `image` カラムを仮想ハッシュ・テーブルに含めることはできません。`text` や `image` データ型に基づくデータ型のカラムも含めることはできません。
- 分割された仮想ハッシュ・テーブルは作成できません。

#### インメモリ・データベースおよびリラックス持続性データベースのテーブルの作成

- `create table` で定義されたテーブルレベルのロギング設定は、`select into` を使用して作成されたテーブルにも適用されます。
- 完全な持続性を使用してデータベースに最低限のロギングが設定されたテーブルを作成できますが、データベースではこれらのテーブルに最低限のロギングを使用しません。`Adaptive Server` では、これらのテーブルを最低限のロギングに設定できるので、持続性が `no_recovery` に設定された他のデータベースのテンプレートとして、これらのデータベースを使用できます。この場合、最低限のロギングは従属するデータベースで有効になります。

#### hash\_factor の値の決定

最初のキーのハッシュ係数を 1 に保つことができます。残りのすべてのキー・カラムに対するハッシュ係数は、そのハッシュ係数で乗算されたハッシュ領域で許可される以前のキーの最大値よりも大きくなります。

`Adaptive Server` では、ページのローをより少なくするために、最初のキー・カラムに対するハッシュ係数が 1 より大きいテーブルを許可しています。たとえば、テーブルに最初のキー・カラムに対してハッシュ係数 5 がある場合、ページの各ローの後、次の 4 つのローの領域は空のままです。これをサポートするために、`Adaptive Server` にはテーブル領域の 5 倍が必要です。

キー・カラムの値が、次のキー・カラムのハッシュ係数と等しいか大きい場合は、ハッシュ領域で衝突が発生しないように、現在のローがオーバフロー・クラスタード領域に挿入されます。

たとえば、`t` は、`id` および `age` キー・カラムと対応するハッシュ係数 (10,1) を持つ仮想ハッシュ・テーブルです。(5, 5) および (2, 35) ローに対するハッシュ値が 55 であるため、ハッシュが衝突する可能性があります。

ただし、値 35 は 10 (次のキー・カラム `id` のハッシュ係数) 以上であるため、ハッシュ領域で衝突しないように、オーバフロー・クラスタード領域に 2 番目のローが格納されます。

また、u は、プライマリ・インデックスと (id1, id2, id3) = (125, 25, 5) と 200 のうち *max hash\_value* のハッシュ係数を持つ仮想ハッシュ・テーブルです。

- ロー (1,1,1) には、ハッシュ値 155 があり、ハッシュ領域に格納されます。
- ロー (2,0,0) には、ハッシュ値 250 があり、オーバフロー・クラスタード領域に格納されます。
- ロー (0,0,6) には、25 以上である 6 x 5 のハッシュ係数があるため、オーバフロー・クラスタード領域に格納されます。
- ロー (0,7,0) には、125 以上である 7 x 25 のハッシュ係数があるため、オーバフロー・クラスタード領域に格納されます。

#### 共有ディスク・クラスタの制限事項

- 同じローカル・テンポラリ・データベースのテーブルからでない場合、ローカル・テンポラリ・データベースのカラムを参照する参照整合性制約を含めることはできません。create table は、別のデータベースのテーブルからローカル・テンポラリ・データベースのカラムへの参照を作成しようとすると失敗します。
- ローカル・テンポラリ・データベースに格納されている暗号化キーを持つカラムは、そのカラムのテーブルが同じローカル・テンポラリ・データベースにないと、暗号化できません。ローカル・テンポラリ・データベースにある暗号化キーを持つカラムを暗号化する場合、そのテーブルが別のデータベースにあると、alter table は失敗します。

#### Java-SQL カラム

- Java がデータベースで実行可能になっている場合は、Java-SQL カラムがあるテーブルを作成できます。詳細については、『Adaptive Server Enterprise における Java』を参照してください。
- Java-SQL カラムの宣言クラス (*datatype*) は、`Serializable` インタフェースまたは `Externalizable` インタフェースのいずれかを実装する必要があります。
- テーブルを作成する場合に、Java-SQL カラムを次のように指定することはできません。
  - 外部キーとして指定する
  - 参照句の中で指定する
  - `UNIQUE` プロパティを持つカラムとして指定する
  - プライマリ・キーとして指定する

- `in row` が指定されている場合、データベース・サーバのページ・サイズとその他の変数によっては、格納される値が 16 キロバイトを超えることができません。
- `off row` が指定されている場合は、次のようになります。
  - 検査制約内では、カラムの参照はできない。
  - `distinct` を指定する `select` では、カラムの参照はできない。
  - 比較演算子、述部、または `group by` 句の中でカラムを指定することはできない。

## 標準規格

ANSI SQL — 準拠レベル：初級レベル

Transact-SQL 拡張機能は次のとおりです。

- テーブル名またはカラム名を修飾するためのデータベース名の使用
- `IDENTITY` カラム
- `not null` カラムのデフォルト
- `asc` および `desc` オプション
- `reservepagegap` オプション
- `lock` 句
- `on segment_name` 句

準拠しているデータ型の詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の「[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。

## パーミッション

すべてのユーザがロギングを無効にしてテンポラリ・テーブルと新しいテーブルを作成できます。

細密なパーミッションの設定によって異なる `create table` のパーミッション・チェックについて、次に説明します。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>create table</code> 権限が必要。他のユーザに対して <code>create table</code> を実行するには、 <code>create any table</code> が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、 <code>sa_role</code> が付与されたユーザであるか、 <code>create table</code> 権限を持つユーザであることが必要。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
10	create	create table	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> <li>• <code>with transfer table [on   off]</code> の <code>with</code> の設定に応じて次のようになります。 <ul style="list-style-type: none"> <li>• <code>on</code> – Adaptive Server は WITH TRANSFER TABLE ON を監査レコードの extrainfo に出力。</li> <li>• <code>off</code> – Adaptive Server は WITH TRANSFER TABLE OFF を出力。</li> </ul> </li> </ul>

## 参照

コマンド [alter table](#), [create existing table](#), [create index](#), [create rule](#), [create schema](#), [create view](#), [drop index](#), [drop rule](#), [drop table](#)

システム・プロシージャ [sp\\_addmessage](#), [sp\\_addsegment](#), [sp\\_addtype](#), [sp\\_bindmsg](#), [sp\\_chgattribute](#), [sp\\_commonkey](#), [sp\\_depends](#), [sp\\_foreignkey](#), [sp\\_help](#), [sp\\_helpjoins](#), [sp\\_helpsegment](#), [sp\\_primarykey](#), [sp\\_rename](#), [sp\\_spaceused](#)

## create thread pool

説明	ユーザ定義のスレッド・プールを作成します。
プロセス・モードの考慮事項	create thread pool はプロセス・モードではサポートされていません。
構文	<pre>create thread pool <i>pool_name</i> with thread count = <i>count</i> [ , pool description = <i>description</i> ] [idle timeout = <i>time_period</i>]</pre>
パラメータ	<p><i>pool_name</i> 作成するプールの名前です。</p> <p>thread count = <i>count</i> プール内のスレッド数です。1 以上にしてください。</p> <p>pool description = <i>description</i> (任意) プールの目的について説明します。256 文字未満にしてください。</p> <p>idle timeout = <i>time_period</i> スレッドがスリープ状態になる前に作業を探すマイクロ秒単位の時間です。デフォルトは 100 マイクロ秒です。-1 の値は、スレッドがスリープ状態にならず、実行する作業がない場合も CPU を消費し続けることを意味します。0 の値は、作業が見つからなかった場合にスレッドがただちにスリープ状態になることを意味します。</p>
例	<p><b>例 1</b> 10 のスレッドを持つ sales_pool という名前のスレッド・プールを作成します。</p> <pre>create thread pool sales_pool with thread count = 10</pre> <p><b>例 2</b> 説明を含む、order_pool という名前のスレッド・プールを作成します。</p> <pre>create thread pool order_pool with thread count = 10, pool description = 'used for handling order entry users'</pre> <p><b>例 3</b> 2つのスレッドと 500 マイクロ秒の idle timeout を持つ、order_pool という名前のスレッド・プールを作成します。</p> <pre>create thread pool order_pool with thread count = 2, idle timeout = 500</pre>
使用法	<ul style="list-style-type: none"><li>sp_adddexeclass を使用して、ユーザが作成したスレッド・プールに負荷を関連付けます。</li></ul>

- Adaptive Server がエンジン・プール内のスレッドすべて (*count* で指定) をオンラインにするには、十分な数のフリー・エンジンが必要です。max online engines はエンジンの合計数を決定します。すべてのエンジン・プール内のアクティブなスレッドの総数が max online engines の値を超えることはできません。
- *pool\_name* の名前を *syb\_* で始めることはできません。これは Sybase が作成したスレッド・プール用に予約されています。
- Transact-SQL 変数をパラメータとして使用して、create thread pool を実行することはできません。
- idle timeout 0 における 0 の値は、作業が見つからなかった場合にスレッドがただちにスリープ状態になることを示します。
- idle timeout -1 における 0 の値は、スレッドがスリープ状態にならず、実行する作業がない場合も CPU を消費し続けることを意味します。
- create thread pool は execute immediate を指定して発行できます。

標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

パーミッション

create thread pool のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、manage any thread pool 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、sa_role が付与されたユーザであることが必要。

監査

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
42			プール名とスレッド数

## create trigger

**説明** 整合性制約の実行によく使用されるストアド・プロシージャのタイプのひとつであるトリガを作成します。ユーザが特定のテーブル上で特定のデータ修正文を実行しようとする、トリガが自動的に実行されます。

**構文**

```
create trigger [owner.]trigger_name
  on [owner.]table_name
  {for {insert , update} | instead of {insert, update, delete}}
  [as
    [if update (column_name)
      [{and | or} update (column_name)]...
      SQL_statements
    [if update (column_name)
      [{and | or} update (column_name)]...
      SQL_statements]...]
```

**パラメータ**

*trigger\_name*  
トリガの名前です。識別子の規則に従っていると同時に、データベース内でユニークでなければなりません。現在のデータベース内で別のユーザが所有しているトリガと同じ名前別のトリガを作成する場合は、所有者名を指定します。*owner*のデフォルト値は現在のユーザです。所有者名を使用してトリガを修飾する場合は、同じ方法でテーブル名を明示的に修飾しなければなりません。

トリガ名に変数を使用することはできません。

*table\_name*  
トリガを作成するテーブル名です。データベース内に同じ名前のテーブルが複数ある場合は、所有者名を指定します。*owner*のデフォルト値は現在のユーザです。

**for | instead of**  
**for** — トリガを作成する対象を示すために **insert**、**delete**、または **update** の前で使用します。

**instead of** — **inserted** **擬似テーブル**と **deleted** **擬似テーブル**を作成してデータを入力します。これらのテーブルは、元の **insert**、**delete**、または **update** クエリで修正されたローを調べるためトリガで使用されます。

**insert, update, delete**  
どの組み合わせにも含めることができますが、**delete** を **if update** 句とともに使用することはできません。



### SQL\_statements

トリガ条件とトリガ動作を指定します。トリガ条件は、実行する `insert`、`update`、または `delete` が、トリガの動作を起動するかどうかを決定します。SQL 文では、`if` キーワードの後ろにサブクエリが続くことがあります。下記の例 2 では、`if` キーワードに続くサブクエリがトリガ条件です。

ユーザ動作 (`insert`、`update`、または `delete`) が実行されると、トリガの動作が実行されます。複数のトリガ動作を指定する場合は、`begin` と `end` でグループ化します。

トリガ定義で使用できない文のリストについては、「[トリガおよびトランザクション](#)」(297 ページ) を参照してください。トリガ定義に使用できる `deleted` および `inserted` 論理テーブルの詳細については、「[deleted 論理テーブルと inserted 論理テーブル](#)」(295 ページ) を参照してください。

#### if update

指定したカラムが `update` 文の `set` リストに含まれるかどうか、または `insert` の影響を受けるかどうかのテストに使用します。`if update` を使用すると、指定したトリガの動作を、指定したカラムの更新に関連付けることができます (例 3 を参照してください)。複数のカラムを指定でき、1 つの `create trigger` 文で複数の `if update` 文を使用できます (例 5 を参照してください)。

例

**例 1** ユーザが `titles` テーブルでデータを追加または変更しようとした場合、メッセージを出力します。

```
create trigger reminder
on titles
for insert, update as
print "Don't forget to print a report for accounting."
```

**例 2** `titles` テーブルに対応する `title_id` がない場合、新しいローが `titleauthor` に挿入されないようにします。

```
create trigger t1
on titleauthor
for insert as
if (select count (*)
    from titles, inserted
    where titles.title_id = inserted.title_id) = 0
begin
print "Please put the book's title_id in the
    titles table first."
rollback transaction
end
```

**例 3** publishers テーブルの pub\_id カラムが変更された場合、titles テーブルでそれに対応する変更を行います。

```
create trigger t2
on publishers
for update as
if update (pub_id) and @@rowcount = 1
begin
    update titles
    set titles.pub_id = inserted.pub_id
    from titles, deleted, inserted
    where deleted.pub_id = titles.pub_id
end
```

**例 4** titleauthor からローが削除された場合、titles テーブルからタイトルを削除します。本の著者が複数の場合は、titleauthor 内のそれらに対する参照も削除します。

```
create trigger t3
on titleauthor
for delete as
begin
    delete titles
    from titles, deleted
    where deleted.title_id = titles.title_id
    delete titleauthor
    from titleauthor, deleted
    where deleted.title_id = titleauthor.title_id
    print "All references to this title have been
    deleted from titles and titleauthor."
end
```

**例 5** 週末にプライマリ・キーが更新されないようにします。また、あるタイトルの総売上げが前渡し金を超えないかぎり、そのタイトルの価格または前渡し金への更新を行えないようにします。

```
create trigger stopupdatetrig
on titles
for update
as
if update (title_id)
and datename (dw, getdate ())
in ("Saturday", "Sunday")
begin
    rollback transaction
    print "We don't allow changes to"
    print "primary keys on the weekend!"
end
```

```

if update (price) or update (advance)
  if (select count (*) from inserted
      where (inserted.price * inserted.total_sales)
            < inserted.advance) > 0
  begin
  rollback transaction
  print "We don't allow changes to price or"
  print "advance for a title until its total"
  print "revenue exceeds its latest advance."
  end

```

**例 6** instead of トリガを使用して union ビューを更新します。

```

create table EmployeeWest (

    empid                int primary key,
    empname              varchar(30),
    empdob              datetime,
    region              char(5)
    constraint region_chk
        check (region='West'))

create table EmployeeEast (

    empid                int primary key,
    empname              varchar(30),
    empdob              datetime,
    region              char(5)
    constraint region_chk
        check (region='East' ))

create view Employees as
    select * from EmployeeEast
    union all
    select * from EmployeeWest

create trigger EmployeesInsertTrig on Employees
instead of insert as
begin

    insert into EmployeeEast select * from inserted where region = "East"

    insert into EmployeeWest select * from inserted where region = "West"
end

--will insert the data into the EmployeeEast table
insert into Employees values (10, 'Jane Doe', '11/11/1967', 'East')

```

```
--will insert the data into the EmployeeWest table
insert into Employees values (11, 'John Smith', '01/12/1977', 'West')

--will insert multiple rows into EmployeeEast and
--EmployeeWest tables.Employee2 table includes employees
--from both East and West.
insert into Employees select * from Employee2
```

**例 7** `instead of` トリガを使用して、暗号化カラムのサポートを実装し、アプリケーションを変更することなく暗号化された形式でデータをデータベースに格納します(ユーザ定義関数 `my_encrypt` および `my_decrypt` はデータの暗号化と復号化を行います)。

```
CREATE TABLE Employee_t (id int PRIMARY KEY, name varchar(20),
    salary binary (64))
--where the id and name columns are stored unencrypted, salary is
--encrypted and id is a primary key.
```

```
create view employee_v as select id, name, my_decrypt (salary)
from employee_t
```

```
CREATE TRIGGER EmployeeInsert
ON employee_v
INSTEAD OF INSERT
AS
BEGIN
    INSERT employee_t SELECT id, name, my_encrypt (salary)
    FROM inserted
END
```

```
CREATE TRIGGER employeeUpdate
ON employee_v
INSTEAD OF UPDATE
AS
BEGIN
    DELETE FROM employee_t WHERE id IN (SELECT id FROM deleted)
    INSERT employee_t SELECT id, name, my_encrypt (salary)
    FROM inserted
END
```

```
CREATE TRIGGER employeeDelete
ON employee_v
INSTEAD OF DELETE
AS
BEGIN
    DELETE FROM employee_t WHERE id IN (SELECT id FROM deleted)
END
```

## 使用法

- 設定の変更による予期しない結果を防ぐには、`create trigger` を実行する前に最初の文で `set rowcount 0` を実行します。`set` のスコープは `create trigger` コマンドのみに制限され、プロシージャが終了すると以前の設定にリセットされます。
- トリガは、1 つのデータ修正文につき 1 回しか起動しません。`while` ループを含む複雑なクエリは、`update` や `insert` を何度も繰り返す場合があります、そのたびにトリガが起動されます。

## トリガおよび参照整合性

- トリガは、一般に、**参照整合性** (テーブルやビューのプライマリ・キーと外部キーの関係に関する整合性の規則) を実現して、削除のカスケードや更新のカスケードを行うために使用されます (参照整合性については例 2、削除のカスケードについては例 3、更新のカスケードについては例 4 を参照してください)。
- データ修正文が完了し、`Adaptive Server` がデータ型、ルール、または整合性の制約の違反を検査した後でなければ、トリガは起動しません。トリガと、トリガを起動する文は、単一のトランザクションとみなされ、そのトリガ内からロールバックできます。重大なエラーが検出されると、トランザクション全体がロールバックされます。
- `create trigger` を使用する代わりに、`create table` 文で定義された制約を使用しても参照整合性を実行できます。整合性制約の詳細については、`create table` と `alter table` を参照してください。

*deleted* 論理テーブルと *inserted* 論理テーブル

- `deleted` および `inserted` テーブルは、論理 (概念) テーブルです。これらは、構造的にはトリガが定義されているテーブルと同様、ユーザ動作が実行されるテーブルであり、ユーザ動作によって変更されるローの古い値または新しい値が保持されています。

---

**注意** `inserted` テーブルも `deleted` テーブルも、トランザクション・ログにはビューとして表示されますが、`syslogs` ではどちらも偽のテーブルです。

---

- `deleted` および `inserted` テーブルは、トリガによってトリガ動作を実行するかどうか、またはその実行方法を確認することはできませんが、トリガ動作によってそのテーブル自体が変更されることはありません。

- `deleted` テーブルは `delete` および `update` とともに使用され、`inserted` テーブルは `insert` および `update` とともに使用されます。`update` では `delete` の後に `insert` が行われます。最初に `deleted` テーブル、次に `inserted` テーブルに作用します。

#### トリガの制約

- トリガを作成できるのは、現在のデータベースに限られます。所有者名を使用してトリガを修飾する場合は、同じ方法でテーブル名を明示的に修飾しなければなりません。トリガは、現在のデータベース以外のオブジェクトを参照することができます。
- トリガを複数のテーブルに適用することはできません。ただし、同じトリガ動作を、同一の `create trigger` 文の複数のユーザ動作 (たとえば `insert` や `update`) に対して定義できます。1つのテーブルには、最大3つのトリガを定義できます。`insert`、`update`、および `delete` のそれぞれに1つずつです。
- 1つのテーブルまたはカラムで、同じ操作 (`insert`、`update`、または `delete`) の新しいトリガを定義すると、古いトリガが上書きされます。上書きの前に警告メッセージは表示されません。
- セッション固有のテンポラリー・テーブルにはトリガを作成できません。
- ビューにはトリガを作成できません。
- システム・テーブルにはトリガを作成できません。
- `inserted` または `deleted` テーブルの `text`、`unitext`、または `image` カラムから選択するトリガは使用できません。
- Sybase では、結果をユーザに返す `select` 文が含まれないトリガをおすすめします。返された結果に対するすべてのアプリケーション・プログラムに、トリガ・テーブルの修正を可能にする特殊な処理を作成する必要があるためです。
- 無効な識別子であるテーブル名、カラム名、またはビュー名を、トリガが参照する場合、`create trigger` コマンドの前に `set quoted_identifier on` を実行し、その識別子名のそれぞれを二重引用符で囲みます。トリガの起動時に `quoted_identifier` オプションが有効になっている必要はありません。

#### トリガとパフォーマンス

- 通常、パフォーマンスに対するトリガのオーバーヘッド影響は非常に小さいものです。トリガ実行の時間のほとんどは、メモリ内またはデータベース・デバイス上の他のテーブルの参照に費やされます。

- トリガが頻繁に参照する `deleted` テーブルおよび `inserted` テーブルは論理テーブルであるため、常にデータベース・デバイス上ではなくメモリ内にあります。トリガが参照する他のテーブルの位置によって、操作にかかる時間が決まります。

#### トリガ内でのオプションの設定

トリガ内で `set` コマンドを使用できます。トリガの実行中は、呼び出した `set` オプションの有効性が保持され、トリガが終了すると以前の設定に戻ります。特に、`self_recursion` オプションはトリガ内で使用でき、トリガ自体によるデータ修正で、トリガを再起動できます。

#### トリガの削除

- トリガが参照するオブジェクトの名前を変更した場合は、トリガを削除してから、作成しなおす必要があります。トリガの名前は、`sp_rename` を使用して変更できます。
- テーブルを削除すると、そのテーブルに関連するトリガも削除されます。

#### トリガを起動できない動作

- `truncate table` コマンドは `delete` トリガの対象にはなりません。`truncate table` 文は、`where` 句のない `delete` (すべてのローが削除される) に似ていますが、データ・ローへの変更がロギングされないため、トリガを起動できません。

`truncate table` コマンドのパーミッションは、デフォルトではテーブル所有者にあり、譲渡はできません。このため、テーブル所有者だけが、`truncate table` 文で `delete` トリガを誤って実行しないように注意する必要があります。

- `writetext` コマンドは、ロギングの有無にかかわらず、トリガを起動しません。

#### トリガおよびトランザクション

- トリガが定義されると、適用されるテーブル上でトリガが指定するアクションは、トリガ自体とともに、常に暗黙的にトランザクションの一部となります。トリガは、エラーが検出された場合に、トランザクション全体をロールバックしたり、特定のデータを変更したときの影響をロールバックしたりするのによく使用されます。
- トリガに `rollback transaction` コマンドが含まれる場合は、ロールバックによってバッチ全体がアポートされ、後続のバッチ文は、どれも実行されない。

- トリガに **rollback trigger** が含まれる場合、ロールバックはトリガを起動したデータ修正だけに影響する。**rollback trigger** コマンドには、**raiserror** 文を含めることができる。バッチ内の後続の文が実行される。
- トリガは、トランザクションの一部として実行されるため、次の文およびシステム・プロシージャをトリガ内に記述することはできません。
  - **create database**、**create default**、**create index**、**create procedure**、**create rule**、**create table**、**create trigger**、**create view** など、すべての **create** コマンド
  - すべての **drop** コマンド
  - **alter database** および **alter table**
  - **truncate table**
  - **grant** および **revoke**
  - **update statistics**
  - **sp\_configure**
  - **load database** および **load transaction**
  - **disk init**、**disk refit**、**disk reinit**、**disk remirror**、**disk unmirror**
  - **select into**
- 求める結果 (計算値など) が、1 つのデータ修正の影響を受けるローの数によって異なる場合は、**@@rowcount** を使用して、複数ローのデータ修正 (**select** 文に基づく **insert**、**delete**、または **update**) についてテストし、適切な処置をとります。**if** 文など、ローを返さない Transact-SQL 文では、**@@rowcount** を 0 に設定して、**@@rowcount** のテストがトリガの最初の部分で行われるようにします。

#### トリガの挿入と更新

- **insert** コマンドまたは **update** コマンドの実行時に、Adaptive Server はトリガ・テーブルと **inserted** テーブルの両方に同時にローを追加します。**inserted** テーブルのローは、常にトリガ・テーブルの 1 つ以上のローの複製になります。



- `update` または `insert` トリガでは、`if update` コマンドを使用して、`update` または `insert` が特定の列を変更したかどうかを確認できます。`select` リストまたは `values` 句の値が列に割り当てられている場合は、常に `insert` 文に対する `if update (column_name)` は `true` になります。明示的な `NULL` またはデフォルトでは値が列に割り当てられるため、トリガが起動されます。暗黙的な `NULL` ではトリガは起動されません。

たとえば、次のようなテーブルとトリガを作成できます。

```
create table junk
  (aaa int null,
   bbb int not null)
create trigger trigtest on junk
for insert as
if update (aaa)
  print "aaa updated"
if update (bbb)
  print "bbb updated"
```

いずれかの列または両方の列に値を挿入すると、列 `aaa` と列 `bbb` の両方に対してトリガが起動します。

```
insert junk (aaa, bbb)
values (1, 2)
aaa updated
bbb updated
```

列 `aaa` に明示的な `NULL` を挿入しても、トリガを起動できません。

```
insert junk
values (NULL, 2)
aaa updated
bbb updated
```

列 `aaa` にデフォルト値があるときも、トリガが起動します。

ただし、列 `aaa` にデフォルト値がなく、明示的に値が挿入されていない場合、`Adaptive Server` では暗黙的な `NULL` を生成し、トリガを起動しません。

```
insert junk (bbb)
values (2)
bbb updated
```

`if update` が `delete` 文で `true` になることはありません。

### トリガのネストおよび再帰

- デフォルトでは、Adaptive Server ではネストされたトリガを使用できません。トリガのネストを防ぐには、`sp_configure` を使用して `allow nested triggers` オプションを 0 (オフ) に設定します。

```
sp_configure "allow nested triggers", 0
```

- トリガは 16 レベルの深さにネストできます。あるトリガが別のトリガが設定されているテーブルを変更すると、2 番目のトリガが起動し、次に 3 番目のトリガが呼び出され、というように続きます。このような連鎖内のトリガが無制限ループを引き起こすと、ネスト・レベルを超えるためトリガはアボートし、トリガ・キューを含むトランザクションがロールバックされます。

---

**注意** トリガはトランザクションに記述されているので、ネストされたトリガのいずれかのレベルで実行に失敗すると、トランザクション全体が取り消されます。その結果、データの変更がすべてロールバックされます。実行に失敗した箇所を確認するために、トリガにメッセージやその他のエラー処理、およびデバッグなどの補助プログラムを作成してください。

---

- グローバル変数 `@@nestlevel` には、現在実行中のネスト・レベルが格納されます。ストアド・プロシージャまたはトリガが別のストアド・プロシージャやトリガを呼び出すたびに、ネスト・レベルは増加します。ネスト・レベルは、キャッシュされる文が作成されたときも 1 つ増えます。最大値の 16 を超えると、トランザクションはアボートします。
- あるトリガが、再びそのトリガを起動するようなアクションを行うストアド・プロシージャを呼び出す場合でも、そのトリガは、ネストされたトリガが使用可能な場合にしか再度アクティブ化されません。トリガ内に再帰の回数を制限する条件がない場合は、これによりネストレベルのオーバーフローが発生します。

たとえば、更新トリガが、更新を実行するストアド・プロシージャを呼び出すと、`allow nested triggers` の設定がオフの場合は、トリガおよびストアド・プロシージャは、1 回しか実行されません。`allow nested triggers` の設定がオンで、しかもトリガまたはプロシージャの条件によって更新回数が制限されていない場合は、プロシージャまたはトリガのループは、ネストの最大値である 16 レベルを超えるまで続きます。

- デフォルトでは、トリガは `allow nested triggers` 設定パラメータの設定には関係なく、そのトリガ内の、同じテーブルに対する 2 番目のデータ変更に応答して自分自身を呼び出すことはありません。 `set` オプションの `self_recursion` では、トリガ内のデータ変更の結果としてトリガを再起動できます。たとえば、テーブルのあるカラムの更新トリガの結果によって別のカラムが更新される場合、更新トリガは `self_recursion` が無効になっていると一度だけ起動されますが、`self_recursion` がオンに設定されていれば 16 回まで起動できます。また、自己再帰を実行するには、`allow nested triggers` 設定パラメータも有効になっている必要があります。

#### instead of トリガと for トリガ

- `instead of` トリガと `for` トリガのネストをインターリーブできます。たとえば、ビューの `instead of update` トリガを持つ `update` 文はトリガを実行します。 `for` トリガを定義してテーブルを更新する SQL 文がトリガに含まれている場合は、そのトリガが起動します。 `for` トリガに、`instead of` トリガを持つ別のビューを更新する SQL 文が含まれていると、次にそのトリガが実行され、以降も同様です。
- ただし、`instead of` トリガと `for` トリガの再帰の動作は異なります。 `for` トリガは再帰をサポートしていますが、`instead of` トリガは再帰をサポートしていません。 `instead of` トリガが起動したビューと同じビューを参照している場合、トリガは再帰的に呼び出されず、トリガ文が直接ビューに適用されます。つまり、文はビューの基盤となるベース・テーブルに対する修正として解決されます。この場合、ビューの定義が更新可能なビューのすべての制約を満たしている必要があります。ビューが更新不可の場合は、エラーが発生します。

たとえば、トリガがビューの `instead of update` トリガとして定義されている場合は、`instead of` トリガ内の同じビューに `update` 文を実行しても、トリガが再実行することはありません。トリガで実行される更新は、ビューに `instead of` トリガがなかった場合と同様に、ビューに対して処理されます。更新によって変更されたカラムは、1 つのベース・テーブルに解決される必要があります。

#### instead of の制限事項：

- 識別子が有効でないテーブル名、カラム名、ビュー名などをトリガが参照する場合は、`create trigger` コマンドの前に `quoted_identifier` を設定し、それぞれの名前を二重引用符で囲む必要があります。トリガが起動するとき、`quoted_identifier` オプションが `on` である必要はありません。角カッコ付きの識別子も使用できます。

- `set cursor rows` コマンドをクライアント・カーソル、つまり `Open Client` 呼び出しや `Embedded SQL™` を使用して宣言されたカーソルとともに使用すると、`positioned delete` および `update` が `instead of` トリガを起動できなくなる可能性があります。`positioned update` 文は、`where current of <cursorname>` 句を含む `SQL update` 文であり、カーソル `<cursorname>` が現在置かれているローのみを更新します。
- ジョインは、`instead of` トリガを起動する検索済み `delete` 文と `update` 文では許可されていません。
- ジョインを使用して定義されたカーソル上の `positioned delete` および `update` は `instead of` トリガを起動しません。

`positioned delete` (または `positioned update`) は、`where current of <cursorname>` 句を含む `SQL delete` (または `update`) 文であり、カーソル `<cursorname>` が現在置かれているローのみを削除(または更新)します。

- `instead of` トリガを起動する `positioned delete` および `update` 文では、カーソルの宣言時に `instead of` トリガが存在している必要があります。

#### トリガに関する情報の取得

- トリガの実行プランは `sysprocedures` に格納されています。
- 各トリガには ID 番号が割り当てられます。ID は、`deltrig` カラムに適用するテーブルのオブジェクト ID とともに `sysobjects` に新しいローとして格納されます。また、適用されるテーブルの `sysobjects` ローの `deltrig`、`instrig`、`updtrig` カラムにもエントリとして格納されます。
- `syscomments` に格納されているトリガのテキストを表示するには、`sp_helptext` を使用します。

システム・セキュリティ担当者が、`Adaptive Server` を評価済み設定で実行するために `sp_configure` の `allow select on syscomments.text column` パラメータをリセットした場合、トリガの作成者またはシステム管理者のみが、`sp_helptext` を使用してトリガのテキストを表示できます

- トリガのレポートを表示するには、`sp_help` を使用します。
- トリガが参照するテーブルやビューのレポートを表示するときは、`sp_depends` を使用します。

## パーミッション

トリガ作成時のオブジェクトに対するパーミッション - トリガの作成時に **Adaptive Server** では、テーブルやビューなど、トリガが参照するオブジェクトに対するパーミッション・チェックは行われません。したがって、そのオブジェクトにアクセスしなくても、正常にトリガを作成できます。すべてのパーミッション検査は、トリガ起動時に行われます。

トリガ実行時のオブジェクトに対するパーミッション - トリガの実行時のオブジェクトに対するパーミッション・チェックは、トリガとそのオブジェクトを同じユーザが所有しているかどうかによって異なります。

- トリガとそのオブジェクトを同じユーザが所有していない場合は、そのトリガを起動するユーザは、オブジェクトに直接アクセスできなければなりません。たとえば、ユーザがアクセスできないテーブルからトリガが選択を行う場合は、トリガの実行は失敗します。また、トリガを起動させるデータ修正はロールバックされます。
- トリガとそのオブジェクトを同じユーザが所有している場合は、特別な規則が適用されます。ユーザが、トリガのオブジェクトに直接アクセスできない場合でも、そのオブジェクトにアクセスできる暗黙的なパーミッションが自動的に付与されます。暗黙のパーミッションの規則の詳細については、『システム管理ガイド』を参照してください。

**instead of** とトリガ **instead of** に対するパーミッション - **instead of** トリガのパーミッションには、**for** トリガと同じ要件が設定されます。**instead of** トリガを使用してビューを作成するには、基本となるテーブルではなく、ビューの **insert/update/delete** パーミッションをユーザに付与する必要があります。

細密なパーミッションの設定によって異なる **create trigger** のパーミッション・チェックについて、次に説明します。

細密なパーミッションが有効

細密なパーミッションが有効の場合、データベース所有者であることが必要。また **create trigger** 権限が無効であってははいけない。他のユーザに対して **create trigger** を実行するには、**create any trigger** 権限が必要。

細密なパーミッションが無効

細密なパーミッションが無効化されている場合は、次のようになる。

トリガ作成パーミッションの付与や取り消しを実行できるのは、システム・セキュリティ担当者だけです。データベース所有者には、ユーザ・テーブルにトリガを作成する暗黙のパーミッションが付与されます。ユーザは、各自が所有しているテーブルにのみトリガを作成できます。

システム・セキュリティ担当者はトリガを作成するユーザ・パーミッションを取り消すことができます。トリガ作成パーミッションの取り消しは、システム・セキュリティ担当者が `revoke` コマンドを発行したデータベースでのみ実行されます。パーミッションを取り消されたユーザに対して、システム・セキュリティ担当者が明示的に `create trigger` パーミッションを付与した場合、`create trigger` コマンドを実行するパーミッションは復元されます。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
12	create	create trigger	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter table](#), [create procedure](#), [drop trigger](#), [rollback trigger](#), [set](#)

システム・プロシージャ [sp\\_commonkey](#), [sp\\_configure](#), [sp\\_depends](#), [sp\\_foreignkey](#), [sp\\_help](#), [sp\\_helptext](#), [sp\\_primarykey](#), [sp\\_rename](#), [sp\\_spaceused](#)

## create view

説明	ビューを作成します。ビューは、1 つまたは複数のテーブルのデータを参照する方法の 1 つです。
構文	<pre>create view [owner.]view_name   [(column_name[, column_name]...)]   as   select [distinct] select_statement   [with check option]</pre>
パラメータ	<p><b>view_name</b> ビューの名前です。この名前は、データベース名を含むことはできません。 <b>set quoted_identifier on</b> を指定していれば、区切り識別子を使用できます。指定していなければ、ビュー名を変数にすることはできず、識別子の規則に従う必要があります。現在のデータベース内で別のユーザが所有しているビューと同じ名前で、別のビューを作成する場合は、所有者名を指定します。 <b>owner</b> のデフォルト値は現在のユーザです。</p> <p><b>column_name</b> ビューのカラムの見出しに使用する名前を指定します。 <b>set quoted_identifier on</b> を指定していれば、区切り識別子を使用できます。指定していない場合は、カラム名は識別子の規則に従ってください。</p> <p>カラム名はいつでも指定できますが、カラム名が必要になるのは次の場合に限られます。</p> <ul style="list-style-type: none"><li>• カラムが算術式、関数、文字列の連結、または定数によって導出される場合</li><li>• 2 つ以上のカラムが (通常はジョインによって) 同じ名前を持っている場合</li><li>• 抽出元のカラムと異なる名前をビューのカラムに指定する場合 (例 3 を参照)</li></ul> <p>カラム名は、 <b>select</b> 文でも指定できます (例 4 を参照)。カラム名が指定されていない場合、ビュー・カラムには <b>select</b> 文のカラムと同じ名前が使用されます。</p> <p><b>select</b> ビューを定義する <b>select</b> 文を開始します。</p> <p><b>distinct</b> 重複ローを含むことができないビューを指定します。</p>

**select\_statement**

ビューを定義する **select** 文を完了します。select 文では複数のテーブルとその他のビューを使用できます。

**with check option**

ビューの選択基準に対してすべてのデータ修正文を検証することを指定します。ビューを通して挿入または更新されるローはすべて、ビューを通して参照できる状態になっていなければなりません。

**例 1** ベース・テーブル **titles** の **title**、**type**、**price**、**pubdate** の各カラムから抽出したビューを作成します。

```
create view titles_view
as select title, type, price, pubdate
from titles
```

**例 2** "old view" から "new view" を作成します。新しいビューでは両方のカラムの名前が変更されます。ブランクが埋め込まれたすべてのビューおよびカラム名は、二重引用符で囲みます。set **quoted\_identifier** on を行ってからビューを作成してください。

```
create view "new view" ("column 1", "column 2")
as select col1, col2 from "old view"
```

**例 3** 価格が 5 ドル未満の書籍のタイトル、前渡し金、満期支払額を含むビューを作成します。

```
create view accounts (title, advance, amt_due)
as select title, advance, price * total_sales
from titles
where price > $5
```

**例 4** 2 つのベース・テーブル、**authors** と **publishers** から抽出したビューを作成します。ビューには、出版社が存在する都市に住む著者の名前と都市名が含まれます。

```
create view cities
(authorname, acity, publishername, pcity)
as select au_lname, authors.city, pub_name,
publishers.city
from authors, publishers
where authors.city = publishers.city
```

**例 5** 前の例と同じ定義でビューを作成します。ただし、**select** 文にカラム見出しを含めます。

```
create view cities2
as select authorname = au_lname,
acity = authors.city, publishername = pub_name, pcity =
publishers.city
```



```
from authors, publishers
where authors.city = publishers.city
```

**例 6** `titleauthor` から著者のユニークな ID コードを抽出する、`author_codes` というビューを作成します。

```
create view author_codes
as select distinct au_id
from titleauthor
```

**例 7** `title` からユニークな書籍価格を抽出する、`price_list` というビューを作成します。

```
create view price_list (price)
as select distinct price
from titles
```

**例 8** カリフォルニア州の書店だけについての情報を含む、`stores` テーブルのビューを作成します。`with check option` 句は、挿入または更新された各ローが、ビューの選択基準に照らして有効かどうかを検証します。`state` に "CA" 以外の値を持つローは拒否されます。

```
create view stores_cal
as select * from stores
where state = "CA"
with check option
```

**例 9** `stores_cal` から抽出した `stores_cal30` というビューを作成します。新しいビューは、チェック・オプションを `stores_cal` から継承します。`stores_cal30` で挿入または更新されたすべてのローは、`state` の値が "CA" になります。`stores_cal30` には `with check option` 句がないため、`stores_cal30` を使用して、`payterms` の値が "Net 30" 以外のローを挿入または更新できます。

```
create view stores_cal30
as select * from stores_cal
where payterms = "Net 30"
```

**例 10** `stores_cal` から抽出した `stores_cal30_check` というビューを作成します。新しいビューは、チェック・オプションを `stores_cal` から継承します。また、新しいビューには、それ自身の `with check option` 句もあります。`stores_cal30_check` によって挿入または更新されたローのそれぞれが、`stores_cal` と `stores_cal30_check` の両方の選択基準に照らして有効かどうか検証されます。"CA" 以外の `state` 値や "Net 30" 以外の `payterms` 値があるローは拒否されます。

```
create view stores_cal30_check
as select * from stores_cal
where payterms = "Net 30"
with check option
```

**例 11** ビューの作成に SQL 抽出テーブルを使用します。

```
create view psych_titles as
  select *
    from (select * from titles
          where type = "psychology") dt_psych
```

#### 使用法

- 基本となるテーブルではなく、ビューに対してパーミッションを付与することによって、ビューをセキュリティ・メカニズムとして使用できます。
- `sp_rename` を使用して、ビューの名前を変更できます。
- ビューを介してクエリを行う場合、文のいずれかの箇所から参照できるデータベース・オブジェクトがすべて存在しているかどうか、オブジェクトが文のコンテキスト内で有効かどうか、データ更新コマンドがデータ整合性の規則に違反していないかが Adaptive Server によって検証されます。この検証の結果、違反しているものがあると、エラー・メッセージが表示されます。検証の結果に問題がなければ、`create view` によってビューが基本となるテーブルに対する動作に変換されます。
- 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。

#### ビューの制約

- 現在のデータベース内でのみ、ビューを作成できます。
- ビューが参照するカラム数が、1024 を超えることはできません。
- テンポラリー・テーブルにはビューを作成できません。
- ビューではトリガの作成やインデックスの構築はできません。
- ビューの `text`、`unitext`、`image` カラムでは、`readtext` や `writetext` を使用できません。
- ビューを定義する `select` 文には `order by` 句、`compute` 句、またはキーワードの `into` を含めることができません。
- `select` 文に `union` 演算子が使用されているビューでは、更新、削除、または挿入を実行できません。
- ローカル変数またはグローバル変数を使用してビューを作成すると、「ローカル変数またはグローバル変数は、ビュー定義では使用できません。」というエラー・メッセージ 7351 が表示されます。

- `create view` 文は、1 つのバッチ内で他の SQL 文と組み合わせることができます。

---

**警告！** `if...else` ブロックまたは `while` ループ内に `create view` コマンドがある場合、`Adaptive Server` では条件が `true` であるかどうかを判定する前に、ビューのスキーマが作成されます。このため、そのビューがすでに存在している場合は、エラーが発生することがあります。このエラーを回避するには、データベース内に同名のビューが存在していないことを確認するか、次のように `execute` 文を使用します。

```
if not exists
    (select * from sysobjects where name="mytable")
begin
execute ("create table mytable (x int)")
end
```

---

### ビューの解析

- カラムを追加または削除してビューの基本となるテーブルの構造を変更しても、ビューを削除して再定義しない限り、`select *` 句で定義されたビューに新しいカラムは表示されません。アスタリスクの省略形は、ビューが最初に作成されたときに解釈され、展開されます。
- 削除されたテーブルまたはビューにビューが従属している場合、そのビューを使用すると `Adaptive Server` はエラー・メッセージを返します。同じ名前とスキーマを持つ新しいテーブルまたはビューを作成して、削除したテーブルまたはビューを置き換えると、ビューは再び使用できるようになります。
- 再定義によって `Adaptive Server` が従属ビューを変換できなくなる場合を除いて、ビューに従属しているその他のビューを再定義しなくてもビューを再定義できます。

### ビューを通じたデータ修正

- 複数のテーブルから構成されるビューでは `delete` 文を使用できません。
- 基本となるテーブルまたはビューのすべての `not null` カラムが、新しいローの挿入を行うビューに含まれていないかぎり、`insert` 文は使用できません。`Adaptive Server` は、基本となるテーブルやビューの `not null` カラムに値を指定できません。

- ビューを通して計算カラムに直接挿入することはできません。計算カラムの値は、常に **Adaptive Server** で内部的に生成されます。
- `insert` 文は、`distinct` または `with check option` で作成したジョイン・ビューで使用できません。
- `update` 文は `with check option` のジョイン・ビューで使用できます。影響を受けるカラムが、複数のテーブルからのカラムを含む式の `where` 句に含まれている場合は、更新が失敗します。
- ジョイン・ビューからローを挿入または更新した場合、影響を受けるすべてのカラムは同一のベース・テーブルに属する必要があります。
- `distinct` 句によって定義されたビューでは、更新や挿入を実行できません。
- データ更新文では、計算のビューにあるカラム、および集約が含まれるビューを変更できません。

#### IDENTITY カラムとビュー

- `column_name = identity (precision)` 構文を使用してビューに新しい **IDENTITY** カラムを追加することはできません。
- 明示的な値を **IDENTITY** カラムに挿入するには、テーブル所有者、データベース所有者、またはシステム管理者が、**IDENTITY** カラムを挿入するビューからではなく、カラムのベース・テーブルに `set identity_insert table_name on` を指定して挿入する必要があります。

#### `group by` 句とビュー

セキュリティ保護のためにビューを作成する場合は、集合関数と `group by` 句を注意して使用してください。Transact-SQL 拡張機能では、`group by` 句にないカラムに名前を付けることができます。`group by` 句にないカラムに名前を付けると、**Adaptive Server** はそのカラムの詳細なデータのローを返します。たとえば、次の Transact-SQL 拡張カラムのクエリでは、ローを (18 ローごとに) 1 つずつというように、予想よりも多くのデータが返されます。

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

それに対して、次の ANSI 準拠のクエリは、それぞれのタイプに対してローが 1 つずつ (6 ロー) 返されます。

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

「[group by 句と having 句](#)」(527 ページ) を参照してください。

#### *distinct* 句とビュー

- *distinct* 句は、ビューを重複ローを含まないデータベース・オブジェクトとして定義します。すべてのカラムの値が、別のローの同じカラムの値と一致している場合、ローは別のローと重複しているとみなされます。null 値が複数ある場合も、他の null 値の重複とみなされます。

ビューのカラムのサブセットを問い合わせると、それが重複するローであるとみなされる可能性があります。カラムのサブセットを選択した場合、カラムのいくつかに同じ値を持つものがあり、その結果重複するローが含まれているように思えることがあります。ただし、この時点でもビューの基本のローはユニークです。Adaptive Server は初めてビューにアクセスするとき (射影や選択を実行する前) にビューの定義に *distinct* 要件を適用し、ビューのローが相互に区別されるようにします。

集合関数の一部または *group by* 句として、*distinct* をビュー定義の *select* 文の中で複数回指定することで、重複ローを削除できます。次に例を示します。

```
select distinct count (distinct title_id), price
from titles
```

- *distinct* のスコープは、そのビューにしか適用されません。*distinct* ビューから派生した新しいビューには適用されません。

#### *with check option* 句とビュー

- ビューを *with check option* で作成する場合は、このビューを使用して挿入または更新される各ローがビューの選択基準を満たしている必要があります。
- ビューを *with check option* 付きで作成する場合は、「ベース」となるビューから抽出したビューのすべてがチェック・オプションを満たしている必要があります。抽出したビューを通して挿入または更新された各ローは、もともっているビューから参照できる状態になっていなければなりません。

#### ビュー情報の取得

- ビューが依存しているテーブルや複数のビューのレポート、またはビューに依存しているオブジェクトのレポートを作成するには、*sp\_depends* を実行します。
- *syscomments* に格納されているビューのテキストを表示するには、ビュー名をパラメータとして指定して、*sp\_helptext* を実行します。

### SQL 抽出テーブルからのビューの作成

- SQL 抽出テーブルを使用してビューを作成するには、`create view` 文の `select` 部分の `from` 句に抽出テーブル式を追加します (例 11 を参照)。
- 抽出テーブル式が更新可能な場合は、SQL 抽出テーブルを使用して作成されたビューも更新できます。抽出テーブル式の更新規則は、`create view` 文の `select` 部分の更新規則に従います。
- 抽出テーブル式の `insert` ルールとパーミッション設定が `create view` 文の `select` 部分の `insert` ルールとパーミッション設定に従っている場合は、SQL 抽出テーブルを含むビューからデータを挿入できます。
- `create view` 文の一部である抽出テーブル式では、テンポラリ・テーブルとローカル変数を使用できません。
- SQL 抽出テーブルには、名前のないカラムを作成できません。
- 抽出テーブル式の詳細については、『*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：初級レベル

複数の `distinct` キーワード、および `select` リストでの “`column_heading = column_name`” の使用は、Transact-SQL 拡張機能です。

### パーミッション

ビュー作成時は、Adaptive Server は、ビューによって参照されるテーブルやビューなどのオブジェクトについてのパーミッション検査を行いません。したがって、そのオブジェクトにアクセスしなくても、正常にビューを作成できます。すべてのパーミッション検査は、ビュー起動時に行われます。

ビューと参照されるすべてのオブジェクトが同じユーザに所有されているかどうかによって、ビューの呼び出し時のオブジェクトに対するパーミッション検査が異なります。

- ビューとそのオブジェクトを同じユーザが所有していない場合は、そのビューを起動するユーザは、オブジェクトに直接アクセスできなければなりません。たとえば、ユーザがアクセスできないテーブルからビューが `select` 文を実行すると、`select` 文は失敗します。

- ビューとそのオブジェクトを同じユーザが所有している場合は、特別な規則が適用されます。ビューを呼び出したユーザは、オブジェクトに直接アクセスできない場合でも、そのオブジェクトにアクセスできる暗黙的なパーミッションが自動的に付与されます。テーブルへの直接アクセスをユーザに認めなくても、ビューによって制限付きのアクセスをユーザに与えることができます。この方法では、ビューはセキュリティ・メカニズムの機能を果たします。たとえば、ビューを起動するユーザは、テーブルの特定のローとカラムにしかアクセスできません。暗黙のパーミッションの規則の詳細については、『システム管理ガイド』を参照してください。
- テーブル内に暗号化されているカラムがあるとき、ビューから選択するには、復号化パーミッションが必要です。ビューとそのオブジェクトの所有者が同じユーザではない場合、ビューから選択するには、そのテーブル内の暗号化カラムに対する復号化パーミッションが必要です。ビューとそのオブジェクトの所有者が同じユーザの場合、そのユーザがビューから選択するには、テーブル内の暗号化カラムに対応するビュー・カラムに対する復号化パーミッションを付与するだけで十分です。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>create view</code> 権限が必要。他のユーザに対して <code>create view</code> を実行するには、 <code>create any view</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、 <code>create view</code> 権限が必要。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	<code>extrainfo</code> の情報
16	<code>create</code>	<code>create view</code>	<ul style="list-style-type: none"> <li><code>Roles</code> — 現在のアクティブな役割</li> <li><code>Keywords</code> または <code>options</code> — NULL</li> <li><code>Previous value</code> — NULL</li> <li><code>Current value</code> — NULL</li> <li><code>Other information</code> — NULL</li> <li><code>Proxy information</code> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

マニュアル 『リファレンス・マニュアル：ビルディング・ブロック』の「第4章 式、識別子、およびワイルドカード文字」の「識別子」

コマンド `create schema`, `drop view`, `select`, `update`

システム・プロシージャ `sp_depends`, `sp_help`, `sp_helptext`, `sp_rename`

## dbcc

### 説明

データベースの一貫性チェッカ (dbcc) は、データベースの論理的および物理的な一貫性の検査を行い、統計、プラン、修復機能を提供します。

一部の dbcc コマンドは共有ディスク・クラスタにのみ適用されます。クラスタそれぞれの dbcc 構文を参照してください。

### 構文

```
dbcc addtempdb( dbid | database_name )
dbcc checkalloc [(database_name[, fix | nofix])]
dbcc checkcatalog [(database_name[, fix])
dbcc checkdb [(database_name[, skip_ncindex])]
dbcc checkindex ({table_name | table_id}, index_id
                [, bottom_up[, partition_name | partition_id]])
dbcc checkstorage [(database_name)
dbcc checktable (table_name | table_id
                [, skip_ncindex | fix_spacebits | "check spacebits" |
                bottom_up | NULL[, partition_name | partition_id])
dbcc checkverify (dbname[, tblname[, ignore_exclusions]])
dbcc complete_xact (xid, [{"commit", "1pc" } | "rollback"])
dbcc dbrepair (database_name, dropdb)
dbcc engine ({offline, [enginenum] | "online"})
dbcc fix_text ({table_name | table_id)
dbcc forget_xact (xid)
dbcc indexalloc (table_name | table_id, index_id
                [, optimized | fast | NULL [, fix | nofix | NULL
                [, partition_name | partition_id]])
dbcc monitor (increment, <group name>)
dbcc monitor (decrement, <group name>)
dbcc monitor (reset, <group name>)
dbcc pravailabletempdbs
dbcc rebuild_text (table_name | table_id | "all"[, column[, text_page
                [, data_partition_name | data_partition_id]])
dbcc reindex ({table_name | table_id)
dbcc serverlimits
dbcc stackused
dbcc tablealloc (table_name | table_id [, full | optimized | fast | NULL
                [, fix | nofix | NULL [, data_partition_name | data_partition_id]])
```



```

dbcc textalloc (table_name | table_id [, full | optimized | fast | NULL
               [, fix | nofix | NULL [, data_partition_name | data_partition_id]])
dbcc {traceon | traceoff} (flag [, flag ...])
dbcc tune ({ascinserts, {0 | 1} , table_name |
           cleanup, {0 | 1} |
           cpuaffinity, start_cpu {, on| off} |
           des_greedyalloc, dbid, object_name,
           " {on | off}" | deviochar vdevno, "batch_size" |
           des_bind, dbid, object_name
           des_unbind, dbid, object_name
           doneinproc {0 | 1})
dbcc upgrade_object [ ( dbid | dbname
                       [, [database.owner].compiled_object_name' |
                          'check' | 'default' | 'procedure' | 'rule' |
                          'trigger' | 'view'
                          [, 'force' ] ] )

```

クラスタ専用の

dbcc 構文 :

```

dbcc nodetraceon(trace_flag_number)
dbcc nodetraceoff(trace_flag_number)

dbcc set_scope_in_cluster("cluster"|"instance"|"scope")
dbcc quorum

```

## パラメータ

### addtempdb

使用可能なテンポラリ・データベースのグローバル・リストにテンポラリ・データベースを追加します。データベースが存在しない場合や、データベースがテンポラリ・データベースでない場合は、エラーが生成されます。データベースがすでにリストに登録されている場合は、情報メッセージが出力されます。

### *dbid*

データベース ID です。

### *database\_name*

検査対象のデータベースの名前です。データベース名を指定しない場合、dbcc は現在のデータベースを使用します。

### checkalloc

指定したデータベースを検査して、すべてのページが正しく割り付けられているか、または割り付けられているページで使用されていないページがないかどうかを確認します。データベース名が指定されていない場合、**checkalloc** は現在のデータベースを検査します。レポート・オプションは常に **optimized** が使用されます(「**tablealloc**」を参照)。

**checkalloc** は割り付けられ、使用されている領域の量をレポートします。

### fix | nofix

**dbcc** が、検出された割り付けエラーを修正するかどうかを設定します。**checkalloc** のデフォルト・モードは **nofix** です。**fix** オプションを使用するには、データベースをシングルユーザ・モードに切り替える必要があります。**Adaptive Server** でのページ割り付けの詳細については、『システム管理ガイド』を参照してください。

### checkcatalog

システム・テーブル内、およびシステム・テーブル間の一貫性を検査します。たとえば、**checkcatalog** では、**syscolumns** と一致するエントリが **systypes** にあるか、**sysobjects** のすべてのテーブルとビューのカラムが **syscolumns** に 1 つ以上あるか、**syslogs** の最終チェックポイントが有効であるかなどが確認されます。「[dbcc checkcatalog に よって実行される検査](#)」(335 ページ)を参照してください。**checkcatalog** はアーカイブ・データベースで使用できますが、**fix** バージョンの **checkcatalog** は使用できません。

**checkcatalog** は、定義されている任意のセグメントについてもレポートします。データベース名を指定しない場合、**checkcatalog** は現在のデータベースを検査します。

### fix

**dbcc** が検出した **sysindexes** エラーを修正するかどうかを指定します。**checkcatalog** のデフォルト・モードではエラーは修正されません。**fix** オプションを使用するには、データベースをシングルユーザ・モードに切り替える必要があります。新しい **sysindexes** 検査では、12.5.2 より前のバージョンの **Adaptive Server** の **dbcc checkcatalog** では発生しなかった新しいエラーが発生する可能性があります。

### checkdb

指定されたデータベース内で、**checktable** と同じ検査を、**syslogs** などのテーブルごとに実行します。データベース名を指定しない場合、**checkdb** は現在のデータベースを検査します。**checkdb** はアーカイブ・データベースで使用できます。

**skip\_ncindex**

このオプションを使用すると、**dbcc checktable** または **dbcc checkdb** は、ユーザ・テーブルのノンクラスタード・インデックスの検査を省略します。デフォルトではすべてのインデックスが検査されます。

**checkindex**

**checktable** と同じ検査を、指定されたインデックスだけを対象として実行します。**checkindex** はアーカイブ・データベースで使用できます。

**bottom\_up**

(データオンリーロック・テーブルのみ) このオプションを **checkindex** とともに指定すると、インデックスをボトムアップ方式で検査します。**bottom\_up** の検査では、各データ・ローに対応するインデックス・ローがあるかどうかも確認されます。

**partition\_name | partition\_id**

検査するデータ・パーティションの名前または ID です。パーティションを指定した場合、**dbcc** はグローバル・インデックスをスキップします。

**checkstorage**

指定されたデータベースを検査して、割り付け、オブジェクト・アロケーション・マップ (OAM) ページ・エントリ、ページの一貫性、テキスト値カラム、テキスト値カラムの割り付け、テキスト・カラム・チェーンを確認します。それぞれの **dbcc checkstorage** オペレーションの結果は、**dbccdb** データベースに格納されます。**dbcc checkstorage** の使い方、および **dbccdb** からのレポートの作成、管理、生成の方法の詳細については、『システム管理ガイド』を参照してください。

**checktable**

指定したテーブルを検査して、インデックスとデータ・ページが正しくリンクされているか、インデックスが正しいソート順になっているか、すべてのポインタが一貫しているか、各ページのデータ情報が妥当か、また、ページのオフセットが妥当かどうかを確認します。**checktable** はアーカイブ・データベースで使用できます。

**dbcc checktable** の変更には、仮想ハッシュ・テーブルに関係するものがあります。

- 通常の検査に加えて、**checktable** はハッシュ領域のデータ・ページと OAM ページのレイアウトが正しいことを確認します。
  - レイアウトごとに OAM ページ用に予約されているエクステンメント内に、データ・ページは割り付けられません。

- OAM ページは、アロケーション・ユニットの最初のエクステンション内のみに割り付けされます。

**table\_name | table\_id**

検査するテーブルの名前またはオブジェクト ID です。

**fix\_spacebits**

**datapages** ロックまたは **datarows** ロックを使用するテーブルを対象としています。空白ビットの妥当性を確認し、不正な空白ビットを修正します。空白ビットは、ページごとに保管され、ページへの挿入に使用できる空間を示します。

**check spacebits**

**datapages** ロックされたテーブルまたは **datarows** ロックされたテーブルを使用するテーブルの空白ビットを検査します。**check spacebits, dbcc** を指定すると、ノンクラスタード・インデックスは検査されません。

**checkverify**

指定したデータベースに使用した **dbcc checkstorage** で最新の実行結果を確認します。**dbcc checkverify** の使い方の詳細については、『システム管理ガイド』を参照してください。

**ignore\_exclusions**

除外リストを有効または無効にします。値は 0 (除外リストを有効にする) または 1 (無効にする) のいずれかです。デフォルト値は 0 です。

**complete\_xact**

作業をコミットするかまたはロールバックすると、自発的にトランザクションが完了します。**Adaptive Server** は、自発的に完了したすべてのトランザクションについての情報を **master.dbo.systransactions** テーブルに保持し、外部トランザクション・コーディネータが、トランザクションがどのように完了したかについて多少の知識を得られるようにします。

---

**警告！** 準備ステータスでトランザクションを自発的に完了すると、分散トランザクション全体の結果に一貫性がなくなる場合があります。システム管理者がトランザクションを自発的にコミットまたはロールバックすることに決定すると、**Adaptive Server** またはプロトコルを調整した場合の決定内容と矛盾する可能性があります。

---

**xid**

**systransactions.xactname** カラムのトランザクション名です。**sp\_transactions** を使用しても、有効な **xid** 値を決定できます。

**1pc**

外部トランザクション・マネージャによって(通常の2フェーズ・コミット・プロトコルではなく)1フェーズ・コミット・プロトコルでの最適化の対象として完了の調整が行われていたトランザクションを自発的に完了します。このオプションを使用すると、準備ステータスでなかったトランザクションを自発的にコミットできます。

**forget\_xact**

自発的に完了したトランザクションの完了ステータスを `master.dbo.systransactions` から削除します。`forget_xact` は、システム管理者が調整を行うサービスにトランザクションに自発的な完了を認識されないようにする場合、または外部コーディネータが `systransactions` のコミット・ステータスをクリアできない場合に使用できます。

---

**警告!** 通常の DTP 環境では `dbcc forget_xact` を絶対に使用しないでください。外部トランザクション・コーディネータによる自発的に完了したトランザクションの検出を許可することになるためです。X/Open XA 標準に準拠するトランザクション・マネージャと Adaptive Server のトランザクション調整サービスは、`systransactions` のコミット・ステータスを自動的にクリアします。

---

**dbrepair (database\_name, dropdb)**

損傷したデータベースを削除します。`drop database` は、損傷したデータベース上では機能しません。

この `dbcc` 文を発行した時点で削除対象となっていたデータベースは、この文を発行したユーザを含めて誰も使用できません。

**engine**

Adaptive Server のエンジンをオフラインにしたり、オンラインに設定したりします。`enginenum` を指定していない場合、`dbcc engine (offline)` は一番大きい番号の付いたエンジンをオフラインにします。『システム管理ガイド』の「マルチプロセッサ・サーバの管理」を参照してください。

**fix\_text**

Adaptive Server の文字セットがいずれかの文字セットから新しいマルチバイト文字セットに変更された後に、`text` 値をアップグレードします。

マルチバイト文字セットに変更すると、**text** データの内部管理がより複雑になります。**text** 値は数ページに渡るほど大きくなることがあるため、**Adaptive Server** ではページ境界をまたがる文字の処理が可能である必要があります。この処理を可能にするには、サーバに **text** ページごとの追加情報が必要となります。システム管理者またはテーブル所有者は、**text** データが含まれる各テーブルに対して **dbcc fix\_text** を実行して、必要な新しい値を計算する必要があります。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。

#### **indexalloc**

指定したデータベースを検査して、すべてのページが正しく割り付けられているか、または割り付けられているページで使用されていないページがないかどうかを確認します。これは **checkalloc** の簡易バージョンで、個々のインデックスについて同様の整合性検査を実行します。**indexalloc** はアーカイブ・データベースで使用できます。

**indexalloc** は、**tablealloc** と同様に **full**、**optimized**、および **fast** の 3 種類のレポートを生成します。種類が指定されていない場合や、**null** が使用されている場合、**Adaptive Server** では **optimized** が使用されます。**fix|nofix** オプションは、**indexalloc** とともに使用する場合も **tablealloc** と使用する場合と同様に機能します。

---

**注意** **fix** または **nofix** は、レポートのタイプの値 (**full**、**optimized**、**fast**、または **null**) を含めた場合にのみ指定できます。

---

#### **table\_name | table\_id**

テーブルの名前またはテーブルのオブジェクト ID です。

#### **indid**

**dbcc indexalloc** の実行時に検査されるインデックスの ID です。

#### **fix\_spacebits**

**datapages** ロック・スキームまたは **datarows** ロック・スキームのテーブルに使用します。空白ビットの妥当性を確認し、不正な空白ビットを修正します。空白ビットは、ページごとに保管され、ページへの挿入に使用できる空間を示します。

#### **check spacebits**

データページロック・テーブルまたはデータローロック・テーブルの空白ビットを検査します。**check spacebits**、**dbcc** を指定すると、ノックラスタード・インデックスは検査されません。

#### **full**

すべての種類の割り付けエラーをレポートします。

**optimized**

インデックスのオブジェクト・アロケーション・マップ (OAM) のページにリストされたアロケーション・ページに基づいてレポートを作成します。このオプションでは、OAM ページにリストされていないアロケーション・ページにある、参照されないエクステン트는レポートしません。また修正することもできません。optimized オプションがデフォルトです。

**fast**

このオプションは、割り付け (アロケーション) に関するレポートは作成しませんが、参照されているエクステン트에割り付けられていないページ (2521 レベルのエラー) の例外レポートを生成します。

**fix | nofix**

テーブル内で検出された割り付けエラーを `indexalloc` が修正するかどうかを指定します。システム・テーブル以外のすべてのインデックスのデフォルトは `fix` で、システム・テーブルのデフォルトは `nofix` です。システム・テーブルに対して `fix` オプションを使用するには、あらかじめデータベースをシングルユーザ・モードに切り替えておく必要があります。

`fix` または `nofix` を指定できるのは、レポートのタイプの値 (`full`、`optimized`、`fast`、または `null`) も指定する場合だけです。

**partition\_name | partition\_id**

パーティション ID を指定すると、(`indid`, `partition id`) によって識別されるパーティションに対して割り付けの検査が実行されます。

**pravailabletempdbs**

使用可能なテンポラリー・データベースのグローバル・リストを出力します。

**rebuild\_text**

`text`、`unitext`、または `image` データの Adaptive Server 12.0 以降の内部データ構造を再構築または作成します。Adaptive Server は、このデータ構造を使って、データのクエリ中にランダム・アクセスや非同期プリフェッチを実行できます。`rebuild_text` は、データベース、単一のテーブル、またはデータ・パーティション内のすべてのテーブルに実行できます。

**table\_name | table\_id | "all"**

テーブル名、テーブルのオブジェクト ID、またはデータベース内のすべてのオブジェクトです。

**column**

テキスト・カラムのカラム ID または名前です。`dbcc rebuild_text` は、このカラムの各テキスト値が持つ内部データ構造を再構築します。

**text\_page**

先頭 text ページの論理ページ番号です。dbcc rebuild\_text は、この text ページの内部データ構造を再構築します。

**data\_partition\_name | data\_partition\_id**

データ・パーティションの名前または ID です。text\_page を指定すると、data\_partition\_name (または data\_partition\_id) は無視されます。

**monitor increment, group name**

increment と decrement コマンドは、指定されたグループのモニタ・カウンタの使用カウントを 1 ずつインクリメントまたはデクリメントします。reset コマンドは、指定されたグループのモニタ・カウンタの使用カウントをゼロに設定します。これで、このグループのモニタリング・データの収集がオフになる。

group name には次のいずれかを指定できます。

- 'all' — all グループ (ほぼすべてのモニタ・カウンタを含む) の使用カウントを確認するには、@@monitors\_active グローバル変数を選択します。
- spinlock\_s — spinlock\_s の使用カウントは dbcc resource コマンドで報告されます。
- appl — appl の使用カウントは dbcc resource コマンドで報告されます。

**reindex**

dbcc checktable の高速バージョンを実行して、ユーザ・テーブルのインデックスの整合性を検査します。このオプションは、テーブル名またはテーブルのオブジェクト ID (sysobjects の id カラム) とともに使用できます。reindex は、1 つ目のインデックス関連のエラーを発見するとメッセージを出力し、次に、suspect (疑わしい) インデックスを削除して再作成します。Adaptive Server のソート順が変更され、インデックスに破壊された疑いがあるとして "suspect" のマークが Adaptive Server によって付けられた場合、システム管理者またはテーブル所有者は dbcc reindex を実行する必要があります。

dbcc は、破壊されたインデックスを検出すると、そのインデックスを削除し、適切なインデックスを再作成します。テーブルのインデックスがすでに適正である場合、またはテーブルのインデックスがない場合、dbcc reindex によるインデックスの再構築は行われません。ただし、情報メッセージが出力されます。



破壊されたデータがテーブルに存在すると疑われる場合、`dbcc reindex` はアボートします。この場合はエラー・メッセージが表示され、`dbcc checktable` を実行するようにユーザに指示します。`dbcc reindex` では、システム・テーブルのインデックスの再作成ができません。システム・インデックスは、**Adaptive Server** がソート順を変更して再起動した後で、自動リカバリの一部として必要に応じて検査され再構築されます。

#### stackused

サーバが最初に起動されてから使用されたスタック・メモリの最大量をレポートします。

#### serverlimits

**Adaptive Server** のさまざまなエンティティの制限を表示します。たとえば、識別子の長さ、テーブル内のカラム数などの各種オブジェクトの最大数、テーブルのインデックス数、ページ・サイズ、ローのオーバーヘッドなどが含まれます。この情報は、**Adaptive Server** プロセスのさまざまなサイズの特徴を確認するために使用します。

#### tablealloc

指定したテーブルまたはデータ・パーティションを検査して、すべてのページが正しく割り付けられているか、または割り付けられているページで使用されていないページがないかどうかを確認します。これは、`checkalloc` の簡易バージョンであり、個々のテーブルに対して同様の整合性検査を実行します。これは、テーブル名またはテーブルのオブジェクト ID (`sysobjects` の `id` カラム) と合わせて使用できます。アーカイブ・データベースで、`tablealloc` を使用できません。`tablealloc` の出力例については、『システム管理ガイド』を参照してください。

`tablealloc` では、`full`、`optimized`、`fast` という 3 種類のレポートを生成できます。種類が指定されていない場合や、`null` が使用されている場合、**Adaptive Server** では `optimized` が使用されます。

#### textalloc

データベースの `text` ページまたは `image` ページの割り付け整合性をチェックします。アーカイブ・データベースで、`dbcc textalloc` を使用できます。

#### full

このオプションは、テーブル・レベルの `checkalloc` に相当します。すべての種類の割り付けエラーをレポートします。

**optimized**

テーブルのオブジェクト・アロケーション・マップ (OAM) のページにリストされたアロケーション・ページに基づいてレポートを作成します。このオプションでは、**OAM** ページにリストされていないアロケーション・ページにある、参照されないエクステントはレポートしません。また修正することもできません。**optimized** オプションがデフォルトです。

**fast**

このオプションは、割り付け (アロケーション) に関するレポートは作成しませんが、参照されていてもエクステントに割り付けられていないページ (2521 レベルのエラー) の例外レポートを生成します。

**fix | nofix**

テーブルで検出された割り付けエラーを **tablealloc** が修正するかどうかを指定します。システム・テーブル以外のすべてのテーブルのデフォルトは、**fix** となります。システム・テーブルのデフォルトは **nofix** です。システム・テーブルに対して **fix** オプションを使用するには、あらかじめデータベースをシングルユーザ・モードに切り替えておく必要があります。

**fix** または **nofix** を指定できるのは、レポートのタイプの値 (**full**、**optimized**、**fast**、または **null**) も指定する場合だけです。

**data\_partition\_name | data\_partition\_id**

検査するデータ・パーティションの名前または **ID** です。パーティションを指定した場合、**dbcc tablealloc** はグローバル・インデックスをスキップします。

**traceon | traceoff**

クエリ最適化中に診断の出力を切り替えます。**3604** の値はトレース出力の送信先をユーザ・セッションに切り替え、**3605** の値はトレース出力の送信先をエラー・ログに切り替えます。

**tune**

特別なパフォーマンス状態に対応するチューニング・フラグを有効または無効にします。**dbcc tune** コマンドは、**Adaptive Server** を再起動するたびに再発行する必要があります。各オプションの詳細については、『パフォーマンス & チューニング・ガイド：基本』を参照してください。

**upgrade\_object**

**syscomments** テーブルに保管されたテキストを使用して、コンパイル済みオブジェクトをアップグレードします。**upgrade\_object** パラメータは次のとおりです。

パラメータ	説明
<b>dbid</b>	データベース ID を指定する。 <b>dbid</b> を指定しない場合は、現在のデータベースのすべてのコンパイル済みオブジェクトがアップグレードされます。
<b>dbname</b>	データベースの名前を指定する。 <b>dbname</b> を指定しない場合は、現在のデータベース内のすべてのコンパイル済みオブジェクトがアップグレードされます。
<b>compiled_object_name</b>	アップグレードする特定のコンパイル済みオブジェクトの名前。完全修飾名を使用する場合、 <b>dbname</b> および <b>database</b> は一致する必要があります。また、完全修飾名は引用符で囲む必要があります。データベースが同じ名前の複数のコンパイル済みオブジェクトを含んでいる場合は、完全修飾名を使用してください。そうしないと、同じ名前を持つすべてのオブジェクトが解析され、エラーがない場合はアップグレードされます。
<b>check</b>	すべての検査制約とルールをアップグレードする。参照整合性制約はコンパイル済みオブジェクトではないので、アップグレードの必要はありません。
<b>default</b>	すべての宣言デフォルトと、 <b>create default</b> コマンドで作成されたデフォルトをアップグレードする。
<b>procedure</b>	すべてのストアド・プロシージャをアップグレードする。
<b>rule</b>	すべてのルールと検査制約をアップグレードする。
<b>trigger</b>	すべてのトリガをアップグレードする。
<b>view</b>	すべてのビューをアップグレードする。
<b>force</b>	指定したオブジェクトに <b>select * </b> 句が含まれる場合も、アップグレードする必要がある場合に指定する。 <b>force</b> は、 <b>select * </b> 文が予期しない結果を返すことがないという確信がないかぎり使用しないでください。 <b>force</b> オプションは、予約語を含んでいるオブジェクト、トランケートまたは削除されたソース・テキストを含んでいるオブジェクト、存在しないテンポラリ・テーブルを参照するオブジェクト、引用符で囲まれた識別子の設定と合わないオブジェクトはアップグレードしません。これらのオブジェクトは、修正してからアップグレードする必要があります。

キーワード **check**、**default**、**procedure**、**rule**、**trigger**、**view** は、アップグレードされるコンパイル済みオブジェクトのクラスを指定します。クラスを指定すると、**dbcc upgrade\_object** がエラーや問題が発生する可能性のある領域を検出しない限り、指定したデータベース内のそのクラスのすべてのオブジェクトがアップグレードされます。

**check**

指定したデータベース内の **syscomments** の指定したコンパイル済みオブジェクトの構文を検査します。**select** のオカレンスではエラーを生成しません。

`upgrade_object` は、すべての検査制約とルールをアップグレードします。参照整合性制約はコンパイル済みオブジェクトではないので、アップグレードの必要はありません。

**force**

`syscomments` のオブジェクトを、アップグレードが不要な場合でも強制的にアップグレードします。

**object\_name**

コンパイル済みオブジェクトの名前です。

**object\_type**

Adaptive Server がコンパイルする次のオブジェクト・タイプのいずれかです。procedure、function、view、trigger、default、rule、condition

**compiled\_object\_name**

アップグレードする特定のコンパイル済みオブジェクトの名前です。完全修飾名を使用する場合、`database_name` および `database` が一致している必要があります。また、完全修飾名は引用符で囲む必要もあります。データベースが同じ名前の複数のコンパイル済みオブジェクトを含んでいる場合は、完全修飾名を使用してください。そうしないと、同じ名前を持つすべてのオブジェクトが解析され、エラーがない場合はアップグレードされます。

**default**

すべての宣言デフォルトと、`create default` コマンドで作成されたデフォルトをアップグレードします。

**procedure**

すべてのストアド・プロシージャをアップグレードします。

**rule**

すべてのルールと検査制約をアップグレードします。

**trigger**

すべてのトリガをアップグレードします。

**view**

すべてのビューをアップグレードします。

キーワード `check`、`default`、`procedure`、`rule`、`trigger`、`view` は、アップグレードされるコンパイル済みオブジェクトのクラスを指定します。クラスを指定すると、`dbcc upgrade_object` がエラーや問題が発生する可能性のある領域を検出しない限り、指定したデータベース内のそのクラスのすべてのオブジェクトがアップグレードされます。

**force**

指定したオブジェクトが **select \* 句** を含んでもアップグレードしたい場合に指定します。**force** は、**select \* 文** が予期しない結果を返すことがないという確信がないかぎり使用しないでください。**force** オプションは、予約語を含んでいるオブジェクト、トランケートまたは削除されたソース・テキストを含んでいるオブジェクト、存在しないテンポラリ・テーブルを参照するオブジェクト、引用符で囲まれた識別子の設定と合わないオブジェクトはアップグレードしません。これらのオブジェクトは、修正してからアップグレードする必要があります。

**trace\_flag\_number**

有効化または無効化するトレース・フラグの番号です。

**cluster**

**dbcc** コマンド範囲をクラスタに設定します。後続の **dbcc** コマンドは、クラスタ全体に影響します。

**instance**

**dbcc** コマンド範囲を現在のインスタンスに設定します。後続の **dbcc** コマンドは、ローカル・インスタンスのみに影響します。

**scope**

**cluster** または **instance** の **dbcc** コマンドの現在のスコープを表示します。

例

**例 1** pubs2 のページ割り付けエラーを検査します。

```
dbcc checkalloc (pubs2)
```

**例 2** pubs2 のデータベースの一貫性を検査し、**dbccdb** データベースに情報を保管します。

```
dbcc checkstorage (pubs2)
```

**例 3** salesdetail テーブルを検査します。

```
dbcc checktable (salesdetail)
```

```
Checking salesdetail
```

```
The total number of pages in partition 1 is 3.
```

```
The total number of pages in partition 2 is 1.
```

```
The total number of pages in partition 3 is 1.
```

```
The total number of pages in partition 4 is 1.
```

```
The total number of data pages in this table is 10.
```

```
Table has 116 data rows.
```

```
DBCC execution completed.If DBCC printed error
```

```
messages, contact a user with system administrator (SA)
```

```
role.
```

**例 4** トランザクション "distributedxact1" を自発的にアボートします。

```
dbcc complete_xact (distributedxact1, "rollback")
```

**例 5** 文字セットの変更後に、blurbs のテキスト値をアップグレードします。

```
dbcc fix_text (blurbs)
```

**例 6** 除外リストを無効にした状態で、データベース my\_db にあるテーブル tab に対して checkverify を実行します。

```
dbcc checkverify(my_db, tab)
```

**例 7** 除外リストを有効にした状態で、データベース my\_db にあるテーブル tab に対して dbcc checkverify を実行します。

```
dbcc checkverify (my_db, tab, 0)
```

**例 8** 除外リストを無効にした状態で、データベース my\_db にあるテーブル tab で dbcc checkverify を実行します。

```
dbcc checkverify (my_db, tab, 1)
```

**例 9** "distributedxact1" トランザクションの情報を master.dbo.systransactions から削除します。

```
dbcc forget_xact (distributedxact1)
```

**例 10** titleauthor テーブルで indid が 2 のインデックスの割り付けに関する完全なレポートを出力し、検出した割り付けエラーをすべて修正します。

```
dbcc indexalloc ("pubs..titleauthor", 2, full)
```

**例 11** 使用可能なテンポラリ・データベースのグローバル・リストを出力します。

```
dbcc pravailabletempdbs
```

```
Available temporary databases are:
```

```
Dbid:2
```

```
Dbid:4
```

```
Dbid:5
```

```
Dbid:6
```

```
Dbid:7
```

```
DBCC execution completed.If DBCC printed error  
messages, contact a user with system administrator (SA)  
role.
```

**例 12** blurbs テーブルの text および image カラムすべての Adaptive Server 内部データ構造を再構築または作成します。

```
dbcc rebuild_text (blurbs)
```

**例 13** `smallsales` パーティション (書籍の総売上額 5000 未満のすべてが格納される) にある `titles` テーブルの一部を検査します。

```
dbcc checktable (titles, NULL, "smallsales")
```

**例 14** `dbcc reindex` が、`titles` テーブル内の 1 つ以上のインデックスが破壊されているのを発見しました。

```
dbcc reindex (titles)
```

```
One or more indexes are corrupt.They will be rebuilt.
```

**例 15** `Adaptive Server` が最初に起動されてから使用されたスタック・メモリの最大量を検査します。

```
dbcc stackused
```

**例 16** `listdb` データベース内のすべてのストアド・プロシージャをアップグレードします。

```
dbcc upgrade_object(listdb, 'procedure')
```

**例 17** `listdb` データベース内のすべてのルールと検査制約をアップグレードします。`rule` が二重引用符で囲まれているのは、`set quoted identifiers` が `off` だからです。

```
dbcc upgrade_object(listdb, list_proc)
```

**例 18** `Adaptive Server` の制限に関するクライアント・セッションへの出力を、さまざまな形式で示す省略形出力を表示します。

```
dbcc serverlimits
```

```
Limits independent of page size:
=====
```

```
Server-wide, Database-specific limits and sizes
```

```
Max engines per server                :128
Max number of logins per server        :2147516416
Max number of users per database       :2146484223
Max number of groups per database      :1032193
Max number of user-defined roles per server :1024
Max number of user-defined roles per (user) session :127
Min database page size                 :2048
Max database page size                 :16384
...
```

```
Database page-specific limits
```

```
APL page header size                  :32
```

```

DOL page header size :44
Max reserved page gap :255
Max fill factor :100

```

#### Table, Index related limits

```

Max number of columns in a table/view :1024
Max number of indexes on a table :250
Max number of user-keys in a single index on an unpartitioned table :31
Max number of user-keys in a single local index on a partitioned table :31
...

```

#### General SQL related

```

Max size of character literals, sproc parameters :16384
Max size of local @variables in T-SQL :16384
Max number of arguments to stored procedures :2048
Max number of arguments to dynamic SQL :2048
Max number of aggregates in a COMPUTE clause :254
...

```

#### Maximum lengths of different Identifiers

```

Max length of server name :30
Max length of host name :30
Max length of login name :30
Max length of user name :30
...

```

#### Limits as a function of the page size:

```

=====

```

```

Item dependent on page size : 2048 4096 8192 16384
-----

```

#### Server-wide, Database-specific limits and sizes

```

Min number of virtual pages in master device :11780 22532 45060 90116
Default number of virtual pages in master device :23556 45060 90116 180228
Min number of logical pages in master device :11776 11264 11264 11264
Min number of logical pages in tempdb :2048 1536 1536 1536

```

#### Table-specific row-size limits

```

Max possible size of a log-record row on APL log page :2014 4062 8158 16350

```



```

Physical Max size of an APL data row, incl row-overheads :1962 4010 8106
16298
Physical Max size of a DOL data row, incl row-overheads :1964 4012 8108
16300

Max user-visible size of an APL data row :1960 4008 8104 16296
Max user-visible size of a DOL data row :1958 4006 8102 16294
Max user-visible size of a fixed-length column in an APL table :1960 4008
8104 16296
Max user-visible size of a fixed-length column in a DOL table :1958 4006
8102 16294
...

```

---

**注意** サーバの制限の完全なリストを表示するには、`dbcc traceon (3604)` を実行してクライアント・セッションへの出力を取得します。

---

**例 19 Adaptive Server** はこのテーブルの割り付けについて最適化レポートを出力しますが、検出した割り付けエラーは修正しません。

```
dbcc tablealloc (publishers, null, nofix)
```

**例 20 smallsales** パーティションに対して割り付け検査を実行します。`smallsales` のすべてのローカル・インデックスはこの検査の対象になりますが、グローバル・インデックスは対象外です。

```
dbcc tablealloc (titles, null, null, smallsales)
```

**例 21 sp\_transactions** を使用して、「準備」ステータスでなかったために自発的にコミットしなかった 1 フェーズ・コミット・トランザクションの名前を判別します。次に、トランザクションを正常にコミットするための `1pc` パラメータの使用方法を示します。

```

sp_transactions

xactkey                type      coordinator starttime
state                 connection dbid spid  loid failover   srvnname namelen
xactname
-----
-----
-----
0xbc0500000b00000030c316480100 External XA          Feb 2 2004 1:07PM
Done-Detached Detached      1    0 2099 Resident Tx NULL      88
28_u7dAc31Wc3800000000000000000000000000000000000000001HFfpfSxkDM000FU_00003M00
    00Y_:SYBBEV0A_LRM
(1 row affected)

(return status = 0)

```

次のトランザクションをコミットしようとする、エラー・メッセージが表示されます。

```
dbcc complete_xact
("28_u7dAc31Wc3800000000000000000000000000000000000000000000000000001HFpfSxkDM000FU_00003M000
0Y_:SYBBEV0A_LRM", "commit")
```

Adaptive Server は、次のエラー・メッセージを返します。

```
Msg 3947, Level 16, State 1:
Server 'PISSARRO_1251_P', Line 1:
A heuristic completion related operation failed.Please
see errorlog for more details.
DBCC execution completed.If DBCC printed error
messages, contact a user with system administrator (SA)
role.
```

トランザクションは “done” 状態であるため、トランザクションがコミットされたことを確認してから、1 フェーズ・コミット・プロトコルでの最適化を使用してトランザクションを自発的に完了できます。次のように `dbcc complete_xact ( "1pc" )` パラメータを使用すると、このトランザクションをコミットできます。

```
dbcc complete_xact
("28_u7dAc31Wc3800000000000000000000000000000000000000000000000000001HFpfSxkDM000FU_00003M000
0Y_:SYBBEV0A_LRM", "commit", "1pc")
DBCC execution completed. If DBCC printed error messages, contact a user
with system administrator (SA) role.
```

`dbcc forget_xact` コマンドを使用すると `systransactions` からトランザクションを削除できます。

```
dbcc forget_xact
("28_u7dAc31Wc3800000000000000000000000000000000000000000000000000001HFpfSxkDM0
00FU_00003M0000Y_:SYBBEV0A_LRM")
DBCC execution completed.If DBCC printed error messages, contact a user with
system administrator (SA) role.
```

`sp_transactions` を再び実行すると、前のトランザクションは表示されません。

```
sp_transactions
xactkey type coordinator starttime state connection dbid spid
      loid failover srvnname namelen xactname
-----
(0 row affected)
```

**例 22** トレース・フラグ 3604 を有効化します。

```
dbcc nodetraceoff(3604)
DBCC execution completed. If DBCC printed error
messages, contact a user with system administrator (SA)
role.
```

**例 23** dbcc スコープをクラスタに設定します。

```
dbcc set_scope_in_cluster('cluster')
```

**例 24** dbcc スコープをインスタンスに設定します。

```
dbcc set_scope_in_cluster('instance')
```

**例 25** dbcc コマンドの現在のスコープを表示します。

```
dbcc set_scope_in_cluster('scope')
```

## 使用法

- **dbcc checkstorage** は、非ハッシュ・テーブルの最初のデータ・ページ以外のデータ・ページが空の場合、ソフト・フォールトをレポートします。ただし、**dbcc checkstorage** は仮想ハッシュ・テーブルのハッシュ領域ではこのソフト・フォールトをレポートしません。仮想ハッシュ・テーブルのハッシュ領域のデータ・ページは空でもかまいません。
- **dbcc** は、データベースがアクティブであっても実行できます。ただし、**dbrepair (database\_name, dropdb)** オプションと、**fix** オプションを指定した **dbcc checkalloc** の場合は例外です。
- **dbcc** は、データベース・オブジェクトの検査時にはデータベース・オブジェクトをロックします。**dbcc** の使用時のパフォーマンス上の問題を最小限にする方法の詳細については、『システム管理ガイド』で **dbcc** に関する説明を参照してください。
- **dbcc** コマンドの実行中、ユーザはアーカイブ・データベースにアクセスできません。**dbcc** コマンドの実行中にアーカイブ・データベースにアクセスしようとする、データベースがシングルユーザ・モードになっていることを示すメッセージが表示されます。
- ほとんどの **dbcc** コマンドは、インメモリ・データベースおよびリラックス持続性データベースで動作します。
- **dbcc** コマンドの変形は、アーカイブ・データベースがオンラインでもオフラインでも使用できます。ただし、**fix** オプションが設定された **dbcc** は、オンラインのアーカイブ・データベースでのみ使用できます。
- ユーザ名かデータベース名でテーブルまたはインデックス名を修飾するには、修飾名を一重引用符か二重引用符で囲みます。次に例を示します。

```
dbcc tablealloc ("pubs2.pogo.testtable")
```

- ユーザ定義のトランザクション内では、**dbcc reindex** を実行できません。
- **dbcc fix\_text** によって大量のログ・レコードが生成され、トランザクション・ログが満杯になる場合があります。**dbcc fix\_text** は、更新が一連の小さなトランザクションで実行されるように設計されています。ログ領域エラーの場合でも、少量の作業分しか失われません。ログの領域が不足した場合は、ログをクリアし、元の **dbcc fix\_text** がエラーになったときにアップグレード中だったテーブルを使用して、**dbcc fix\_text** を再起動します。
- 複写データベースを使用している場合に、以前のバージョンの **Adaptive Server** からダンプをロードする必要があるときは、**dbcc dbrepair** を使用します。次に例を示します。
  - 旧バージョンの **Adaptive Server** の運用システムから最新バージョンの **Adaptive Server** のテスト・システムに、ダンプをロードする。
  - ウォーム・スタンバイ・アプリケーションで、最新バージョンの **Adaptive Server** のスタンバイ・データベースを、旧バージョンの **Adaptive Server** のアクティブ・データベースからのデータベース・ダンプで初期化する。
- マルチバイトの文字セットに変更した後で **dbcc fix\_text** を実行する前に **text** 値に対して **select**、**readtext**、または **writetext** を使用しようとすると、そのコマンドは失敗します。次にエラー・メッセージが表示され、テーブルに対して **dbcc fix\_text** を実行するように指示されます。ただし、文字セットの変更後に **dbcc fix\_text** を実行しなくても、**text** ローの削除はできます。
- **dbcc** の出力は、結果ローとしてではなく、メッセージまたはエラーとして送信されます。クライアント・プログラムとスクリプトが、該当するエラー・ハンドラを検査します。
- テーブルが分割されている場合は、**dbcc checktable** によってそれぞれの分割についての情報が返されます。
- **Adaptive Server** バージョン 12.x 以降にアップグレードされている **text** および **image** データは、元の記憶フォーマットへの自動的なアップグレードが行われません。クエリのパフォーマンスを改善し、このデータのプリフェッチを可能にするには、アップグレードされている **text** および **image** カラムに対して **rebuild\_text** キーワードを使用します。

- 過去に使用されたスタック・メモリ量は、将来のニーズの可能性を示すだけです。過去の使用量を上回るスタック・メモリが必要になる場合もあります。`dbcc stackused` を定期的に行って、現在のスタック・メモリ使用量を調べてください。
- `dbcc upgrade_object check` は、Adaptive Server をアップグレードする前に発生した Adaptive Server の異常が原因で起こった `syscomments` テキストの破損を検出するために使用します。アップグレード・プロセスが失敗するため、この `syscomments` テキストの破損は重大です。
- `dbcc upgrade_object check` でエラーが報告された場合は、`compiled_object` を削除して、再作成します。

#### *dbcc complete\_xact*

`dbcc complete_xact` を使用すると、システム管理者は外部トランザクション・コーディネータが機能しない状態で、分散トランザクションをコミットまたはロールバックできます。15.0 より前のバージョンの Adaptive Server では、トランザクションは「準備」状態でないかぎり自発的にコミットできませんでした。トランザクション・コーディネータでは、トランザクションをコミットするために 2 フェーズ・コミット・プロトコルが使用されていました。ただし、トランザクションのコミットに 1 フェーズ・コミット・プロトコルを使用するのが最適な場合もあります。

`1pc` を使用すると、外部トランザクション・マネージャによって (通常の 2 フェーズ・コミット・プロトコルではなく) 1 フェーズ・コミット・プロトコルでの最適化の対象として完了の調整が行われていたトランザクションが自発的に完了します。このようなトランザクションを自発的にコミットするには、トランザクションが "done" 状態 (`sp_transactions` でレポートされる) であることが必要です。

---

**注意** トランザクションを自発的に完了する前に、システム管理者は調整を行っているトランザクション・マネージャが分散トランザクションをコミットまたはロールバックしたかどうかをあらゆる方法で判別する必要があります。

---

#### *dbcc checkcatalog* によって実行される検査

`dbcc checkcatalog` は、次の検査を実行します。

- 範囲分割、ハッシュ分割、リスト分割のテーブルにマップされる `sysindexes` の各ローについては、`sysindexes.conditionid` が `sysobjects.id` に等しい場合に `sysobjects` にローが 1 つ存在するかどうかを検査されます。また、`dbcc checkcatalog` は、分割条件が存在するラウンドロビン分割テーブルにマップされる `sysindexes` の各ローに対してもこのチェックを実行します。
- 範囲分割テーブル、ハッシュ分割テーブル、またはリスト分割テーブルにマップする `sysindexes` 内の各ローについては、`sysindexes.conditionid` が `sysprocedures.id` に等しい場合に `sysprocedures` にローが 1 つ以上存在するかどうかを検査されます。また、`dbcc checkcatalog` は、分割条件があるラウンドロビン分割テーブルにマップされる `sysindexes` の各ローに対してもこの検査を実行します。
- 範囲分割、ハッシュ分割、またはリスト分割のテーブルにマップされる `sysindexes` 内の各ローについては、次の条件が `true` の場合に `syspartitionkeys` にローが 1 つ存在するかが検査されます。この条件とは、`sysindexes.id` が `syspartitionkeys.id` に等しく、`sysindexes.indid` が `syspartitionkeys.indid` に等しいことです。また `dbcc checkcatalog` は、分割条件が存在するラウンドロビン分割テーブルにマップされる `sysindexes` 内の各ローに対してもこの検査を実行します。
- `sysindexes` 内の各ローについては、次の条件の両方が `true` の場合に、`syspartitions` 内にローが 1 つ以上存在するかどうかを検査されます。この条件とは、`sysindexes.id` が `syspartitions.id` に等しく、かつ、`sysindexes.indid` が `syspartitions.indid` に等しいことです。
- 型が N である `sysobjects` 内の各ローについて、`sysindexes.conditionid` が `sysobjects.id` に等しい 1 つのローが `sysindexes` にあるかどうか。
- `syspartitions` 内の各ローについては、次の条件が `true` の場合に、`sysindexes` 内にローが 1 つ存在するかどうかを検査されます。この条件とは、`syspartitions.id` が `sysindexes.id` に等しく、`syspartitions.indid` が `sysindexes.indid` に等しいことです。
- `syspartitionkeys` 内の各ローについては、次の条件が `true` の場合に、`sysindexes` 内にローが 1 つ存在するかどうかを検査されます。この条件とは、`syspartitionkeys.id` が `sysindexes.id` に等しく、`syspartitionkeys.indid` が `sysindexes.indid` に等しいことです。
- `syspartitions` 内の各ローについては、次の条件が `true` の場合に、`syssegments` 内にローが 1 つ存在するかどうかを検査されます。この条件とは、`syspartitions.segments` が `syssegments.segment` に等しいことです。

- `systabstats` 内の各ローについては、次の条件が `true` の場合に、`syspartitions` 内にローが存在するか同かが検査されます。この条件とは、`syspartitions.id` が `systabstats.id` に等しく、`syspartitions.indid` が `systabstats.indid` に等しく、`syspartitions.partitionid` が `systabstats.partitionid` に等しいことです。
- テキスト・インデックス (`indid=255`) は、`systabstats` にエントリを持ちません。
- `sysstatistics` 内の各ローについては、次の条件が `true` の場合に、`sysobjects` 内にローが 1 つ存在するかどうかを検査されます。この条件とは、`sysstatistics.id` が `sysobjects.id` に等しいことです。
- `sysobjects` の暗号化キーのローごとに、Adaptive Server は、キーを定義するローを `sysencryptkeys` でチェックします。
- 暗号化が指定された `syscolumns` 内の各ローについては、Adaptive Server によって `sysobjects` および `sysencryptkeys` でキーの存在が検証されます。
- `dbcc checkcatalog` は、次を確認します。
  - `sysencryptkeys` のすべてのキー・コピーについて、対応するベース・キーが `sysencryptkeys` に存在する。ベース・キーが存在しない場合、Adaptive Server はエラーを発行する。
  - すべてのキー・コピーについて、対応する `uid` が `sysusers` に存在する。`uid` が存在しない場合、Adaptive Server はエラーを発行する。
  - カラムで定義されたすべての復号化デフォルトについて、対応する復号化デフォルトが `sysobjects` と `sysattributes` に存在する。対応する復号化デフォルトが存在しない場合、Adaptive Server はエラーを発行する。

#### dbcc checktable の使用

ログ・セグメントが専用のデバイス上にある場合は、`syslogs` テーブルで `dbcc checktable` を実行すると、使用されているログおよび空き領域がレポートされます。次に例を示します。

```
Checking syslogs
```

```
The total number of data pages in this table is 1.
```

```
*** NOTICE:Space used on the log segment is 0.20 Mbytes, 0.13%.
```

```
*** NOTICE:Space free on the log segment is 153.4 Mbytes, 99.87%.
```

```
DBCC execution completed.  If dbcc printed error messages, see your system administrator.
```

ログ・セグメントが専用のデバイス上にない場合は、次のメッセージが表示されます。

```
*** NOTICE: Notification of log space used/free cannot be reported because the log segment is not on its own device.
```

実行される通常のチェックに加えて、**checktable** は、テーブル作成時に実行された事前割り付けが正しいかどうかを確認します。

- 事前割り付けされたページ数は、特定の**最大ハッシュ・キー**値に割りつけられる必要のあるデータ・ページの総数と一致します。
- 事前割り付けスキームによってオブジェクト・アロケーション・マップ (OAM) ページのみが使用できるように指定されるエクステンション内では、データ・ページは事前割り付けされません。
- OAM ページは、アロケーション・ユニットの最初のエクステンション内だけに割り付けされます。

#### dbcc nodetraceoff および dbcc nodetraceon ( クラスタのみ ) の使用

**dbcc traceon** および **dbcc traceoff** は、クラスタ全体にトレース・フラグを適用しますが、**dbcc nodetraceoff** および **dbcc nodetraceon** は、トレース・フラグをローカルに適用します。

#### dbcc quorum ( クラスタのみ ) の使用

- **dbcc quorum** 出力の送信先は次のとおりです。
  - デフォルトでは **Adaptive Server** を起動したターミナル
  - **トレース・フラグ 3604** または **3605** がオンの場合はクライアント・セッション
- **dbcc quorum** は、出力するビュー・レコードの数の整数パラメータを受け入れます。たとえば、最新の上位 20 のビュー・レコードを出力するには、次のコマンドを使用します。

```
dbcc quorum(20)
```

- パラメータを指定しない場合、**dbcc quorum** は、最新の上位 10 のビュー・レコードを出力します。
- すべてのレコードを表示するには、**dbcc quorum (-1)** を発行します。

#### dbcc checkstorage での共有ディスク・クラスタの制限事項

- インスタンス専用の名前付きキャッシュを **-dbcc checkstorage** に含めることはできません。含めると、**dbcc checkstorage** は、次のエラー・メッセージを出力します。

```
The cache %1! cannot be used because it is an instance only cache
```



- ローカル・テンポラリ・データベースに対して `dbcc checkstorage` を実行するには、ローカル・テンポラリ・データベースを所有する同じインスタンスからコマンドを実行する必要があります。
- パフォーマンス上の理由により、Cluster Edition の `dbcc checkstorage` はクラスタのページの最新バージョンに対するクエリを行わない場合があります。このため、Cluster Edition ではレポートされるソフト・フォールトの数が他のバージョンより多くなる可能性があります。

単一インスタンスが 1 つのデータベースを更新するように適切に分割されたアプリケーションの場合、`dbcc checkstorage` は、Cluster Edition 以外の Adaptive Server の以前のバージョンと同様に動作します。

## 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

## パーミッション

`dbcc` コマンドのパーミッション・チェックは、細密なパーミッションの設定によって異なります。`dbcc` コマンドそれぞれのパーミッション要件については、表 1-17 を参照してください。

**注意** `dbcc` コマンドは、表内で各コマンドに記載された要件（権限または所有権）のいずれかを保持している場合に実行できます。

表 1-17: `dbcc` コマンドのパーミッション要件

DBCC コマンド名	パーミッション要件	
	細密なパーミッションが無効	細密なパーミッションが有効
<code>addtempdb</code>	<ul style="list-style-type: none"> <li>データベース所有者</li> <li><code>sa_role</code></li> </ul>	<ul style="list-style-type: none"> <li>データベース所有者</li> <li><code>own database</code></li> </ul>
<code>checkalloc</code>	<ul style="list-style-type: none"> <li><code>dbcc checkalloc</code></li> <li>データベース所有者</li> <li><code>sa_role</code></li> </ul>	<ul style="list-style-type: none"> <li><code>dbcc checkalloc</code></li> </ul>
<code>checkcatalog</code>	<ul style="list-style-type: none"> <li><code>dbcc checkcatalog</code></li> <li>データベース所有者</li> <li><code>sa_role</code></li> </ul>	<ul style="list-style-type: none"> <li><code>dbcc checkcatalog</code></li> </ul>
<code>checkdb</code>	<ul style="list-style-type: none"> <li><code>dbcc checkdb</code></li> <li>データベース所有者</li> <li><code>sa_role</code></li> </ul>	<ul style="list-style-type: none"> <li><code>dbcc checkdb</code></li> </ul>
<code>checkindex</code>	<ul style="list-style-type: none"> <li><code>dbcc checkindex</code></li> <li>テーブル所有者</li> <li><code>sa_role</code></li> </ul>	<ul style="list-style-type: none"> <li><code>dbcc checkindex</code></li> <li>テーブル所有者</li> </ul>

DBCC コマンド名	パーミッション要件	
	細密なパーミッションが無効	細密なパーミッションが有効
checkstorage	<ul style="list-style-type: none"> <li>• dbcc checkstorage</li> <li>• データベース所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• dbcc checkstorage</li> </ul>
checktable	<ul style="list-style-type: none"> <li>• dbcc checktable</li> <li>• テーブル所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• dbcc checktable</li> <li>• テーブル所有者</li> </ul>
checkverify	<ul style="list-style-type: none"> <li>• dbcc checkverify</li> <li>• データベース所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• dbcc checkverify</li> </ul>
complete_xact	<ul style="list-style-type: none"> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• manage server</li> </ul>
dbrepair dropdb	<ul style="list-style-type: none"> <li>• データベース所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• データベース所有者</li> <li>• own database</li> </ul>
engine	<ul style="list-style-type: none"> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• manage server</li> </ul>
fix_text	<ul style="list-style-type: none"> <li>• dbcc fix_text</li> <li>• オブジェクト所有者 (object owner)</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• dbcc fix_text</li> <li>• オブジェクト所有者 (object owner)</li> </ul>
forget_xact	<ul style="list-style-type: none"> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• manage server</li> </ul>
indexalloc	<ul style="list-style-type: none"> <li>• dbcc indexalloc</li> <li>• テーブル所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• dbcc indexalloc</li> <li>• テーブル所有者</li> </ul>
monitor	<ul style="list-style-type: none"> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• manage server</li> </ul>
rebuild_text	<ul style="list-style-type: none"> <li>• テーブル所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• テーブル所有者</li> <li>• manage database</li> </ul>
reindex	<ul style="list-style-type: none"> <li>• dbcc reindex</li> <li>• テーブル所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• dbcc reindex</li> <li>• テーブル所有者</li> </ul>
stackused	<ul style="list-style-type: none"> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• manager server</li> </ul>
tablealloc	<ul style="list-style-type: none"> <li>• dbcc tablealloc</li> <li>• テーブル所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• dbcc tablealloc</li> <li>• テーブル所有者</li> </ul>
textalloc	<ul style="list-style-type: none"> <li>• dbcc textalloc</li> <li>• テーブル所有者</li> <li>• sa_role</li> </ul>	<ul style="list-style-type: none"> <li>• dbcc textalloc</li> <li>• テーブル所有者</li> </ul>
traceon   traceoff nodetraceon   nodetraceoff		

DBCC コマンド名	パーミッション要件	
	細密なパーミッションが無効	細密なパーミッションが有効
trace flags — 3604, 3605	<ul style="list-style-type: none"> <li>sa_role</li> </ul>	<ul style="list-style-type: none"> <li>347 = set tracing</li> <li>monitor qp performance</li> <li>set switch</li> </ul>
その他のすべてのトレース・フラグ	<ul style="list-style-type: none"> <li>sa_role</li> </ul>	<ul style="list-style-type: none"> <li>set switch</li> </ul>
tune ascinserts	<ul style="list-style-type: none"> <li>dbcc tune</li> <li>テーブル所有者</li> <li>sa_role</li> </ul>	<ul style="list-style-type: none"> <li>dbcc tune</li> <li>テーブル所有者</li> </ul>
他のすべてのパラメータに対する tune	<ul style="list-style-type: none"> <li>dbcc tune</li> <li>sa_role</li> </ul>	<ul style="list-style-type: none"> <li>dbcc tune</li> </ul>
upgrade_object	<ul style="list-style-type: none"> <li>オブジェクト所有者 (object owner)</li> <li>データベース所有者</li> <li>sa_role</li> </ul>	<ul style="list-style-type: none"> <li>manage database</li> <li>オブジェクト所有者 (object owner)</li> </ul>
set_scope_in_cluster	<ul style="list-style-type: none"> <li>sa_role</li> </ul>	<ul style="list-style-type: none"> <li>manage cluster</li> </ul>
quorum	<ul style="list-style-type: none"> <li>sa_role</li> </ul>	<ul style="list-style-type: none"> <li>manage cluster</li> </ul>

## 監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
81	dbcc	dbcc	<ul style="list-style-type: none"> <li><i>Roles</i> — 現在のアクティブな役割</li> <li><i>Keywords</i> または <i>options</i> — checkstorage などの dbcc のキーワードとそのキーワードのオプション</li> <li><i>Previous value</i> — NULL</li> <li><i>Current value</i> — NULL</li> <li><i>Other information</i> — NULL</li> <li><i>Proxy information</i> — set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [drop database](#)

システム・プロシージャ [sp\\_configure](#), [sp\\_helpdb](#)

## deallocate cursor

説明	カーソルをアクセスできない状態にし、そのカーソルにコミットされているメモリ・リソースをすべて解放します。
構文	<code>deallocate [cursor] cursor_name</code>
パラメータ	<code>cursor_name</code> 割り付けを解除するカーソルの名前です。
例	<b>例 1</b> "authors_csr" という名前のカーソルの割り付けを解除します。 <pre>deallocate cursor authors_csr</pre> <b>例 2</b> また、"authors_csr" という名前のカーソルの割り付けを解除しますが、 <code>cursor</code> を構文から省きます。 <pre>deallocate authors_csr</pre>
使用法	<ul style="list-style-type: none"><li>• アーカイブ・データベースで、<code>deallocate cursor</code> を使用できます。</li><li>• そのカーソルが存在しない場合、Adaptive Server はエラー・メッセージを返します。</li><li>• カーソルの割り付けを解除してからでないと、そのカーソル名を別の <code>declare cursor</code> 文の一部として使用できません。</li><li>• ストアド・プロシージャまたはトリガで指定しても、<code>deallocate cursor</code> はメモリ・リソースの使用状況に影響しません。</li><li>• カーソルがオープンしていてもクローズしていても、<code>deallocate</code> を実行できます。</li></ul>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>deallocate cursor</code> の使用にパーミッションは必要ありません。
参照	コマンド <a href="#">close</a> , <a href="#">declare cursor</a>

## deallocate locator

説明	メモリに格納されているラージ・オブジェクト (LOB) を削除し、その LOB ロケータを無効にします。
構文	<code>deallocate locator locator_descriptor</code>
パラメータ	<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	@v によって参照されている LOB の割り付けを解除します。 <pre>deallocate locator @v</pre>
使用法	<ul style="list-style-type: none"><li>• <code>deallocate locator</code> はトランザクション内で使用します。Adaptive Server はトランザクションの終了時に各ロケータの割り当てを自動的に解除します。</li><li>• <code>deallocate locator</code> を使用すると、メモリを節約できます。多数の LOB ロケータがトランザクション内で作成される場合は、<code>deallocate locator</code> を使用することで、不要になった LOB とロケータを個別に削除できます。</li></ul>
パーミッション	すべてのユーザが <code>deallocate locator</code> を実行できます。
参照	コマンド : <code>truncate lob</code> Transact-SQL 関数 : <code>locator_literal</code> 、 <code>locator_valid</code> 、 <code>return_lob</code> 、 <code>create_locator</code>

## declare

**説明** バッチまたはプロシージャのローカル変数の名前とデータ型を宣言します。

**構文** 変数宣言

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

変数割り当て

```
select @variable = {expression | select_statement}
        [, @variable = {expression | select_statement} ...]
        [from table_list]
        [where search_conditions]
        [group by group_by_list]
        [having search_conditions]
        [order by order_by_list]
        [compute function_list [by by_list]]
```

**パラメータ**

*@variable\_name*

@ で開始し、識別子の規則に従ってください。

*datatype*

システム・データ型またはユーザ定義データ型のいずれかです。

**例** **例 1** 2つの変数を宣言し、それら変数の値に従って文字列を出力します。

```
declare @one varchar (18), @two varchar (18)
select @one = "this is one", @two = "this is two"
if @one = "this is one"
    print "you got one"
if @two = "this is two"
    print "you got two"
else print "nope"

you got one
you got two
```

**例 2** **titles** テーブルの書籍の最高価格が 20.00 ドルを超えている場合、"Ouch!" が出力されます。

```
declare @veryhigh money
select @veryhigh = max (price)
        from titles
if @veryhigh > $20
    print "Ouch!"
```

**使用法**

- ローカル変数に値を代入するには、**select** 文を使用します。

- 1つのプロシージャ内で使用できるパラメータの最大数は 2048 です。ローカル変数またはグローバル変数の数は、使用可能なメモリによってのみ制限されます。@ 記号は変数名を表しています。
- ローカル変数は、while ループまたは if...else ブロックのカウンタとして頻繁に使用されます。ストアド・プロシージャでは、プロシージャの実行時に、非対話形式で自動的に使われるようにローカル変数を宣言します。ローカル変数は、そのローカル変数が宣言されているバッチまたはプロシージャ内で使用してください。
- 通常、値をローカル変数に割り当てる select 文は、1つの値を返します。複数の値を返す場合は、変数に最後の値が割り当てられません。変数に値を割り当てる select 文を使用して、同じ文内でデータ検索を行うことはできません。
- print と raiserror コマンドは、引数としてローカル変数を使用できます。
- ユーザはグローバル変数を作成できません。また、select 文でグローバル変数の値を直接更新することもできません。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

パーミッション

declare の使用にパーミッションは必要ありません。

参照

コマンド [print](#), [raiserror](#), [select](#), [while](#)

## declare cursor

説明	<p>select 文をカーソル名に関連付けることで、カーソルを定義します。アーカイブ・データベースで、declare cursor を使用できます。</p>
構文	<pre>declare cursor_name [semi_sensitive   insensitive] [scroll   no scroll] [release_locks_on_close] cursor for select_statement [for {read only   update [of column_name_list]}]</pre>
パラメータ	<p><b>cursor_name</b> 定義されるカーソルの名前です。</p> <p><b>select_statement</b> カーソル結果セットを定義するクエリです。詳細については、「select」を参照してください。</p> <p><b>semi_sensitive</b> カーソルとは無関係に行われるデータ変更がカーソルの結果セットに表示される可能性があることを指定します。無関係に行われたデータ変更が表示されるかどうかは、オプティマイザで選択されるクエリ・プランで決まります。プランでワーク・テーブルが作成されない場合、データ変更は結果セットに表示されます。デフォルトは、semi_sensitive です。</p> <p><b>insensitive</b> カーソルとは無関係に行われるデータ変更がカーソルの結果セットに表示されないことを指定します。この引数を指定しない場合、デフォルト値は semi_sensitive になります。insensitive を指定したカーソルは更新できません。</p> <p><b>scroll   no scroll</b> スクロール可能カーソルとして宣言するかどうかを指定します。スクロール可能カーソルを使用すると、カーソル結果セットをランダムな順序でフェッチしたり、カーソルを前後にスキャンしたりできます。スクロール可能カーソルは更新できません。</p> <p><b>release_locks_on_close</b> 各カーソルのロック解放動作を設定して、トランザクションがアクティブでもカーソルをクローズしたときに共有ロックを解放できるようにします。このオプションはどのタイプのカーソルにも適用されます。</p> <p><b>for read only</b> カーソル結果セットを更新できないように指定します。</p> <p><b>for update</b> カーソル結果セットを更新できるように指定します。</p>



of *column\_name\_list*

更新可能として定義されたカーソル結果セット (*select\_statement* によって指定) のカラムのリストです。また、Adaptive Server では、カーソルの *select\_statement* カラムのリストには指定がなくても、*select\_statement* で指定したテーブルの一部であるカラムを含める (または、結果セットから除外する) ことができます。

例

**例 1 authors** テーブルから、カリフォルニア州以外に在住しているすべての著者を含む、*authors\_crsr* カーソルの結果セットを定義します。

```
declare authors_crsr cursor
for select au_id, au_lname, au_fname
from authors
where state != 'CA'
```

**例 2 titles** テーブルから、ビジネス関連の書籍を含む、*titles\_crsr* カーソルに読み込み専用の結果セットを定義します。

```
declare titles_crsr cursor
for select title, title_id from titles
where title_id like "BU%"
for read only
```

**例 3 publishers** テーブルのすべてのローを含む、更新可能な *pubs\_crsr* カーソルの結果セットを定義します。各出版社の所在地 (*city* および *state* カラム) を更新の対象として定義します。

```
declare pubs_crsr cursor
for select pub_name, city, state
from publishers
for update of city, state
```

**例 4** カリフォルニア州の書店に関する情報を含む *stores\_scrollcrsr* の、*insensitive* なスクロール可能結果セットを定義します。

```
declare stores_scrollcrsr insensitive scroll cursor
for select stor_id, stor_name
from stores where state = 'CA'
```

**例 5** カリフォルニア州の書店に関する情報を含む *stores\_scrollcrsr* の、*insensitive* な非スクロール可能結果セットを定義します。

```
declare stores_scrollcrsr insensitive no scroll cursor
for select stor_id, stor_name
from stores where state = 'CA'
```

使用法

カーソルの制限事項

- `declare cursor` 文は、そのカーソルの `open` 文の前に置く必要があります。

- `declare cursor` を指定した他の文を、同じ Transact-SQL のバッチに記述できません。
- クライアントの `declare cursor` 文の `update` 句に組み込むことができるカラム数は、最大で 1024 です。
- `cursor_name` は Adaptive Server の有効な識別子であり、長さは 30 文字以内でなければなりません。
- `declare cursor` 文の `for update` 句に暗号化カラムを含めることはできません。
- スクロール可能カーソルは更新できません。
- `insensitive` を指定したカーソルは更新できません。

### カーソル `select` 文

- `select_statement` は、次のような制限のもとに、Transact-SQL `select` 文の構文とセマンティクスをすべて使用できます。
  - `from` 句を含めなければならない。
  - `compute`、`for browse`、または `into` 句は指定できない。
  - `holdlock` キーワードを使用できる。
- `select_statement` には、Transact-SQL パラメータ名か、言語を除くすべての種類のカーソルに対する Transact-SQL ローカル変数への参照を含めることができます。これらの名前は、`declare cursor` 文が含まれるプロシージャ、トリガ、またはバッチで定義されている Transact-SQL パラメータとローカル変数を参照する必要があります。

`declare cursor` 文内で参照されるパラメータとローカル変数には、カーソルがオープンされるまで有効な値が含まれている必要はありません。

- `select_statement` には、トリガで使用される `inserted` と `deleted` のテンポラリ・テーブルに対する参照を含めることができます。

### カーソル・スコープ

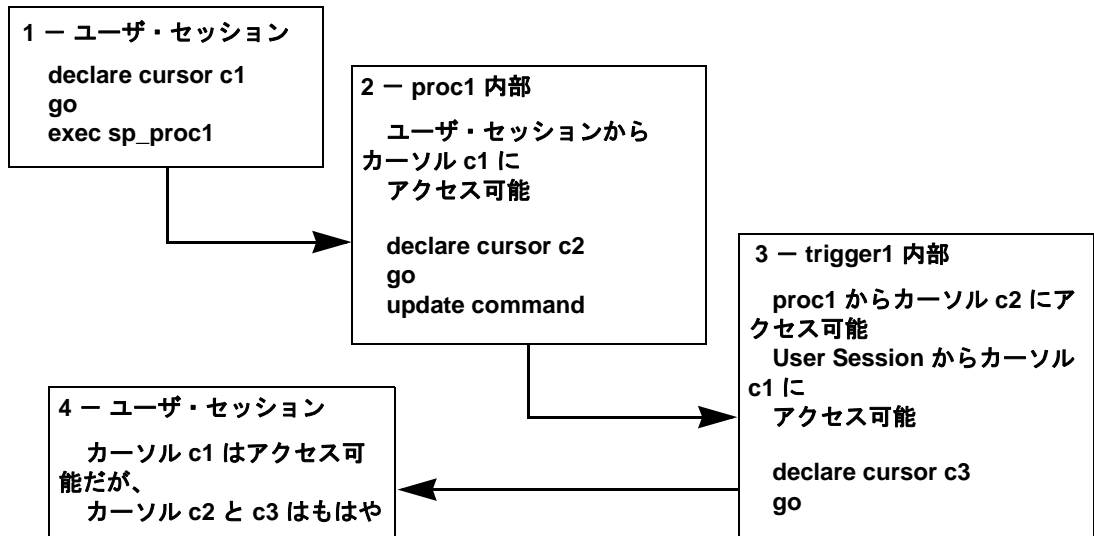
- カーソルの存在は、カーソルのスコープ (有効範囲) によって異なります。スコープとは、カーソルがユーザ・セッション内、ストアド・プロシージャ内、またはトリガ内で動作する有効範囲のことで、カーソルが使用されるコンテキスト (つまり実行状態) に適用されます。

ユーザ・セッション内では、カーソルはユーザがセッションを終了するまでの間しか存在しません。カーソルは、他のユーザによって開始された追加セッションでは存在しません。ユーザがログオフすると、Adaptive Serverはそのセッションの間に作成されたカーソルの割り付けを解除します。

`declare cursor` 文がストアード・プロシージャまたはトリガの一部である場合、その中に作成されたカーソルは、ストアード・プロシージャまたはトリガの範囲に適用され、またプロシージャやトリガを設定し格納している範囲に適用されます。`inserted` または `deleted` テーブルのトリガ内で宣言されたカーソルは、ネストされたストアード・プロシージャやトリガからアクセスできません。ただし、`inserted` テーブルや `deleted` テーブルのトリガ内で宣言されたカーソルは、トリガの範囲内ではアクセスできます。ストアード・プロシージャまたはトリガが完了すると、Adaptive Serverは内部で作成されたカーソルの割り付けを解除します。

図 1-3 は、スコープ間でどのようにカーソルが動作するのかを示します。

図 1-3: スコープ間でのカーソルの動作



- カーソル名は、指定の範囲内でユニークでなければなりません。Adaptive Server は、実行時にのみ特定の範囲内で名前の重複を検出します。ストアド・プロシージャまたはトリガは、カーソルが 1 つだけ実行されている場合に、同一名の 2 つのカーソルを定義できます。たとえば、次のようなストアド・プロシージャでは、その範囲内で定義されているカーソルが `names_crsr` 1 つだけであるため、このストアド・プロシージャは正常に動作します。

```
create procedure proc2 @flag int
as
if @flag > 0
    declare names_crsr cursor
    for select au_fname from authors
else
    declare names_crsr cursor
    for select au_lname from authors
return
```

#### 結果セット

- カーソル結果セットのローは、実際のベース・テーブルのローの値を反映しないことがあります。たとえば、`order by` 句を指定して宣言したカーソルは、通常、カーソル結果セットとしてローを並べ替えるための内部テーブルの作成を必要とします。Adaptive Server は、内部テーブルのローと一致するベース・テーブルのローをロックしません。これによって、他のクライアントはこれらのベース・テーブルのローを更新できます。この場合には、カーソル結果セットからクライアントに返されたローは、ベース・テーブルのローと一致しません。
- カーソル結果セットは、そのカーソルの `fetch` によってローが返されるときに生成されます。つまり、カーソルの `select` クエリは、通常の `select` クエリのように処理されます。この処理は**カーソル・スキャン**と呼ばれ、処理結果を受け取るまでにかかる時間 (ターンアラウンド・タイム) を短縮し、アプリケーションが要求しないローを読み込む必要をなくします。

カーソル・スキャンの制限事項の 1 つに、カーソル・スキャンが 1 つのテーブルのユニーク・インデックスしか使用できないことがあります。ただし、カーソル結果セットによって参照されるベース・テーブルが、カーソルと同じロック・スペース内の別の処理によって更新されない場合は、この制限は不要となります。Adaptive Server では、ユニーク・インデックスのないテーブルでカーソルの宣言を実行できますが、同じロック領域でそれらのテーブルを更新しようとする、そのテーブルのカーソルはすべてクローズします。

### 更新可能なカーソル

- `declare cursor` を使用してカーソルを定義した後に、Adaptive Server はカーソルが更新可能か読み取り専用かを判断します。つまり、次のようになります。
  - カーソルは更新可能 – カーソルを使用して、つまり `cursor_name` を使用してローを位置付け更新 (`position update`) または位置付け削除 (`position delete`) できる。
  - カーソルは読み込み専用 – `cursor_name` を使用して位置付けの更新 (`update`) または削除 (`delete`) を実行できない。
- `for update` 句または `for read only` 句を使用して、カーソルが更新可能か、読み込み専用かを明示的に定義します。ただし、`select_statement` が以下の構成要素のいずれかを含む場合、更新可能なカーソルは定義できません。
  - `distinct` オプション
  - `group by` 句
  - 集合関数
  - サブクエリ
  - `union` 演算子
  - `at isolation read uncommitted` 句

`for update` 句または `read only` 句をどちらかも指定しない場合は、Adaptive Server でカーソルが更新可能かどうかを確認されます。

また、`select_statement` の一部として `order by` 句を含む言語タイプまたはサーバ・タイプのカーソルを宣言した場合にのみ、Adaptive Server でカーソルが読み込み専用として定義されます。クライアント・タイプまたは実行タイプのカーソルに対しては、Adaptive Server による更新処理方法が異なるため、この制限は無視されます。

### カーソル・クローズ時のロックの解放

カーソル・スキャンが独立性レベル 1 で起こる場合、`release_locks_on_close` は何も影響を及ぼしません。

カーソルが閉じられる前にトランザクションがコミットまたはロールバックされた場合の独立性レベル 2 および 3 におけるデフォルトの動作は、その時点までにカーソルによって取得された共有ロックを Adaptive Server が解放することです。ただし、最後にフェッチされたローのロックは例外です。`release_on_locks_close` を使用する場合、カーソルによって取得された共有ロックは、カーソルが閉じられるまで存在します。

カーソルが `release_on_locks_close` パラメータを使用して宣言されたかどうかを特定するには、`sp_cursorinfo` を使用します。

```
1) sp_cursorinfo
2> go
Cursor name 'c' is declared at nesting level '0'.
The cursor is declared as NON-SCROLLABLE
RELEASE_LOCKS_ON_CLOSE cursor.
The cursor id is 917505.
The cursor has been successfully opened 0 times.
The cursor will remain open when a transaction is
committed or rolled back.
```

### 更新可能なカーソルと全ページ・ロック

- `for update` 句を使って `column_name_list` を指定しない場合、クエリ内で指定されているカラムはすべて更新可能になります。Adaptive Server は、ベース・テーブルをスキャンするときに、更新可能なカーソルのユニーク・インデックスを利用しようとしません。カーソルに関しては Adaptive Server は、`IDENTITY` カラムを含むインデックスを、たとえそれが宣言されていない場合でも、ユニークであるとみなします。
- `for update` 句が指定されていないと、Adaptive Server は任意のユニーク・インデックスを選択します。ただし、指定されたテーブル・カラムのユニーク・インデックスが存在しない場合、他のインデックスやテーブル・スキャンを使用することもできます。`for update` 句が指定されると、Adaptive Server はベース・テーブルをスキャンするための 1 つまたは複数のカラム用に定義されたユニーク・インデックスを使用します。ユニーク・インデックスが存在しないときは、エラーを返します。
- ほとんどの場合は、`for update` 句の `column_name_list` に更新対象のカラムだけを指定します。テーブルにユニーク・インデックスが 1 つしかないときは、そのカラムを `for update column_name_list` に組み込む必要はありません。カーソル・スキャンを実行するときに Adaptive Server によって検出されます。テーブルに複数のユニーク・インデックスがある場合は、そのうちの少なくとも 1 つを `for update column_name_list` から除外します。

こうすることによって、Adaptive Server でユニーク・インデックスをカーソル・スキャンに使用できるようになり、「ハロウィーン問題」と呼ばれる、更新異常を防止できます。ハロウィーン問題防止には、`unique auto_identity index` データベース・オプションを使用してテーブルを作成する方法もあります。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。

ハロウィーン問題は、ベース・テーブルから返されるローの順序を定義するカーソル結果セット・ローのカラムをクライアントが更新するときに発生します。たとえば、**Adaptive Server** がインデックスを使用してベース・テーブルにアクセスし、インデックス・キーがクライアントによって更新される場合は、更新されたインデックス・ローは、インデックス内で移動し、カーソルによって再び読み込まれます。これは、カーソル結果セットを論理的にのみ作成する更新可能なカーソルの結果です。カーソル結果セットは、実際には、そのカーソルを取り出すベース・テーブルです。

`read only` オプションを指定すると、カーソル名を使用して `delete` や `update` 文を実行し、カーソル結果セットを更新することはできません。

#### スクロール可能なカーソルの使用

- `declare cursor` を実行するときに `insensitive` または `semi_sensitive` を指定すると、変更反映の可否はデフォルトで暗黙的な設定となるため、カーソルの変更反映の可否はオプティマイザで選択されたクエリ・プランによって決まります。クエリ・プランでワーク・テーブルが作成されない場合、カーソルは `insensitive` になります。
- カーソルの変更反映の可否に `semisensitive` を指定した場合も、変更反映の可否はクエリ・プランによって決まります。
- `insensitive` を指定すると、カーソルは `read_only` になります。カーソル宣言では、`for update` 句を使用できません。
- カーソルのスクロール可能性 (`scrollability`) を指定しないと、`no scroll` が暗黙的に設定されます。
- すべてのスクロール可能カーソルは読み込み専用です。カーソル宣言では、`for update` 句を使用できません。

#### 標準規格

ANSI SQL — 準拠レベル：初級レベル

#### パーミッション

`declare cursor` の使用にパーミッションは必要ありません。

#### 参照

コマンド [open](#)

## delete

説明 テーブルからローを削除します。

構文

```
delete
    [top unsigned_integer]
    [from] [[database.]owner.]{view_name|table_name}
    [where search_conditions]
    [plan "abstract plan"]

delete [[database.]owner.]{table_name | view_name}
    [from [[database.]owner.]{view_name [readpast]]
        table_name
            [(index {index_name | table_name}
                [prefetch size][lru|mru])]
            [readpast]
    [, [[database.]owner.]{view_name [readpast]]
        table_name
            [(index {index_name | table_name}
                [prefetch size][lru|mru])]
            [readpast]} ...]
    [where search_conditions]]
    [plan "abstract plan"]

delete [from] [[database.]owner.]{table_name|view_name}
    where current of cursor_name
```

パラメータ

from (delete の後)

SQL の他のバージョンとの互換性保持に使用される、オプションのキーワードです。

view\_name | table\_name

ローの削除元のビューまたはテーブルの名前です。ビューまたはテーブルが別のデータベースの中にあるときは、データベース名を指定します。データベース内に同名のビューまたはテーブルが複数あるときは、所有者名を指定します。**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。

where

標準の **where** 句です。詳細については、[where 句](#) を参照してください。

from (table\_name または view\_name の後)

削除するローを指定するときに、**where** 句で使用する複数のテーブルまたはビューの名前を指定できます。この **from** 句を使用すると、他のテーブルに格納されたデータを基準にしてテーブルからローを削除でき、埋め込み **select** 文の機能を活用できます。



**top *unsigned\_integer***

結果セットのローの数を、*unsigned\_integer* で指定されたロー数に制限するために使用します。

**readpast**

**delete** コマンドが、互換性のないロックが保持されているページまたはローのすべてを、ロックまたはタイムアウトを待たないでスキップするように指定します。データページ・ロック・テーブルの場合、**readpast** は、互換性のないロックが保持されているページ上のすべてのローをスキップします。データロー・ロック・テーブルの場合は、互換性のないロックが保持されているすべてのローをスキップします。

**index *index\_name***

*table\_name* へのアクセスに使用するインデックスを指定します。ビューから削除するときはこのオプションを使用できません。

**prefetch size**

大容量 I/O が設定されているキャッシュにバインドされるテーブルの I/O サイズを、キロバイト単位で指定します。ビューからの削除の場合は、**prefetch** は使用できません。**sp\_helpcache** は、オブジェクトがバインドされているキャッシュまたはデフォルトのキャッシュの有効サイズを示します。

**prefetch** を使用してプリフェッチ・サイズ (*size*) を指定するとき、最小値は 2K、および 16K までの各論理ページ・サイズに基づいた 2 の累乗になります。キロバイト単位の **prefetch** サイズ・オプションは、次のとおりです。

論理ページ・サイズ	プリフェッチ・サイズ・オプション
2	2, 4, 8 16
4	4, 8, 16, 32
8	8, 16, 32, 64
16	16, 32, 64, 128

クエリで指定される `prefetch` サイズは、1つの候補にすぎません。サイズ設定を有効にするには、そのサイズでデータ・キャッシュを設定してください。データ・キャッシュを特定のサイズに設定していない場合、デフォルトの `prefetch` サイズが使用されます。

データ・キャッシュ・サイズを設定するには、`sp_cacheconfigure` を使用します。

---

**注意** コンポーネント統合サービスが有効な場合は、リモート・サーバに対して `prefetch` を使用できません。

---

#### `lru` | `mru`

テーブルに対して使用するバッファ置換方式を指定します。

**MRU/LRU** (最も最近に使用された/最も長い間使用されていない) チェーン上のキャッシュへのテーブルの読み込みを最適化するために強制的に実行させるには、`lru` を使用してください。キャッシュからバッファを廃棄し、それをテーブルの次のバッファとして置き換えるには、`mru` を使用します。ビューから削除するときはこのオプションを使用できません。

#### `plan "abstract plan"`

クエリを最適化するために使用する抽象プランを指定します。抽象プラン言語で指定された完全プランまたは部分プランを指定できます。『パフォーマンス&チューニング・シリーズ：最適化と抽象プラン』の「抽象プランの作成と使用」を参照してください。

#### `where current of cursor_name`

Adaptive Server が、`cursor_name` の現在位置によって示されるテーブルまたはビューのローを削除します。

例

**例 1** `authors` テーブルのローをすべて削除します。

```
delete authors
```

**例 2** `authors` テーブルから 1 つ以上のローを削除します。

```
delete from authors where au_lname = "McBadden"
```

**例 3** `titles` テーブルから、Bennet の著書のローを削除します。

```
delete titles
from titles, authors, titleauthor
where authors.au_lname = 'Bennet'
and authors.au_id = titleauthor.au_id
and titleauthor.title_id = titles.title_id
```

pubs2 データベースには、sales テーブルに記録されたタイトルの削除を防ぐトリガ (deltitle) があります。この例が正常に機能するように、このトリガを削除します。

**例 4** 現在 title\_crsr カーソルが示している titles テーブルから、ローを削除します。

```
delete titles where current of title_crsr
```

**例 5** IDENTITY カラムに 4 の値があるローを確認し、そのローを authors テーブルから削除します。IDENTITY カラムの実際の名前ではなく、syb\_identity キーワードが使用されていることに注意してください。

```
delete authors where syb_identity = 4
```

**例 6** authors からローを削除します。ただし、ロックされたローはスキップします。

```
delete from authors from authors readpast
where state = "CA"
```

**例 7** stores からローを削除します。ただし、ロックされたローはスキップします。authors にロックされたローがある場合、クエリはこれらのローでブロックされ、ロックが解除されるのを待ちます。

```
delete stores from stores readpast, authors
where stores.city = authors.city
```

## 使用法

- delete 文では最大 15 個のテーブルを参照できます。
- 12.5.2 より前のバージョンの Adaptive Server では、クエリで、union all 句を含むビューに対して update や delete を使用すると、ワーク・テーブルを使用せずに解析され、結果に誤りが生じることがありました。Adaptive Server 12.5.2 以降では、union all 句を含むビューに対して update や delete を使用するクエリは、必ず tempdb のワーク・テーブルを使用して解析されます。

## 制限事項

- update または insert を使用できるビューであっても、複数テーブルのビュー (from 句で複数のテーブルが指定されているもの) の場合は delete を使用できません。複数のテーブルで構成されるビューを介してローを削除すると、複数のテーブルが変更されることになるため、許可されません。ビューのベース・テーブル 1 つにしか影響しない insert 文や update 文は、使用できます。
- Adaptive Server では、delete で同一のテーブルに対して 2 つの異なる指定をした場合、2 つのテーブルとして処理されます。たとえば、次のように pubs2 で delete を発行すると、discounts は 2 つのテーブル (discounts と pubs2.discounts) として指定されます。

```
delete discounts
from pubs2..discounts, pubs2..stores
where pubs2..discounts.stor_id =
      pubs2..stores.stor_id
```

この場合、ジョインに `discounts` が含まれないため、`where` 条件はすべてのローに対して `true` で保持されます。Adaptive Server では `discounts` のすべてのローが削除されます (これは求めていた結果ではありません)。この問題を回避するには、文全体にわたって、テーブルに同じ指定を使用してください。

- 参照制約によって他のテーブルから参照されるテーブルのローを削除する場合、Adaptive Server は参照元のすべてのテーブルを検査してから、削除を許可します。削除しようとしているロー内に、参照元テーブルのいずれかによって外部キーとして使用されているプライマリ・キーがある場合は、そのローを削除することはできません。

#### テーブルからのすべてのローの削除

- `where` 句を使用しない場合、`delete [from]` の後に指定されたテーブルのローがすべて削除されます。テーブルはデータが空になっても、`drop table` コマンドが発行されるまで存在します。
- ロー指定をしない場合、`truncate table` と `delete` の機能が同じになりますが、`truncate table` のほうが高速です。`delete` は一度に 1 つずつローを削除し、そのトランザクションをログに記録します。`truncate table` はデータ・ページ全体を削除し、ローはログに記録されません。

`delete` と `truncate table` は両方とも、データとそれに対応するインデックスが占有していた領域を再利用します。

- 分割されたテーブル上では `truncate table` コマンドを使用できません。分割されたテーブルからすべてのローを削除するには、`where` 句を指定しないで `delete` コマンドを使用します。または、テーブルの分割を解除してから `truncate table` を実行します。

#### `delete` およびトランザクション

連鎖トランザクション・モードでは、現在アクティブなトランザクションがなければ、各 `delete` 文によって暗黙的に新しいトランザクションが開始されます。削除を完了するには `commit` を使用し、変更を取り消すには `rollback` を使用します。次に例を示します。

```
delete from sales where date < ' 01/01/06'
if exists (select stor_id
          from stores
          where stor_id not in
```

```
(select stor_id from sales))
rollback transaction
else
commit transaction
```

このバッチでは、(連鎖トランザクション・モードを使用して)トランザクションを開始し、**sales** テーブルから 2006 年 1 月 1 日より前の日付のローを削除します。これによって 1 つの店に関連するすべての販売エントリが削除される場合は、**sales** に対するすべての変更内容がロールバックされ、トランザクションが終了します。そうでない場合は、削除がコミットされて、トランザクションが終了します。連鎖モードの詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

#### **delete** トリガ

指定したテーブルで **delete** コマンドが発行されたときに指定の動作を実行するトリガを定義できます。

#### **delete where current of** の使用

- カーソルとともに **where current of** 句を使用します。まず **declare cursor** でカーソルを定義し、**open** 文を使用してカーソルをオープンしてから、**where current of** 句を使用してローを削除します。1 つ以上の **fetch** 文を使用して、削除するローにカーソルを配置します。カーソル名には、Transact-SQL パラメータやローカル変数を使用できません。カーソルは更新可能でなければなりません。更新可能でない場合、Adaptive Server はエラーを返します。また、カーソルの結果セットに対する削除は、カーソル・ローが取り出されたベース・テーブルのローにも影響します。カーソルを使用して削除できるのは、一度に 1 つのローだけです。
- カーソルの **select** 文にジョイン句がある場合、カーソルが更新可能でも、カーソルの結果セットのローは削除できません。**delete...where current of** で指定された **table\_name** または **view\_name** は、カーソルを定義する **select** 文の最初の **from** 句で指定されたテーブルまたはビューにします。
- カーソルの結果セットからローを削除した後、カーソルは、カーソルの結果セットの次のローの前に配置されます。次のローにアクセスするには、**fetch** を発行します。削除されたローがカーソルの結果セットの最終ローである場合、カーソルは結果セットの最終ローの後に配置されます。次に、**delete** の影響を受けるオープン・カーソルの位置とその動作を説明します。

- クライアントが (別のカーソルまたは一般的な **delete** を使用して) ローを削除し、そのローが同じクライアントが所有する他のオープン・カーソルの現在位置を示す場合は、影響が及ぶ各カーソルの位置が次に使用可能なローの前に暗黙的に置かれる。ただし、カーソルが別のクライアントの現在位置を示している場合は、ローを削除できない。
- ジョイン演算によって定義され、同一クライアントが所有する別のカーソルの現在位置を示すローをクライアントが削除する場合、**Adaptive Server** は **delete** 文を受け入れる。ただし、ジョインによって定義されたカーソルは暗黙的にクローズされる。

#### *readpast* の使用

- **readpast** を使用すると、他のタスクが保持している互換性のないロックによってブロックされることなく、データオンリーロック・テーブルに対して **delete** コマンド使用できます。
- データローロック・テーブルで **readpast** は、他のタスクによって共有ロック、更新ロック、排他ロックが保持されているすべてのローをスキップします。
- データページロック・テーブルで **readpast** は、他のタスクが共有ロック、更新ロック、排他ロックを保持しているすべてのページをスキップします。
- **readpast** を指定するコマンドは、排他テーブル・ロックがあるとブロックします。
- **readpast** オプションを全ページロック・テーブルに指定した場合、**readpast** オプションは無視されます。コマンドは、互換性のないロックを見つけるとただちにブロックします。
- セッション-ワイドな独立性レベルが 3 の場合、**readpast** オプションは暗黙的に無視されます。コマンドはレベル 3 で実行されません。コマンドは、互換性のないロックが設定されたローまたはページ上でブロックされます。
- セッションのトランザクション独立性レベルが 0 の場合、**readpast** を使用する **delete** コマンドから警告メッセージが発行されません。データページロック・テーブルでは、**readpast** が指定された **delete** によって、互換性のないロックによるロックのない全ページのすべてのローが変更されます。データロー・ロック・テーブルの場合、このコマンドは、互換性のないロックによってロックされていないすべてのローに影響します。

- 複数の `text` カラムがあるローに `delete` コマンドが適用され、`text` カラムに互換性のないロックが設定されている場合、`readpast` ロックはそのローをスキップします。

#### `index`、`prefetch`、または `lru | mru` の使用

`index`、`prefetch`、`lru | mru` の各オプションは、Adaptive Server オプティマイザが行う選択を無効にします。これらのオプションは注意して使用し、`set statistics io on` でパフォーマンスへの影響を常にチェックしてください。『パフォーマンス&チューニング・ガイド：モニタリングと分析』の「`set statistics` コマンドの使用」を参照してください。

#### 標準規格

ANSI SQL 準拠レベル：初級レベル `from` 句での複数のテーブルの使用、およびデータベース名によるテーブル名の修飾は、Transact-SQL の拡張機能です。

`readpast` は Transact-SQL の拡張機能の 1 つです。

#### パーミッション

`set ansi_permissions` を `on` に設定した場合は、`delete` 文に必要な通常のパーミッションの他に、`where` 句に表示されるすべてのカラムに対する `select` パーミッションが必要です。デフォルトでは、`ansi_permissions` は `off` です。

細密なパーミッションの設定によって異なる `delete` のパーミッション・チェックについて、次に説明します。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブルまたはビューの所有者であるか <code>delete</code> または <code>delete any table</code> の権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>delete</code> パーミッションを持っているか、テーブルの所有者であるか、 <code>sa_role</code> を持つユーザであることが必要。

#### 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
18	<code>delete</code>	テーブルからの <code>delete</code>	<ul style="list-style-type: none"> <li><code>Roles</code> — 現在のアクティブな役割</li> <li><code>Keywords</code> または <code>options</code> — <code>delete</code></li> <li><code>Previous value</code> — NULL</li> <li><code>Current value</code> — NULL</li> <li><code>Other information</code> — NULL</li> <li><code>Proxy information</code> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
19	delete	ビューからの delete	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – delete</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照

コマンド [create trigger](#), [drop table](#), [drop trigger](#), [truncate table](#), [where 句](#)



## delete statistics

説明	<code>sysstatistics</code> システム・テーブルから統計情報を削除します。
構文	<pre>delete [shared] statistics <i>table_name</i>       [<i>partition data_partition_name</i>]       [(<i>column_name</i>[, <i>column_name</i>] ...)]</pre>
パラメータ	<p><code>shared</code></p> <p>master データベースの <code>sysstatistics</code> から、シミュレートした統計情報を削除します。</p> <p><i>table_name</i></p> <p>テーブル内にあるすべてのカラムに対する統計を削除します。</p> <p><i>data_partition_name</i></p> <p>データ・パーティションのすべての統計を削除します。グローバル統計は削除されません。</p> <p><i>column_name</i></p> <p>指定したカラムに対する統計を削除します。</p>
例	<p><b>例 1</b> <code>titles</code> テーブルのすべてのカラムに対する密度、選択性、ヒストグラムを削除します。</p> <pre>delete statistics titles</pre> <p><b>例 2</b> <code>titles</code> テーブルの <code>pub_id</code> カラムに対する密度、選択性、ヒストグラムを削除します。</p> <pre>delete statistics titles (pub_id)</pre> <p><b>例 3</b> <code>titles</code> テーブルの <code>smallsales</code> パーティションに対する密度、選択性、ヒストグラムを削除します。</p> <pre>delete statistics titles partition smallsales</pre> <p><b>例 4</b> シングルカラムの <code>pub_id</code> またはシングルカラムの <code>pubdate</code> に関する統計に影響を与えることなく、<code>pub_id</code>、<code>pubdate</code> の密度、選択性、ヒストグラムを削除します。</p> <pre>delete statistics titles (pub_id, pubdate)</pre> <p><b>例 5</b> カラム <code>pub_id</code> およびデータ・パーティション <code>smallsales</code> の密度、選択性、ヒストグラムを削除します。</p> <pre>delete statistics titles partition smallsales (pub_id)</pre>
使用法	<ul style="list-style-type: none"> <li>• <code>delete statistics</code> は <code>sysabstats</code> テーブルの統計情報には影響しません。</li> <li>• <code>delete statistics</code> をデータ・パーティションに対して実行しても、グローバル統計は削除されません。</li> </ul>

- **drop table** コマンドを発行すると、**sysstatistics** 内の対応するローが削除されます。**drop index** を使用すると、**sysstatistics** 内のローは削除されません。このコマンドによって、クエリ・オブティマイザは、テーブルのインデックスを保守するオーバーヘッドを発生させずに、インデックス統計を引き続き使用できます。

---

**警告！** 密度、選択性、ヒストグラムは、クエリの完全な最適化に不可欠のものです。**delete statistics** コマンドは、オブティマイザが使用しない統計を削除するツールとして提供されています。クエリの最適化に必要な統計情報を誤って削除した場合は、テーブル、インデックス、カラムに対して **update statistics** を実行します。

---

- シミュレートした統計を **optdiag** ユーティリティ・コマンドを使用してロードすると、**master.sysstatistics** テーブルに少数のローが追加されます。シミュレートした統計が現在使用されていない場合は、**delete shared statistic** コマンドを使用して、**master.sysstatistics** の情報を削除します。

#### 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

#### パーミッション

**delete statistics** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であるか <b>delete statistics</b> パーミッションを持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者、 <b>sa_role</b> が付与されたユーザ、 <b>delete statistics</b> パーミッションを持つユーザのいずれかであることが必要。 <b>delete statistics</b> パーミッションは、テーブル所有者またはシステム所有者が任意のユーザに付与または譲渡できます。

#### 参照

コマンド **create index**, **grant**, **revoke**, **update**

ユーティリティ **optdiag**

## disk init

**説明** 物理デバイスまたはファイルを Adaptive Server で使用できるようにします。

**構文**

```
disk init
    name = "device_name",
    physname = { 'physical_name' | 'cache_name' }
    skip_alloc={true | false},
    [vdevno = virtual_device_number,]
    size = number_of_blocks
    [, type = 'inmemory' ]
    [, vstart = virtual_address
        , cntrltype = controller_number]
    [, dsync = {true | false}]
    [, directio = {true | false}]
    [, instance = "instance_name"]
```

**パラメータ**

*device\_name*  
データベースのデバイス名またはファイル名です。この名前は識別子の規則に従い、一重引用符または二重引用符で囲んでください。この名前は、[create database](#) コマンドと [alter database](#) コマンドで使用されます。

*physical\_name*  
データベース・デバイスまたはデータベース・キャッシュのフル・パス名です。この名前は、一重引用符または二重引用符で囲んでください。物理デバイス・パスが相対パスである場合、`disk init` は警告を返します。

*cache\_name*  
ディスクを作成するキャッシュの名前です。

*inmemory*  
メモリ内デバイスを作成することを示します。

*skip\_alloc*  
`disk init` コマンドのブール値パラメータです。このパラメータは、Windows 以外のファイル・システムおよび Windows ロー・システムで作成されたデバイスでサポートされています。`skip_alloc` を `true` に設定すると、ページは 0 初期化されません。`skip_alloc` のデフォルト設定は `false` です。

**vdevno**

仮想デバイス番号です。Adaptive Server 関連のデータベース・デバイス内でユニークでなければなりません。デバイス番号 0 は、マスタ・デバイスに予約されています。それ以外の有効なデバイス番号は 1 ~ 2,147,483,647 です。

仮想デバイス番号を確認するには、sp\_helpdevice レポートの device\_number カラムを調べ、次の未使用の整数を使用してください。

**size**

新しいデバイスに割り付ける領域の量です。指定できる単位は、'k' または "K" (キロバイト)、"m" または 'M' (メガバイト)、"g" または "G" (ギガバイト)、および 't' または 'T' (テラバイト) です。これらには大文字と小文字の区別はなく、一重引用符と二重引用符のどちらも使用できます。常に単位を指定することをおすすめします。単位を含めない場合は、引用符を省略することができます。ただし、単位を付加する場合は、必ず引用符を付けてください。受け入れられる値は次のとおりです。

- 5120 = 10MB
- "5120" = 10MB
- "10M" = 10MB

**vstart**

開始仮想アドレス、または開始オフセットで、Adaptive Server の場合はデータベース・デバイスを使用して開始します。指定できる単位は、'k' または "K" (キロバイト)、"m" または 'M' (メガバイト)、"g" または "G" (ギガバイト)、および 't' または 'T' (テラバイト) です。これらには大文字と小文字の区別はなく、一重引用符と二重引用符のどちらも使用できます。常に単位を指定することをおすすめします。単位を含めない場合は、引用符を省略することができます。ただし、単位を付加する場合は、必ず引用符を付けてください。

オフセットのサイズは、vstart の値の入力状態によって異なります。

- 単位を指定しなかった場合、vstart は 2K ページをその開始アドレスとして使用します。たとえば、vstart = 13 と指定すると、Adaptive Server では 13 \* 2K ページが開始アドレスのオフセットとして使用されます。
- 単位を指定すると、vstart はそれを開始アドレスとして使用します。たとえば、vstart = "13M" と指定すると、Adaptive Server では 13MB で開始アドレスのオフセットが設定されます。

`vstart` のデフォルト値 ( および通常は優先値 ) は 0 です。指定したデバイスで、`vstart + size` の合計ブロックが利用できない場合は、`disk init` コマンドは失敗します。AIX オペレーティング・システム上で論理ボリューム・マネージャを実行している場合は、`vstart` の値を 2 にしてください。Sybase 製品の保守契約を結んでいるサポート・センタの指示があった場合のみ、`vstart` を指定してください。

#### `cntrltype`

ディスク・コントローラを指定します。デフォルト値は 0 です。Sybase 製品の保守契約を結んでいるサポート・センタの指示があった場合のみ、`cntrltype` を再設定してください。

#### `dsync`

データベース・デバイスへの書き込みが記憶メディアにフラッシュされるか、UNIX オペレーティング・システムのファイルを使用するときのみバッファに入れられるかを指定します。このオプションはオペレーティング・システムのファイルを初期化するときだけに意味があります。ロー・パーティションのデバイスの初期化に影響はありません。デフォルトでは、オペレーティング・システムのファイルはすべて、`true` に設定されている `dsync` によって初期化されます。

#### `directio`

オペレーティング・システムのバッファ・キャッシュを回避して、データをディスクに直接転送できます。`directio` は、静的パラメータであるため、このパラメータを有効にするには、Adaptive Server を再起動する必要があります。

`directio` は、デフォルトで、ノンクラスタード Adaptive Server では `false` に設定され、クラスタード Adaptive Server では `true` に設定されます。

ロー・デバイスの場合、`directio` パラメータは無視されます。

#### `instance = "instance_name"`

( クラスタのみ ) デバイスをプライベートとして指定し、所有インスタンスを `instance_name` に設定します。

#### 例

##### 例 1 ページを 0 初期化しません。

```
disk init name="d2",
physname="/usr/sybase/devices/d3.dat",
skip_alloc="true",
size="10G"
```

`skip_alloc` を `true` に設定した場合、Adaptive Server は、ディスクを初期化するとき領域を割り当てません。

**例 2** UNIX システムでディスクの 10MB を初期化します。

```
disk init
name = "user_disk",
physname = "/dev/rxyla",
vdevno = 2, size = 5120
```

**例 3** UNIX オペレーティング・システムのファイルでディスクの 10MB を初期化します。Adaptive Server は、dsync の設定によってデバイスのファイルを開きます。このファイルへの書き込みは、記憶メディアに対して直接行われます。

```
disk init
name = "user_file",
physname = "/usr/u/sybase/data/userfile1.dat",
vdevno = 2, size = 5120, dsync = true
```

**例 4** directio を使用してデータをディスクに直接書き込む "user\_disk" という名前のデバイスを作成します。

```
disk init
name = "user_disk",
physname = "/usr/u/sybase/data/userfile1.dat",
size = 5120,
directio= true
```

**例 5** inmemory\_dev という名前のデバイスを作成します。

```
disk init name = inmemory_dev,
physname = 'imdb_cache',
size = '3G',
type = 'inmemory'
```

## 使用法

- ブロック・デバイスは、HP-UX, Windows および Linux のプラットフォームのみでデータベース・デバイスとして使用することをおすすめします。Sybase がおすすめしないプラットフォーム上でブロック・デバイスを作成しようとすると、disk init と disk reinit で、警告メッセージが表示されます。
- NT 以外のファイル・システムと NT ロー・システム上でクラッシュを短時間で復旧させるには、skip\_alloc を使用します。また、skip\_alloc を directio 機能とともに使用すると、デバイスの作成時間が短縮され、更新の持続性が向上します。使用可能な領域の程度にかかわらず、skip\_alloc は必ず、今後必要となる領域を Adaptive Server に確保するように要求する警告メッセージを出力します。
- マスタ・デバイスは、インストール・プログラムによって初期化されています。disk init を使用してマスタ・デバイスを初期化する必要はありません。

- `disk init` で作成されたデバイスには、パーミッションに制限があります。
- ディスクの初期化を正常完了するには、"sybase" ユーザに初期化されるデバイスに対する適切なオペレーティング・システム・パーミッションが必要とされます。
- `size` を float データ型として指定できますが、このサイズは 2K の最も近い倍数に丸められます。
- サイズに単位を指定しない場合、`disk init` は 2K の仮想ページ・サイズを使用します。
- `disk init` を使用して初期化できるディスク区分の最小サイズは、次のいずれか大きいほうになります。
  - 1 メガバイト
  - サーバの論理ページ・サイズの 1 アロケーション・ユニット
- `directio` と `dsync` は互いに排他的です。デバイスの `dsync` が `true` に設定されている場合、そのデバイスの `directio` を `true` に設定することはできません。デバイスの `directio` を有効にするには、`dsync` の設定を `false` に変更しておく必要があります。
- `directio` は、すべてのプラットフォームで使用できるわけではありません。サポートされていないプラットフォームで、`directio` パラメータを使用して `disk init` を発行すると、Adaptive Server は、`No such parameter: 'directio'` というメッセージを出力します。
- 新しいデータベース・デバイスのそれぞれに `disk init` を使用します。`disk init` が発行されるたびに、`master.sysdevices` にローが 1 つ追加されます。新しいデータベース・デバイスが、デフォルトのデータベース記憶領域のプールの一部に自動的になることはありません。データベース・デバイスにデフォルト・ステータスを割り当てるには、`sp_diskdefault` を使用します。
- `disk init` の使用後は、必ず `dump database` または `dump transaction` コマンドを使用して `master` データベースのバックアップをとります。このようにすれば、`master` が損傷した場合でも、簡単かつ確実にリカバリできます。( `disk init` でデバイスを追加し、`master` のバックアップをとれなかった場合は、まず `disk reinit` を適用してから、Adaptive Server をいったん停止して、再起動すると、変更内容をリカバリできる場合もあります)。
- `create database` または `alter database` コマンドで `on` 句を使用して、ユーザ・データベースをデータベース・デバイスに割り当てます。

- データベースのトランザクション・ログ (システム・テーブル `syslogs`) を、残りのデータベースの部分が格納されているデバイスとは異なるデバイスに置く場合には、`create database` に `log on` 拡張機能を使用する方法をおすすめします。この他に、データベースの作成時に最低 2 つのデバイスの名前を指定してから、`sp_logdevice` を実行する方法もあります。また、`alter database` を使用して別のデバイスにデータベースを拡張してから、`sp_logdevice` を実行する方法もあります。`log on` 拡張を実行すると、すぐにログ全体が別のデバイスに移動されます。`sp_logdevice` を使用する方法では、トランザクションの動作によって移行が完了するまで、システム・ログの一部が元のデータベース・デバイスにそのまま保持されます。
- システム上のすべての `Adaptive Server` デバイス (データベース・デバイスとダンプ・デバイスの両方) のレポートを取得するには、`sp_helpdevice` を実行します。
- データベース・デバイスは、`sp_dropdevice` を使用して削除します。最初にそのデバイス上の既存のデータベースをすべて削除しなければなりません。

#### `dsync` の使用

**注意** 重要なデータを格納しているデバイスでは、`dsync` を `false` に設定しないでください。唯一の例外は `tempdb` で、`dsync` が `false` に設定されているデバイスでも安全に格納できます。

- (UNIX のみ) ロー・デバイスでは以下の実行ができません。
  - `directio` または `dsync` の `true` 設定
  - `sp_deviceattr` ストアド・プロシージャによる、`directio` または `dsync` の `true` 設定

**注意** HPUX の場合、`dsync` オプションのみが適用されます。

これを行うと、次のようなメッセージが返されます。

```
You cannot set option dsync for raw device 'dev/raw/raw235'
```

または

```
You cannot set attribute dsync for raw device 'myrawdsk1'
```

- `dsync` がオンの場合、データベース・デバイスへの書き込みは、物理記憶メディアに対して行われます。また、システム障害が発生した場合は、`Adaptive Server` によってデバイス上のデータがリカバリされます。



- `dsync` がオフの場合、データベース・デバイスへの書き込みは UNIX ファイル・システムによってバッファされることがあります。UNIX ファイル・システムは、物理メディアがまだ修正されていない場合でも、更新が完了したものとしてマークを付けることがあります。このため、システム障害の発生時に、データの更新が物理メディア上で行われる保証がなくなり、Adaptive Server がデータベースをリカバリできないことがあります。
- `dsync` は、マスタ・デバイス・ファイルに対して常にオンに設定されます。
- `dsync` の値は、システム障害後にデバイス上のデータベースをリカバリする必要がない場合にのみオフにします。たとえば、`tempdb` データベースのみを格納するデバイスでは、`dsync` をオフに設定することもできます。
- Adaptive Server では、ロー・パーティションに格納されたデバイスの `dsync` 設定は無視されます。`dsync` の設定に関係なく、これらのデバイスへの書き込みは物理記憶メディアに対して行われます。
- `master` データベースに損傷がある場合、または `master` の最終ダンプ以降にデバイスが追加されている場合は、`disk reinit` によって `master.sysdevices` が正しいことが確認されます。

#### インメモリ・データベースおよびリラックス持続性データベースのデバイスの作成

- メモリ内デバイスの論理名は、デバイスが作成されるキャッシュと同じ名前にはできません。
- メモリ内デバイスは、割り当てられる最初のインメモリ・データベースのために予約されます。
- 物理デバイス名をメモリ内デバイスの論理名として使用する必要があります。
- メモリ内デバイスの論理名は、デバイス全体でユニークであることが必要です。
- 次の処理はできません。
  - メモリ内デバイスの作成時の、次のパラメータの使用。`vstart`、`cntrltype`、`dsync`、`directio`、および `skip_alloc`。
  - メモリ内デバイスを作成するキャッシュより大きな容量のメモリ内デバイスを作成する。
  - デフォルトのデータ・キャッシュを含む通常の名前付きキャッシュからメモリ内デバイスを作成する。

- いったん作成したインメモリ・デバイスのサイズの拡大。ただし、`sp_cacheconfig` を使用すると、インメモリ・キャッシュのサイズを増やし、`disk init` を使用して新しいインメモリ・デバイスを作成できます。
- インメモリ・デバイスに対する、次のコマンドの使用。`disk resize`、`disk mirror`、`disk remirror`、`disk unmirror`、`disk refit`、および `disk reinit`。
- ディスク・デバイス上での `sp_deviceattr` および `sp_diskdefault` システム・プロシージャの使用。

標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

パーミッション

`disk init` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。`disk init` を使用するには、`master` データベースを使用している必要があります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage disk</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> が付与されたユーザであることが必要。 <code>disk init</code> パーミッションは譲渡できません。

監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
20	disk	disk init	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — <code>disk init</code></li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — ディスクの名前</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照

コマンド [alter database](#), [create database](#), [disk refit](#), [disk reinit](#), [dump database](#), [dump transaction](#), [load database](#), [load transaction](#)

システム・プロシージャ [sp\\_diskdefault](#), [sp\\_dropdevice](#), [sp\\_helpdevice](#), [sp\\_logdevice](#)

## disk mirror

**説明** プライマリ・デバイスの障害発生時に、ただちに処理を引き継ぐソフトウェア・ミラーを作成します。

**構文**

```
disk mirror
    name = "device_name",
    mirror = "physicalname"
    [, writes = {serial | noserial}]
    [clear = {TRUE | FALSE}]
```

### パラメータ

#### name

ミラーリングするデータベース・デバイスの名前です。これは `sysdevices` テーブルの `name` カラムに記録されます。この名前は、一重または二重引用符で囲んでください。

#### mirror

セカンダリ・デバイスとなるデータベース・ミラー・デバイスのフル・パス名です。この名前は、一重引用符または二重引用符で囲んでください。セカンダリ・デバイスがファイルである場合、`physicalname` は、Adaptive Server が作成するファイルを明確に識別できるパス指定にしてください。`physicalname` の値に既存のファイル名は使用できません。

#### writes

デバイスへの逐次書き込みを実行するかどうかを選択できます。デフォルト (`serial`) では、プライマリ・データベース・デバイスへの書き込みが終了してから、セカンダリ・デバイスへの書き込みが開始されるように保証されています。プライマリおよびセカンダリ・デバイスが別々の物理デバイスにある場合は、停電の場合も、逐次書き込みにより、少なくとも 1 つのディスクが影響を受けないように保証されます。一般的に、`serial` の方が、`noserial` よりも書き込みは遅くなります。

#### clear

ミラー デバイスを 0 初期化し、基本となるファイルシステムによるミラー・デバイスのスペース予約を保証します。デフォルト値 `FALSE` はミラーをクリアしません。デバイスへの書き込みは、ファイル・システムの領域不足のために失敗する可能性があります。`TRUE` を指定すると、ミラーがクリアされ、ファイル・システムがデバイスのための領域の予約を強制します。

例 **例 1** tranlog がロー・デバイスの論理デバイス名です。tranlog デバイスは、disk init によって初期化されており、トランザクション・ログ・デバイスとして使用されています (create database...log on tranlog)。次のコマンドは、このトランザクション・ログ・デバイスのミラーリングを行います。

```
disk mirror
  name = "tranlog",
  mirror = "/dev/rxyle"
```

**例 2** データベース・デバイス user\_disk のソフトウェア・ミラーを mirror.dat ファイルに作成します。

```
disk mirror
  name = "user_disk",
  mirror = "/server/data/mirror.dat"
```

## 使用法

- ディスク・ミラーリングによって、ユーザ・データベース・デバイス、マスタ・データベース・デバイス、またはユーザ・データベースのトランザクション・ログに使用されるデータベース・デバイスのソフトウェア・ミラーが作成されます。データベース・デバイスに障害が発生した場合、そのミラーがただちに処理を引き継ぎます。

ディスク・ミラーリングは、データベースで進行中のアクティビティを妨げることはありません。Adaptive Server を停止しなくても、データベース・デバイスのミラーリング、またはミラーリングの解除ができます。

- disk mirror の使用後は、必ず dump database コマンドを使用して、master データベースのバックアップをとります。このようにすれば、master が損傷した場合でも、簡単かつ確実にリカバリできます。
- ミラーリングされたデバイスの読み込みまたは書き込みを正しく実行できなかった場合は、Adaptive Server が不良デバイスのミラーリングを自動的に解除し、エラー・メッセージを出力します。Adaptive Server は、ミラーリングが解除された状態で処理を継続します。ミラーリングを再開するには、システム管理者が [disk remirror](#) コマンドを使用する必要があります。
- NT プラットフォームで使用する場合、このコマンドの clear オプションは無効です。
- マスタ・デバイスをミラーリングする他に、データやトランザクション・ログを格納するデバイスもミラーリングできます。ただし、ダンプ・デバイスをミラーリングすることはできません。

- ミラーリングされるのはデバイスであり、データベースではありません。
- デバイス 1 つとそのミラーによって、1 つの論理デバイスが構成されます。Adaptive Server は、`sysdevices` テーブルの `mirrorname` カラムに、ミラー・デバイスの物理名を格納します。ミラー・デバイスの物理名は、`sysdevices` に独立したエントリが必要なく、`disk init` を使用して初期化する必要もありません。
- 非同期 I/O を続けて使用するには、常に非同期 I/O 機能があるデバイスを、非同期 I/O 機能がある他のデバイスにミラーリングします。ほとんどの場合、これはロー・デバイス間のミラーリング、およびオペレーティング・システム・ファイル間のミラーリングを意味します。

ファイルに対する非同期 I/O を実行できないオペレーティング・システムの場合は、ロー・デバイスを通常のファイルにミラーリングするとエラー・メッセージが表示されます。通常のファイルをロー・デバイスにミラーリングすることはできますが、その場合、非同期 I/O は使用されません。

- 非同期 I/O をサポートするシステムでは、`writes` オプションを使用すると、最初のデバイスへの書き込みが終了してから 2 番目のデバイスへの書き込みを始めるかどうかを指定することができます (`serial` による逐次書き込み)。または、両方の I/O 要求を、両側のミラーリング・デバイスに 1 つずつ直接キューイングさせるかどうかを指定することができます (`noserial` による非逐次書き込み)。どちらの場合も、書き込みを完了できないときは I/O エラーが発生し、不良デバイスのミラーリングが解除されます。
- `create database` または `alter database` コマンドがデフォルト・リスト内のデータベース・デバイスに影響を及ぼす場合でも保護できるように、すべてのデフォルト・データベース・デバイスのミラーを作成します。
- 保護をより強化するには、トランザクション・ログに使用されるデータベース・デバイスのミラーを作成します。
- ユーザ・データベースのトランザクション・ログは、常に別のデータベース・デバイスに配置してください。データベースのトランザクション・ログ (`syslogs` システム・テーブル) を、残りのデータベースの部分が格納されているデバイス以外のデバイスに配置するには、データベースの作成時にデータベース・デバイスとログ・デバイスを指定します。また、`alter database` を使用して別のデバイスにデータベースを拡張してから、`sp_logdevice` を実行する方法もあります。

- **master** データベースのデータベース・デバイスのミラーを作成する場合、**dataserver** ユーティリティ・プログラムを使用して **Adaptive Server** を再起動するときに、**-r** オプションと **UNIX** 用のミラー名を使用できます。そのサーバの **RUN\_servername** ファイルにこのオプションとミラー名を追加して、**startserver** ユーティリティ・プログラムで認識できるようにします。たとえば、マスタ・デバイス **master.dat** と、そのミラー **mirror.dat** を起動するには、次のように入力します。

```
dataserver -dmaster.dat -rmirror.dat
```

『ユーティリティ・ガイド』の「**dataserver**」と「**startserver**」を参照してください。

- 割り付けられていない領域 (追加の **create database** と **alter database** 文がそのデバイスの一部を割り付けるための空き領域) があるデータベース・デバイスのミラーを作成する場合、**disk mirror** コマンドが発行されたときではなく、割り付けが行われたときに、**disk mirror** によってミラーリングが開始されます。
- 使用しているシステム上のすべての **Adaptive Server** デバイス (ユーザ・データベース・デバイスとそのミラー、およびダンプ・デバイス) についてのレポートを取得するには、**sp\_helpdevice** を実行します。

## 標準規格

ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

## パーミッション

**disk mirror** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。**disk mirror** を使用するには、**master** データベースを使用している必要があります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <b>manage disk</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <b>sa_role</b> が付与されたユーザであることが必要。 <b>disk init</b> パーミッションは譲渡できません。

## 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オブション	監査対象のコマンドまたはアクセス	extrainfo の情報
23	disk	disk mirror	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – disk mirror</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – ディスクの名前</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter database](#), [create database](#), [disk init](#), [disk refit](#), [disk reinit](#), [disk remirror](#), [disk unmirror](#), [dump database](#), [dump transaction](#), [load database](#), [load transaction](#)

システム・プロシージャ [sp\\_diskdefault](#), [sp\\_helpdevice](#), [sp\\_logdevice](#)

ユーティリティ [dataserver](#)、[startserver](#)

## disk refit

**説明** sysdevices に格納されている情報に基づいて、master データベースの sysusages と sysdatabases システム・テーブルを再構築します。

**構文** disk refit

**使用方法**

- disk refit によってシステム・テーブルが再構築された後は、Adaptive Server が自動的に停止します。
- [disk reinit](#) の後に master データベースをリストアする手順の一部として disk refit を使用します。

---

**注意** disk refit を実行する前に、トレース・フラグ 3608 を使用して Adaptive Server を起動する必要があります。ただし、トレース・フラグを使用して Adaptive Server を起動するときは、事前に『トラブルシューティング&エラー・メッセージ・ガイド』の情報を理解しておく必要があります。

---

**標準規格** ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

**パーミッション** disk refit のパーミッション・チェックは、細密なパーミッションの設定によって異なります。disk refit を使用するには、master データベースを使用している必要があります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、manage disk 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、sa_role が付与されたユーザであることが必要。 disk refit パーミッションは譲渡できません。

**監査** sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
21	disk	disk refit	<ul style="list-style-type: none"> <li>• Roles — 現在のアクティブな役割</li> <li>• Keywords または options — disk refit</li> <li>• Previous value — NULL</li> <li>• Current value — NULL</li> <li>• Other information — ディスクの名前</li> <li>• Proxy information — set proxy が有効な場合は元のログイン名</li> </ul>

**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 [disk init](#), [disk reinit](#)

システム・プロシージャ [sp\\_addumpdevice](#), [sp\\_helpdevice](#)



## disk reinit

説明	master データベースの <code>sysdevices</code> システム・テーブルを再構築します。 <code>disk reinit</code> は、master データベースをリストアする手順の一部として使用します。
構文	<pre> disk reinit     name = "device_name",     physname = "physicalname" ,     [vdevno = virtual_device_number ,]     size = number_of_blocks     [, vstart = virtual_address       , cntrltype = controller_number]     [, dsync = {true   false}]     [, directio = {true   false}]     [, instance = "instance_name"] </pre>
パラメータ	<p><b>name</b> データベース・デバイスの名前です。この名前は識別子の規則に従い、一重引用符または二重引用符で囲んでください。この名前は、<a href="#">create database</a> コマンドと <a href="#">alter database</a> コマンドで使用されます。</p> <p><b>physname</b> データベース・デバイスの名前です。物理名は、一重引用符または二重引用符で囲んでください。</p> <p><b>vdevno</b> 仮想デバイス番号です。Adaptive Server 関連のデータベース・デバイス内でユニークでなければなりません。デバイス番号 0 は、マスタ・デバイスに予約されています。それ以外の有効なデバイス番号は 1 ~ 2,147,483,647 です。</p> <p>仮想デバイス番号を確認するには、<code>sp_helpdevice</code> レポートの <code>device_number</code> カラムを調べ、次の未使用の整数を使用してください。</p> <p><b>size</b> 再初期化するデバイスの現在のサイズです。指定できる単位は、'k' または "K" (キロバイト)、"m" または 'M' (メガバイト)、"g" または "G" (ギガバイト)、および 't' または 'T' (テラバイト) です。これらには大文字と小文字の区別はなく、一重引用符と二重引用符のどちらも使用できます。常に単位を指定することをおすすめします。単位を含めない場合は、引用符を省略することができます。ただし、単位を付加する場合は、必ず引用符を付けてください。</p>

**vstart**

開始仮想アドレス、または開始オフセットで、Adaptive Server の場合はデータベース・デバイスを使用して開始します。指定できる単位は、'k' または "K" (キロバイト)、"m" または 'M' (メガバイト)、"g" または "G" (ギガバイト)、および 't' または 'T' (テラバイト) です。これらには大文字と小文字の区別はなく、一重引用符と二重引用符のどちらも使用できます。常に単位を指定することをおすすめします。単位を含めない場合は、引用符を省略することができます。ただし、単位を付加する場合は、必ず引用符を付けてください。size の単位を指定しなかった場合、値はメガバイト単位であると見なされます。オフセットのサイズは、vstart の値の入力状態によって異なります。

- 単位を指定しなかった場合、vstart は 2K ページをその開始アドレスとして使用します。たとえば、vstart = 13 と指定すると、Adaptive Server では 13 \* 2K ページが開始アドレスのオフセットとして使用されます。
- 単位を指定すると、vstart はそれを開始アドレスとして使用します。たとえば、vstart = "13M" と指定すると、Adaptive Server では 13MB で開始アドレスのオフセットが設定されます。

vstart のデフォルト値 (および通常は優先値) は 0 です。指定したデバイスで、vstart + size の合計ブロックが利用できない場合は、disk reinit は失敗します。

---

**注意** AIX オペレーティング・システム上で論理ボリューム・マネージャを実行している場合は、vstart の値を 2 にしてください。

---

Sybase 製品の保守契約を結んでいるサポート・センタの指示があった場合のみ、vstart を指定してください。

**cntrltype**

ディスク・コントローラを指定します。デフォルト値は 0 です。Sybase 製品の保守契約を結んでいるサポート・センタの指示があった場合のみ、このオプションを再設定してください。

**dsync**

データベース・デバイスへの書き込みが記憶メディアにフラッシュされるか、UNIX オペレーティング・システムのファイルを使用するときのみバッファに入れられるかを指定します。このオプションはオペレーティング・システムのファイルを初期化するときだけに意味があります。ロー・パーティションのデバイスの初期化に影響はありません。デフォルトでは、オペレーティング・システムのファイルはすべて、**true** に設定されている **dsync** によって初期化されます。

**directio**

オペレーティング・システムのバッファ・キャッシュを回避して、データをディスクに直接転送できます。**directio** は、静的パラメータであるため、このパラメータを有効にするには、**Adaptive Server** を再起動する必要があります。

**directio** は、デフォルトで、ノンクラスタード **Adaptive Server** では **false** に設定され、クラスタード **Adaptive Server** では **true** に設定されます。

ロー・デバイスの場合、**directio** パラメータは無視されます。

**instance = "instance\_name"**

(クラスタのみ) デバイスをプライベートとして指定し、所有インスタンスを **instance\_name** に設定します。

## 例

**例 1** **sysdevices** テーブルに新しいローを追加します。この新しいローには、再初期化する既存のデバイスの特性が含まれています。

```
disk reinit
name = "user_file",
physname = "/usr/u/sybase/data/userfile1.dat",
vdevno = 2, size = 5120, dsync = true
```

**例 2** 新しいローを **sysdevices** テーブルに追加し、データをディスクに直接転送します。この新しいローには、再初期化する既存のデバイスの特性が含まれています。

```
disk reinit
name = "user_disk",
physname = "/usr/u/sybase/data/userfile1.dat",
size = 5120, directio= true
```

## 使用方法

- ブロック・デバイスは、**HP-UX**、**Windows** および **Linux** のプラットフォームのみでデータベース・デバイスとして使用することをおすすめします。**Sybase** がおすすめしないプラットフォーム上でブロック・デバイスを作成しようとすると、**disk init** と **disk reinit** では、警告メッセージが表示されます。

- `master` データベースがダメージを受けているか、または最後に `master` をダンプしてからデバイスが追加された場合、`disk reinit` は `master.sysdevices` が正しいことを確認します。
- `disk reinit` は `disk init` と似ていますが、データベース・デバイスの初期化は行いません。
- `size` を `float` データ型として指定できますが、このサイズは 2K の最も近い倍数に丸められます。
- `size` に単位を使用しない場合、`disk reinit` は 2K の仮想ページ・サイズを使用します。
- `directio` オプションのデフォルト値は、すべてのプラットフォームで `false` (オフ) に設定されます。
- `master` データベースのリストア方法の詳細については、『システム管理ガイド』を参照してください。

#### `dsync` の使用

---

**注意** 重要なデータを格納しているデバイスでは、`dsync` を `false` に設定しないでください。唯一の例外は `tempdb` で、`dsync` が `false` に設定されているデバイスでも安全に格納できます。

---

- `dsync` がオンの場合、データベース・デバイスへの書き込みは、物理記憶メディアに対して行われることが保証されます。また、システム障害が発生した場合は、Adaptive Server によってデバイス上のデータがリカバリされます。
- `directio` と `dsync` は互いに排他的です。デバイスの `dsync` を "true" に設定した場合、そのデバイスの `directio` を "true" に設定することはできません。デバイスの `directio` を有効にするには、`dsync` の設定を "false" に変更しておく必要があります。
- `dsync` がオフの場合、データベース・デバイスへの書き込みは UNIX ファイル・システムによってバッファされることがあります。UNIX ファイル・システムは、物理メディアがまだ修正されていない場合でも、更新が完了したものとしてマークを付けることがあります。このため、システム障害の発生時に、データの更新が物理メディア上で行われる保証がなくなり、Adaptive Server がデータベースをリカバリできないことがあります。
- `dsync` は、マスタ・デバイス・ファイルに対して常にオンに設定されます。

- `dsync` の値は、システム障害後にデバイス上のデータベースをリカバリする必要がない場合にのみオフにします。たとえば、`tempdb` データベースのみを格納するデバイスでは、`dsync` をオフに設定することもできます。
- `Adaptive Server` では、ロー・パーティションに格納されたデバイスの `dsync` 設定は無視されます。`dsync` の設定に関係なく、これらのデバイスへの書き込みは物理記憶メディアに対して行われます。
- `dsync` の設定は、`Windows NT` プラットフォームでは使用されません。
- `master` データベースがダメージを受けている場合、または最後に `master` をダンプしてからデバイスが追加されている場合、`disk reinit` は `master.sysdevices` が正しいことを確認します。

標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

パーミッション

`disk reinit` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。`disk reinit` を使用するには、`master` データベースを使用している必要があります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage disk</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> が付与されたユーザであることが必要。 <code>disk reinit</code> パーミッションは譲渡できません。

監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
22	disk	disk reinit	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — <code>disk reinit</code></li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — ディスクの名前</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照

コマンド [alter database](#), [create database](#), [dbcc](#), [disk init](#), [disk refit](#)  
 システム・プロシージャ [sp\\_addumpdevice](#), [sp\\_helpdevice](#)

## disk remirror

説明	ディスク・ミラーリングが、ミラーリングされたデバイスの障害により停止したか、または <code>disk unmirror</code> コマンドによって一時的に無効になった後に、ディスク・ミラーリングを再開します。
構文	<pre>disk remirror     name = "device_name"</pre>
パラメータ	<p><b>name</b></p> <p>再ミラーリングするデータベース・デバイスの名前です。この名前は <code>sysdevices</code> テーブルの <code>name</code> カラムに記録されます。一重引用符か二重引用符で名前を囲む必要があります。</p>
例	<p>データベース・デバイス <code>user_disk</code> のソフトウェア・ミラーリングを再開します。</p> <pre>disk remirror     name = "user_disk"</pre>
使用法	<ul style="list-style-type: none"><li>ディスク・ミラーリングによって、ユーザ・データベース・デバイス、マスタ・データベース・デバイス、またはユーザ・データベースのトランザクション・ログに使用されるデータベース・デバイスのソフトウェア・ミラーが作成されます。データベース・デバイスに障害が発生した場合、そのミラーがただちに処理を引き継ぎます。<p>ミラーリングされたデバイスの障害によりミラーリングが一時的に停止した後、または <code>disk unmirror</code> コマンドの <code>mode = retain</code> オプションによって一時的に無効化された後は、<code>disk remirror</code> コマンドを使用してミラーリングを回復します。<code>disk remirror</code> コマンドは、保持されたディスクのデータをミラーにコピーします。</p></li><li><code>disk remirror</code> の使用後は、必ず <code>dump database</code> コマンドを使用して、<code>master</code> データベースのバックアップをとります。このようにすれば、<code>master</code> が損傷した場合でも、簡単かつ確実にリカバリできます。</li><li><code>mode = remove</code> オプションによってミラーリングが永続的に無効になっている場合は、ミラーを含むオペレーティング・システム・ファイルを削除してから <code>disk remirror</code> を使用してください。</li><li>ミラーリングされるのはデータベース・デバイスであり、データベースではありません。</li><li><code>Adaptive Server</code> を停止しなくてもデータベース・デバイスのミラーリング、再ミラーリング、またはミラーリング解除を実行できます。ディスク・ミラーリングは、データベースで進行中のアクティビティを妨げることはありません。</li></ul>

- ミラーリングされたデバイスの読み込みまたは書き込みを正常に完了できなかった場合は、Adaptive Server が、不良デバイスのミラーリングを解除し、エラー・メッセージを出力します。Adaptive Server は、ミラーリングが解除された状態で処理を継続します。ミラーリングを再開するには、システム管理者が `disk remirror` を使用します。
- ユーザ・データベース・デバイスをミラーリングするだけでなく、ユーザ・データベースのトランザクション・ログを常に別のデータベース・デバイスに保管するようにしてください。トランザクション・ログに使用されるデータベース・デバイスも、さらに保護を強化するためにミラーリングできます。データベースのトランザクション・ログ (syslogs システム・テーブル) を、データベースの残りの部分が格納されるデバイスとは別のデバイスに配置するには、データベースの作成時にデータベース・デバイスとログ・デバイスを指定します。また、別のデバイスへの `alter database` を適用してから、`sp_logdevice` を実行することもできます。
- master データベースのデータベース・デバイスのミラーを作成する場合、`dataserver` ユーティリティ・プログラムを使用して Adaptive Server を再起動するときに、`-r` オプションと UNIX 用のミラー名を使用できます。そのサーバの `RUN_servername` ファイルにこのオプションを追加すると、`startserver` ユーティリティ・プログラムで認識されます。たとえば、次のコマンドでは、マスタ・デバイス `master.dat` と、そのミラー `mirror.dat` が起動されます。

```
dataserver -dmaster.dat -rmirror.dat
```

『ユーティリティ・ガイド』の「`dataserver`」と「`startserver`」を参照してください。

- 使用しているシステム上のすべての Adaptive Server デバイス (ユーザ・データベース・デバイスとそのミラー、およびダンプ・デバイス) についてのレポートを取得するには、`sp_helpdevice` を実行します。

## 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

## パーミッション

`disk remirror` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。`disk remirror` を使用するには、master データベースを使用している必要があります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage disk</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> が付与されたユーザであることが必要。 <code>disk remirror</code> パーミッションは譲渡できません。

## 監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
25	disk	disk remirror	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – disk remirror</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – ディスクの名前</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter database](#), [create database](#), [disk init](#), [disk mirror](#), [disk refit](#), [disk reinit](#), [disk unmirror](#), [dump database](#), [dump transaction](#), [load database](#), [load transaction](#)

システム・プロシージャ [sp\\_diskdefault](#), [sp\\_helpdevice](#), [sp\\_logdevice](#)

ユーティリティ [dataserver](#)、[startserver](#)



## disk resize

**説明** Adaptive Server が使用するデバイスのサイズを動的に拡大します。

**構文**

```
disk resize
    name = "device_name" ,
    size = additional_space
```

**パラメータ** *name*  
サイズを拡大するデバイスの名前です。

*additional\_space*  
デバイスに追加する領域のサイズです。

**例** testdev のサイズを 4MB 増やすには、次のように入力します。

```
disk resize
name = "test_dev" ,
size = "4M"
```

**使用法**

- `disk resize` コマンドを使用すると、ディスクのサイズを動的に増やすことができます。

- デバイスのサイズを変更したら、`sysdevices` テーブルでデバイスのサイズを管理するマスタ・デバイスをダンプします。マスタ・デバイスの古いダンプからリカバリしても、`sysdevices` テーブルに格納される情報は最新のものになりません。

- デバイ스에設定されたプロパティは、サイズを拡大した後も設定は解除されずに残ります。

- ディスクの物理的な初期化中にディスク領域が不足してエラーが発生すると、`disk resize` コマンドはエラー発生前の時点のサイズにデータベース・デバイスを拡張します。

たとえば、4K の論理ページを使用するサーバでデバイスのサイズを 40MB 増やそうとしたときに、空きディスク領域が 39.5MB しかない場合、デバイスは 39.5MB 分だけ拡張されます。拡張サイズ (39.5MB) のうち、Adaptive Server が使用するのは 39MB だけです。4K のサーバは最低 1MB 単位でデバイスを設定するため、残りの 0.5MB は割り付けられますが使用はされません。

残りの 0.5MB を利用するには、デバイスに少なくともあと 1.5MB の使用可能な領域があることを確認してから、インクリメンタル・サイズとして 1.5MB を指定して、`disk resize` コマンドを再実行します。

- `disk resize` を使用して、デバイスのサイズの縮小はできません。
- *device\_name* には、有効な識別子が必要です。デバイスは `disk init` コマンドで初期化し、`ダンプ・デバイス` や `ロード・デバイス` ではなく、有効な Adaptive Server デバイスを参照している必要があります。

- 指定できる単位は、'k' または "K" (キロバイト)、"m" または 'M' (メガバイト)、"g" または "G" (ギガバイト)、および 't' または 'T' (テラバイト) です。これらには大文字と小文字の区別はなく、一重引用符と二重引用符のどちらも使用できます。常に単位を指定することをおすすめします。単位の使用は任意ですが、実際に割り付けられるページ数と混同しないように、**disk resize** コマンドを使用するときには必ず単位を入力するようにおすすめします。

単位指定子は、一重引用符または二重引用符で囲んでください。単位を指定しないと、デフォルトでサイズはディスク・ページの数に設定されます。

- サイズ変更操作の実行中はミラーリングを永続的に無効にします。サイズ変更操作が完了したら、ミラーリングを回復できます。

#### 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

#### パーミッション

**disk resize** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <b>manage disk</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <b>sa_role</b> が付与されたユーザであることが必要。

#### 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
100	disk	disk resize	<ul style="list-style-type: none"> <li><b>Roles</b> — 現在のアクティブな役割</li> <li><b>Keywords</b> または <b>options</b> — NULL</li> <li><b>Previous value</b> — NULL</li> <li><b>Current value</b> — NULL</li> <li><b>Other information</b> — インデックス名</li> <li><b>Proxy information</b> — <b>set proxy</b> が有効な場合は元のログイン名</li> </ul>

#### 参照

コマンド **create database**、**disk init**、**drop database**、**load database**  
 システム・プロシージャ **sp\_addsegment**、**sp\_dropsegment**、**sp\_helpdb**、**sp\_helpsegment**、**sp\_logdevice**、**sp\_renamedb**、**sp\_spaceused**

## disk unmirror

説明	<code>disk mirror</code> コマンドによって開始されたディスク・ミラーリングを中断して、ハードウェアのメンテナンス、またはハードウェア・デバイスの変更を可能にします。
構文	<pre>disk unmirror     name = "device_name"     [, side = {"primary"   secondary}]     [, mode = {retain   remove}]</pre>
パラメータ	<p><b>name</b> ミラーリングを解除するデータベース・デバイスの名前です。この名前は、一重または二重引用符で囲んでください。</p> <p><b>side</b> <code>primary</code> デバイスまたは <code>secondary</code> デバイス (ミラー) を無効にするかどうかを指定します。デフォルトでは、セカンダリ・デバイスのミラーリングが解除されます。</p> <p><b>mode</b> ミラーリングの解除が一時的 (<code>retain</code>) か、または永続的 (<code>remove</code>) かを決定します。デフォルトでは、ミラーリングの解除は一時的です。</p> <p>後で同じ設定でデータベース・デバイスを再度ミラーリングする場合は、<code>retain</code> を指定します。このオプションは、プライマリ・デバイスに障害が発生した場合の動作と同じように機能します。</p> <ul style="list-style-type: none"> <li>• I/O は、ミラーリングが解除されていないデバイスだけを対象とします。</li> <li>• <code>sysdevices</code> の <code>status</code> カラムは、ミラーリングが無効化されていることを示します。<code>remove</code> は、ミラー・デバイスに対する <code>sysdevices</code> のリファレンスすべてを削除します。</li> <li>• <code>status</code> カラムは、ミラーリング機能が無視されることを示します。</li> <li>• プライマリ・デバイスが停止される場合は、<code>phyname</code> カラムの内容は、<code>mirrorname</code> カラム内のセカンダリ・デバイスの名前で置き換えられます。</li> <li>• <code>mirrorname</code> カラムは <code>null</code> に設定されます。</li> </ul>
例	<p><b>例 1</b> <code>user_disk</code> データベース・デバイスのソフトウェア・ミラーリングを中断します。</p> <pre>disk unmirror     name = "user_disk"</pre>

**例 2** セカンダリ側の `user_disk` データベース・デバイスのソフトウェア・ミラーリングを中断します。

```
disk unmirror name = "user_disk", side = secondary
```

**例 3** `user_disk` データベース・デバイスのソフトウェア・ミラーリングを中断し、ミラー・デバイスのすべてのデバイス・リファレンスを削除します。

```
disk unmirror name = "user_disk", mode = remove
```

## 使用法

- ディスク・ミラーリングによって、ユーザ・データベース・デバイス、マスタ・データベース・デバイス、またはユーザ・データベースのトランザクション・ログに使用されるデータベース・デバイスのソフトウェア・ミラーが作成されます。データベース・デバイスに障害が発生した場合、そのミラーがただちに処理を引き継ぎます。

`disk unmirror` は、オリジナルのデータベース・デバイスまたはそのミラーのいずれかを、一時的にあるいは永続的に無効化します。その結果、**Adaptive Server** でそのデバイスを読み込みや書き込みを使用できなくなります。このオプションによって、関連したファイルがオペレーティング・システムから削除されることはありません。

- ディスクのミラーリングを解除することによって、`master` データベースの `sysdevices` テーブルが変更されます。`disk unmirror` の使用後は、必ず `dump database` コマンドを使用して、`master` データベースのバックアップをとります。このようにすれば、`master` が損傷した場合でも、簡単かつ確実にリカバリできます。
- データベース・デバイスが使用中でも、ミラーリングを解除することができます。
- `dump database`、`load database`、または `load transaction` の実行中は、いずれのデータベース・デバイスもミラーリングを解除できません。**Adaptive Server** は、ダンプまたはロードをアボートするか、あるいはダンプまたはロードが完了するまで `disk unmirror` を遅延させるかどうかを尋ねるメッセージを表示します。

- `dump transaction` の実行中は、データベースのログ・デバイスのミラーリングを解除できません。Adaptive Server は、ダンプをアポートするか、またはダンプが完了するまで `disk unmirror` を遅延させるかどうかを尋ねるメッセージを表示します。

---

**注意** `dump transaction` ログ・デバイスのミラーリングが解除されていても、`with truncate_only` と `dump transaction with no_log` は影響を受けません。

---

- `create` または `alter database` コマンドがデフォルト・リスト内のデータベース・デバイスに影響を及ぼす場合でも保護できるように、すべてのデフォルト・データベース・デバイスのミラーを作成します。
- ミラーリングされたデバイスの読み込みまたは書き込みを正しく実行できなかった場合は、Adaptive Server が不良デバイスのミラーリングを自動的に解除し、エラー・メッセージを出力します。Adaptive Server は、ミラーリングが解除された状態で処理を継続します。ミラーリングは、システム管理者が `disk remirror` コマンドを使用して再起動してください。
- システム上のすべての Adaptive Server デバイス (ユーザ・データベース・デバイス、そのミラー、およびダンプ・デバイス) のレポートを取得するには、`sp_helpdevice` を実行します。
- `disk unmirror` コマンドの `mode = retain` オプションによって一時的にミラーリングが停止された後にミラーリングを回復するには、`disk remirror` を使用します。`mode = remove` オプションによってミラーリングが永続的に無効になっている場合は、ミラーを含むオペレーティング・システム・ファイルを削除してから、`disk remirror` を使用します。

#### 標準規格

ANSI SQL – 準拠レベル : Transact-SQL 拡張機能

#### パーミッション

`disk unmirror` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。`disk unmirror` を使用するには、`master` データベースを使用している必要があります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage disk</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> が付与されたユーザであることが必要。 <code>disk unmirror</code> パーミッションは譲渡できません。

#### 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
24	disk	disk unmirror	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – disk unmirror</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – ディスクの名前</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter database](#), [create database](#), [disk init](#), [disk mirror](#), [disk refit](#), [disk reinit](#), [disk remirror](#), [dump database](#), [dump transaction](#), [load database](#), [load transaction](#)

システム・プロシージャ [sp\\_diskdefault](#), [sp\\_helpdevice](#), [sp\\_logdevice](#)

ユーティリティ [dataserver](#)、[startserver](#)

## drop database

説明	アーカイブ・データベースを含む 1 つ以上のデータベースを Adaptive Server から削除します。
構文	<code>drop database database_name [, database_name] ...</code>
パラメータ	<p><code>database_name</code>          削除するデータベースの名前です。データベースのリストを取得するには、<code>sp_helpdb</code> を使用します。</p>
例	<p><b>例 1</b> <code>publishing</code> データベースとその内容すべてを削除します。</p> <pre>drop database publishing</pre> <p><b>例 2</b> <code>key_db</code> は暗号化キーを含むデータベース、<code>col_db</code> は暗号化カラムを含むデータベースです。この例では、Adaptive Server によってエラーが生成され、<code>key_db</code> の削除が失敗します。<code>col_db</code> の削除は成功します。両方のデータベースを削除するには、次のように <code>col_db</code> を最初に削除します。</p> <pre>drop database col_db, key_db</pre>
使用法	<ul style="list-style-type: none"> <li>• アーカイブ・データベースを削除すると、そのデータベースのすべてのローが、スクラッチ・データベースの <code>sysaltusages</code> テーブルから削除されます。これにより、スクラッチ・データベースにログ領域が必要になります。</li> <li>• データベースを削除すると、データベースとそのすべてのオブジェクトが削除され、データベースの記憶領域の割り付けが解放されます。さらに <code>master</code> データベースにある <code>sysdatabases</code> と <code>sysusages</code> の各システム・テーブルからデータベースのエントリが消去されます。</li> <li>• <code>drop database</code> は、削除されたデータベースに付属した <code>suspect</code> (疑わしい) ページのエントリを <code>master..sysattributes</code> からクリアします。</li> </ul> <p><b>暗号化カラムと <code>drop database</code></b></p> <p>データベースに他のデータベースのカラムの暗号化に現在使用されているキーが含まれる場合、キーが誤って失われないようにするために、<code>drop database</code> が失敗します。データベースを削除するには、次のようにします。</p> <ul style="list-style-type: none"> <li>• カラムを復号化するか、<code>alter table</code> を使用して異なるキーによる暗号化のためにカラムを変更します。</li> <li>• 暗号化カラムを含んでいるテーブルまたはデータベースを削除します。</li> </ul>

## 制限事項

- データベースを削除するには、**master** データベースを使用している必要があります。
- (ユーザの誰かが読み込みまたは書き込みのためにオープンして) 使用中であるデータベースは削除できません。
- **drop database** を、他のデータベース内のテーブルが参照しているデータベースの削除に使用することはできません。どのテーブルと外部データベースが、現在のデータベース内のプライマリ・キー・テーブルで外部キー制約を持っているかを判断するには、次のクエリを実行します。

```
select object_name (tableid), frgndbname
      from sysreferences
      where frgndbname is not null
```

**alter table** を使用して、これらのデータベース間の制約を削除してから、**drop database** コマンドを再発行します。

- **drop database** は、損傷したデータベースの削除に使用できます。データベースがダメージを受けたために **drop database** が実行されない場合は、**dbcc dbrepair** を使用して、データベースを修復します。

```
dbcc dbrepair (database_name, dropdb)
```

- 監査機能が有効になっているときは、**sybsecurity** データベースを削除できません。監査機能が無効の場合、**sybsecurity** を削除できるのは、システム・セキュリティ担当者だけです。

## 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

## パーミッション

**drop database** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、データベース所有者であるか、データベースに対する <b>own database</b> 権限が必要。 <b>sybsecurity</b> を削除するには、データベース所有者であるか、 <b>manage auditing</b> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、 <b>sa_role</b> が付与されたユーザであることが必要。また、 <b>sybsecurity</b> については、 <b>sso_role</b> が付与されていることが必要。

## 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。



イベント	監査オブション	監査対象のコマンドまたはアクセス	extrainfo の情報
26	drop	drop database	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter database](#), [create database](#), [dbcc](#), [use](#)

プロシージャ [sp\\_changedbowner](#), [sp\\_helpdb](#), [sp\\_renamedb](#), [sp\\_spaceused](#)

## drop default

説明	ユーザ定義のデフォルトを削除します。
構文	<code>drop default [owner.]default_name [, [owner.]default_name] ...</code>
パラメータ	<p><i>default_name</i></p> <p>既存のデフォルトの名前です。既存のデフォルトのリストを表示するには、<code>sp_help</code> を実行します。現在のデータベースで別のユーザが所有している同じ名前のデフォルトを削除する場合は、所有者名を指定します。<i>owner</i> のデフォルト値は現在のユーザです。</p>
例	<p><code>datedefault</code> というユーザ定義のデフォルトをデータベースから削除します。</p> <pre>drop default datedefault</pre>
使用法	<ul style="list-style-type: none"> <li>カラムまたはユーザ定義のデータ型に現在バインドされているデフォルトは、削除できません。<code>sp_unbindefault</code> を使用して、デフォルトのバインドを解除してから削除します。</li> <li>新しいデフォルトは、現在のデフォルトのバインドを解除しなくても、カラムまたはユーザ定義のデータ型にバインドできます。新しいデフォルトは、古いデフォルトを上書きします。</li> <li>NULL カラムのデフォルトを削除すると、NULL がカラムのデフォルト値になります。NOT null カラムのデフォルトを削除した場合、データ挿入時にユーザが明示的に値をそのカラムに入力しないと、エラー・メッセージが表示されます。</li> </ul>
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>drop default</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、デフォルト所有者であるか <code>drop any default</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、デフォルト所有者であるか <code>sa_role</code> が付与されたユーザであることが必要。
	<code>drop default</code> のパーミッションは、デフォルトの所有者に対してデフォルトで設定され、譲渡はできません。
監査	<code>sysaudits</code> の <code>event</code> カラムと <code>extrainfo</code> カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンド またはアクセス	extrainfo の情報
31	drop	drop default	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [create default](#)システム・プロシージャ [sp\\_help](#), [sp\\_helptext](#), [sp\\_unbindefault](#)

## drop encryption key

説明	キー所有者は名前付き暗号キーを削除できます。
構文	<pre>drop encryption key [[database.][owner].]keyname</pre> <p>外部ログイン・パスワードのサービス・キーを明示的に削除する構文は次のとおりです。</p> <pre>drop encryption key syb_extpasswdkey with password encryption downgrade</pre> <p>非表示のサービス・キーを明示的に削除する構文は次のとおりです。</p> <pre>drop encryption key syb_syscommkey_dddddd</pre> <p>または</p> <pre>drop encryption key syb_syscommkey with text encryption downgrade</pre>
パラメータ	<p><i>database</i> データベースの名前を指定します。</p> <p><i>owner</i> 所有者です。</p> <p><i>keyname</i> キーの名前です。</p> <p><i>syb_extpasswdkey</i> サービス・キーの名前です。</p> <p><i>with password encryption downgrade</i> を指定すると、15.7 より前のバージョンで使用されているアルゴリズムで外部ログイン・パスワードが再設定され、Replication Agent のパスワード、およびコンポーネント統合サービスと RTMS の外部ログイン・パスワードが無効な値に再設定されます。</p> <p>キーが削除された後で、管理者は手動でパスワードを再入力することにより、対応するサービスを再開する必要があります。</p> <p><i>syb_syscommkey_dddddd</i> 削除する個々の syscomments サービス・キーの明示的な名前です。</p> <p><i>syb_syscommkey with text encryption downgrade</i> Adaptive Server は、15.7 より前のバージョンで使用されているアルゴリズムを使用して、syscomments 内のすべての非表示テキストを再暗号化します。</p>
例	<p>暗号化キー <code>cc_key</code> を削除します。</p> <pre>drop encryption key cust.dbo.cc_key</pre>

- 使用法**
- キーがキー・コピーを持っている場合、コピーはベース・キーとともに削除されます。
  - いずれかのデータベースのいずれかのカラムが、削除するキーで暗号化されている場合、コマンドは失敗します。
  - **drop encryption key** では、アーカイブされたデータベース、**suspect** のマークが付けられたデータベース、オフラインのデータベース、リカバリされていないデータベース、またはキーによって暗号化されているカラムに現在ロードされているデータベースはチェックできません。このコマンドによって、使用不可のデータベースの名前を示した警告メッセージが発行されますが、コマンド自体は正常に実行されます。該当するデータベースがオンラインになると、削除されたキーによって暗号化されていたカラムを持つすべてのテーブルが使用不可になります。このキーをリストアするには、システム管理者が、キーが削除された時間より前の時点の、削除されたキーのデータベース・ダンプをロードする必要があります。

**パーミッション** **drop encryption key** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、キー所有者か <b>manage any encryption key</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、キー所有者であるか <b>sa_role</b> が付与されたユーザであることが必要。

**監査** **sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
109		drop encryption key	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — <b>set proxy</b> が有効な場合は元のログイン名</li> </ul>

**参照** [create encryption key](#), [alter encryption key](#), [sp\\_encryption](#), [sp\\_help](#)

## drop function

説明	1 つ以上のユーザ定義関数を現在のデータベースから削除します。
構文	<code>drop function{ [ owner_name .]function_name } [ ,...n ]</code>
パラメータ	<p><i>owner_name</i> ユーザ定義関数を所有するユーザ ID の名前です。既存のユーザ ID にする必要があります。</p> <p><i>] function_name</i> 削除されるユーザ定義関数の名前です。所有者名の指定はオプションですが、サーバ名とデータベース名は指定できません。</p>
例	この例では、 <code>bonus</code> 関数が削除されます。  <code>drop function bonus</code>
使用法	<code>drop function</code> は、現在のデータベースから SQL のユーザ定義関数を削除します。
パーミッション	<code>drop function</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、関数所有者であるか <code>drop any function</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、関数所有者であるか <code>sa_role</code> を持つユーザであることが必要。 パーミッションは譲渡できません。

## drop function (SQLJ)

説明	SQLJ 関数を削除します。
構文	drop func[tion] [owner.]function_name [, [owner.]function_name ] ...
パラメータ	[owner.]function_name SQLJ 関数の SQL 名です。
例	SQLJ 関数の square_root を削除します。 <pre>drop function square_root</pre>
使用法	drop function は、現在のデータベースからユーザ作成の関数だけを削除します。システム関数は削除しません。
パーミッション	drop function (SQLJ) のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、関数所有者であるか drop any function 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、関数所有者であるか sa_role を持つユーザであることが必要。 パーミッションは譲渡できません。

監査 sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
98	drop	drop function	<ul style="list-style-type: none"> <li>• Roles – 現在のアクティブな役割</li> <li>• Keywords または options – NULL</li> <li>• Previous value – NULL</li> <li>• Current value – NULL</li> <li>• Other information – NULL</li> <li>• Proxy information – set proxy が有効な場合は元のログイン名</li> </ul>

参照 マニュアル SQLJ 関数の詳細については、『Adaptive Server Enterprise における Java』を参照してください。

コマンド [create function \(SQLJ\)](#)

## drop index

説明	現在のデータベースのテーブルからインデックスを削除します。
構文	<pre>drop index <i>table_name.index_name</i> [, <i>table_name.index_name</i>] ...</pre>
パラメータ	<p><i>table_name</i> インデックス・カラムが配置されているテーブルです。テーブルは現在のデータベースになければなりません。</p> <p><i>index_name</i> 削除するインデックスです。Transact-SQL では、インデックス名をテーブル内でユニークとしますが、データベース内でユニークである必要はありません。</p>
例	<p>authors テーブルから au_id_ind を削除します。</p> <pre>drop index authors.au_id_ind</pre>
使用法	<ul style="list-style-type: none"> <li>• drop index が発行されると、インデックスが前に占有していたすべての領域を回復できます。この領域は、どのデータベース・オブジェクトにも使用できます。</li> <li>• システム・テーブルでは drop index を使用できません。</li> <li>• drop index では、一意性制約をサポートしているインデックスを削除できません。このようなインデックスを削除するには、alter table を使用して制約を削除するか、テーブルを削除します。インデックスの一意性制約の詳細については、「create table」を参照してください。</li> <li>• オープンしているカーソルが現在使用しているインデックスを削除することはできません。オープンしているカーソル、およびカーソルが使用しているインデックスの情報を取得するには、sp_cursorinfo を使用します。</li> <li>• テーブルに存在するインデックスの情報を取得するには、次のように入力します。この中の objname はテーブル名を示します。  <pre>sp_helpindex objname</pre> </li> </ul>
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	drop index のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者であることが必要。



監査 sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
105		drop index	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

参照 コマンド [create index](#), [setuser](#)

システム・プロシージャ [sp\\_cursorinfo](#), [sp\\_helpindex](#), [sp\\_spaceused](#)

## drop login

説明	1つまたは複数のログイン・アカウントを削除します。
構文	<code>drop login login_name [, login_name_list] [ with override ]</code>
パラメータ	<code>login_name</code> 削除するログイン・アカウントの名前を指定します。 <code>login_name_list</code> 削除するログイン・アカウントのリストを指定します。 <code>with override</code> ログイン参照をチェックできない使用不可のデータベースがある場合でも、ログインを削除します。
例	ログイン・アカウント ravi および vinod を削除します。 <pre>drop login ravi, vinod</pre>
使用法	<ul style="list-style-type: none"><li>• <code>drop login</code> を実行すると、ユーザのログインが Adaptive Server から消去され、そのユーザのエントリが <code>master.dbo.syslogins</code> から削除されます。</li><li>• Adaptive Server は、削除されたログインのサーバ・ユーザ ID を再使用しますが、アカウントの信頼性はなくなります。アカウントの完全な削除を避けるには、代わりに <code>sp_locklogin</code> を実行して、使用されなくなるアカウントをロックしてください。</li><li>• ログインを削除しなければならない場合は、必ず各ログインのイベントを監査し (<code>sp_audit</code> を使用)、各ログインの記録を取ってください。</li><li>• <code>drop login</code> は、削除されたログインに対応するすべてのリソース制限を削除します。</li><li>• 削除されるログインが、サーバ上のデータベース・ユーザである場合、<code>drop login</code> は実行できません。データベースからユーザを削除するには、<code>sp_dropuser</code> を使用してください。そのユーザがデータベース内でオブジェクトを所有している場合は、データベースからユーザを削除することはできません。</li><li>• 削除されるログインがシステム・セキュリティ担当者の場合、<code>drop login</code> は、ロックされていないシステム・セキュリティ担当者のアカウントが、他に少なくとも1つ存在していることを確認します。存在しない場合は、<code>drop login</code> は実行できません。同様に <code>drop login</code> は、ロックされていないシステム管理者のアカウントが少なくとも1つは常に存在していることを確認します。</li></ul>

標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	drop login のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、manage any login 権限を持つユーザーであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、sso_role が付与されたユーザーであることが必要。

**監査** sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
139	login_admin	drop login	キーワードの内容は次のとおりです。 loginname1[, ...[, loginnameN ]]

**参照** コマンド create login profile、create login、drop login profile、alter login、alter login profile

マニュアル ログイン・アカウントの削除の詳細については、『セキュリティ管理ガイド』を参照してください。

関数 lprofile\_id、lprofile\_name

システム・プロシージャ sp\_passwordpolicy、sp\_displaylogin、sp\_displayroles、sp\_locklogin

## drop login profile

説明	1つまたは複数のログイン・プロファイルを削除します。
構文	<pre>drop login profile <i>login_profile_name</i> [, <i>login_profile_name_list</i>] [ with override ]</pre>
パラメータ	<p><code>login_profile_name</code> 削除するログイン・プロファイルの名前を指定します。</p> <p><code>login_profile_name_list</code> 削除するログイン・プロファイルのリストを指定します。</p> <p><code>with override</code> ログイン・アカウントにバインドされているログイン・プロファイルを強制的に削除します。削除したログイン・プロファイルと関連付けられているログイン・アカウントは、デフォルトのログイン・プロファイルと関連付けられます。</p>
例	<p><b>例 1</b> ログイン・プロファイル <code>group1_login_profile</code> がログイン・アカウントにバインドされていない場合に、このログイン・プロファイルを削除します。</p> <pre>drop login profile group1_login_profile</pre> <p><b>例 2</b> <code>sa_login_profile</code> が 1 つ以上のログイン・アカウントにバインドされているため、エラーが生成されます。</p> <pre>drop login profile sa_login_profile</pre> <pre>Msg 11193, Level 16, State 1: Line 1: The specified login profile is the default login profile and/or associated with one or more login accounts. Remove the default property and/or associations or use the WITH OVERRIDE clause to drop the login profile.</pre> <p><b>例 3</b> <code>sa_login_profile</code> は、1 つ以上のログイン・アカウントにバインドされている場合でも削除されます。</p> <pre>drop login profile sa_login_profile with override</pre>
使用法	<ul style="list-style-type: none"><li>• コマンド <code>drop login profile</code> は、ログイン・アカウントにバインドされていないログイン・プロファイルを削除します。</li></ul>

- ログイン・アカウントにバインドされているログイン・プロファイルを強制的に削除するには、`drop login profile with override` を使用します。削除するログイン・プロファイルがログイン・アカウントにバインドされている場合、このログイン・アカウントはデフォルトのログイン・アカウントにバインドされます（デフォルトのログイン・アカウントが存在する場合）。デフォルトのログイン・プロファイルも存在しない場合は、Adaptive Server バージョン 15.7 より前の優先度の規則がログイン・アカウントに適用されます。

**標準規格**

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

**パーミッション**

`drop login profile` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage any login profile</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sso_role</code> が付与されたユーザであることが必要。

**監査**

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
141	<code>security_profile</code>	<code>drop login profile</code>	キーワードの内容は次のとおりです。 <code>login_profile_name [WITH OVERRIDE]</code>

**参照**

**コマンド** `create login profile`、`create login`、`alter login profile`、`alter login`、`drop login`

**マニュアル** ログイン・プロファイルの削除の詳細については、『セキュリティ管理ガイド』を参照してください。

**関数** `lprofile_id`、`lprofile_name`

**システム・プロシージャ** `sp_passwordpolicy`、`sp_displaylogin`、`sp_displayroles`、`sp_locklogin`

## drop precomputed result set

説明	事前計算済み結果セットを削除します。
構文	<pre>drop {precomputed result set   materialized view}       [owner_name.]prs_name</pre>
パラメータ	<p>materialized view   precomputed result set</p> <p>マテリアライズされたビューまたは事前計算済み結果セットを削除します。</p> <p><i>prs_name</i></p> <p>事前計算済み結果セットの名前。完全修飾された <i>prs_name</i> にサーバまたはデータベースの名前を含めることはできません。</p>
例	<pre>authors_prs 事前計算済み結果セットを削除します。  drop precomputed result set authors_prs</pre>
使用法	
標準規格	create precomputed result set コマンドは、Transact-SQL 拡張機能で、SQL 規格には含まれません。
パーミッション	事前計算済み結果セットの所有者である必要があります。
監査	事前計算済み結果セットの削除は監査対象ではありません。

## drop procedure

説明	プロシージャを削除します。
構文	<pre>drop proc[edure] [owner.]procedure_name [, [owner.]procedure_name] ...</pre>
パラメータ	<p><i>procedure_name</i></p> <p>削除する Transact-SQL または SQLJ プロシージャの名前です。現在のデータベースで別のユーザが所有している同じ名前のプロシージャを削除する場合は、所有者名を指定します。<i>owner</i> のデフォルト値は現在のユーザです。</p>
例	<p><b>例 1</b> showind ストアド・プロシージャを削除します。</p> <pre>drop procedure showind</pre> <p><b>例 2</b> 拡張ストアド・プロシージャの xp_echo の登録を解除します。</p> <pre>drop procedure xp_echo</pre>
使用法	<ul style="list-style-type: none"> <li>• <code>drop procedure</code> は、ユーザ定義のストアド・プロシージャ、システム・プロシージャ、拡張ストアド・プロシージャ (ESP) を削除します。</li> <li>• Adaptive Server は、ユーザまたはプログラムがプロシージャを実行するたびに、そのプロシージャの存在を確認します。</li> <li>• プロシージャ・グループ (名前が同じでも <code>number</code> サフィックスが異なる複数のプロシージャ) を、1 つの <code>drop procedure</code> 文だけで削除できます。たとえば、<code>orders</code> アプリケーションで使用するプロシージャが <code>orderproc;1</code>、<code>orderproc;2</code> などと命名されている場合、次の文でグループ全体を削除できます。</li> </ul> <pre>drop proc orderproc</pre> <p>プロシージャをグループ化したら、グループ内のプロシージャを個別に削除することはできません。たとえば、次の文は使用できません。</p> <pre>drop procedure orderproc;2</pre> <p>拡張ストアド・プロシージャをプロシージャ・グループとして削除することはできません。</p> <ul style="list-style-type: none"> <li>• <code>sp_helptext</code> は、プロシージャのテキストを表示します。このテキストは <code>syscomments</code> に格納されています。</li> <li>• <code>sp_helpextendedproc</code> は、ESP とそれに対応する DLL を表示します。</li> </ul>

- ESP を削除すると、システム・テーブルから ESP が削除され、プロシージャの登録が解除されます。これは、基本となる DLL には影響しません。
- drop procedure は、現在のデータベースからユーザ作成のストア・プロシージャだけを削除します。

標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

パーミッション

drop procedure のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、プロシージャ所有者であるか drop any procedure 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、プロシージャ所有者であるか sa_role が付与されたユーザであることが必要。

監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
28	drop	drop procedure	<ul style="list-style-type: none"> <li>• Roles — 現在のアクティブな役割</li> <li>• Keywords または options — NULL</li> <li>• Previous value — NULL</li> <li>• Current value — NULL</li> <li>• Other information — NULL</li> <li>• Proxy information — set proxy が有効な場合は元のログイン名</li> </ul>

参照

コマンド [create procedure](#), [create procedure \(SQLJ\)](#)

システム・プロシージャ [sp\\_depends](#), [sp\\_droptext](#), [sp\\_helpextendedproc](#), [sp\\_helpextendedproc](#), [sp\\_helptext](#), [sp\\_rename](#)



## drop role

説明	ユーザ定義の役割を削除します。
構文	<code>drop role <i>role_name</i> [with override]</code>
パラメータ	<p><i>role_name</i> 削除する役割の名前です。</p> <p><b>with override</b> 役割を削除するときのあらゆる制限事項を無効にします。 <b>with override</b> オプションを使用すると、各データベースで役割パーミッションが削除されているかどうかを確認しなくても、すべての役割を削除できます。</p>
例	<p><b>例 1</b> すべてのデータベース内のパーミッションがすべて取り消されている場合に、指定された役割だけを削除します。システム管理者またはオブジェクト所有者は、それぞれのデータベースに付与されているパーミッションを取り消してから、役割を削除する必要があります。この手順を踏まないとコマンドは失敗します。</p> <pre>drop role doctor_role</pre> <p><b>例 2</b> 指名された役割を削除し、その役割に関連したパーミッション情報やリファレンスをすべてのデータベースから取り除きます。</p> <pre>drop role doctor_role with override</pre>
使用法	<ul style="list-style-type: none"> <li>役割の削除前にメンバシップを削除する必要はありません。役割を削除すると、<b>with override</b> オプションを使用するかしないかに関わりなく、その役割内のメンバシップが自動的に削除されます。</li> <li><code>drop role</code> は、<b>master</b> データベースから使用します。</li> </ul>
制限事項	<ul style="list-style-type: none"> <li>システム役割の削除には、<code>drop role</code> を使用できません。</li> </ul>
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>drop role</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、 <b>manage roles</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <b>ssr_role</b> が付与されたユーザであることが必要。
監査	<code>sysaudits</code> の <code>event</code> カラムと <code>extrainfo</code> カラムの値は次のとおりです。

イベント	監査オブション	監査対象のコマンドまたはアクセス	extrainfo の情報
85	roles	create role、drop role、alter role、grant role、または revoke role	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter role](#), [create role](#), [grant](#), [revoke](#), [set](#)

システム・プロシージャ [sp\\_activeroles](#), [sp\\_displaylogin](#), [sp\\_displayroles](#), [sp\\_helprotect](#),

## drop rule

説明	ユーザ定義のルールを削除します。
構文	<code>drop rule [owner.]rule_name[, [owner.]rule_name] ...</code>
パラメータ	<code>rule_name</code> 削除するルールの名前です。現在のデータベースで別のユーザが所有している同じ名前のルールを削除する場合は、所有者名を指定します。 <code>owner</code> のデフォルト値は現在のユーザです。
例	現在のデータベースから <code>pubid_rule</code> ルールを削除します。  <code>drop rule pubid_rule</code>
使用法	<ul style="list-style-type: none"> <li>ルールは、<code>sp_unbindrule</code> を使用してバインドを解除してから削除してください。ルールのバインドが解除されていないと、エラー・メッセージが表示され、<code>drop rule</code> コマンドは失敗します。</li> <li>新しいルールは、現在のルールのバインドを解除しなくても、カラムまたはユーザ定義のデータ型にバインドできます。新しいルールは、古いルールを上書きします。</li> <li>ルールが削除されると、<code>Adaptive Server</code> は、前にルールが適用されていたカラムに、制約なしで新しいデータを入力します。既存のデータには影響ありません。</li> </ul>
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>drop rule</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、ルール所有者であるか <code>drop any rule</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、ルール所有者であるか <code>sa_role</code> が付与されたユーザであることが必要。
監査	<code>sysaudits</code> の <code>event</code> カラムと <code>extrainfo</code> カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
30	drop	drop rule	<ul style="list-style-type: none"><li>• <i>Roles</i> – 現在のアクティブな役割</li><li>• <i>Keywords</i> または <i>options</i> – NULL</li><li>• <i>Previous value</i> – NULL</li><li>• <i>Current value</i> – NULL</li><li>• <i>Other information</i> – NULL</li><li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li></ul>

## 参照

コマンド [create rule](#)システム・プロシージャ [sp\\_bindrule](#), [sp\\_help](#), [sp\\_helptext](#), [sp\\_unbindrule](#)

## drop service

説明	drop service コマンドは、現在のデータベースからユーザ定義 Web サービスを削除します。メタデータと対応するストアド・プロシージャの両方が削除されます。
構文	drop service <i>service-name</i>
パラメータ	<i>service-name</i> ユーザ定義 Web サービスの名前です。サービス名には、ストアド・プロシージャとして有効な任意の名前を指定できます。存在しないサービスの名前を指定すると、例外が発生します。現在別のセッションで使用中のサービスは削除できません。
例	この例では <code>sp_who_service</code> という名前のユーザ定義 Web サービスを削除します。  <pre>drop service sp_who_service</pre>
使用法	ユーザ定義 Web サービスを削除する前に、展開を解除する必要があります。
パーミッション	drop role のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、サービス所有者であるか drop any procedure 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、sa_role が付与されたユーザであることが必要。
参照	コマンド create service ストアド・プロシージャ sp_webservices マニュアル Web Services ユーザーズ・ガイド

## drop table

説明	データベースからテーブルの定義とそのデータ、インデックス、パーティション・プロパティ、トリガ、暗号化プロパティ、パーミッションすべてを削除します。
構文	<pre>drop table [[database.]owner.]table_name           [, [[database.]owner.]table_name] ...</pre>
パラメータ	<p><i>table_name</i></p> <p>削除するテーブルの名前です。テーブルが別のデータベース内にある場合はデータベース名を指定し、データベース内に同じ名前のテーブルが複数ある場合は所有者名を指定します。<i>owner</i> のデフォルト値は現在のユーザで、<i>database</i> のデフォルト値は現在のデータベースです。</p>
例	<p>現在のデータベースから、<code>roysched</code> テーブル、そのテーブルのデータとインデックスを削除します。</p> <pre>drop table roysched</pre>
使用法	<ul style="list-style-type: none"><li>• <code>drop table</code> を使用すると、テーブルのすべてのルールやデフォルトのバインドが失われ、関連付けられていたすべてのトリガが自動的に削除されます。テーブルを再作成するときは、適切なルールおよびデフォルト値をバインドしなおし、トリガも再作成する必要があります。</li><li>• テーブルを削除すると、そのテーブルに関連するパーティション条件も削除されます。</li><li>• テーブルを削除すると、テーブルのカラムに関連付けられている復号化デフォルト値と、カラムの暗号化プロパティも削除されます。</li><li>• テーブルを削除したときに影響を受けるシステム・テーブルは、<code>sysobjects</code>、<code>syscolumns</code>、<code>sysindexes</code>、<code>sysprotects</code>、<code>syscomments</code>、<code>syspartitions</code>、<code>syspartitionkeys</code>、<code>sysprocedures</code> です。</li><li>• コンポーネント統合サービスが有効であり、削除するテーブルが <code>create existing table</code> を使用して作成されていた場合は、そのテーブルはリモート・サーバから削除されません。代わりに <code>Adaptive Server</code> が、そのテーブルに対するリファレンスをシステム・テーブルから削除します。</li></ul> <p><b>制限事項</b></p> <ul style="list-style-type: none"><li>• システム・テーブルでは <code>drop table</code> コマンドを使用できません。</li></ul>

- どのデータベースのテーブルでも、そのテーブルの所有者であれば削除できます。たとえば、`otherdb` データベースにあるテーブル `newtable` を削除するには、次のいずれかの文を使用します。

```
drop table otherdb..newtable
drop table otherdb.yourname.newtable
```

- `delete` コマンドでテーブル内のすべてのローを削除したり、`truncate table` コマンドを使用しても、`drop` コマンドを実行してテーブルを削除するまで、そのテーブルは存在します。

#### データベース間の参照整合性制約を持つテーブルの削除

- データベース間の制約を作成すると、各データベースの `sysreferences` システム・テーブルに次の情報が格納されます。

表 1-18: 参照整合性の制約について保管される情報

sysreferences に格納される情報	参照先テーブルについての情報を持つカラム	参照元テーブルについての情報を持つカラム referencing table
キー・カラム ID	refkey1 ~ refkey16	fokey1 ~ fokey16
テーブル ID	reftabid	tableid
データベース名	pmrydbname	frgndbname

- 参照元テーブルは参照先テーブルの情報に依存しているため、Adaptive Server で次の操作は実行できません。
  - 参照先テーブルを削除する
  - 参照先テーブルを含む外部データベースを削除する
  - `sp_renamedb` の使用によるいずれかのデータベースの名前変更
- `sp_helpconstraint` を使用すると、削除対象のテーブルをどのテーブルが参照しているかを確認できます。`alter table` を使用して制約を削除してから `drop table` を再発行してください。
- 参照元テーブルまたは参照元テーブルのデータベースを削除できます。Adaptive Server は、外部キー情報を参照先データベースから自動的に削除します。

- データベース間の制約を追加または削除した場合や、データベース間の制約を含むテーブルを削除した場合は、影響を受けたデータベースを両方ともダンプしてください。

**警告！** これらのデータベースの以前のダンプをロードすると、データベースが破壊される可能性があります。データベース間の参照整合性制約を使用してデータベースをロードする方法の詳細については、『システム管理ガイド』を参照してください。

標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

パーミッション

drop table のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であるか <b>drop any table</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者であるか <b>sa_role</b> が付与されたユーザであることが必要。

監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
27	drop	drop table	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — set proxy が有効な場合は元のログイン名</li> </ul>

参照

コマンド [alter table](#), [create table](#), [delete](#), [truncate table](#)

システム・プロシージャ [sp\\_depends](#), [sp\\_help](#), [sp\\_spaceused](#)



## drop thread pool

説明	ユーザ定義のプールを削除します。
プロセス・モードの考慮事項	drop thread pool はプロセス・モードではサポートされていません。
構文	drop thread pool <i>pool_name</i>
パラメータ	<i>pool_name</i> 削除するプールの名前です。
例	sales_pool という名前のプールを削除します。 <pre>drop thread pool sales_pool</pre>
使用法	<ul style="list-style-type: none"> <li>削除したスレッド・プールと関連付けられているタスクは、Adaptive Server によって syb_default_pool に再度割り当てられます。</li> <li>実行クラスの定義を現在使用しているスレッド・プールは削除できません。実行クラスを削除するには、sp_dropexclass を使用します。</li> <li>システムによって作成されたスレッド・プール (syb_ で始まるスレッド・プール) は削除できません。</li> <li>スレッド・プールを削除するには、現在実行中のタスクが停止するまで待つ必要があります。そのため、プールを削除する際に Adaptive Server で多少の遅延が生じる場合があります。</li> <li>削除するスレッド・プールで実行されているタスクは syb_default_pool にマイグレートされます。</li> <li>Transact-SQL 変数をパラメータとして使用して、drop thread pool を実行することはできません。</li> <li>drop thread pool は execute immediate を指定して発行できます。</li> </ul>
標準規格	ANSI SQL – 準拠レベル：Transact-SQL 拡張機能
パーミッション	drop thread pool のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、manage any thread pool 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、sa_role が付与されたユーザであることが必要。

### 監査

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
144			プール名

## drop trigger

説明	トリガを削除します。
構文	<code>drop trigger [owner.]trigger_name [, [owner.]trigger_name] ...</code>
パラメータ	<i>trigger_name</i> 削除するトリガの名前です。現在のデータベースで別のユーザが所有している同じ名前のトリガを削除する場合は、所有者名を指定します。 <i>owner</i> のデフォルト値は現在のユーザです。
例	現在のデータベースから、 <code>trigger1</code> を削除します。 <pre>drop trigger trigger1</pre>
使用法	<ul style="list-style-type: none"> <li>• <code>drop trigger</code> は、現在のデータベースにあるトリガを削除します。</li> <li>• 同じオペレーション (<code>insert</code>、<code>update</code>、<code>delete</code>) を実行する新しいトリガを作成するときに、テーブルからトリガを明示的に削除する必要はありません。テーブルまたはカラム内では、同じオペレーションを実行する新しいトリガは、それぞれ古いものを上書きします。</li> <li>• テーブルが削除されると、Adaptive Server はそのテーブルに関連付けられたトリガを自動的に削除します。</li> </ul>
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>drop trigger</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、トリガ所有者であるか <code>drop any trigger</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、トリガ所有者か <code>sa_role</code> が付与されたユーザであることが必要。

監査 `sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
29	drop	drop trigger	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照 コマンド [create trigger](#)  
システム・プロシージャ [sp\\_depends](#), [sp\\_help](#), [sp\\_helptext](#)

## drop view

説明	1 つまたは複数のビューを現在のデータベースから削除します。		
構文	<code>drop view [owner.]view_name [, [owner.]view_name] ...</code>		
パラメータ	<p><i>view_name</i></p> <p>削除するビューの名前です。現在のデータベースで別のユーザが所有している同じ名前のビューを削除する場合は、所有者名を指定します。<i>owner</i> のデフォルト値は現在のユーザです。</p>		
例	<p>現在のデータベースから <code>new_price</code> ビューを削除します。</p> <pre>drop view new_price</pre>		
使用法	<ul style="list-style-type: none"> <li>• <code>drop view</code> を使用すると、ビューの定義や、権限などのその他のビュー情報がシステム・テーブル <code>sysobjects</code>、<code>syscolumns</code>、<code>syscomments</code>、<code>sysdepends</code>、<code>sysprocedures</code>、および <code>sysprotects</code> から削除されます。</li> <li>• 別のビューまたはストアド・プロシージャなどによって、ビューが参照されるたびにビューの存在が確認されます。</li> </ul>		
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能		
パーミッション	<code>drop view</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。		
細密なパーミッションが有効	細密なパーミッションが有効の場合、ビュー所有者であるか <code>drop any view</code> 権限を持つユーザであることが必要。		
細密なパーミッションが無効	細密なパーミッションが無効の場合、ビュー所有者であるか、 <code>sa_role</code> が付与されていることが必要。		
監査	<code>sysaudits</code> の <code>event</code> カラムと <code>extrainfo</code> カラムの値は次のとおりです。		
イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
33	drop	drop view	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> </ul>
参照	<p>コマンド <a href="#">create view</a></p> <p>システム・プロシージャ <a href="#">sp_depends</a>, <a href="#">sp_help</a>, <a href="#">sp_helptext</a></p>		

## dump configuration

説明	Adaptive Server の設定ファイルのバックアップを、指定されたダンプ・ディレクトリに作成します。コピーは Backup Server ではなく Adaptive Server によって作成されます。
構文	<pre>dump config[uration]     [with {         file = (file_option, file_option...)}     ]]</pre>
パラメータ	<p>with file = (file_option, file_option...)}          指定したオプションに従って 1 つ以上のファイルのバックアップを作成できるようにします。</p> <p>有効な <i>file_option</i> 値は以下のとおりです。</p> <ul style="list-style-type: none"> <li>• server_config[uration] — サーバ設定ファイル</li> <li>• dump_history — ダンプ履歴ファイル</li> <li>• cluster_config[uration] — クラスタ設定ファイル</li> </ul> <p>Adaptive Server Cluster Edition のクラスタ設定では、インスタンスのそれぞれに独自のサーバ設定ファイルを設定できます。<b>dump configuration</b> コマンドは、インスタンスの最初のサーバ設定ファイルをダンプします。クラスタ設定ファイルは、クォーラム・デバイスから生成され、現在のタイムスタンプを現在のタイムスタンプが追加された <i>cluster.cfg</i> で指定されます。</p> <ul style="list-style-type: none"> <li>• 'all' — (デフォルトの) リストされた (既知の) すべての設定ファイル</li> </ul> <p><i>file_option</i> を省略すると、既存のすべての設定ファイルがバックアップされます。</p>
例	<p><b>例 1</b> サーバ設定ファイルのバックアップを作成し、履歴ファイルをダンプします。</p> <pre>dump configuration     with file = (server_config, dump_history)</pre> <p><b>例 2</b> リストされた (既知の) すべての設定ファイルのバックアップを作成します。</p> <pre>dump configuration</pre>
使用法	<ul style="list-style-type: none"> <li>• <i>list_option</i> の値 'all' は、一重または二重引用符で囲む必要があります。</li> </ul>
標準規格	ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

パーミッション	dump configuration のパーミッション・チェックは、細密なパーミッションの設定によって異なります。	
細密なパーミッションが有効	細密なパーミッションが有効の場合、manage dump configuration 権限を持つユーザであることが必要。	
細密なパーミッションが無効	細密なパーミッションが無効の場合、oper_role または sa_role が付与されたユーザであることが必要。	

**監査** sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
147	cs_config、 .....	dump configuration	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – ユーザが指定した設定ファイル・オプション</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

**参照** マニュアル 『システム管理ガイド』の「ユーザ・データベースのバックアップとリストア」

コマンド dump database、load database、load transaction、online database  
システム・プロシージャ sp\_addumpdevice、sp\_dboption、sp\_dropdevice、sp\_helpdevice、sp\_hidetext、sp\_logdevice、sp\_volchanges

## dump database

### 説明

`load database` で読み込むことができる形式で、トランザクション・ログを含むデータベース全体のバックアップ・コピーを作成します。ダンプとロードは `Backup Server` で行います。

圧縮データをダンプしない限り、`load database` オペレーションのターゲット・プラットフォームは、`dump database` オペレーションを実行するソース・プラットフォームと同じプラットフォームである必要はありません。圧縮データのダンプとロードは同じプラットフォームで行われる必要があります。ただし、`dump database` と `load database` はビッグ・エンディアン・プラットフォームからリトル・エンディアン・プラットフォームに対して (またはリトル・エンディアン・プラットフォームからビッグ・エンディアン・プラットフォームに対して) 実行されます。

サイトで `Tivoli Storage Manager` がライセンスされている場合の `dump database` 構文については、『`IBM Tivoli Storage Manager` と `Backup Server` の使用』を参照してください。

### 構文

```
dump database database_name
  using config[uration] = config_name
  [with {
    verify[ = header | full]
  }]
  to [compress::[compression_level::]]stripe_device
    [at backup_server_name]
    [density = density_value,
    blocksize = number_bytes,
    capacity = number_kilobytes,
    dumpvolume = volume_name,
    file = file_name]
    [with shrink_log]
    with verify[= header | full]
  [stripe on [compress::[compression_level::]]stripe_device
    [at backup_server_name]
    [density = density_value,
    blocksize = number_bytes,
    capacity = number_kilobytes,
    dumpvolume = volume_name,
    file = file_name]]
  [[stripe on [compress::[compression_level::]]stripe_device
    [at backup_server_name]
    [density = density_value,
    blocksize = number_bytes,
    capacity = number_kilobytes,
    dumpvolume = volume_name,
    file = file_name]]...]
  [with {
```

```

density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
compression = compress_level
dumpvolume = volume_name,
file = file_name,
[dismount | nodismount],
[nounload | unload],
passwd = password,
retaindays = number_days,
[noinit | init],
notify = {client | operator_console}
}

```

(Tivoli Storage Manager) Tivoli Storage Manager からバックアップ・サービスが提供されている場合は、次の構文を使用してデータベースをコピーします。

```

dump database database_name
to "syb_tsm::object_name"
[blocksize = number_bytes]
[stripe on "[syb_tsm::object_name"
[blocksize = number_bytes]...]
[with {
blocksize = number_bytes,
compression = compress_level,
passwd = password,
[noinit | init],
notify = {client | operator_console},
verify[ = header | full]
}]

```

## パラメータ

`config[uration]` = *config\_name*

指定されたダンプ設定を読み込み、指定された値を使用してダンプ操作を実行します。

ダンプ設定を使用する場合は、コマンドのパラメータとしてストライプ・ディレクトリを指定することはできません。Adaptive Server によって、ダンプ設定によって指定されたストライプ・ディレクトリにダンプ・ファイルが作成されます。ダンプ・ファイルの名前には、次の命名規則が使用されます。

*database\_name.dump\_type.date-timestamp.stripeID*

ダンプ設定で指定されたパラメータ値は、明示的に指定されたコマンド・パラメータによって上書きされます。

**database\_name**

データのコピー元であるデータベースの名前です。データベース名は、リテラル、ローカル変数、またはストア・プロシージャのパラメータとして指定できます。

**compress::compression\_level**

非推奨になりました。古いアプリケーションとの互換性保持のみを目的として残されています。この代わりに、圧縮には "**compression = compress\_level**" を使用します。compress オプションの詳細については、『システム管理ガイド 第2巻』の「ユーザ・データベースのバックアップとリストア」を参照してください。

---

**注意** ネイティブの "**compression = compress\_level**" オプションを使用し、古い "**compress::compression\_level**" オプションは使用しないことを推奨します。ネイティブのオプションを使用すると、ローカルとリモートの両方のダンプを圧縮できます。このオプションで作成されるダンプは、ロード時に圧縮レベルを示します。古いオプションは、古いアプリケーションとの互換性のために維持されています。

---

**to stripe\_device**

データのコピー先のデバイス名です。ダンプ・デバイス名の指定時に使用する形式については、「[ダンプ・デバイスの指定](#)」(437 ページ) を参照してください。

**at backup\_server\_name**

Backup Server の名前です。デフォルトの Backup Server にダンプする場合は、このパラメータを指定しないでください。このパラメータは、リモート Backup Server にネットワークを介してダンプする場合だけ指定してください。このオプションでは、リモート Backup Server を 32 個まで指定できます。ネットワークを介してダンプする場合、ダンプ・デバイスが接続されるマシンで稼働しているリモート Backup Server の *network name* を指定します。interfaces ファイルを使用するプラットフォームでは、interfaces ファイルで *backup\_server\_name* が示されている必要があります。

**density = density\_value**

テープ・デバイスのデフォルトの記録密度を上書きします。有効な密度は、800、1600、6250、6666、10000、38000 です。すべての値がどのテープ・ドライブに対しても有効というわけではありません。使用しているテープ・ドライブにあった記録密度を選択してください。



**blocksize = number\_bytes**

ダンプ・デバイスのデフォルトのブロック・サイズを上書きします。ブロック・サイズは、少なくとも 1 データベース・ページ (ほとんどのシステムでは 2048 バイト) の大きさと、データベース・ページ・サイズの倍数でなければなりません。最適のパフォーマンスを得るには、**blocksize** を 2 の累乗 (65536、131072、262144 など) で指定します。

**capacity = number\_kilobytes**

デバイスが単一のテープ・ボリュームに書き込むことのできるデータの最大量です。収容データ量は、少なくとも 5 データベース・ページ分で、デバイスに推奨される最大値より少ない値にしてください。

処理能力計算の一般的な規則は、デバイス製造者仕様の最大能力の 70% を使用することです。30% はレコードのギャップ、テープ・マークなどのオーバーヘッドのための余裕分として空けておきます。最大能力とはドライブ上のデバイスの能力で、ドライブ自体の能力ではありません。この規則はほとんどの場合に当てはまりますが、ベンダ間およびデバイス間でオーバーヘッドが異なるため、すべての場合に当てはまるわけではありません。

テープ終了マーカを確実に検出できない UNIX プラットフォームでは、テープへのダンプが可能なキロバイト数を指示します。物理パス名として指定したダンプ・デバイスには、**capacity** を指定してください。ダンプ・デバイスを論理デバイス名として指定した場合、容量が指定されていない限り、Backup Server では、**sysdevices** システム・テーブルに格納されている **size** パラメータが使用されます。

**compression = compress\_level**

圧縮レベルを 0 ~ 9、100、または 101 の数字で指定します。1 桁の圧縮レベルでは、0 は圧縮なし、9 は圧縮の最高レベルです。圧縮レベル 100 および 101 は、より高速で効率的な圧縮モードです。

100 はより高速の圧縮、101 はより効率的な圧縮を行います。**compress\_level** を指定しない場合、ダンプは圧縮されません。

---

**注意** ネイティブの "**compression = compress\_level**" オプションを使用し、古い "**compress::compression\_level**" オプションは使用しないことを推奨します。古いオプションは、古いアプリケーションとの互換性のために維持されています。

---

**dumpvolume = volume\_name**

ボリュームに割り当てる名前を指定します。*volume\_name* の最大長は 6 文字です。既存のダンプへの上書き、新しいテープへのダンプ、または内容が認識できないテープへのダンプを行う場合、Backup Server は ANSI テープ・ラベルに *volume\_name* を書き込みます。load database コマンドはラベルを確認し、誤ったボリュームがロードされた場合はエラー・メッセージを表示します。

---

**警告！** オペレータが正しいテープをロードできるように、テープ・ボリューム作成時に必ず各テープ・ボリュームにラベルを付けてください。

---

**with shrink\_log**

alter database log off コマンドを使用したログの縮小時にデータベースに穴が作成される可能性がある場合に使用されます。このコマンドは、データベースがダンプ・シーケンスにない場合にデータベースの最後にある穴を自動的に削除します。同様に、dump database は、データベースがダンプ・シーケンスにない場合（つまり、最低限のログを取るコマンドが実行されるときなど、dump transaction が許可されていないために dump database を実行することが強制される場合）に、データベースの最後にある穴を自動的に削除します。dump database の with shrink\_log オプションは、データベースがダンプ・シーケンスにあるかどうかにかかわらず、データベースの最後にある穴を削除します。

**with verify[= header | full]**

バックアップ・サーバで、アーカイブにコピーするデータ・ページに対して最小限のヘッダー検査またはロー構造検査を実行します。この時点では、gam、oam、allocation pages、indexes、text、log ページに構造検査は実行されません。ページ番号がページ・ヘッダと一致するページに対して、他の検査だけが実行されます。

**stripe on stripe\_device**

追加のダンプ・デバイスです。to stripe\_device 句で指定したデバイスを含め、32 個までのデバイスを使用できます。Backup Server はデータベースをほぼ等分し、各部分を異なるデバイスに送信します。ダンプはすべてのデバイスで同時に行われるので、ダンプに必要な時間が減少し、ダンプ中のボリュームの交換が少なくなります。「[ダンプ・デバイスの指定](#)」(437 ページ) を参照してください。

**dismount | nodismount**

論理的なマウント解除をサポートするプラットフォームで、テープをマウントしたままにするかどうかを指定します。デフォルトでは、ダンプの完了時に、ダンプに使用したすべてのテープのマウントが解除されます。追加のダンプまたはロードに引き続きテープを使用できるようにするには、**nodismount** を使用します。

**nounload | unload**

ダンプの完了後にテープを巻き戻すかどうかを指定します。デフォルトでは、テープは巻き戻されません。したがって、同じテープ・ボリュームにダンプを追加できます。マルチダンプ・ボリュームに追加される最後のダンプ・ファイルには、**unload** を指定します。この指定によって、ダンプの完了時にテープの巻き戻しおよびアンロードが行われます。

**passwd = password**

不正なユーザからダンプ・ファイルを保護するために指定するパスワードです。パスワードの長さは、6～30文字にする必要があります。パスワードに変数は使用できません。『システム管理ガイド 第 1 巻』の「Adaptive Server のログイン、データベース・ユーザ、クライアント接続の管理」を参照してください。

**retaindays = number\_days**

(UNIX システム) ディスクにダンプするときに、ダンプが上書きされないように Backup Server が保護する日数を指定します。指定した期限が切れる前にダンプを上書きしようとする、期限内のボリュームに上書きを行う前に、Backup Server が確認を要求します。

---

**注意** このオプションは、ディスクにダンプする場合にのみ適用され、テープ・ダンプには適用されません。

---

**number\_days** には、正の整数を指定してください。また、すぐにダンプを上書きできるようにする場合は 0 を指定してください。  
**retaindays** 値を指定しない場合、Backup Server は **sp\_configure** によって設定される **tape retention in days** 値を使用します。

**noinit | init**

既存のダンプ・ファイルにダンプを追加するか、またはダンプによってテープ・ボリュームの再初期化(上書き)を行うかどうかを指定します。デフォルトでは、**Adaptive Server** は前回のテープ終了マークの後からダンプを追加するため、追加のデータベースを同じボリュームにダンプできます。新しいダンプは、マルチボリューム・ダンプの最後のボリュームだけに追加できます。内容を上書きするには、テープにダンプする最初のデータベースに対して **init** を使用します。

**Backup Server** によってテープ設定ファイルのテープ・デバイス特性が保管または更新されるようにするには、**init** を使用します。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。

**file = file\_name**

ダンプ・ファイルの名前です。名前は 17 文字までで、オペレーティング・システムのファイル命名規則に従っていなければなりません。「[ダンプ・ファイル](#)」(439 ページ) を参照してください。

**notify = {client | operator\_console}**

デフォルトのメッセージ送信先を上書きします。

オペレータ端末機能を提供するオペレーティング・システムでは、ボリューム交換メッセージは、**Backup Server** が実行されているマシンのオペレータ端末に常に送信されます。**dump database** を開始した端末セッションに **Backup Server** の他のメッセージを送信するには、**client** を使用します。

UNIX などのオペレータ端末機能を提供しないオペレーティング・システムでは、メッセージは **dump database** を開始したクライアントに送信されます。**Backup Server** が稼動している端末にメッセージを送信するには、**operator\_console** を使用します。

**syb\_tsm::obj\_name**

**Backup Server** と **Tivoli Storage Manager** 間の通信を可能にする *libsyb\_tsm.so* モジュールを起動するキーワードです。

**object\_name**

TSM サーバのバックアップ・オブジェクトの名前です。

**例 1** *dmp\_cfg2* ダンプ設定を使用して、データベースをダンプします。

```
dump database testdb using config=dmp_cfg2
```

**例 2** *dmp\_cfg2* ダンプ設定を使用して、データベースをダンプします。ダンプ操作の一環として作成されたアーカイブ・ファイルはパスワードで保護されています。

```
dump database testdb using config=dmp_cfg2
with passwd='mypass01'
```

---

**注意** パスワードは、一重または二重引用符で囲む必要があります。

---

**例 3** *dmp\_cfg2* ダンプ設定を使用し、圧縮レベル 6 を明示的に指定することで、*dmp\_cfg2* の圧縮レベルを上書きして、データベース・ダンプを実行します。

```
dump database testdb using config=dmp_cfg2
with compression=6
```

**例 4** テープ・デバイスに *pubs2* データベースをダンプします。テープに ANSI テープ・ラベルがある場合、*init* オプションが指定されていないため、このダンプは *dump database* コマンドによってテープの既存ファイルに追加されます。

```
dump database pubs2
to "/dev/nrmt0"
```

**例 5** (UNIX のみ) *REMOTE\_BKP\_SERVER Backup Server* を使用して、*pubs2* データベースをダンプします。3 つのダンプ・デバイスがコマンドによって指定されるため、*Backup Server* は各デバイスにデータベースの約 1/3 ずつダンプします。このコマンドは、テープの既存のファイルにダンプを追加します。*retaindays* オプションは、テープが 14 日間上書きできないように指定します。

```
dump database pubs2
to "/dev/rmt4" at REMOTE_BKP_SERVER
stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
stripe on "/dev/nrmt0" at REMOTE_BKP_SERVER
with retaindays = 14
```

**例 6** *init* オプションは、テープ・ボリュームを初期化し、既存のファイルを上書きします。

```
dump database pubs2
to "/dev/nrmt0"
with init
```

**例 7** ダンプの完了時にダンプ・ボリュームを巻き戻します。

```
dump database pubs2
to "/dev/nrmt0"
with unload
```

**例 8** (UNIX のみ) `notify` 句は、デフォルト・ロケーションである Backup Server マシンのコンソールにではなく、ダンプ要求を開始したクライアントに対して、ボリュームの変更を要求する Backup Server メッセージを送信します。

```
dump database pubs2
  to "/dev/nrmt0"
  with notify = client
```

**例 9** ローカル・ファイル `dmp090100.dmp` に、`pubs2` データベースの圧縮ダンプを圧縮レベル 4 で作成します。

```
dump database pubs2 to
  "compress::4::/opt/bin/Sybase/dumps/dmp090100.dmp"
```

または、ローカル・ファイル `dmp090100.dmp` に、`pubs2` データベースの圧縮ダンプを、`compression = compression_level` 構文を使用して圧縮レベル 100 で作成できます。

```
dump database pubs2 to "/opt/bin/Sybase/dumps/dmp090100.dmp"
  with compression = 100
```

**例 10** `pubs2` データベースを圧縮レベル 4 で “remotemachine” という名前のリモート・マシンにダンプします。

```
dump database pubs2 to "/Syb_backup/mydb.db" at remotemachine
  with compression = "4"
```

**例 11** `pubs2` データベースを TSM バックアップ オブジェクト “obj1.1” にダンプします。

```
dump database pubs2 to "syb_tsm::obj1.1"
```

**例 12** 複数のストライプを使用して、`pubs2` データベースを TSM バックアップ オブジェクト “obj1.2” にダンプします。

```
dump database pubs2 to "syb_tsm::obj1.2"
  stripe on "syb_tsm::obj1.2"
  stripe on "syb_tsm::obj1.2"
  stripe on "syb_tsm::obj1.2"
  stripe on "syb_tsm::obj1.2"
```

**例 13** `sales_db1` の最後の部分 (データベースの最後にあるデータベースの穴) を削除します。

`select *` は、データベースの最後に穴があることを示します。

```
select * from sysusages where dbid=db_id("sales_db1")
go
```

```
dbid segmap lstart  size vstart location unreservedpgs  crdate          vdevno
-----
```

```

5      3      0 1536  1536      0      598 May 5 2011 2:59PM      3
5      4    1536 1536  1536      0    1530 May 5 2011 2:59PM      4
5      0    3072 1536  3072      4    1526 May 5 2011 2:59PM     -5

```

```

dump database sales_db1 to "/tmp/sales_db1.dmp" with shrink_log
go

```

```

Backup Server session id is:42.Use this value when executing the 'sp_volchanged'
system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server:4.41.1.1:Creating new disk file /tmp/sales_db1.dmp.
Backup Server:6.28.1.1:Dumpfile name 'sales_db1111250D8E6 ' section number 1 mounted
on disk file '/tmp/sales_db1.dmp'
Backup Server:4.188.1.1:Database sales_db1:892 kilobytes (55%) DUMPED.
Backup Server:4.188.1.1:Database sales_db1:934 kilobytes (100%) DUMPED.
Backup Server:3.43.1.1:Dump phase number 1 completed.
Backup Server:3.43.1.1:Dump phase number 2 completed.
Backup Server:3.43.1.1:Dump phase number 3 completed.
Backup Server:4.188.1.1:Database sales_db1:942 kilobytes (100%) DUMPED.
Backup Server:3.42.1.1:DUMP is complete (database sales_db1).

```

`select *` を実行して、この部分が正常に削除されていることを確認します。

```

select * from sysusages where dbid=db_id("sales_db1")
go
dbid segmap lstart  size vstart  location unreservedpgs  crdate                vdevno
-----
5      3      0 1536  1536      0      598 May 5 2011 2:59PM      3
5      4    1536 1536  1536      0    1530 May 5 2011 2:59PM      4

```

## 使用法

- `sp_hidetext` の後で、プラットフォーム間の `dump` と `load` を使用した場合は、すべての隠しオブジェクトを手動で削除して再作成する必要があります。
- `dump database` コマンドは、3 段階で実行されます。各段階が完了すると、それを知らせる進行メッセージが表示されます。ダンプの完了時には、実行中に行われたすべての変更 (段階 3 で開始された変更を除く) が反映されます。
- 表 1-19 に、データベースのバックアップに使用するコマンドおよびシステム・プロシージャを示します。

表 1-19: データベースおよびログのバックアップに使用するコマンド

目的	使用
トランザクション・ログを含む、データベース全体の定期的なダンプを作成する。	<code>dump database</code>

目的	使用
トランザクション・ログの日常ダンプを作成し、非アクティブの部分をトランケートする。dump transaction が dump database と同時に実行される場合、トランザクション・ログの非アクティブ部分はトランケートされない。	dump transaction
データベース・デバイスの障害後、トランザクション・ログをダンプする。	dump transaction with no_truncate
バックアップを作成しないでログをトランケートし、データベース全体をコピーする。	dump transaction with truncate_only dump database
ログ領域の不足が原因で通常の方法が失敗した場合に、ログをトランケートしてから、データベース全体をコピーする。	dump transaction with no_log dump database
Backup Server のボリューム変更メッセージに応答する。	sp_volchanged

### 制限事項

- 物理デバイスのファイル・パスまたはファイル名の最大サイズは 127 文字です。
- プロキシ・テーブルがデータベースにある場合は、このテーブルはデータベース・セーブ・セットに組み込まれます。プロキシ・テーブルの内容は保存されません。ポインタのみが保存およびリストアされます。
- データベースの最後でない穴は、with shrink\_log オプションを使用して削除できません。
- Sybase ダンプと Sybase 以外のデータ (UNIX アーカイブなど) を同じテーブルに格納することはできません。
- データベースにデータベース間の参照整合性制約がある場合、sysreferences システム・テーブルに外部データベースの ID 番号ではなく、名前が保存されます。Adaptive Server では、load database を使用してデータベース名を変更したり、別のサーバにデータベースをロードしたりすると、参照整合性は保証されません。

---

**警告！** データベースを別の名前でロードしたり、データベースを別の Adaptive Server に移動するためにデータベースをダンプするには、事前に alter table を使用して外部参照整合性制約をすべて削除します。

---

- ユーザ定義のトランザクションでは、dump database を使用できません。



- `dump transaction` がすでに進行中のデータベースで `dump database` を発行すると、トランザクション・ダンプが完了するまで `dump database` がスリープします。
- 1/4 インチのカートリッジ・テープを使用している場合、1 本のテープにつきダンプできるデータベースまたはトランザクション・ログは 1 つだけです。
- オフライン・ページがあるデータベースはダンプできません。オフライン・ページを強制的にオンラインにするには、`sp_forceonline_db` または `sp_forceonline_page` を使用します。
- プラットフォーム間のダンプとロードでは、`dump database` を実行する前に、データベースをトランザクションが実行されていない状態に移行する必要があります。
  - a `dbcc checkdb` と `dbcc checkalloc` を実行して、データベースの整合性に問題がないことを確認します。
  - b `dump database` の実行中に他のプロセスからオープン・トランザクションの同時更新が行われないように、`dump database` でデータベースをシングルユーザ・モードに移行します。
  - c `sp_flushstats` で統計値を `systabstats` へフラッシュします。
  - d 10 ~ 30 秒待ちます。この期間はデータベースのサイズおよびアクティビティによって異なります。
  - e データベースに対して `checkpoint` を実行して、更新済みのページをフラッシュします。
  - f `dump database` を実行します。
- `dump transaction` と `load transaction` をプラットフォーム間で実行することはできません。
- リモート `backupserver` に対する `dump database` と `load database` をプラットフォーム間で実行することはできません。
- パスワードで保護されたダンプ・ファイルをプラットフォーム間でロードすることはできません。
- 解析済み XML オブジェクトに対して `dump database` と `load database` を実行する場合は、`load database` の実行後にテキストを再度解析する必要があります。
- Adaptive Server は、`binary`、`varbinary`、または `image` カラムとして格納されている埋め込みデータ構造を変換することはできません。

- プラットフォーム間で master データベースに対する load database を実行することはできません。
- スタアド・プロシージャやその他のコンパイル済みオブジェクトは、load database の実行後最初に実行されるとき syscomments 内の SQL テキストから再コンパイルされます。

テキストから再コンパイルするパーミッションがない場合は、パーミッションを持つ人が dbcc upgrade\_object でテキストから再コンパイルしてオブジェクトをアップグレードする必要があります。

---

**注意** master データベースの syslogins システム・テーブル内のログイン・レコードを Solaris から Linux にマイグレートする場合は、bcp -c 文字フォーマットのバルク・コピーを実行できます。Solaris からのログイン・パスワードは Linux と互換性があります。これ以外のプラットフォームの組み合わせについては、パスワードが適合しないのでログイン・レコードを作り直す必要があります。

---

#### ダンプのスケジューリング

- Adaptive Server データベースのダンプは動的に実行されるため、データベースがアクティブであってもダンプを実行できます。ただし、この動的なダンプによってシステムの処理速度が多少遅くなる恐れがあります。したがって、dump database はデータベースの更新量が少ないときに実行するようにしてください。
- master データベースは、定期的に、また頻繁にバックアップしてください。通常のバックアップの他に、create database、alter database、disk init コマンドを発行した後は、必ず master データベースをダンプしてください。
- データベースに変更を加えるたびに model データベースをバックアップしてください。
- データベースの作成後はただちに dump database を使用して、データベース全体のコピーを作成してください。dump database を実行するまでは、新しいデータベースで dump transaction を実行することはできません。
- データベース間の制約を追加または削除した場合や、データベース間の制約を含むテーブルを削除した場合は、影響を受けたデータベースを両方ともダンプしてください。

---

**警告！** これらのデータベースの以前のダンプをロードすると、データベースが破壊される可能性があります。

---

- ユーザ・データベースおよびそのトランザクション・ログを定期的にバックアップするスケジュールを作成してください。
- バックアップ手順を自動化するにはスレッシュホールドを使用します。Adaptive Server のラストチャンス・スレッシュホールドを活用するには、データ・セグメントとは異なるデバイス上のログ・セグメントを指定してユーザ・データベースを作成します。スレッシュホールドの詳細については、『システム管理ガイド』を参照してください。

#### システム・データベースのダンプ

- master、model、および sybsystemprocs データベースには、トランザクション・ログ用の独立したセグメントはありません。dump transaction with truncate\_only を使用してログを消去してから、dump database を使用してデータベースをバックアップしてください。
- master データベースに影響する障害が発生した場合のリカバリ手順では、master データベースのバックアップも必要になります。master データベースのバックアップとリストアの各手順の詳細については、『システム管理ガイド』を参照してください。
- リムーバブル・メディアをバックアップに使用するときは、ボリューム変更メッセージに対応できる別の Adaptive Server がある場合を除き、master データベース全体を単独のボリュームに収容できる容量が必要です。

#### ダンプ・デバイスの指定

- ダンプ・デバイスは、リテラル、ローカル変数、またはストアド・プロシージャへのパラメータとして指定できます。
- null デバイスにはダンプできません (UNIX の場合は、/dev/null)。
- 複数のストライプへのダンプは、テープ・デバイスとディスク・デバイスに対応しています。1つのデバイスに対する複数のダンプは、テープ・デバイスの場合だけにサポートされています。
- ローカル・ダンプ・デバイスは、次のように指定できます。
  - sysdevices システム・テーブルの論理デバイス名
  - 絶対パス名
  - 相対パス名

Backup Server は、Adaptive Server の現在の作業ディレクトリを使用して相対パス名を解決します。

- ネットワーク間でダンプを行う場合は、ダンプ・デバイスの絶対パス名を指定してください。パス名は、Backup Server が起動しているマシンで有効なものでなければなりません。名前に文字、数字、またはアンダースコア (\_) 以外の文字が含まれる場合は、引用符で囲んでください。
- ダンプ・デバイスでの所有権とパーミッションの問題によって、dump コマンドが使用できないことがあります。sp\_addumpdevice によってデバイスをシステム・テーブルに追加できますが、そのデバイスにダンプできるか、またはダンプ・デバイスとしてファイルを作成できるかどうかは保証されていません。
- 各ダンプに別のダンプ・デバイスを使用すると、複数のダンプ (またはロード) を同時に実行できます。
- デバイス・ファイルがすでにある場合は、Backup Server によって上書きされますが、既存のデバイス・ファイルのトランケートは行われません。たとえば、データベースをデバイス・ファイルにダンプした結果、デバイス・ファイルが 10 MB になったとします。そのデバイスに次回ダンプしたときにデータベースのサイズが 10MB より小さくても、デバイス・ファイルのサイズは 10 MB のままです。

#### 圧縮データのダンプ

- 1つのプラットフォームで作成した圧縮テーブルのダンプを、別のプラットフォームにロードすることはできません。
- 圧縮データはアーカイブの場所に直接ダンプされます。
- あらゆる形式の圧縮ローまたは非圧縮ローを含む圧縮テーブルに対する create index コマンドは load transaction 時に完全にリカバリされます。

#### テーブル・デバイス特性の確定

- init 修飾子を指定しないで dump コマンドを発行し、Backup Server がデバイス・タイプを確定できない場合は、dump コマンドは失敗します。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。

#### Backup Server

- Adaptive Server と同じマシン上で稼動する Backup Server が必要です。また、Backup Server は master.syssservers テーブルにリストされている必要があります。このエントリはインストール時またはアップグレード時に作成されます。削除しないでください。

- 別のマシンにバックアップ・デバイスがある場合は、ネットワーク間でダンプをするために、リモート・マシンにも **Backup Server** をインストールしなければなりません。

#### ダンプ・ファイル

- `init` オプションを使用してデータベースをダンプすると、テープまたはディスクの既存のファイルがすべて上書きされます。
- テープ・デバイスへのダンプを 2 つ実行した場合、その両方のダンプに同じファイル名 (`FILENAME` パラメータで指定) を使用すると、**Adaptive Server** では、2 つ目のダンプをアーカイブ・デバイスに追加します。**Adaptive Server** では、指定したファイル名のダンプ・イメージの最初のインスタンスの位置を確認し、このイメージをリストアするため、2 つ目のダンプをリストアできません。**Adaptive Server** では、同じ名前の後続のダンプ・イメージが検索されません。
- **Backup Server** は、`with notify` 句で指定されたロケーションにダンプ・ファイル名を送信します。オペレータは、バックアップ・テープを保管する前に、データベース名、ファイル名、日付、および他の関連情報をテープにラベル付けする必要があります。識別ラベルのないテープをロードする場合は、`with headeronly` オプションと `with listonly` オプションを使用して内容を確認してください。

#### ファイル名とアーカイブ名

- ダンプ・ファイル名によって、ダンプされたデータベースとダンプが実行された時刻が識別されます。ただし構文内の `file_name` は、ディスクにダンプするか、または **UNIX** のテープにダンプするかによって、意味が異なります。

```
file = file_name
```

ディスクにダンプする場合は、ディスク・ファイルのパス名がディスク・ファイルの名前にもなります。

**UNIX** のテープにダンプする場合は、パス名はファイル名ではありません。ファイル交換の **ANSI** 準拠フォーマットには、**HDR1** ラベル内にファイル名のフィールドがあります。テープが **ANSI** 仕様に準拠している場合、ラベル内のこのフィールドでファイル名が識別されます。**ANSI** 仕様は、これらのラベルをテープだけに適用します。ディスク・ファイルには適用されません。

これによって、次の 2 つの問題が発生します。

- UNIX はテープ・ファイル名の ANSI 規則に従わない。したがって、UNIX では、テープのデータがラベル付けされていないとみなされる。このデータは複数のファイルに分けることができるが、各ファイルに名前は付与されない。
- Backup Server では、アーカイブ情報の格納に ANSI のテープ・ラベルが使用され、ANSI の意味が無効になる。したがって、ディスク・ファイルには ANSI ラベルも付加される。アーカイブ名がディスク・ファイル内に保管されるためである。

*filename* の意味は、実行するダンプの種類によって変わります。たとえば次のような構文があるとします。

```
dump database database_name to 'filename' with file='filename'
```

- つ目の *filename* は、ファイルを表示するために入力するパス名です。
- 2 つ目の *filename* は実際のアーカイブ名です。これはアーカイブ内の HDR1 ラベルに格納される名前です。ユーザは dump コマンドまたは load コマンドの `file=filename` パラメータで指定できます。

アーカイブ名が指定されると、サーバはデータベースのロード中にこの名前を使用して、選択されたアーカイブの保管場所を決定します。

アーカイブ名が指定されていない場合、サーバは最初に検出するアーカイブをロードします。

いずれの場合も、HDR1 ラベルに格納される名前は `file=archivename` によって指定され、後続の load はこの名前を使用して、正しいデータを参照していることを確認します。

アーカイブ名が指定されていない場合、dump がアーカイブ名を作成し、load は最初に検索された名前を使用します。

to 'filename' 句内の *filename* の意味は、ダンプがディスクとテープのどちらに対して行われるかによって異なります。

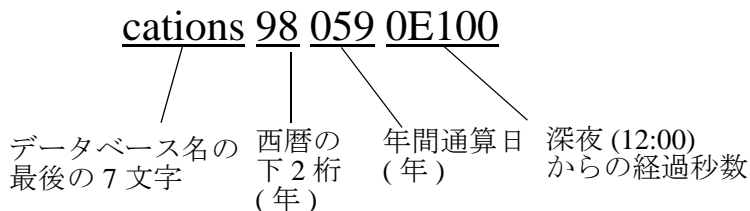
- テープへのダンプの場合、'filename' はテープ・デバイスの名前です。
- ディスクへのダンプの場合は、ディスク・ファイルの名前になります。

ディスク・ダンプの場合に 'filename' が完全パスでなければ、サーバの現在作業中のディレクトリがファイル名の前に付加されます。

- テープにダンプするときファイル名を指定していない場合、Backup Server で次の要素を連結してデフォルトのファイル名が作成されます。
  - データベース名の最後の 7 文字
  - 西暦の下 2 桁
  - 3 桁の年間通算日 (1 ~ 366)
  - ダンプ・ファイルが作成された時刻の 16 進数値

たとえば、*cations980590E100* というファイルには、1998 年の 59 日目に作成された *publications* データベースのコピーが含まれます。

図 1-4: テープへのデータベース・ダンプのファイル命名規則



#### ボリューム名

- ダンプ・ボリュームは、ANSI テープ・ラベル標準に準拠してラベル付けされます。ラベルには、論理ボリューム番号とストライプ・セット内のデバイスの位置が含まれています。
- Backup Server はロード中にテープ・ラベルを使用して、ボリュームが正しい順番でマウントされているかどうかを確認します。これにより、ダンプ時に使用したデバイス数より少ないデバイスからロードできます。

---

**注意** ネットワーク間でダンプおよびロードを実行する場合は、各オペレーションに同じ数のストライプ・デバイスを指定しなければなりません。

---

#### ダンプ・ボリュームの変更

- (UNIX システム) テープ容量に到達すると、Backup Server がボリューム変更を要求します。別のボリュームをマウントした後、オペレータは Backup Server に通知するために、Backup Server と通信できる Adaptive Server 上で、*sp\_volchanged* システム・プロシージャを実行します。

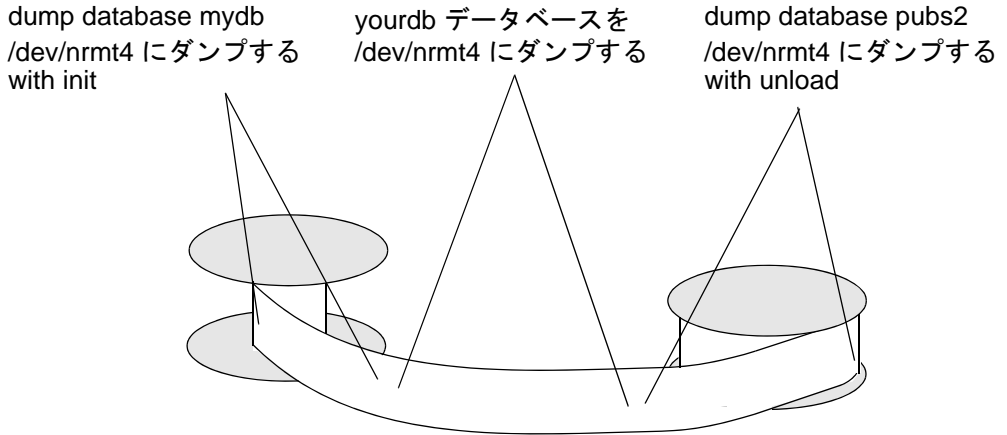
- Backup Server は、現在マウントされているボリュームで問題を検出すると、クライアントかクライアントのオペレータ・コンソールのいずれかにメッセージを送信して、ボリューム変更を要求します。オペレータは `sp_volchanged` システム・プロシージャを使用して、これらのメッセージに応答できます。

#### ボリュームへの追加または上書き

- デフォルト (`noinit`) では、Backup Server は大容量テープ・メディアを効率的に使用して、一連のダンプを 1 つのテープ・ボリュームに書き込みます。データは、前回のテープ終了マークの後に追加されます。新しいダンプは、マルチボリューム・ダンプの最後のボリュームだけに追加できます。Backup Server は、テープに書き込む前に、最初のファイルが期限切れになっていないかどうかを確認します。テープに Sybase 以外のデータがある場合、Backup Server では、重要な情報が破壊される可能性を避けるために、そのデータを受け付けません。
- ボリュームを再初期化するには、`init` オプションを使用します。`init` を指定すると、Backup Server では既存の内容が上書きされます。この上書きは、テープに Sybase 以外のデータがある場合や最初のファイルの期限が切れていない場合、またはテープに ANSI アクセス制限がある場合でも実行されます。
- [図 1-5](#) は、次のコマンドを使用して 3 つのデータベースを 1 つのボリュームにダンプする方法を示しています。
  - `init` を使用して、最初のダンプのテープを初期化する。
  - `noinit` (デフォルト) を使用して、後続のダンプを追加する。
  - `unload` を使用して、最後のダンプ後にテープを巻き戻してアンロードする。



図 1-5: 複数データベースの単一ボリュームへのダンプ



### 32 ビット OS から 64 ビット OS へのダンプ

32 ビット版の Adaptive Server から同じプラットフォーム上の 64 ビット版の Adaptive Server へのデータベース・ダンプ、またその逆方向のダンプは、完全な相互互換性があります。

### デバイスがミラーされているデータベースのダンプ

- `dump database` の開始時に、Adaptive Server は、Backup Server にすべてのデータベースのプライマリ・デバイス名と、ログ・デバイス名を渡します。プライマリ・デバイスのミラーリングが解除されている場合、Adaptive Server は代わりにセカンダリ・デバイス名を渡します。Backup Server によるデータ転送が完了する前に、指名されたデバイスに障害が発生すると、Adaptive Server はダンプをアボートします。
- `dump database` の進行中に、指定されたデータベース・デバイスのいずれかのミラーリングを解除しようとする、Adaptive Server からメッセージが表示されます。`disk unmirror` コマンドを実行しているユーザは、ダンプの完了後までの間、ダンプをアボートするか、`disk unmirror` を遅延させることができます。

### パフォーマンスに関する注意

`dataserver` 内のインデックスは最適な検索パスを与えるように設計されているため、インデックス・ローはテーブルのデータ・ローに高速にアクセスできるように並べ替えられます。ユーザ・テーブルへの高速なアクセスを実現するために、ロー識別子 (RID) を持つインデックス・ローはバイナリとして扱われます。

同じアーキテクチャのプラットフォーム内であればインデックス・ローの順序は相変わらず有効で、与えられた選択基準に対する検索順序はいつものパスをとります。しかし、インデックス・ローが異なるアーキテクチャ間で変換されると最適化が行われた順序が無効になり、プラットフォーム間のダンプとロードでユーザ・テーブルに無効なインデックスが作成されます。

ビッグ・エンディアンからリトル・エンディアンというように、異なるアーキテクチャのデータベース・ダンプをロードすると、一部のインデックスがサスペクトとマークされます。

- APL テーブルのノンクラスタード・インデックス
- DOL テーブルのクラスタード・インデックス
- DOL テーブルのノンクラスタード・インデックス

ターゲット・システムのインデックスを修復するには、アーキテクチャの異なるダンプのロード後、次のいずれかを行うことができます。

- すべてのインデックスを削除して作り直す。
- `sp_post_xpload` を使用する。『リファレンス・マニュアル：プロシージャ』の「システム・プロシージャ」を参照してください。

大規模なテーブルのインデックスの再作成は処理に時間がかかる場合があります。`sp_post_xpload` は、インデックスを検証し、無効なインデックスを削除し、削除したインデックスを作り直すまでの一連の操作を単一のコマンドで実行します。

`sp_post_xpload` を使用すると、インデックスを 1 つずつ削除して作り直す方法よりも時間がかかることがあります。10GB を超えるようなデータベースに対しては、インデックスを削除して作り直す方法を使用することをおすすめします。

#### アーカイブ・データベースの圧縮ダンプ

アーカイブ・データベースの圧縮ダンプを使用するには、次の操作を行う必要があります。

- `dump database` コマンドまたは `dump tran` コマンドの `with compression = <compression level>` オプションを使用して、圧縮ダンプを作成。

- アーカイブ・データベースにアクセスするためのメモリ・プールを作成。

---

**注意** `compress::` を使用して生成されたダンプは、アーカイブ・データベースにロードできません。そのため、この章で圧縮に言及している場合は、`with compression = <compression level>` オプションを使用して生成されたダンプのことを指しています。

従来のデータベースでこの圧縮オプションを使用する場合、ダンプに関する互換性の問題は発生しません。

---

### 圧縮ダンプの互換性の問題

`with compression = compression_level` オプションを使用して生成された圧縮ダンプのフォーマットが変更されました。バージョン 15.0 ESD #2 以降の Backup Server では、旧バージョンと異なるフォーマットの圧縮ダンプが生成されます。そのため、次の結果が生じます。

- Backup Server バージョン 15.0 ESD #2 以降を使用して作成された圧縮済みダンプは、Backup Server バージョン 15.0 ESD #2 以降を使用した 15.0 ESD #2 以前のインストールにだけロードできる。
- 15.0 ESD #2 以前のインストールを使用中にダンプをアーカイブ・データベースに使用する場合、Backup Server バージョン 15.0 ESD #2 以上を使用して圧縮済みデータベース・ダンプを作成する。

---

**注意** 15.0 ESD #2 の Backup Server はダンプとロードのいずれにも使用できます。

---

### 暗号化カラムと `dump database`

`dump` と `load` は、暗号化カラムの暗号テキストに対して実行されます。この動作により、ディスク上で暗号化カラムのデータが暗号化された状態を保つことが保証されます。

`dump` と `load` はデータベース全体に対応します。デフォルト・キーと、同じデータベースに作成されているその他のキーは、それらが対応するデータと一緒にダンプおよびロードされます。

キーとそのキーで暗号化するカラムが別のデータベースにある場合は、次の方法を推奨します。

- 暗号化カラムを含むデータベースをダンプするときは、対応するキーが作成されたデータベースもダンプします。これは、最後のダンプ以降に新しいキーが追加されている場合に必要です。

- 暗号化キーを含むデータベースをダンプするときは、そのキーで暗号化されたカラムを含むすべてのデータベースをダンプします。これにより、暗号化されたデータと対応するキーの同期が保たれます。
- 暗号化キーを含むデータベースおよび暗号化カラムを含むデータベースをロードした後、両方のデータベースを同時にオンラインにします。

キーのデータベースで暗号化カラムのメタデータに依存関係が存在するため、キー データベースを異なる名前データベースにロードする場合は、次の手順に従います(キーと同じデータベースにデータを格納する場合は、この手順に従う必要はありません)。

- 1 暗号化カラムを含むデータベースをダンプする前に、**alter table** を使用してデータを復号化します。
- 2 キーおよび暗号化カラムを含むデータベースをダンプします。
- 3 データベースをロードした後、**alter table** を使用して、新しい名前データベースのキーでデータを再暗号化します。

暗号化キーと暗号化カラムの一貫性の問題は、データベース間の参照整合性の問題と似ています。詳細については、『システム管理ガイド』の「データベース間の制約とデータベースのロード」を参照してください。

#### dump database と Tivoli Storage Manager のサポート

TSM がサイトでサポートされている場合のバックアップの作成の詳細については、Tivoli Storage Manager に関するユーザ向けマニュアル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

#### パーミッション

**create database** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、データベース所有者であるか、データベースに対する <b>dump database</b> 権限または <b>own database</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、次の役割が付与されたユーザであることが必要。 <ul style="list-style-type: none"> <li>• <b>sa_role</b>、または</li> <li>• <b>replication_role</b>、または</li> <li>• <b>oper_role</b></li> </ul>

#### 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
34	dump	dump database	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

マニュアル 『システム管理ガイド』の「ユーザ・データベースのバックアップとリストア」

コマンド [dump transaction](#), [load database](#), [load transaction](#)

システム・プロシージャ [sp\\_addthreshold](#), [sp\\_addumpdevice](#), [sp\\_dropdevice](#), [sp\\_droptreshold](#), [sp\\_helpdb](#), [sp\\_helpdevice](#), [sp\\_helpthreshold](#), [sp\\_hidetext](#), [sp\\_logdevice](#), [sp\\_spaceused](#), [sp\\_volchanged](#)

## dump transaction

**説明** dump transaction コマンドが他の dump database と同時に実行されない場合に、トランザクション・ログのコピーを作成し、ログの非アクティブ部分を削除します。

サイトで Tivoli がライセンスされている場合の dump transaction 構文については、Tivoli Storage Manager (TSM) 構文を参照してください。

**構文** 定期的なログのダンプを作成する場合 :

```
dump tran[saction] database_name
to [compress::[compression_level::]]stripe_device
[at backup_server_name]
[density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
dumpvolume = volume_name,
file = file_name]
[stripe on [compress::[compression_level::]]stripe_device
[at backup_server_name]
[density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
dumpvolume = volume_name,
file = file_name]]
[[stripe on [compress::[compression_level::]]stripe_device
[at backup_server_name]
[density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
dumpvolume = volume_name,
file = file_name]]...]
[with {
density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
compression = compress_level,
dumpvolume = volume_name,
file = file_name,
[dismount | nodismount],
[nounload | unload],
retaindays = number_days,
[noinit | init],
notify = {client | operator_console},
standby_access}]
```

バックアップ・コピーを作成しないでログをトランケートする場合 :

```
dump tran[saction] database_name
with truncate_only
```

容量が満杯になったログをトランケートする場合。ログの内容が失われるため、最後の手段としてのみ使用してください。

```
dump tran[saction] database_name
with no_log
```

データベース・デバイスで障害が発生した後にログをバックアップする場合：

```
dump tran[saction] database_name
to [compress::[compression_level:]]stripe_device
[at backup_server_name]
[density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
dumpvolume = volume_name,
file = file_name]
[stripe on [compress::[compression_level:]]stripe_device
[at backup_server_name]
[density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
dumpvolume = volume_name,
file = file_name]]
[[stripe on [compress::[compression_level:]]stripe_device
[at backup_server_name]
[density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
dumpvolume = volume_name,
file = file_name]]...]
[with {
density = density_value,
blocksize = number_bytes,
capacity = number_kilobytes,
compression = compress_level
dumpvolume = volume_name,
file = file_name,
[dismount | nodismount],
[nounload | unload],
retaindays = number_days,
[noinit | init],
no_truncate,
notify = {client | operator_console}}]
```

Tivoli Storage Manager からバックアップ サービスが提供される状況でトランザクション ログをコピーする場合：

```
dump transaction database_name
to "syb_tsm::object_name"
[blocksize = number_bytes]
[stripe on "[syb_tsm::]object_name"
```

```
[blocksize = number_bytes]...]  
[with {  
    blocksize = number_bytes,  
    compression = compress_level,  
    passwd = password,  
    [noinit | init],  
    notify = {client | operator_console},  
    verify[ = header | full]  
}]
```

ダンプ設定ファイルで指定された設定に従ってトランザクションをダンプする場合：

```
dump transaction database_name  
using config = configuration_name  
[with {  
    verify[ = header | full]  
}]
```

#### パラメータ

##### *database\_name*

データのコピー元であるデータベースの名前です。データベース名は、リテラルやローカル変数として指定できます。または、ストアド・プロシージャに対するパラメータとしても指定できます。

##### *configuration\_name*

ユニークなダンプ設定名。ダンプ設定で指定されたパラメータ値がダンプ操作の実行に使用されます。

コマンドで他のパラメータが明示的に指定されていると、ダンプ設定での指定値が上書きされます。

ダンプ設定を使用する場合は、コマンドのパラメータとしてストライプ・ディレクトリを指定することはできません。Adaptive Serverによって、ダンプ設定によって指定されたストライプ・ディレクトリにダンプ・ファイルが作成されます。ダンプ・ファイル名に使用される命名規則は次のとおりです。

*Database\_Name.Dump\_Type.Date-Timestamp.StripeID*

##### *compress::compression\_level*

圧縮レベルを 0～9、100、または 101 の数字で指定します。1 桁の圧縮レベルでは、0 は圧縮なし、9 は圧縮の最高レベルです。圧縮レベル 100 および 101 は、より高速で効率的な圧縮モードです。100 はより高速の圧縮、101 はより効率的な圧縮を行います。*compression\_level* を指定しない場合、ダンプは圧縮されません。



`compress` オプションの詳細については、『システム管理ガイド』の「ユーザ・データベースのバックアップとリストア」を参照してください。

---

**注意** `compression = compress_level` オプションを使用すると、ダンプ・ファイルをローカル・マシンとリモート・マシンの両方で圧縮できます。ローカル・マシンでしかダンプ・ファイルを圧縮できない `compress::compression_level` オプションとは、この点が異なります。

Adaptive Server バージョン 15.0 以降、ネイティブの `compression = compression_level` 構文がサポートされ、この構文の使用が推奨されています。

---

#### `truncate_only`

ログの非アクティブ部分を、バックアップ・コピーを作成しないで削除します。ログ・セグメントが、データ・セグメントとは異なるデバイス上にないデータベースで使用します。ダンプ・デバイスまたは Backup Server の名前は指定しないでください。

#### `no_log`

バックアップ・コピーを作成することなく、またプロシージャをトランザクション・ログに記録することなく、ログの非アクティブ部分を削除します。`no_log` の使用は、ログ領域がまったくなくなり、通常の `dump transaction` コマンドを実行できない場合のみに限定してください。`no_log` は、最後の手段として、`dump transaction with truncate_only` が失敗した後に一度だけ使用してください。

#### `to stripe_device`

データのダンプ先のデバイス名です。ダンプ・デバイス名の指定時に使用する形式については、「[ダンプ・デバイスの指定](#)」(437 ページ)を参照してください。

#### `at backup_server_name`

Backup Server の名前です。デフォルトの Backup Server にダンプする場合は、このパラメータを指定しないでください。このパラメータは、ネットワークを介してリモート Backup Server にダンプする場合にのみ指定してください。このオプションを使用して、リモート Backup Server を 32 個まで指定できます。ネットワークを介してダンプする場合、ダンプ・デバイスの接続先のマシンで稼動しているリモート Backup Server の `network name` を指定します。`interfaces` ファイルを使用するプラットフォームでは、`backup_server_name` は `interfaces` ファイル内になければなりません。

**density = density\_value**

テープ・デバイスのデフォルトの記録密度を上書きします。有効な密度は、800、1600、6250、6666、10000、38000 です。すべての値がどのテープ・ドライブに対しても有効というわけではありません。使用しているテープ・ドライブにあった記録密度を選択してください。

**blocksize = number\_bytes**

ダンプ・デバイスのデフォルトのブロック・サイズを上書きします。ブロック・サイズは、少なくとも 1 データベース・ページ (ほとんどのシステムでは 2048 バイト) の大きさで、データベース・ページ・サイズの倍数でなければなりません。

---

**注意** 可能なかぎり、デフォルトのブロック・サイズを使用してください。これがシステムに最適なブロック・サイズです。

---

**capacity = number\_kilobytes**

デバイスが単一のテープ・ボリュウムに書き込むことのできるデータの最大量です。収容データ量は、少なくとも 5 データベース・ページ分で、デバイスに推奨される最大値より少ない値にしてください。

処理能力計算の一般的な規則は、デバイス製造者仕様の最大能力の 70% を使用することです。30% はレコードのギャップ、テープ・マークなどのオーバーヘッドのための余裕分として空けておきます。この規則はほとんどの場合に当てはまりますが、ベンダ間およびデバイス間でオーバーヘッドが異なるため、すべての場合に当てはまるわけではありません。

テープ終了マーカを確実に検出できない UNIX プラットフォームでは、テープへのダンプが可能なキロバイト数を指示します。物理パス名として指定したダンプ・デバイスには、**capacity** を指定してください。ダンプ・デバイスを論理デバイス名として指定した場合、容量が指定されていない限り、**Backup Server** では、**sysdevices** システム・テーブルに格納されている *size* パラメータが使用されます。

**compression = *compress\_level***

圧縮レベルを 0 ~ 9、100、または 101 の数字で指定します。1 桁の圧縮レベルでは、0 は圧縮なし、9 は圧縮の最高レベルです。圧縮レベル 100 および 101 は、より高速で効率的な圧縮モードです。100 はより高速の圧縮、101 はより効率的な圧縮を行います。*compression\_level* を指定しない場合、ダンプは圧縮されません。

---

**注意** ネイティブの "**compression = *compress\_level***" オプションを使用し、古い "**compress::*compression\_level***" オプションは使用しないことを推奨します。ネイティブのオプションを使用すると、ローカルとリモートの両方のダンプを圧縮できます。このオプションで作成されるダンプは、ロード中に圧縮レベルを示します。古いオプションは、古いアプリケーションとの互換性のために維持されています。

---

**dumpvolume = *volume\_name***

ボリュームに割り当てる名前を指定します。*volume\_name* の最大長は 6 文字です。既存のダンプへの上書き、新しいテープへのダンプ、または内容が認識できないテープへのダンプを行う場合、Backup Server は ANSI テープ・ラベルに *volume\_name* を書き込みます。load transaction コマンドはラベルを確認し、誤ったボリュームがロードされている場合は、エラー・メッセージを生成します。

**stripe on *stripe\_device***

追加のダンプ・デバイスです。to *stripe\_device* 句で指定したデバイスを含め、32 個までのデバイスを使用できます。Backup Server はログをほぼ等分し、各分割部分を異なるデバイスに送信します。ダンプはすべてのデバイスで同時に行われます。したがって、ダンプに必要な時間が減少し、ダンプ中のボリュームの交換が少なくなります。「[ダンプ・デバイスの指定](#)」(437 ページ) を参照してください。

**dismount | nodismount**

(論理的なマウント解除をサポートするプラットフォームで) テープをマウントしたままにするかどうかを指定します。デフォルトでは、ダンプの完了時に、ダンプに使用したすべてのテープのマウントが解除されます。追加のダンプまたはロードに引き続きテープを使用できるようにするには、**nodismount** を使用します。

**nounload | unload**

ダンプの完了後にテープを巻き戻すかどうかを指定します。デフォルトでは、テープは巻き戻されません。したがって、同じテープ・ボリュームにダンプを追加できます。マルチダンプ・ボリュームに追加される最後のダンプ・ファイルには、**unload** を指定します。この指定によって、ダンプの完了時にテープの巻き戻しおよびアンロードが行われます。

**retaindays = number\_days**

(UNIX プラットフォームの場合) ダンプが上書きされないように Backup Server が保護する日数を指定します。指定した期限が切れる前にダンプを上書きしようとする、期限内のボリュームに上書きを行う前に、Backup Server が確認を要求します。

---

**注意** このオプションは、ディスク、1/4 インチ・カートリッジ、および単一ファイル・メディアに対して有効です。マルチファイル・メディアでは、最初のボリューム以外のすべてのボリュームで有効です。

---

*number\_days* には、正の整数を指定してください。また、すぐにダンプを上書きできるようにする場合は 0 を指定してください。*retaindays* 値を指定しない場合、Backup Server は、*sp\_configure* によってサーバーワイドに設定される **tape retention in days** 値を使用します。

**noinit | init**

既存のダンプ・ファイルにダンプを追加するか、またはダンプによってテープ・ボリュームの再初期化(上書き)を行うかどうかを指定します。デフォルトでは、Adaptive Server は前回のテープ終了マークの後からダンプを追加するため、追加のデータベースを同じボリュームにダンプできます。新しいダンプは、マルチボリューム・ダンプの最後のボリュームの末尾にしか追加できません。内容を上書きするには、テープにダンプする最初のデータベースで **init** を使用します。

Backup Server によってテープ設定ファイルのテープ・デバイス特性が保管または更新されるようにするには、**init** を使用します。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。

**file = file\_name**

ダンプ・ファイルの名前です。名前は 17 文字までで、オペレーティング・システムのファイル命名規則に従っていなければなりません。ファイル名を指定しないと、Backup Server はデフォルトのファイル名を作成します。「**ダンプ・ファイル**」(439 ページ)を参照してください。

**no\_truncate**

**master** データベースのトランザクション・ログに対するポインタを使用して、データベースのデータ・セグメントを含むディスクにアクセスできない場合でも、トランザクション・ログをダンプします。トランザクション・ログがダメージを受けていないデバイスに常駐している場合、および **master** データベースとユーザ・データベースが異なる物理デバイスに常駐している場合は、**with no\_truncate** オプションで、最新のログをリカバリできます。

**dump tran with no\_truncate** を使用する場合は、その後ろに他の **dump tran** ではなく、**dump database** を続けます。**no\_truncate** オプションを使用して生成したダンプをロードすると、その後のダンプをロードできなくなります。

**notify = {client | operator\_console}**

デフォルトのメッセージ送信先を上書きします。

- オペレータ端末機能を提供するオペレーティング・システムでは、ボリューム交換メッセージは、**Backup Server** が実行されているマシンのオペレータ端末に常に送信されます。**dump database** を開始した端末セッションに **Backup Server** の他のメッセージを送信するには、**client** を使用します。
- オペレータ端末機能が備わっていないオペレーティング・システム (UNIX など) では、**dump database** を開始したクライアントにメッセージが送信されます。**Backup Server** が実行されている端末にメッセージを送信するには、**operator\_console** を使用してください。

**with standby\_access**

完了したトランザクションだけをダンプするように指定します。トランザクションが完了したポイントの中で検索可能な最も遠いポイントまで、またアクティブなトランザクションが他にないポイントまで、ダンプが続行します。

**syb\_tsm**

**Backup Server** と **TSM** 間の通信を有効にする **libsyb\_tsm.so** モジュールを起動するキーワードです。

**object\_name**

**TSM** サーバのバックアップ・オブジェクトの名前です。

*configuration\_name*

ユニークなダンプ設定名。ダンプ設定で指定されたパラメータ値がダンプ操作の実行に使用されます。

コマンドで他のパラメータが明示的に指定されていると、ダンプ設定での指定値が上書きされます。

ダンプ設定を使用する場合は、コマンドのパラメータとしてストライプ・ディレクトリを指定することはできません。Adaptive Serverによって、ダンプ設定によって指定されたストライプ・ディレクトリにダンプ・ファイルが作成されます。ダンプ・ファイルの名前には、次の命名規則が使用されます。

*Database Name.Dump Type.Date-Timestamp.StripeID*

**例 1** `init` オプションが指定されていないため、トランザクション・ログが、テープのファイルに追加されてダンプされます。

```
dump transaction pubs2
to "/dev/nrmt0"
```

**例 2** Backup Server の `REMOTE_BKP_SERVER` を使用して、`mydb` データベースのトランザクション・ログをダンプします。ここではコマンドが2つのダンプ・デバイスを指定しているため、Backup Server は各デバイスにログを約 1/2 ずつダンプします。`init` オプションを使用すると、テープの既存のファイルがすべて上書きされます。`retaindays` オプションは、テープが 14 日間上書きできないように指定します。

```
dump transaction mydb
to "/dev/nrmt4" at REMOTE_BKP_SERVER
stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
with init, retaindays = 14
```

**例 3** `inventory_db` トランザクション・ログ・ファイルから `dev1` デバイスに完了したトランザクションをダンプします。

```
dump tran inventory_db to dev1 with standby_access
```

**例 4** `pubs2` データベースのトランザクションログを TSM バックアップオブジェクト `"demo2.2"` に圧縮レベル 100 でダンプします。

```
dump transaction pubs2 to "syb_tsm::demo2.2"
with compression = 100
```

**例 5** `"dmp_cfg2"` 設定を使用して、データベースをダンプします。ダンプ時に作成されたアーカイブ・ファイルはパスワードで保護されています。

```
dump transaction testdb using config = 'dmp_cfg2'
with passwd = 'my_pass01'
go
```

**例 6** "dmp\_cfg2" 設定を使用し、圧縮レベル 6 で (つまり、"dmp\_cfg2" で指定された圧縮レベルを上書きして)、データベースをダンプします。

```
dump transaction testdb using config = 'dmp_cfg2'
      with compression = 6
go
```

#### 使用法

- sp\_hidetext の後で、プラットフォーム間の dump と load を使用した場合は、すべての隠しオブジェクトを手動で削除して再作成する必要があります。
- 表 1-20 に、データベースとログのバックアップに使用するコマンドおよびシステム・プロシージャを示します。

表 1-20: データベースおよびログのバックアップに使用するコマンド

目的	使用
トランザクション・ログを含む、データベース全体の定期的なダンプを作成する。	dump database
トランザクション・ログの日常ダンプを作成し、非アクティブの部分をトランケートする。dump transaction が dump database と同時に実行される場合、ログの非アクティブ部分はトランケートされない。	dump transaction
データベース・デバイスの障害後、トランザクション・ログをダンプする。	dump transaction with no_truncate
バックアップ・コピーを作成しないでログをトランケートする。 次にデータベース全体をコピーする。	dump transaction with truncate_only dump database
ログの領域不足のために通常のダンプが失敗した場合に、その後でログをトランケートする。 次にデータベース全体をコピーする。	dump transaction with no_log dump database
Backup Server のボリューム変更メッセージに応答する。	sp_volchanged

#### 制限事項

- 物理デバイスのファイル・パスまたはファイル名の最大サイズは 127 文字です。
- null デバイスにはダンプできません (UNIX の場合は、/dev/null)。
- トランザクション内では dump transaction コマンドを使用できません。
- 1/4 インチのカートリッジ・テープを使用している場合、1 本のテープにつきダンプできるデータベースまたはトランザクション・ログは 1 つだけです。
- dump transaction database\_name to を実行するには、まず新しく作成されたデータベースを完全にダンプする必要があります。

- ログを取らないオペレーションが一度データベースで実行されると、`dump transaction database_name to` は使用できません。
- `trunc log on chkpt` データベース・オプションが有効の場合、また、`select into/bulk copy/plsort` を有効にして、`select into` によって、ログを最小限変更する変更をデータベースに加えた後、高速のバルク・コピー操作の後、デフォルトのログを取らない `writetext` 操作の後、および並列ソートの後は、`dump the transaction log` を発行できません。この代わりに `dump database` を使用してください。

---

**警告！** `delete`、`update`、または `insert` コマンドを使用して、ログ・テーブル `syslogs` を変更しないでください。

---

- データベースのログ・セグメントがデータ・セグメントとは別のデバイス上でない場合、`dump transaction` を使用したログのコピーや、トランケートはできません。
- `dump database` または他の `dump transaction` が進行中のデータベース上で、ユーザ・プロシージャまたはスレッショルド・プロシージャから `dump transaction` コマンドを発行すると、最初のコマンドが終了するまで、2つ目のコマンドがスリープします。
- データベースをリストアするには、`load database` を使用して、最新のデータベースのダンプをロードします。次に `load transaction` を使用して後続のトランザクション・ログのダンプを作成順にロードします。
- データベース間の制約を追加または削除した場合や、データベース間の制約を含むテーブルを削除した場合は、影響を受けたデータベースを両方ともダンプしてください。

---

**警告！** これらのデータベースの以前のダンプをロードすると、データベースが破壊される可能性があります。

---

- Sybase ダンプと Sybase 以外のデータ (UNIX アーカイブなど) を同じテープに格納することはできません。
- データベースにオフライン・ページがある場合、`with no_log` または `with truncate_only` のトランザクションはダンプできません。

### `with no_truncate` オプションの使用に関する制限

通常は、次のような場合、Adaptive Server はエラー・メッセージを返します。



- 新しく作成されたデータベースを完全にダンプする前に、`dump transaction database_name to` を実行した場合 :

```
This database has not been dumped since it was
created or upgraded or a transaction dump may have
been loaded using the UNTIL_TIME clause.You must
perform a DUMP DATABASE before you can dump its
transaction log.
```

- 最小限にしかログに記録されないオペレーションをデータベースで実行し、`dump transaction database_name to` を使用した場合 :

```
Dump transaction is not allowed because a
non-logged operation was performed on the database.
Dump your database or use dump transaction with
truncate_only until you can dump your database.
```

詳細については、「完全にリカバリ可能な DDL と `dump transaction`」(467 ページ) を参照してください。

- `dump transaction with truncate_only` を実行後、`dump transaction database_name to` を使用した場合 :

```
DUMP TRANSaction to a dump device is not allowed
where a truncate-only transaction dump has been
performed after the last DUMP DATABASE.Use DUMP
DATABASE instead.
```

`dump transaction database_name to dump_file` コマンド内で、`with no_truncate` オプションを使用しても、Adaptive Server はデータベースの検査を行わないため、これらのエラー・メッセージは返されません。Adaptive Server では、ディスクのクラッシュなどによりデータベースからデータが失われたためにアクセスできなものとみなされます。

ただし、トランザクションをロードしようとする、エラー・メッセージが表示されます。次のエラー・メッセージが表示されて、`load transaction` 処理が失敗することがあります。

```
指定されたファイル 'dump device' はシーケンス外です。current
現在のタイム・スタンプは <x> ですが、ダンプは <y> からです。
```

#### デバイス障害後のログのコピー

- デバイスの障害が発生した後は、`dump transaction with no_truncate` を使用し、トランケートしないでログをコピーします。このオプションが使用できるのは、別のセグメントにログがあり、master データベースにアクセス可能な場合のみです。

- `dump transaction with no_truncate` によって作成されたバックアップは、ログの最新のダンプです。データベースをリストアする際は、このダンプを最後にロードします。

#### 独立したログ・セグメントを持たないデータベースのダンプ

- データベースのログ・セグメントがデータ・セグメントとは別のデバイス上でない場合、`dump transaction with truncate_only` を使用して、バックアップ・コピーを作成しないで、ログからコミットされたトランザクションを削除します。

---

**警告！** `dump transaction with truncate_only` を使用すると、データベースをリカバリする手段がなくなります。確実にリカバリできるようにするために、できるだけ早い時期に `dump database` を実行してください。

---

- `master`、`model` および `sysystemprocs` データベースでは、ログ・セグメントがデータ・セグメントとは異なるデバイス上でないため、`with truncate_only` を使用します。
- また、同じデバイスにトランザクション・ログおよびデータを保管できるデータベースが非常に小さい場合にも、`with truncate_only` を使用できます。
- ミッション・クリティカルなユーザ・データベースのログ・セグメントは、データ・セグメントとは異なるデバイス上に配置してください。`create database` の `log on` 句を使用して、ログ・セグメントが独立したデータベースを作成するか、または `alter database` および `sp_logdevice` を使用して、別のデバイスにログを転送します。

#### 完了トランザクションのみのダンプ

- `with standby_access` オプションを使用して、データベースのウォーム・スタンバイ・サーバとして機能するサーバにロードするトランザクション・ログをダンプします。
- `with standby_access` を使用してトランザクション・ログをダンプすると、以前のトランザクションすべてが完了した、オープン・トランザクションに属する記録がない、最も遠いポイントまでダンプ処理が行われます。
- 複数のトランザクション・ログを順番にロードするとき、ロードとロードの間にデータベースをオンラインの状態にする場合は、常に `dump tran[saction]...with standby_access when` を使用する必要があります。

- `with standby_access` オプションによって作成されたダンプをロードした後は、`online database` コマンドを `for standby_access` オプションと一緒に使用して、データベースをアクセス可能にします。

---

**警告！** トランザクション・ログにオープン・トランザクションが含まれているときに、`with standby_access` オプションを使用しないでそのトランザクション・ログのダンプを行うと、そのログをロードしてデータベースをオンライン化し、その後で以降のトランザクション・ダンプをロードすることはできなくなります。複数のトランザクション・ダンプを連続してロードする場合、元々 `with standby_access` を使用してダンプしたものをロードしてから、または連続したダンプの全体をロードしてからのみ、データベースをオンラインにできます。

---

### ログなしのダンプ

---

**警告！** `dump transaction with no_log` は、ログ領域の不足によってトランザクション・ダンプの通常の方法 (`dump transaction` または `dump transaction with truncate_only`) が失敗した場合にのみ最終的な手段として使用してください。`dump transaction with no_log` を使用すると、データベースをリカバリする手段がなくなります。確実にリカバリできるようにするために、できるだけ早い時期に `dump database` を実行してください。

---

- `dump transaction...with no_log` は、ダンプ・トランザクションのイベントのログを取らないでログをトランケートします。データをコピーしないため、データベース名だけを必要とします。
- `dump transaction...with no_log` は、使用するたびにエラーとみなされ、Adaptive Server のエラー・ログに記録されます。
- ログ・セグメントをデータ・セグメントとは異なるデバイスに配置してデータベースを作成し、トランザクション・ログを頻繁にダンプするラストチャンス・スレッショルド・プロシージャを書き込み、ログとデータベースに十分な領域が割り付けられている状態であれば、`with no_log` を使用する必要はありません。`with no_log` の使用が必要な場合は、ダンプの頻度とログ領域を増やしてください。

### ダンプのスケジューリング

- トランザクション・ログのダンプは動的に実行されます。つまりデータベースがアクティブであってもダンプを実行できます。ただし、この動的なダンプによってシステムの処理速度が多少遅くなる可能性があります。したがって、データベースの更新量が少ないときにダンプを実行してください。
- ユーザ・データベースおよびそのトランザクション・ログを定期的にバックアップするスケジュールを作成してください。
- `dump transaction` が使用する記憶領域と時間は、`dump database` よりも少なくなります。通常、トランザクション・ログのダンプは、データベース・ダンプよりも頻繁に行われます。

### スレッシュホールドを使用した `dump transaction` の自動化

- バックアップ手順を自動化するにはスレッシュホールドを使用します。Adaptive Server のラストチャンス・スレッシュホールドを活用するには、ログ・セグメントをデータ・セグメントとは異なるデバイスに配置してユーザ・データベースを作成します。
- ログ・セグメントの領域がラストチャンス・スレッシュホールドを下回ると、Adaptive Server はラストチャンス・スレッシュホールド・プロシージャを実行します。ラストチャンス・スレッシュホールド・プロシージャに `dump transaction` コマンドを含めると、ログ領域の不足を防ぐことができます。詳細については、「`sp_thresholdaction`」を参照してください。
- `sp_addthreshold` を使用すると、ログ領域をモニタする 2 つ目のスレッシュホールドを追加できます。スレッシュホールドの詳細については、『システム管理ガイド』を参照してください。

### ダンプ・デバイスの指定

- ダンプ・デバイスは、リテラル、ローカル変数、またはストアド・プロシージャへのパラメータとして指定できます。
- ローカル・ダンプ・デバイスは、次のように指定できます。
  - `sysdevices` システム・テーブルの論理デバイス名
  - 絶対パス名
  - 相対パス名

Backup Server は、Adaptive Server の現在の作業ディレクトリを使用して相対パス名を解決します。

- 複数のストライプへのダンプは、テープ・デバイスとディスク・デバイスに対応しています。1つのデバイスに対する複数のダンプは、テープ・デバイスの場合だけにサポートされています。
- ネットワーク間でダンプを行う場合は、ダンプ・デバイスの絶対パス名を指定します。パス名は、Backup Server が起動しているマシンで有効なものでなければなりません。名前に文字、数字、アンダースコア ( \_ ) 以外の文字が含まれる場合は、名前を引用符で囲んでください。
- ダンプ・デバイスでの所有権とパーミッションの問題によって、dump コマンドが使用できないことがあります。sp\_addumpdevice によってデバイスをシステム・テーブルに追加できますが、そのデバイスにダンプできるか、またはダンプ・デバイスとしてファイルを作成できるかどうかは保証されていません。
- それぞれに別のダンプ・デバイスを使用すると、複数のダンプ ( またはロード ) を同時に実行できます。

#### テープ・デバイス特性の確定

init 修飾子を指定しないで dump transaction コマンドを発行したときに、Backup Server がデバイス・タイプを確定できない場合、dump transaction コマンドは失敗します。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。

#### Backup Server

- Backup Server は、Adaptive Server が稼働しているマシン上で実行する必要があります。また、Backup Server は master.syssservers テーブルにリストされている必要があります。このエント리는インストールまたはアップグレード中に作成されます。削除しないでください。
- 別のマシンにバックアップ・デバイスがある場合は、ネットワーク間でダンプをするために、リモート・マシンにも Backup Server をインストールしなければなりません。

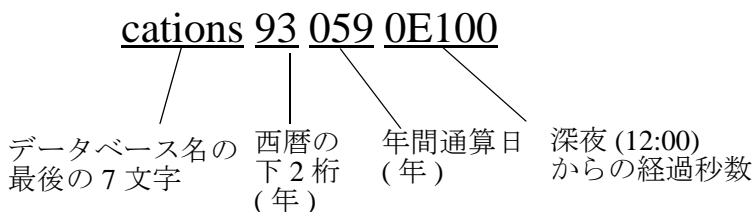
#### ダンプ・ファイル

- init オプションを使用してログをダンプすると、テープまたはディスクの既存のファイルが上書きされます。
- ダンプされたデータベースと、ダンプが行われた時刻は、ダンプ・ファイル名によって識別できます。ファイル名を指定していない場合、Backup Server は次の要素を連結してデフォルトのファイル名を作成します。
  - データベース名の最後の7文字

- 西暦の下2桁
- 3桁の年間通算日 (1 ~ 366)
- ダンプ・ファイルが作成された時刻の16進数値

たとえば、`cations930590E100` ファイルは、1993年の59日目に作成された `publications` データベースのコピーを含みます。

図 1-6: トランザクション・ログ・ダンプのファイルの命名規則



- `Backup Server` は、`with notify` 句で指定されたロケーションにダンプ・ファイル名を送信します。オペレータは、バックアップ・テープを保管する前に、データベース名、ファイル名、日付、および他の関連情報をテープにラベル付けする必要があります。識別ラベルのないテープをロードする場合は、`with headeronly` オプションと `with listonly` オプションを使用して内容を確認してください。

#### ボリューム名

- ダンプ・ボリュームは、ANSI テープ・ラベル標準に準拠してラベル付けされます。ラベルには、論理ボリューム番号とストライプ・セット内のデバイスの位置が含まれています。
- `Backup Server` はロード中にテープ・ラベルを使用して、ボリュームが正しい順番でマウントされているかどうかを確認します。これにより、ダンプ時に使用したデバイス数より少ないデバイスからロードできます。

---

**注意** ネットワーク間でダンプおよびロードを実行する場合は、各オペレーションに同じ数のストライプ・デバイスを指定しなければなりません。

---

### ダンプ・ボリュームの変更

- (UNIX システム) テープ容量がいっぱいになると、Backup Server がボリュームの変更を要求します。別のボリュームをマウントした後、オペレータは Backup Server に通知するために、Backup Server と通信できる Adaptive Server 上で、sp\_volchanged システム・プロシージャを実行します。
- Backup Server は、現在マウントされているボリュームで問題を検出すると (たとえば、誤ったボリュームがマウントされている場合)、クライアントまたはオペレータ・コンソールにメッセージを送ってボリュームの変更を要求します。オペレータは、sp\_volchanged システム・プロシージャを実行して、これらのメッセージに応答します。

### ボリュームへの追加または上書き


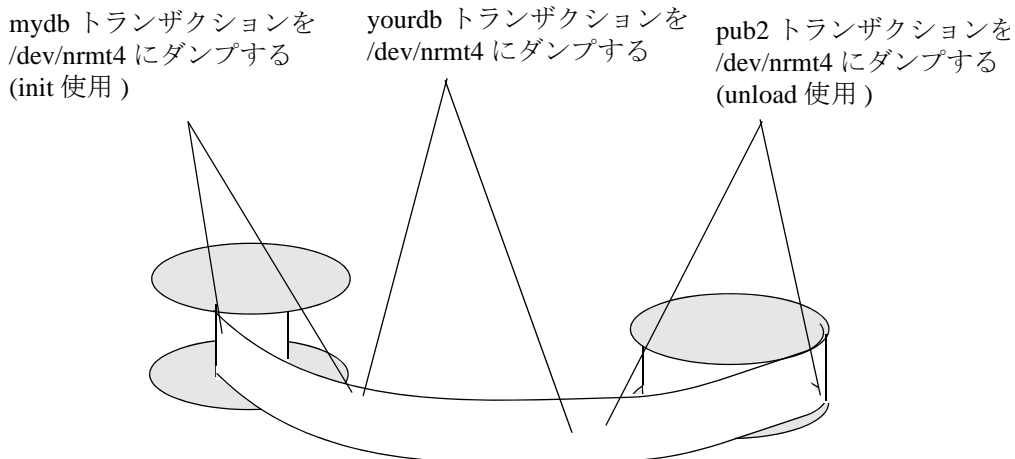
- デフォルト (noinit) の場合、Backup Server は大容量テープ・メディアを効率的に使用して、一連のダンプを 1 つのテープ・ボリュームに書き込みます。データは、前回のテープ終了マークの後に追加されます。新しいダンプは、マルチボリューム・ダンプの最後のボリュームだけに追加できます。Backup Server は、テープに書き込む前に、最初のファイルが期限切れになっていないかどうかを確認します。テープに Sybase 以外のデータがある場合、Backup Server では、重要な情報が破壊される可能性を避けるために、そのデータを受け付けません。
- ボリュームを再初期化するには、init オプションを使用します。init を指定すると、Backup Server では既存の内容が上書きされます。この上書きは、テープに Sybase 以外のデータがある場合や最初のファイルの期限が切れていない場合、またはテープに ANSI アクセス制限がある場合でも実行されます。
-  図 1-7 に、単一ボリュームに 3 つのトランザクション・ログをダンプする方法を示します。次のコマンドを使用します。
  - init を使用して、最初のダンプのテープを初期化する。
  - noinit (デフォルト) を使用して、後続のダンプを追加する。
  - unload を使用して、最後のダンプ後にテープを巻き戻してアンロードする。

図 1-7:3 つのトランザクション・ログの単一ボリュームへのダンプ



## ミラーされたデバイスに格納されたログのダンプ

- `dump transaction` の最初の段階で、Adaptive Server は、各論理ログ・デバイスのプライマリ・デバイス名を Backup Server に渡します。プライマリ・デバイスのミラーリングが解除されている場合、Adaptive Server は代わりにセカンダリ・デバイス名を渡します。Backup Server によるデータ転送が完了する前に、指名されたデバイスが失敗すると、Adaptive Server はダンプをアボートします。
- `dump transaction` の進行中に、指名されているログ・デバイスのミラーリングを解除しようとする、Adaptive Server はメッセージを表示します。`disk unmirror` コマンドを実行しているユーザは、ダンプが完了するまでにダンプをアボートするか、`disk unmirror` を遅延させることができます。
- `dump transaction with truncate_only` と `dump transaction with no_log` は、Backup Server を使用しません。デバイスの失敗または `disk unmirror` コマンドによってミラーリングが解除されても、これらのコマンドは影響を受けません。
- `dump transaction` は、ログ・セグメントだけをコピーします。データオンリー・デバイスが、デバイスの障害または `disk unmirror` コマンドによってミラーリングを解除されても、このオプションは影響を受けません。



## 完全にリカバリ可能な DDL と dump transaction

15.7 より前のバージョンの Adaptive Server で、一部の操作は最小限のログしか記録されません。このため最小限のログ記録操作後は `dump transaction` を使用できず、この制約による影響を次のように受けます。

- 大容量データベース (VLDB) のインストールのリカバリ性とオペレーション面でのスケーラビリティ。この場合、`dump database` に多大な時間がかかる可能性があります。
- データベースを最新のものにリカバリできるかどうか。最小限にしかログに記録されないオペレーションをサーバ障害から完全にリカバリできる場合でも、データ・デバイスの故障やデータベースの破損の際には、トランザクション・ダンプが最後に成功した後で加えられた変更は失われる可能性があります。最小限にしかログに記録されないオペレーションの後は、`dump tran with no_truncate` を使用してログをダンプし、ダンプされたトランザクション・ログを使用してデータベースをリカバリする操作を行えなくなります。
- `dump transaction` を使用してから `load tran with until_time` を使用しても、データベースを特定の時点の状態にリストアできません。

Adaptive Server 15.7 以降では、`dump transaction` を使用することで、以前のバージョンの Adaptive Server では最小限にしかログに記録されなかった以下のオペレーションを完全にリカバリできます。

- `select into` (プロキシ・テーブルへの挿入を含む)
- データ移動を要する `alter table` コマンド
- `reorg rebuild`

デフォルトでは最小限にしかログに記録されないコマンドのフル・ログをとるには、`sp_dboption` を master データベースで使用します。

## dump transaction と Tivoli Storage Manager

TSM がサイトでサポートされている場合のバックアップの作成の詳細については、Tivoli Storage Manager に関するユーザ向けマニュアル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

## パーミッション

`dump transaction` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

細密なパーミッションが有効の場合、データベース所有者であるか、データベースに対する `dump database` 権限または `own database` 権限を持つユーザであることが必要。

細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、次の役割が付与されたユーザであることが必要。 <ul style="list-style-type: none"> <li>• sa_role、または</li> <li>• replication_role、または</li> <li>• oper_role</li> </ul>
---------------	--

**監査** sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
35	dump	dump transaction	<ul style="list-style-type: none"> <li>• Roles – 現在のアクティブな役割</li> <li>• Keywords または options – NULL</li> <li>• Previous value – NULL</li> <li>• Current value – NULL</li> <li>• Other information – NULL</li> <li>• Proxy information – set proxy が有効な場合は元のログイン名</li> </ul>

**参照** マニュアル 『システム管理ガイド』の「ユーザ・データベースのバックアップとリストア」

コマンド [dump database](#), [load database](#), [load transaction](#), [online database](#)

システム・プロシージャ [sp\\_addumpdevice](#), [sp\\_dboption](#), [sp\\_dropdevice](#), [sp\\_helpdevice](#), [sp\\_hidetextsp\\_logdevice](#), [sp\\_volchanged](#)

## execute

説明	プロシージャを実行したり、Transact-SQL コマンドを動的に実行します。
構文	<pre>[exec[ute]] [@return_status =]   [[[server .]database.]owner.]procedure_name[;number]   [[@parameter_name =] value      [@parameter_name =] @variable [output]   [, [@parameter_name =] value      [@parameter_name =] @variable [output]...]]   [with recompile]</pre> <p>または</p> <pre>exec[ute] ("string"   char_variable   [+ "string"   char_variable]...)</pre>
パラメータ	<p><b>execute   exec</b></p> <p>ストアド・プロシージャまたは拡張ストアド・プロシージャ (ESP) の実行に使用します。このキーワードは、バッチ内に複数の文がある場合に必要です。</p> <p><b>execute</b> は、Transact-SQL を含む文字列の実行にも使用します。</p> <p><b>@return_status</b></p> <p>ストアド・プロシージャのリターン・ステータスを保管するオプションの整数変数です。<b>@return_status</b> は、<b>execute</b> 文で使用する前に、バッチまたはストアド・プロシージャの中で宣言してください。</p> <p><b>server</b></p> <p>リモート・サーバの名前です。別の <b>Adaptive Server</b> でプロシージャを実行するには、そのサーバを使用するパーミッションと、そのデータベースでプロシージャを実行するパーミッションが必要です。サーバ名だけを指定してデータベース名を指定しなければ、<b>Adaptive Server</b> はデフォルトのデータベースでプロシージャを探します。</p> <p><b>database</b></p> <p>データベース名です。プロシージャが別のデータベースにある場合に指定します。<b>database</b> のデフォルト値は現在のデータベースです。別のデータベースでプロシージャを実行するには、そのデータベースの所有権か、そのデータベースにおける実行パーミッションが必要です。</p>

**owner**

プロシージャの所有者名です。データベースに同じ名前のプロシージャが複数ある場合に区別するために指定します。**owner**のデフォルト値は現在のユーザです。データベース所有者がプロシージャを所有している場合や、自分がプロシージャを所有している場合には、所有者名を省略できます。

**procedure\_name**

**create procedure** で定義するプロシージャの名前です。

**number**

同じ名前プロシージャをグループ化するための任意の整数です。この整数を使用すると、1つの **drop procedure** 文によってこれらのプロシージャをまとめて削除できます。同じアプリケーションで使用するプロシージャは、この方法でグループ化されます。たとえば、**orders** アプリケーションで使用するプロシージャが **orderproc;1**、**orderproc;2** などと指定されている場合に、次の文によってグループ全体を削除できます。

```
drop proc orderproc
```

プロシージャをグループ化すると、グループ内のプロシージャを個別に削除することはできません。たとえば、次の文は使用できません。

```
drop procedure orderproc;2
```

**parameter\_name**

**create procedure** に定義されている、プロシージャに対する引数名です。パラメータ名の前に **@** 記号を付けなければなりません。

“**@parameter\_name = value**”の形式を使用した場合、パラメータ名および定数は **create procedure** で定義している順序で指定する必要はありません。ただし、パラメータにこの形式を使用する場合は、後続のすべてのパラメータにもこの形式を使用しなければなりません。

**value**

プロシージャに対するパラメータまたは引数の値です。

“**@parameter\_name = value**”の形式を使用しない場合、パラメータ値は、**create procedure** に定義されている順序で指定してください。

**@variable**

リターン・パラメータの保管に使用する変数名です。

**output**

ストアド・プロシージャがリターン・パラメータを返すように指示します。ストアド・プロシージャ内の対応するパラメータも、**output** キーワードで作成されている必要があります。

**output** キーワードは、**out** と省略できます。

**with recompile**

新しいプランのコンパイルを強制的に行います。指定するパラメータが通常と異なる場合、またはデータに多くの変更が加えられている場合は、このオプションを使用してください。変更されたプランは、次の実行時に使用されます。拡張システム・プロシージャの実行時、Adaptive Server はこのオプションを無視します。

---

**注意** `execute procedure with recompile` を何度も使用すると、プロシージャ・キャッシュのパフォーマンスが低下することがあります。`with recompile` を使用するたびに新しいプランが生成されるため、新しいプランを格納する領域が足りない場合は、有用なパフォーマンス・プランがキャッシュから押し出される場合があります。

---

**string**

実行する Transact-SQL コマンドの部分を含む、リテラル文字列です。リテラル文字列で指定される文字数に制限はありません。

**char\_variable**

Transact-SQL コマンドのテキストを指定する変数名です。

例

**例 1** 上記の 3 つの文では、`showind` が `titles` パラメータ値を使用して実行されます。

```
execute showind titles
exec showind @tabname = titles
```

バッチやファイル内に他の文がない場合は、次のように指定できます。

```
showind titles
```

**例 2** リモート・サーバ GATEWAY 上で `checkcontract` を実行します。成功か失敗かを示すリターン・ステータスが `@retstat` に保管されます。

```
declare @retstat int
execute @retstat = GATEWAY.pubs.dbo.checkcontract
"409-56-4008"
```

**例 3** `roy_check` を実行し、3 つのパラメータを渡します。3 番目のパラメータ `@pc` は、`output` パラメータです。プロシージャの実行後、戻り値が `@percent` 変数に返されます。

```
declare @percent int
select @percent = 10
execute roy_check "BU1032", 1050, @pc = @percent output
select Percent = @percent
```

**例 4** このプロシージャは、ユーザがパラメータを指定しなければ、システム・テーブルに関する情報を表示します。

```

create procedure
showsystind @table varchar (30) = "sys%"
as
select sysobjects.name, sysindexes.name, indid
from sysindexes, sysobjects
where sysobjects.name like @table
and sysobjects.id = sysindexes.id

```

**例 5** xp\_echo を実行し、"Hello World!" という値を渡します。拡張ストア・プロシージャのリターン値は、*result* という名前の変数に格納されます。

```

declare @input varchar (12), @in varchar (12),
        @out varchar (255), @result varchar (255)
select @input="Hello World!"
execute xp_echo @in = @input, @out= @result output

```

**例 6** 最終の execute コマンドが文字列の値と文字変数を連結して Transact-SQL コマンドを発行します。

```

select name from sysobjects where id=3

declare @tablename char(20)
declare @columnname char(20)
select @tablename="sysobjects"
select @columnname="name"
execute ('select ' + @columnname + ' from ' + @tablename
+ ' where id=3')

```

**例 7** sp\_who を実行します。

```

declare @sproc varchar(255)
select @sproc = "sp_who"
execute @sproc

```

## 使用法

- アーカイブ・データベースを参照する文がアーカイブ・データベース内で使用可能であれば、アーカイブ・データベースで **execute** を実行できます。ストア・プロシージャ内部または外部でのトランザクションは、**execute** コマンドでは使用できません。
- プロシージャの実行結果は、それを実行したデータベースによって異なります。たとえば、ユーザ定義のシステム・プロシージャである **sp\_foo** は、**db\_name()** システム関数を実行し、それが実行されるデータベースの名前を返します。**pubs2** データベースから実行すると、値 "**pubs2**" が返されます。

```

exec pubs2..sp_foo
-----
pubs2
(1 row affected, return status = 0)

```

sybsystemprocs から実行すると、値 “sybsystemprocs” が返されます。

```
exec sybsystemprocs..sp_foo
-----
sybsystemprocs
(1 row affected, return status = 0)
```

- パラメータの指定方法には 2 種類あります。1 つは位置を指定する方法、もう 1 つは次の方法です。

```
@parameter_name = value
```

2 番目の形式を使用する場合は、`create procedure` に定義されている順序でパラメータを指定する必要はありません。

`output` キーワードを使用し、バッチまたはプロシージャの追加文でリターン・パラメータを使用する場合は、パラメータの値を変数として渡す必要があります。次に例を示します。

```
parameter_name = @variable_name
```

拡張ストアド・プロシージャを実行すると、名前または値のいずれかですべてのパラメータを渡します。ESP の `execute` コマンドの 1 つの呼び出しの中に、値で渡したパラメータと名前で渡したパラメータを混在させることはできません。

- `exec (@parameter_name)` の動的 SQL の構文も有効です。ただし、キー入力が増えます。たとえば、動的 SQL のコマンド `exec (@sproc="7")` は、プロシージャに整数値 7 を渡しますが、これは `exec @sproc 7` でも実行できます。
- `text`、`unitext`、`image` カラムを、ストアド・プロシージャのパラメータ、またはパラメータに渡す値として使用することはできません。
- `create procedure` でリターン・パラメータとして定義されていないパラメータに対して `output` を指定してプロシージャを実行すると、エラーが発生します。
- `output` を使用してストアド・プロシージャに定数を渡すことはできません。リターン・パラメータには変数名が必要です。プロシージャの実行前に、変数のデータ型の宣言、および変数への値の割り当てを行ってください。リターン・パラメータには `text`、`unitext`、または `image` データ型を使用することはできません。
- バッチの最初の文には、`execute` キーワードを使用する必要はありません。バッチとは、行に単独で入力されているワード "go" で終了する入力ファイルのセグメントです。

- プロシージャの実行プランは最初の実行時に保管されるため、次の実行時には、同じ文のセットを単独で実行するよりかなり時間が短縮されます。
- 1つのストアド・プロシージャが別のストアド・プロシージャを呼び出すと、ネストが発生します。ネスト・レベルは、呼び出されたプロシージャが実行を開始するときに増加し、終了するときに減少します。ネスト・レベルは、キャッシュされる文が作成されたときも1つ増えます。ネスト・レベルの最大値である16レベルを超えると、トランザクションが失敗します。現在のネスト・レベルは、`@@nestlevel` グローバル変数に格納されます。
- 戻り値0と-1～-14までは、現在 Adaptive Server によって、ストアド・プロシージャの実行状態を示すために使用されています。15～99の値は将来の使用のために予約されています。値のリストについては、「return」を参照してください。
- パラメータはトランザクションの一部ではないため、後でロールバックされるトランザクション内でパラメータが変更された場合でも、パラメータの値は前の値に復元されません。呼び出し側に返される値が、常にプロシージャが返されるときの値になります。
- `select *` をストアド・プロシージャで使用する場合、プロシージャは (with recompile オプションを指定した場合でも) `alter table` を使用してテーブルに追加した新しいカラムを取得しません。これを行うには、ストアド・プロシージャを削除して再作成する必要があります。そうしなければ、`select *` をベースとする `insert` を実行すると、結果がエラーになる可能性があります。新しく追加したカラムにデフォルトがバインドされていても、新しく追加したカラムでの `insert` の結果は NULL になります。

ストアド・プロシージャを削除して再作成する際、またはデータベースの再ロード時に、ターゲット・テーブルのカラム定義が `select *` の結果と一致しないと、エラー・メッセージが表示されます。
- リモート・プロシージャ・コールを使用して実行したコマンドをロールバックすることはできません。
- Adaptive Server が拡張ストアド・プロシージャを実行すると、with recompile オプションは無視されます。

#### Transact-SQL の動的な実行

- `string` オプションまたは `char_variable` オプションを指定して使用すると、`execute` は、指定された文字列と変数を連結し、その連結結果の Transact-SQL コマンドを実行します。`execute` コマンドのこの形式は、SQL バッチ、プロシージャ、トリガで使用されます。



- 次のコマンドの実行時は、*string* オプション、および *char\_variable* オプションは指定できません。use、exec(*string*) (execute ストアド・プロシージャではない)、connect、begin transaction、rollback、commit、および dbcc。
- *string* オプションまたは *char\_variable* オプションの内容は、SQL バッチまたはプロシージャ内で宣言されているローカル変数を参照できません。
- *string* オプションと *char\_variable* オプションを連結して新しいテーブルを作成できます。ただし、同じ SQL バッチまたはプロシージャ内では、execute によって作成されたテーブルは、他の execute コマンドからのみ参照できます。SQL バッチまたはプロシージャが完了した後は、動的に作成されたテーブルは永続的なテーブルになり、他のコマンドから参照できます。

## 標準規格

## ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

## パーミッション

プロシージャが実行モード "dynamic ownership chain" を使用して設定された場合を除き、*string* オプションまたは *char\_variable* オプションを使用して定義された Transact-SQL コマンドの実行パーミッションは、コマンドを実行するユーザに照らして検査されます。「sp\_procxmode」を参照してください。

execute のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、プロシージャ所有者であるか、プロシージャに対する execute パーミッションを持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、プロシージャに対する create procedure 権限が必要。

## 監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
38	exec_procedure	プロシージャの実行	<ul style="list-style-type: none"> <li>• Roles — 現在のアクティブな役割</li> <li>• Keywords または options — NULL</li> <li>• Previous value — NULL</li> <li>• Current value — NULL</li> <li>• Other information — すべての入力パラメータ</li> <li>• Proxy information — set proxy が有効な場合は元のログイン名</li> </ul>

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
39	exec_trigger	トリガの実行	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [create procedure](#), [drop procedure](#), [return](#)

システム・プロシージャ [sp\\_addextendedproc](#), [sp\\_depends](#),  
[sp\\_dropextendedproc](#), [sp\\_helptext](#), [sp\\_procxmode](#)

## fetch

説明	カーソル結果セットからローまたはロー・セットを返します。
構文	<pre>fetch [next   prior   first   last   absolute       fetch_offset   relative fetch_offset]       [from] cursor_name       [into fetch_target_list]</pre>
パラメータ	<p><b>next   prior   first   last   absolute   relative</b>          フェッチ方向を指定するキーワードです。非スクロール可能カーソルの場合、フェッチ方向を指定する必要はありません。フェッチ方向を指定する場合、他のオプションを使用してスクロール可能カーソルのローにアクセスできます。<b>absolute</b> または <b>relative</b> を使用するときは、<b>fetch_offset</b> を指定する必要があります。</p> <p><b>[from] cursor_name</b>          カーソルの名前です。<b>from</b> はオプションです。</p> <p><b>fetch_offset</b>          特定の位置からのオフセット値を指定します。<b>absolute</b> または <b>relative</b> を使用するときは、<b>fetch_offset</b> を指定する必要があります。<b>fetch_offset</b> には、位取りが 0 の符号付き数値リテラル、または位取りが 0 の数値を含む整数型または数値型の Transact-SQL 変数を使用できます。「スクロール可能なカーソルの位置付け規則」(481 ページ)を参照してください。</p> <p><b>fetch_target_list</b>          カーソル結果を格納するパラメータ、またはローカル変数をカンマで区切ったリストです。パラメータと変数は、<b>fetch</b> より前に宣言してください。</p>
例	<p><b>例 1</b> <code>authors_crshr</code> カーソルによって定義されたカーソル結果セットから、情報ローを返します。</p> <pre>fetch authors_crshr</pre> <p><b>例 2</b> <code>pubs_crshr</code> カーソルによって定義されたカーソル結果セットから、情報ローを変数の <code>@name</code>、<code>@city</code>、<code>@state</code> に返します。</p> <pre>fetch pubs_crshr into @name, @city, @state</pre> <p><b>例 3</b> スクロール可能カーソルの使用時は、数値リテラル・オフセットを方向キーワード <b>absolute</b> とともに使用できます。次の例では、25 番目のローを指定します。次のように入力します。</p> <pre>fetch absolute 25 from pubs_crshr into @name, @city, @state</pre>

**例 4 25** 番目のローを表す Transact-SQL 変数を使用するには、次のように入力します。

```
declare @offset int
select @offset = 25
fetch absolute @offset from c1
```

## 使用法

### 制限事項

- カーソルを宣言して `open` を実行してから、`fetch` を使用してください。
- アーカイブ・データベースで、`fetch` を使用できます。
- `cursor_name` に、Transact-SQL パラメータまたはローカル変数を指定することはできません。
- 非スクロール可能カーソルの場合、すでにフェッチされたローでは `fetch` は実行できません。結果セットを通して前の状態に戻ることにはできません。ただし、カーソルをクローズし、もう一度オープンして、カーソル結果セットを再作成すれば、始めからやり直すことができます。
- Adaptive Server では、`fetch_target_list` 内の変数と、カーソルを定義する `select` 文によって指定されたターゲット・リストの式が 1 対 1 で対応していると想定されます。変数またはパラメータのデータ型は、カーソル結果セットのカラムのデータ型と互換性がなければなりません。
- Adaptive Server は、連鎖トランザクション・モードが設定されると、現在アクティブ状態のトランザクションがない場合、`fetch` 文を使用してトランザクションを暗黙に開始します。ただし、`open` 文もトランザクションを自動的に開始するため、このような状態になるのは、`close on endtran` オプションを設定し、最初にカーソルをオープンしたトランザクションの終了後もカーソルがオープンしているときだけです。

### カーソル位置

- 非スクロール可能カーソルの場合、ローをすべてフェッチした後、カーソルは結果セットの最後のローを指します。さらにフェッチすると、Adaptive Server は、データがないことやカーソル位置が結果セットの終わりの位置より下に移動したことを示す値を `@@sqlstatus` および `@@fetch_status` グローバル変数に格納することで警告を発します。この場合、現在のカーソル位置から `update` または `delete` を実行できなくなります。

- `fetch into` の使用時に、`fetch_target_list` の変数の数と、カーソルを定義するクエリによって指定されたターゲット・リストの式の数が異なると、エラーが発生して、Adaptive Server はカーソルの位置を進めることができません。ただし、カーソル結果セットの変数のデータ型とカラムのデータ型との間に互換性エラーが発生しても、カーソル位置を進めることができます。

#### フェッチされるロー数の指定

- 一度に 1 つまたは複数のローの `fetch` を実行できます。`set` コマンドの `cursor rows` オプションで、`fetch` を実行するロー数を指定してください。

#### フェッチ情報の取得

- `@@sqlstatus` グローバル変数は、`fetch` 文の実行結果であるステータス情報 (例外の警告) を保管します。この値には、フェッチされた最後のカーソルが反映されます。

値	説明
0	<code>fetch</code> 文が正常に終了したことを示す。
1	<code>fetch</code> 文によってエラーが発生したことを示す。
2	結果セットにこれ以上データがないことを示す。現在のカーソル位置が結果セットの最終ローにあり、クライアントがそのカーソルの <code>fetch</code> 文を実行すると、この警告が出される。

- `@@fetch_status` グローバル変数は、スクロール可能カーソルで `fetch` が正常に実行されたかどうかを示します。

値	説明
0	<code>fetch</code> 文が正常に終了したことを示す。
-1	フェッチ操作が失敗したこと、つまりフェッチ対象のローが結果セットにないことを示す。
-2	今後のために予約済み。

- `fetch` 文だけが `@@sqlstatus` および `@@fetch_status` を設定できます。これ以外の SQL 文は、`@@sqlstatus` または `@@fetch_status` に作用しません。
- `@@rowcount` 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ります。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、`@@rowcount` の値は、結果セットにフェッチされたローの総数に等しくなるまで、前方のみに 1 つずつ増えます。

カーソル結果セットからすべてのローが読み込まれると、`@@rowcount` はカーソル結果セットのローの総数を示します。フェッチ後に、そのフェッチで指定したカーソルで読み込まれたロー数を `@@rowcount` から取得します。

スクロール可能カーソルの `@@rowcount` に最大値はありません。`@@rowcount` の詳細については、『リファレンス・マニュアル：ビルディング・ブロック』を参照してください。

### スクロール可能なカーソルの使用

`fetch_direction` の値に応じて、次のようになります。

- 値を指定しない場合、デフォルト値は `next` になる。
- `next` を指定しない場合、カーソルはスクロール可能で宣言する必要がある。
- `fetch_offset` は、位取り 0 の符号付き真数値でなければならない。
- カーソルを末尾のローの後または先頭のローの前に移動した場合、データは返されず、エラーも生成されない。
- `absolute` を指定すると、`fetch_offset > 0` の場合に、オフセットは結果セットの先頭ローの前の位置から計算される。`fetch_offset < 0` の場合に、オフセットは結果セットの末尾ローの後の位置から計算される。
- `relative` を指定すると、`fetch_offset n > 0` の場合に、カーソルは現在の位置から  $n$  ロー後に配置される。`fetch_offset n < 0` の場合に、カーソルは現在の位置より  $\text{abs}(n)$  ロー前に配置される。

結果セット内のローの番号は、1 からカウントされます。つまり、先頭のローは 1 番です。

### 複数ローのフェッチ

デフォルトでは、`fetch` を 1 回実行するたびに 1 つのローがクライアントに返されます。次のように入力することで、1 回の `fetch` で返されるローの数を別の数に変更できます。

```
set cursor rows number for cursor_name
```

ここで `number` は、カーソルで実行する `fetch` ごとのローの数を指定します。この数には、小数点のない数字リテラルまたは `integer` 型のローカル変数を指定できます。`cursor rows` が 1 より大きい場合、`fetch` 後に複数のローがクライアントに返されます。カーソルの位置によっては、`fetch` で返されるローが指定したロー数より少ないこともあります。現在のカーソルの位置は、常に 1 つのローです。

#### スクロール可能なカーソルの位置付け規則に使用する用語

これらの用語は、次の「スクロール可能なカーソルの位置付け規則」で使用されます。

- `curRowsetStart` – カーソルの現在の位置。
- `new_CurRowsetStart` – カーソルの新しい現在の位置。
- `total_rows` – カーソル結果セット内のローの総数。
- `before_first` – カーソル結果セットの最初のローの前のロー位置。この変数の値は 0 です。
- `after_last` – カーソル結果セットの最後のローの後のロー位置。この変数の値は、`total_rows + 1` です。
- `first_row` – カーソル結果セットの最初のローの位置。この変数の値は 1 です。
- `last_row` – カーソル結果セットの最後のローの位置。この変数の値は、`total_rows` と同じです。
- `fetchSize` – `fetch` オペレーションごとに要求されるローの数。

#### スクロール可能なカーソルの位置付け規則

この規則は、カーソル・ローをフェッチするときに `fetch_orientation` オプションで指定するカーソルの位置付けに適用されます。フェッチ実行時、`curPos` はカーソル位置を指します。`fetch_orientation` オプションの構文を参照してください。

`Fetch first CurRowsetStart` の位置や `fetchSize` の値とは関係なく、`new_CurRowsetStart` は、常に `first_row` へ移動する。

#### Fetch last

- `total_rows >= fetchSize` の場合、  

$$\text{new\_CurRowsetStart} = \text{total\_rows} - \text{fetchSize} + 1$$
 になります。
- `total_rows < fetchSize` の場合、  
`new_CurRowsetStart` は `first_row` に位置付けられます。

## Fetch next

- *CurRowsetStart* が *before\_first* の場合、*new\_CurRowsetStart* は *first\_row* に位置付けられます。
- $curPos = (CurRowsetStart + fetchSize)$  とすると、
  - $curPos \leq total\_rows$  の場合、*new\_CurRowsetStart* = *curPos* である。
  - $curPos > total\_rows$  の場合、*new\_CurRowsetStart* は *after\_last* である。
- *CurRowsetStart* が *after\_last* ローの場合、*new\_CurRowsetStart* は *after\_last* のままになります。

## Fetch prior

- 次の条件のいずれかが true の場合、*new\_CurRowsetStart* は *before\_first* となります。
  - $((CurRowsetStart \geq 1) \ \&\& \ (CurRowsetStart - fetchSize \leq 0))$  の場合
  - *CurRowsetStart* が *before\_first* の場合
- $curPos = CurRowsetStart - fetchSize$  に設定し、かつ  $1 \leq curPos \leq total\_rows$  の場合にのみ、*new\_CurRowsetStart* = *curPos* になります。
- (*CurRowsetStart* が *after\_last*) の場合、 $curPos = total\_rows - fetchSize + 1$  になります。  
*curPos* > 0 の場合、*new\_CurRowsetStart* = *curPos* に、*curPos* ≤ 0 の場合、*new\_CurRowsetStart* は *before\_first* になります。

## Fetch relative

- (*CurRowsetStart* が *before\_first*) && (*fetch\_offset* > 0) の場合、*new\_CurRowsetStart* = *fetch\_offset* になります。
- 次の条件のいずれかが成立する場合、*new\_CurRowsetStart* は *before\_first* になります。
  - ((*CurRowsetStart* が *before\_first*) であり、(*fetch\_offset* < 0) である。
  - ((*CurRowsetStart* が *first\_row* 位置付けられる)、さらに (*fetch\_offset* < 0) である。
  - (*CurRowsetStart* が *after\_last*) であり、 $((CurRowsetStart + fetch\_offset + 1) \leq 0)$  である



- $(1 < CurRowsetStart \leq total\_rows)$  の場合に、  
 $curPos = CurRowsetStart + fetch\_offset$  とすると、
  - $(curPos < 1)$  であり、 $(abs(fetch\_offset) \leq fetchSize)$  である場合のみ、 $new\_CurRowsetStart$  は  $first\_row$  に位置付けられる。
  - $(curPos < 1) \ \&\& \ (abs(fetch\_offset) > fetchSize)$  である場合のみ、 $new\_CurRowsetStart$  は  $first\_row$  の前に位置付けられる。
  - $(0 < curPos \leq total\_rows)$  の場合にのみ  $new\_CurRowsetStart = curPos$
  - $curPos > total\_rows$  である場合に限り、 $new\_CurRowsetStart$  は  $after\_last$  になります。
- $(CurRowsetStart$  が  $after\_last$ ) の場合に、  
 $curPos = CurRowsetStart + fetch\_offset + 1$  とすると、
  - $(1 \leq curPos \leq total\_rows)$  の場合にのみ  $new\_CurRowsetStart = curPos$  になります
  - $curPos \leq 0$  の場合のみ  $new\_CurRowsetStart$  は  $before\_first$  になる
  - $curPos > total\_rows$  の場合に限り、 $new\_CurRowsetStart$  は  $after\_last$  になる

#### Fetch absolute

- $fetch\_offset = 0$  の場合、 $new\_CurRowsetStart$  は  $before\_first$  になります。
- $fetch\_offset > total\_rows$  の場合、 $new\_CurRowsetStart$  は  $after\_last$
- $0 < fetch\_offset \leq total\_rows$  の場合、 $new\_CurRowsetStart = fetch\_offset$
- $(fetch\_offset < 0) \ \&\& \ (abs(fetch\_offset) > total\_rows)$  の場合、  
 $abs\_offset = abs(fetch\_offset)$  とすると、  
 $abs\_offset > fetchSize$  の場合のみ  $new\_CurRowsetStart$  は  $before\_first$   
 $abs\_offset \leq fetchSize$  の場合のみ  $new\_CurRowsetStart$  は  $first\_row$  に位置付けられる。
- $(fetch\_offset < 0) \ \&\& \ (abs(fetch\_offset) \leq total\_rows)$  の場合、  
 $new\_CurRowsetStart = total\_rows + fetch\_offset + 1$

標準規格	ANSI SQL – 準拠レベル：初級レベル 複数ローのフェッチは、Transact-SQL 拡張機能です。
パーミッション	fetch パーミッションは、デフォルトではすべてのユーザに付与されています。
参照	コマンド <a href="#">declare cursor</a> , <a href="#">open</a> , <a href="#">set</a>

## goto label

説明	ユーザ定義のラベルに分岐します。
構文	<i>label:</i> <i>goto label</i>
例	restart ラベルの使い方を示します。 <pre>declare @count smallint select @count = 1 restart:     print "yes" select @count = @count + 1 while @count &lt;=4     goto restart</pre>
使用法	<ul style="list-style-type: none"><li>ラベル名は、識別子の規則に従って命名してください。宣言するときは、後ろにコロン(:)を付けてください。<b>goto</b> とともに使用する場合は、コロンを付けません。</li><li><b>goto</b> とラベルとの間で無限ループが発生しないようにするには、<b>goto</b> を <b>if</b>、<b>while test</b>、またはその他の条件に依存させます。</li></ul>
標準規格	ANSI SQL — 準拠レベル: Transact-SQL 拡張機能
パーミッション	<b>goto</b> の使用にパーミッションは必要ありません。
参照	コマンド <a href="#">if...else</a> 、 <a href="#">while</a>

## grant

**説明** 個々のユーザ、ユーザ・グループ、役割にパーミッションを割り当てます。

**構文** データベース・オブジェクトへのアクセス・パーミッションを付与する場合：

```
grant {all [privileges] | permission_list}
    on {table_name as [correlation_name][(column_list)]
        | view_name(column_list)
        | stored_procedure_name | SQL_function_name}
        | keyname}
    [where search_conditions [as pred_name]]
    to {public | name_list | role_list}
    [with grant option]
    [granted by grantor]
```

組み込み関数を使用するパーミッションを付与する場合：

```
grant select
    on [builtin] builtin
    to {name_list | role_list}
    [granted by grantor]
```

特定のコマンドを実行するシステム権限を付与する場合：

```
grant {all [privileges] | privilege_list}
    to {public | name_list | role_list}
    [granted by grantor]
```

dbcc 権限を付与する場合：

```
grant {dbcc_privilege [on database ]
    [, dbcc_privilege [on database ], ...]}
    to {user_list | role_list}
    [granted by grantor]
```

特定のシステム・テーブルにデフォルトのパーミッションを付与する場合：

```
grant default permissions on system tables
```

他のサーバ・ログインへのユーザ ID の切り替えを許可するパーミッションを付与し、切り替え先のログインの役割に従ってその使用を制限する場合：

```
grant set proxy to name_list
    [restrict role role_list | all | system]
    [granted by grantor]
```

## パラメータ

## all

データベース・オブジェクトへのアクセス・パーミッションの割り当てに使用すると、**all** では、**decrypt** を除き、指定されたオブジェクトに適用可能なすべてのパーミッションが付与されます。すべてのオブジェクト所有者は、オブジェクト名に **grant all** を指定して、自分のオブジェクトにパーミッションを付与できます。**decrypt** パーミッションは別途付与する必要があります。

細密なパーミッションが有効でない場合、システム管理者またはデータベース所有者は **grant all** を使用して、データベース・オブジェクト作成の権限を割り当てることができます(「特定のコマンドを実行するシステム権限を付与する場合」の構文を参照)。システム管理者が使用すると、**grant all** では、すべての **create** 権限 (**create database**、**create default**、**create procedure**、**create rule**、**create table**、**create function**、**create view**) を割り当てることができます。データベース所有者が **grant all** を使用するか、または **master** データベース以外で **grant all** を実行すると、**Adaptive Server** によって **create database** 以外のすべての **create** 権限が付与され、情報メッセージが出力されます。

細密なパーミッションが有効化されている場合、**grant all** を使用した **create** 権限の付与のすべてを実行できることにはなりません。詳細については、『セキュリティ管理ガイド』の「細密なパーミッションの使用」を参照してください。

**where** 句が含まれる場合は、**grant** 文で **all** を使用できません。

*permission\_list*

付与するオブジェクト・アクセス・パーミッションのリストです。複数のパーミッションをリストするにはカンマで区切ってください。このテーブルは、各タイプのオブジェクトに対して付与できるアクセス・パーミッションを示します。

オブジェクト	permission_list に指定できるパーミッション
カラム	select、update、references、decrypt カラム名は column_list で指定できる。
暗号化キー	select
ストアド・プロシージャ	execute
SQL 関数	execute
テーブル	select、insert、delete、update、references、update statistics、delete statistics、truncate table、decrypt、transfer table、identity_insert *、identity_update *
ビュー	select、insert、delete、update、decrypt、identity_insert *、identity_update *

**注意** アスタリスク (\*) が付けられたパーミッションは、細密なパーミッションが有効化されている場合のみ付与できます。

*correlation\_name*

grant ... where で、where 句の *table\_name* のカラムを参照するためのエイリアスとしてのみ使用されます。

*table\_name*

パーミッションを付与するテーブルの名前です。テーブルは、現在のデータベースになければなりません。grant 文ごとにリストできるオブジェクトは 1 つだけです。

*column\_list*

パーミッションを適用する 1 つ以上の指定カラムです。リストはカンマで区切ります。カラムが指定されている場合は、select、references、decrypt、update のパーミッションのみを付与できます。

where 句を使用して 1 つ以上の指定カラムに対して grant が実行される場合、Adaptive Server では次のようにユーザの select コマンド、update コマンドまたは delete コマンドでロー・レベルのアクセスを実施します。

- grant select 文の 1 つ以上の指定カラムがターゲットのリストまたはユーザの where 文の where 句で参照されます。

- `grant update` 文で指定された 1 つ以上のカラムがユーザの `update` 文のターゲット・リストで参照されます。
- セッションで `ansi_permissions` をオンに設定している場合、`grant select` の 1 つ以上のカラムがユーザの `update` または `delete` 文の `where` 句で参照されます。

**view\_name**

パーミッションを付与するビューの名前です。ビューは、現在のデータベースになければなりません。

**stored\_procedure\_name**

パーミッションを付与するストアド・プロシージャの名前です。ストアド・プロシージャは、現在のデータベースになければなりません。

**key\_name**

アクセス権限を付与する暗号化キーの名前です。`key_name` は、現在のデータベースになければなりません。

**SQL\_function\_name**

パーミッションを付与する SQL 関数の名前です。ストアド関数は、現在のデータベースになければなりません。`grant` 文ごとにリストできる関数は 1 つだけです。

**where search\_conditions**

ロー・フィルタとして機能し、`select` 文、`update` 文、または `delete` 文で指定されたすべての `where` 句と結合します。`where` 構文は、特定のテーブルに対する `select`、`update`、および `delete` 権限を付与する場合のみ使用できます。`search_conditions` は、汎用の `where` 句で許容されるすべての構文を利用できます。`where` 句が付与されたテーブルとは別のテーブルにアクセスする場合は、サブクエリを使用する必要があります。`grant` 文における `where` 句の使用の詳細については、『セキュリティ管理ガイド』の「述部付き権限の付与」を参照してください。

**as pred\_name**

述部の名前です。この名前は現在のデータベースの付与者が所有する他のオブジェクトの名前の中でユニークであり、識別子の規則に従っている必要があります。`pred_name` を省略した場合、Adaptive Server は `grant` 述部にユニークな名前を割り当て、この名前は `sp_helprotect` を使用して表示できます。`pred_name` を `where` 句が指定されていない `grant` 文で使用することはできません。述部は `revoke` コマンドによって名前で参照することができます。

**public**

すべてのユーザです。オブジェクトのアクセス・パーミッションについては、**public** にオブジェクト所有者は含まれていません。オブジェクト作成パーミッションまたは **set proxy** 権限については、**public** にデータベース所有者は含まれていません。

**name\_list**

ユーザ名とグループ名をカンマで区切ったリストです。

**role\_list**

パーミッションを付与するシステム定義の役割名またはユーザ定義の役割名のリストです。

**with grant option**

**name\_list** で指定したユーザが、他のユーザにオブジェクトのアクセス・パーミッションを付与できるようにします。**with grant option** でパーミッションを付与できる対象は個々のユーザだけで、**"public"**、グループ、または役割に対して付与することはできません。**with grant option** を使用して述部付き権限を付与することはできません。

**granted by grantor**

付与者をユーザがコマンドを実行しているデータベースとは異なるデータベースのユーザとして示します。

**grantor**

現在のデータベースの有効なユーザ名で、実行者のユーザ ID の代わりに付与者のユーザ ID がシステムカタログ **sysprotects** に付与者として記録されます。

**builtin**

組み込み関数です。組み込み関数名の前にキーワード **builtin** を指定すると、同じ名前のテーブルと付与可能な組み込み関数を区別できます。付与可能な **builtin** 関数は、**set\_appcontext**、**get\_appcontext**、**list\_appcontext**、**authmech**、**rm\_appcontext** および **next\_identity** (**IDENTITY** カラムの **select** パーミッションが必要) です。

**privilege\_list**

付与可能なシステム権限のリストです。システム権限にはサーバワイド、およびデータベースワイドの権限が含まれます。付与可能なシステム権限のリストについては、[表 1-21](#)、[表 1-22](#) を参照してください。また、システム権限の付与方法の詳細については、「[使用法](#)」のセクションも参照してください。複数のコマンドは、カンマで区切って使用します。



**dbcc\_privilege**

付与する dbcc 権限の名前です。変数は使用できません。表 1-21 と表 1-22 には、付与可能なサーバワイドの dbcc 権限とデータベースワイドの dbcc 権限が記載されています。

---

**注意** public またはグループに対して dbcc 権限を付与したり、取り消したりすることはできません。

---

**database**

パーミッションを付与するデータベースの名前です。データベースワイドの dbcc 権限の付与で使用されます。on database 句はオプションで、データベースは現在のデータベースである必要があります。付与対象者は、ターゲット・データベース内の有効なユーザです。database は識別子の規則に従い、変数は指定できません。

1 つのコマンドで複数のアクションのパーミッションを付与する場合、database はユニークでなければなりません。

**set proxy**

ユーザが別のユーザと同一化するパーミッションをユーザに付与します。付与対象者に role\_list 内の役割がまだ割り当てられていない場合、ターゲットのログインに role\_list 内のいずれかの役割が付与されていると、ターゲットのログインに対する set proxy が失敗します。

**system**

付与対象者は、所有していないシステム標準の役割を所有するユーザと ID を切り替えることはできません。system を使用するには、set proxy パラメータを指定しなければなりません。

**restrict role role\_list**

付与対象者とターゲット・ログインに role\_list に含まれる役割が設定されている場合のみ、付与対象者は ID を切り替えることができます。

**all**

付与対象者は、同じ役割セットを所有するユーザに各自の ID を付与できます。つまり、付与対象者は set proxy コマンドを実行しても新しい役割を継承することはできません。

**default permissions on system tables**

「システム・テーブルのデフォルト・パーミッションの付与」(516 ページ) にリストしたシステム・テーブルのデフォルトのパーミッションを付与します。

**例** **例 1** ユーザ Mary と "sales" グループに titles テーブルの insert と delete コマンドを使用するパーミッションを付与します。

```
grant insert, delete
on titles
to mary, sales
```

**例 2** get\_appcontext 関数の select パーミッションを "public" (すべてのユーザを含む) に付与する場合は、次のように入力します。

```
grant select on builtin get_appcontext to public
```

この例文を、テーブル get\_appcontext の select パーミッションを付与する次の例文と比較します (その名前のテーブルが存在する場合)。

```
grant select on get_appcontext to public
```

関数と同じ名前のテーブル (この例では get\_appcontext 関数と get\_appcontext テーブル) を誤って選択しないように、builtin 引数を grant 文に指定します。

**例 3** titles テーブルの price と advance カラムの update パーミッションを "public" (すべてのユーザを含む) に割り当てる 2 つの方法は、次のとおりです。

```
grant update
on titles (price, advance)
to public
```

または

```
grant update (price, advance)
on titles
to public
```

**例 4** ユーザ Mary に titles テーブルの transfer table パーミッションを付与します。

```
grant transfer table on titles to mary
```

**例 5** ユーザ Mary と John に、create database コマンドと create table コマンドを使用するパーミッションを付与します。Mary と John の create table パーミッションは、master データベースのみに適用されます。

```
grant create database, create table
to mary, john
```

**例 6** すべてのユーザに、titles テーブルの decrypt パーミッション以外のすべてのアクセス・パーミッションを付与します。

```
grant all on titles
to public
```

**例 7** ユーザ **Mary** に対して、**authors** テーブルでの **update** コマンドの使用と、他のユーザへのパーミッションの付与を許可します。

```
grant update on authors
to mary
with grant option
```

**例 8** ユーザ **Bob** に対して、**titles** テーブルの **price** カラムでの **select** および **update** コマンドの使用と、他のユーザへのパーミッションの付与を許可します。

```
grant select, update on titles (price)
to bob
with grant option
```

**例 9** すべてのシステム・セキュリティ担当者に、**new\_sproc** ストアド・プロシージャの実行パーミッションを付与します。

```
grant execute on new_sproc
to sso_role
```

**例 10** ユーザ **James** に対して、**titles** テーブルの **price** カラムを参照する別のテーブルに参照整合性の制約を作成するパーミッションを付与します。

```
grant references on titles (price)
to james
```

---

**注意** 他のユーザのテーブルを参照する参照整合性の制約を含むテーブルを作成する場合は、そのテーブルを参照するための **references** パーミッションが事前に付与されている必要があります。そのテーブルの参照先のカラムには、ユニークな制約またはユニークなインデックスがなければなりません。参照整合性制約の詳細については、「[create table](#)」を参照してください。

---

**例 11** キー所有者が実行する場合、**ssn\_key** を使用してカラムの暗号化を指定するパーミッションをデータベース所有者に付与します。データベース所有者が **create table**、**alter table**、または **select into** で参照するには、**ssn\_key** の **select** パーミッションが必要です。

```
grant select on ssn_key to dbo
```

**例 12** ユーザ **Bob** に暗号化キーの作成パーミッションを付与します。

```
grant create encryption key to Bob
```

**例 13** **customer** テーブルのすべての暗号化カラムに対する **decrypt** パーミッションを付与します。

```
grant decrypt on customer to accounts_role
```

**例 14** Joe に master の dump any database 権限を付与して、すべてのデータベースについてダンプを実行できるようにします。

```
1>use master
2> go
1> grant dump any database to joe
2> go
```

**例 15** Joe にデータベース pubs2 の create any object 権限を付与し、Joe が自己の成り代わりまたは pubs2 の他のユーザの代理として任意のオブジェクト権限を作成できるようにします。

```
1> use pubs2
2> go
1> grant create any object to joe
2> go
```

**例 16** Alex に manage roles を付与します。サーバワイドの権限では、master が現在のデータベースである必要があるため、エラーが返されます。

```
1> use pubs2
2> go
1> grant manage roles to alex
2> go
Msg 4627, Level 16, State 1:
Line 1:
The user must be in the master database to GRANT/REVOKE
this command.
```

**例 17** システム管理者は役割を使用して、ユーザ Carlos が有効なユーザに設定されているすべてのデータベースまたは "guest" ユーザを許可するデータベースで dbcc checkalloc コマンドを実行できるように許可します。

---

**注意** master データベースにユーザ "guest" がすでに存在している場合は、Carlos を master データベースに実際のユーザとして追加する必要はありません。

---

```
1>use master
2> go
1> create role checkalloc_role
2> go
1> grant dbcc checkalloc any database to checkalloc_role
2> go
```

```
1> create login carlos with password carlospassword
2> go
1> grant role checkalloc_role to carlos
2> go
```

**例 18** master データベース内の有効なユーザである Frank に、サーバ内のすべてのデータベースに対して `dbcc checkdb` を実行するパーミッションを付与します。

```
1>use master
2> go
1> create login frank with password frankpassword
2> go

Password correctly set.
Account unlocked.
New login created.
(return status = 0)

1>sp_adduser fred
2> go

New user added.
(return status = 0)

1> grant dbcc checkdb any database to frank
2> go
```

これで、ユーザ Frank は、自分が有効なユーザとして設定されているサーバ内の各データベースで `dbcc checkdb` コマンドを実行できます。

```
% isql -Ufrank -Pfrankpassword -SSERVER
1> dbcc checkdb (tempdb)
2> go

Checking tempdb:Logical pagesize is 2048 bytes
Checking sysobjects:Logical pagesize is 2048 bytes
...
The total number of data pages in this table is 1. DBCC
execution completed.If DBCC printed error messages,
contact a user with system administrator (SA) role.
```

---

**注意** `public` またはグループに対して `dbcc` 権限を付与したり、取り消したりすることはできません。

---

**例 19** ユーザ Walter を `pubs2` のメンテナンス・ユーザに指定して、それ以外の場所では管理者レベルの権限を Walter に付与しない場合、システム管理者は次のコマンドを実行できます。

```
1> use pubs2
2> go
1> grant dbcc checkdb on pubs2 to walter
2> go
```

---

**注意** システム管理者はターゲット・データベース (この場合は pubs2) 内からコマンドを実行します。Walter はターゲット・データベース内の有効なユーザであることが必要です。on pubs2 句はオプションです。

---

これで Walter は、エラーを生じることなく customers データベースに対する dbcc checkdb コマンドを実行できます。

**例 20** grant dbcc コマンドと revoke dbcc コマンドがパブリックまたはグループに誤って適用されています。

```
1> grant dbcc tablealloc on pubs2 to public

Msg 4629, Level 16, State 1:
Line 1:
GRANT/REVOKE DBCC does not apply to groups or PUBLIC.

1> sp_addgroup gr

New group added.
(return status = 0)

1> grant dbcc tablealloc on pubs2 to gr

Msg 4629, Level 16, State 1:
Line 1:
GRANT/REVOKE DBCC does not apply to groups or PUBLIC.
```

**例 21** grant オプションを使用してシステム権限を付与することはできません。

```
grant change password to alex with grant option

Msg 156, Level 15, State 1:
Line 1:
Incorrect syntax near the keyword 'with'.
```

**例 22** ユーザ Harry が authors テーブルに対して truncate table と updates statistics を使用できるようにします。

```
grant truncate table on authors to harry
grant update statistics on authors to harry
```

**例 23** ユーザ Billy が authors テーブルに対して delete statistics コマンドを使用できるようにします。

```
grant delete statistics on authors to billy
```

**例 24** `oper_role` を持つすべてのユーザに `truncate table`、`update`、`delete statistics` の各権限を付与します (ユーザ `Billy` と `Harry` は、`oper_role` を持っている、これらのコマンドを `authors` に対して実行できるようになります)。

```
grant truncate table on authors to oper_role
grant update statistics on authors to oper_role
grant delete statistics on authors to oper_role
```

**例 25** ストアド・プロシージャを使用して、`truncate table`、`delete statistics`、`update statistics` のパーミッションを暗黙に付与します。たとえば、`Billy` が `authors` テーブルを所有している場合、`Billy` は次を実行すると、`authors` に対して `truncate table` と `update statistics` を実行する権限を `Harry` に付与できます。

```
create procedure sprocl
as
truncate table authors
update statistics authors
go
grant execute on sprocl to harry
go
```

また、ストアド・プロシージャを使用してカラム・レベルで `update statistics` と `delete statistics` のパーミッションを暗黙に付与することもできます。

**例 26** ユーザ `Harry` と `Billy` に `set proxy` または `set session authorization` を実行するパーミッションを付与し、サーバ内の他のユーザと同一化できるようにします。

```
grant set proxy to harry, billy
```

**例 27** `sso_role` のユーザに、`set proxy` または `set session authorization` を実行するパーミッションを付与し、サーバ内の他のユーザと同一化できるようにします。

```
grant set session authorization to sso_role
```

**例 28** `set proxy` をユーザ `Joe` に付与しますが、`Joe` が `ID` を、`sa`、`sso`、または `admin` の役割を持つユーザに切り替えることは制限します (ただし、`Joe` がすでにこれらの役割を持っている場合は、これらの役割を持つユーザに対して `set proxy` を実行できます)。

```
grant set proxy to joe
restrict role sa_role, sso_role, admin_role
```

ユーザ Joe が `admin_role` を持つユーザ (この例では `Our_admin_role`) に ID を切り替えようとした場合、Joe が `admin_role` を持っていないかぎりコマンドは失敗します。

```
set proxy Our_admin_role
```

```
Msg 10368, Level 14, State 1:
```

```
Server 's', Line 2:Set session authorization permission denied because the target login has a role that you do not have and you have been restricted from using.
```

ユーザ Joe が `admin_role` を付与された後でコマンドを再試行すると成功します。

```
grant role admin_role to joe
set proxy Our_admin_role
```

**例 29** ID を切り替えるときにユーザ Joe に新しい役割が付与されないようにします。

```
grant set proxy to joe
restrict role all
```

ユーザ Joe は、自分と同じか権限の低い役割を持つユーザにしか `set proxy` を付与できません。

**例 30** `set proxy` を使用するとき Joe が新しいシステム標準の役割を取得できないようにします。

```
grant set proxy to joe
restrict role system
```

Joe が持っていないシステム役割をターゲット・ログインが持っているとき、`set proxy` は失敗します。

**例 31** 学生には各自の評価に関する情報の表示のみが許可されます。

```
grant select on grades
where user_name(uid) = USER as predicate_grades
to public
```

**例 32** 登録済みの学生に、すべてのコースに関する情報の表示を許可します。最初の `grant` では、提供されているコースおよびセクションの閲覧を全員に許可します。2 番目の `grant` では、コースにおける各自の登録のみ、表示を許可します。

```
grant select on enrollment
(course_id, quarter, section_id)
to public
```



```
grant select on enrollment as e
  (uid, with_honors)
where e.uid in
  (select r.uid from
   registered_students r
   where USER = user_name(r.uid))
to public
```

登録済みの学生が次のクエリを入力すると、各自のコースの表示が制限されます (`with_honors` カラムが選択されているため)。

```
select course_id, quarter, with_honors
  from enrollment
```

同様にして、登録済みの学生が次のクエリでコースの受講者数を表示しようとしています。

```
select course_id, count(uid) from enrollment
  group by course_id
```

Adaptive Server は、ユーザが登録しているコースのカウンタを示す 1 行を返します。

**例 33** ユーザ `Smith` は、テーブル所有者 `Mary` を付与者として、`mary.books` に対する `select` パーミッションをユーザ `John` に付与します。

```
grant select on mary.books to john
  granted by mary
```

**例 34** ユーザ `Smith` は、`dbo` を付与者として、`create table` パーミッションをユーザ `John` に付与します。

```
grant create table to john
  granted by dbo
```

**例 35** 細密なパーミッションが無効の場合、システム権限 `manage any login` を付与するとエラーが発生します。

```
1>sp_configure "enable granular permissions"
2>go
```

Parameter Name	Default	Memory Used	Config Value	Run Value	Unit	Type
enable granular permissions	0	0	0	0	switch	dynamic

(1 row affected)

```
(return status = 0)

>grant manage any login to smith
>go
Msg 16325, Level 15, State 87:
Line 1:
Cannot GRANT/REVOKE permission 'MANAGE ANY LOGIN'.Verify that the granular
permissions option is enabled.
```

**例 36** システム権限 `own database` を付与する際は `on database` 句を指定する必要があります。

```
1>grant own database to smith
2>go

Msg 156, Level 15, State 2:
Line 1:
Incorrect syntax near the keyword 'to'.

1>grant own database on tdbl to smith
2>go
```

## 使用法

### 構文の代入

`grant` 構文の語句では `to` の代わりに `from` を使用できます。

### `set fipsflagger` の使用

`set fipsflagger` オプションが有効になっているときに `grant dbcc` を発行すると、次の警告が発行されます。

```
SQL statement on line number 1 contains Non-ANSI
text.The error is caused due to the use of DBCC.
```

### Privileges

サーバワイドのシステム権限 [表 1-21](#) は、付与可能なサーバワイドのシステム権限すべてのリストです。サーバワイドの権限は、`master` データベースで付与する必要があります。各権限の実行が許可される操作の詳細については、『セキュリティ管理ガイド』の章「細密なパーミッションの使用」にある「[表 8-15 サーバワイドの権限](#)」を参照してください。

---

**注意** [表 1-21](#) で、細密なパーミッションが無効の場合は、アスタリスク (\*) がマークされた権限のみを付与できます。

`dbcc` 権限構文 `dbcc dbcc_subcmd on all` は、`dbcc dbcc_subcmd any database` のエイリアスです。どちらの構文も使用できます。

---

表 1-21: 付与可能なサーバワイドのシステム権限

カテゴリ	権限
権限管理	<ul style="list-style-type: none"> <li>• manage security permissions</li> <li>• manage server permissions</li> </ul>
監査管理	<ul style="list-style-type: none"> <li>• manage auditing</li> </ul>
ログインおよび役割管理	<ul style="list-style-type: none"> <li>• allow exceptional login</li> <li>• change password</li> <li>• manage any login</li> <li>• manage any login profile</li> <li>• manage remote login</li> <li>• manage roles</li> </ul>
データベース管理	<ul style="list-style-type: none"> <li>• checkpoint (on <i>database</i>)</li> <li>• checkpoint any database</li> <li>• create database*</li> <li>• dump any database</li> <li>• dump database (on <i>database</i>)</li> <li>• load any database</li> <li>• load database (on <i>database</i>)</li> <li>• manage any database</li> <li>• mount any database</li> <li>• online any database</li> <li>• online database (on <i>database</i>)</li> <li>• own any database</li> <li>• own database (on <i>database</i>)</li> <li>• quiesce any database</li> <li>• unmount any database</li> </ul>
サーバ管理	<ul style="list-style-type: none"> <li>• manage any thread pool</li> <li>• manage cluster</li> <li>• manage disk</li> <li>• manage security configuration</li> <li>• manage server</li> <li>• manage server configuration</li> <li>• shutdown</li> </ul>
DBCC 権限	<ul style="list-style-type: none"> <li>• dbcc checkalloc any database*</li> <li>• dbcc checkcatalog any database*</li> <li>• dbcc checkdb any database*</li> <li>• dbcc checkindex any database*</li> <li>• dbcc checkstorage any database*</li> <li>• dbcc checktable any database*</li> <li>• dbcc checkverify any database*</li> <li>• dbcc fix_text any database*</li> <li>• dbcc indexalloc any database*</li> <li>• dbcc reindex any database*</li> <li>• dbcc tablealloc any database*</li> <li>• dbcc textalloc any database*</li> <li>• dbcc tune*</li> </ul>
アプリケーション管理	<ul style="list-style-type: none"> <li>• manage any execution class</li> <li>• manage any ESP</li> <li>• manage data cache</li> <li>• manage dump configuration</li> <li>• manage lock promotion threshold</li> <li>• monitor qp performance</li> <li>• manage resource limit</li> </ul>
その他のデッドロック	<ul style="list-style-type: none"> <li>• connect*</li> <li>• kill</li> <li>• kill any process</li> <li>• map external file</li> <li>• monitor server replication</li> <li>• set proxy</li> <li>• set tracing*</li> <li>• set tracing any process</li> <li>• set switch</li> <li>• show switch</li> <li>• use any database</li> <li>• use database</li> </ul>

データベースワイドの権限 表 1-22 は、付与可能なデータベースワイドのシステム権限すべてのリストです。データベースワイドの権限は、権限実行の対象とされるデータベース内で付与する必要があります。各権限の実行が許可される操作の詳細については、『セキュリティ管理ガイド』の章「細密なパーミッションの使用」にある「表 8-16 データベースワイドの権限」を参照してください。

**注意** 表 1-22 で、細密なパーミッションが無効の場合は、アスタリスク (\*) がマークされた権限のみを付与できます。

表 1-22: 付与可能なデータベースワイドの権限

カテゴリ	権限
権限管理	<ul style="list-style-type: none"> <li>• manage any object permission</li> <li>• manage database permissions</li> </ul>
ユーザ管理	<ul style="list-style-type: none"> <li>• manage any user</li> </ul>
ユーザの設定	<ul style="list-style-type: none"> <li>• setuser *</li> </ul>
複写管理	<ul style="list-style-type: none"> <li>• manage replication</li> </ul>
データベース管理	<ul style="list-style-type: none"> <li>• manage database</li> </ul>
クエリ・プラン管理	<ul style="list-style-type: none"> <li>• manage abstract plans</li> </ul>
DBCC 権限	<ul style="list-style-type: none"> <li>• dbcc checkalloc *</li> <li>• dbcc checkcatalog*</li> <li>• dbcc checkdb*</li> <li>• dbcc checkindex*</li> <li>• dbcc checkstorage*</li> <li>• dbcc checktable*</li> <li>• dbcc checkverify *</li> <li>• dbcc fix_text *</li> <li>• dbcc indexalloc*</li> <li>• dbcc reindex*</li> <li>• dbcc tablealloc*</li> <li>• dbcc textalloc*</li> <li>• manage checkstorage</li> <li>• report checkstorage</li> </ul>
システム・カタログ	<ul style="list-style-type: none"> <li>• select any audit table</li> <li>• select any system catalog</li> <li>• truncate any audit table</li> </ul>
一般オブジェクト	<ul style="list-style-type: none"> <li>• alter any object owner</li> <li>• create any object</li> <li>• drop any object</li> </ul>
暗号化キー	<ul style="list-style-type: none"> <li>• create encryption key *</li> <li>• manage any encryption key</li> <li>• manage column encryption key</li> <li>• manage master key</li> <li>• manage service key</li> </ul>
デフォルト	<ul style="list-style-type: none"> <li>• create default *</li> <li>• create any default</li> <li>• drop any default</li> </ul>

カテゴリ	権限
関数	<ul style="list-style-type: none"> <li>• create function *</li> <li>• create any function</li> <li>• drop any function</li> <li>• execute any function</li> </ul>
インデックス	<ul style="list-style-type: none"> <li>• create any index</li> </ul>
プロシージャ	<ul style="list-style-type: none"> <li>• create procedure *</li> <li>• create any procedure</li> <li>• execute any procedure</li> <li>• drop any procedure</li> </ul>
ルール	<ul style="list-style-type: none"> <li>• create rule *</li> <li>• create any rule</li> <li>• drop any rule</li> </ul>
テーブル	<ul style="list-style-type: none"> <li>• alter any table</li> <li>• create any table</li> <li>• create table*</li> <li>• decrypt any table</li> <li>• delete any table</li> <li>• drop any table</li> <li>• identity_insert any table</li> <li>• identity_update any table</li> <li>• insert any table</li> <li>• manage any statistics</li> <li>• references any table</li> <li>• reorg any table</li> <li>• select any table</li> <li>• transfer any table</li> <li>• truncate any table</li> <li>• update any table</li> </ul>
トリガ	<ul style="list-style-type: none"> <li>• create trigger *</li> <li>• create any trigger</li> <li>• drop any trigger</li> </ul>
ビュー	<ul style="list-style-type: none"> <li>• create view *</li> <li>• create any view</li> <li>• drop any view</li> </ul>

権限リスト 表 1-23 は、付与可能な権限とパーミッションのすべてをアルファベット順にリストしています。"\*" が示された権限については、細密なパーミッションを有効にする必要はありません。

表 1-23: 権限のアルファベット順リスト

権限名	権限タイプ	管理 ( 細密なパーミッションの有効時 )	暗黙設定
allow exceptional login	サーバ	manage server permissions	
alter any object owner	データベース	manage database permissions	
alter any table	データベース	manage database permissions	
change password	サーバ	manage security permissions	
checkpoint any database	サーバ	manage server permissions	
checkpoint (on database)	サーバ	manage server permissions	checkpoint any database

権限名	権限タイプ	管理 ( 細密なパーミッションの有効時 )	暗黙設定
checkpoint (on sybsecurity)	サーバ	manage security permissions	
connect*	サーバ	manage server permissions	
create any default	データベース	manage database permissions	create any object
create any function	データベース	manage database permissions	create any object
create any index	データベース	manage database permissions	create any object
create any object	データベース	manage database permissions	
create any procedure	データベース	manage database permissions	create any object
create any rule	データベース	manage database permissions	create any object
create any table	データベース	manage database permissions	create any object
create any trigger	データベース	manage database permissions	create any object
create any view	データベース	manage database permissions	create any view
create database*	サーバ	manage database permissions	
create default*	データベース	manage database permissions	create any default
create encryption key*	データベース	manage security permissions	manage column encryption key
create function*	データベース	manage database permissions	create any function
create index*	データベース	manage database permissions	create any index
create procedure*	データベース	manage database permissions	create procedure
create rule*	データベース	manage database permissions	create any rule
create table*	データベース	manage database permissions	create any table
create trigger*	データベース	manage database permissions	create any trigger
create view*	データベース	manage database permissions	create any view

権限名	権限タイプ	管理 ( 細密なパーミッションの有効時 )	暗黙設定
dbcc checkalloc*	データベース	manage database permissions	dbcc checkalloc any database
dbcc checkalloc any database*	サーバ	manage server permissions	
dbcc checkcatalog*	データベース	manage database permissions	dbcc checkcatalog any database
dbcc checkcatalog any database*	サーバ	manage server permissions	
dbcc checkdb*	データベース	manage database permissions	dbcc checkdb any database
dbcc checkdb any database*	サーバ	manage server permissions	
dbcc checkindex*	データベース	manage database permissions	dbcc checkindex any database
dbcc checkindex any database*	サーバ	manage server permissions	
dbcc checkstorage*	データベース	manage database permissions	dbcc checkstorage any database
dbcc checkstorage any database*	サーバ	manage server permissions	
dbcc checktable*	データベース	manage database permissions	dbcc checktable any database
dbcc checktable any database*	サーバ	manage server permissions	
dbcc checkverify*	データベース	manage database permissions	dbcc checkverify any database
dbcc checkverify any database*	サーバ	manage server permissions	
dbcc fix_text*	データベース	manage database permissions	dbcc fix_text any database
dbcc fix_text any database*	サーバ	manage server permissions	
dbcc indexalloc*	データベース	manage database permissions	dbcc indexalloc any database
dbcc indexalloc any database*	サーバ	manage server permissions	
dbcc reindex*	データベース	manage database permissions	dbcc reindex any database
dbcc reindex any database*	サーバ	manage server permissions	
dbcc tablealloc*	データベース	manage database permissions	dbcc tablealloc any database
dbcc tablealloc any database*	サーバ	manage server permissions	
dbcc textalloc*	データベース	manage database permissions	dbcc textalloc any database
dbcc textalloc any database*	サーバ	manage server permissions	
dbcc tune*	サーバ	manage server permissions	

権限名	権限タイプ	管理 ( 細密なパーミッションの有効時 )	暗黙設定
decrypt*	オブジェクト ( カラム )	manage any object permission/object owner	decrypt any table
decrypt any table	データベース	manage database permissions	
delete*	オブジェクト	manage any object permission/object owner	delete any table
delete any table	データベース	manage database permissions	
delete statistics*	オブジェクト	manage any object permission/object owner	manage any statistics
drop any default	データベース	manage database permissions	drop any object
drop any function	データベース	manage database permissions	drop any object
drop any object	データベース	manage database permissions	
drop any procedure	データベース	manage database permissions	drop any object
drop any rule	データベース	manage database permissions	drop any object
drop any table	データベース	manage database permissions	drop any object
drop any trigger	データベース	manage database permissions	drop any object
drop any view	データベース	manage database permissions	drop any object
dump any database	サーバ	manage server permissions	
dump database ( on database )	サーバ	manage server permissions	dump any database
dump database ( on sybsecurity )	サーバ	manage security permissions	
execute*	オブジェクト	manage any object permission/object owner	execute any function ( udf ) execute any procedure ( システム・プロシージャ )
execute any function	データベース	manage database permissions	
execute any procedure	データベース	manage database procedures	
identity_insert	オブジェクト	manage any object permission/object owner	



権限名	権限タイプ	管理 ( 細密なパーミッションの有効時 )	暗黙設定
identity_insert any table	データベース	manage database permissions	
identity_update	オブジェクト	manage any object permission/object owner	
identity_update any table	データベース	manage database permissions	
insert*	オブジェクト	manage any object permission/object owner	insert any table
insert any table	データベース	manage database permissions	
kill	サーバ	manage server permissions	kill any process
kill any process	サーバ	manage server permissions	
load any database	サーバ	manage server permissions	
load database (on database)	サーバ	manage server permissions	load any database
load database (on sybsecurity)	サーバ	manage security permissions	
manage abstract plans	データベース	manage database permissions	
manage any database	サーバ	manage server permissions	
manage any encryption key	データベース	manage security permissions	
manage any ESP	サーバ	manage server permissions	
manage any execution class	サーバ	manage server permissions	
manage any login	サーバ	manage security permissions	
manage any login profile	サーバ	manage security permissions	
manage any remote login	サーバ	manage security permissions	
manage any statistics	データベース	manage database permissions	
manage any thread pool	サーバ	manage server permissions	
manage any user	データベース	manage database permissions	
manage auditing	サーバ	manage security permissions	
manage checkstorage	データベース	manage database permissions	
manage cluster	サーバ	manage server permissions	
manage column encryption key	データベース	manage security permissions	manage any encryption key
manage data cache	サーバ	manage server permissions	

権限名	権限タイプ	管理 ( 細密なパーミッションの有効時 )	暗黙設定
manage database	データベース	manage database permissions	manage any database
manage database permissions	データベース	manage security permissions	
manage disk	サーバ	manage server permissions	
manage dump configuration	サーバ	manage server permissions	
manage lock promotion threshold	サーバ	manage server permissions	
manage master key	データベース	manage security permissions	manage any encryption key
manage replication	サーバ	manage server permissions	
manage resource limit	サーバ	manage server permissions	
manage roles	サーバ	manage security permissions	
manage security configuration	サーバ	manage security permissions	
manage security permissions	サーバ	manage security permissions	
manage server	サーバ	manage server permissions	
manage server configuration	サーバ	manage server permissions	
manage server permissions	サーバ	manage server permissions	
manage service key	データベース	manage security permissions	manage any encryption key
map external file	サーバ	manage server permissions	
monitor qp performance	サーバ	manage server permissions	
monitor server replication	サーバ	manage server permissions	
mount any database	サーバ	manage server permissions	
own any database	サーバ	manage server permissions	
online any database	サーバ	manage server permissions	
online database (on database)	サーバ	manage server permissions	online any database
online database (on sybsecurity)	サーバ	manage security permissions	
own database (on database)	サーバ	manage server permissions	
own database (on sybsecurity)	サーバ	manage security permissions	
quiesce any database	サーバ	manage server permissions	
references*	オブジェクト ( カラム )	manage any object permission/object owner	references any table
references any table	データベース	manage database permissions	
report checkstorage	データベース	manage database permissions	

権限名	権限タイプ	管理 ( 細密なパーミッションの有効時 )	暗黙設定
reorg any table	データベース	manage database permissions	
select*	オブジェクト ( カラム )	manage any object permission/object owner	select any table ( ユーザ・テーブルまたはビュー ) select any audit table ( 監査テーブル ) select any system catalog ( システム・テーブル )
select any audit table	データベース	manage database permissions	
select any system catalog	データベース	manage database permissions	
select any table	データベース	manage database permissions	
set proxy*	サーバ	manage security permissions	
set switch	サーバ	manage server permissions	
set tracing*	サーバ	manage server permissions	set tracing for any process
set tracing any process	サーバ	manage server permissions	
setuser *	データベース	manage database permissions	
show switch	サーバ	manage server permissions	
shutdown	サーバ	manage server permissions	
transfer any table	データベース	manage database permissions	
transfer table	オブジェクト	manage any object permission/object owner	transfer any table
truncate any audit table	データベース	manage database permissions	
truncate any table	データベース	manage database permissions	
truncate table*	オブジェクト	manage any object permission/object owner	truncate any table
unmount any database	サーバ	manage server permissions	
update*	オブジェクト ( カラム )	manage any object permission/object owner	update any table
update any security catalog	サーバ	manage security permissions	
update any table	データベース	manage database permissions	

権限名	権限タイプ	管理 (細密なパーミッションの有効時)	暗黙設定
update statistics	オブジェクト	manage any object permission/object owner	manage any statistics
use any database	サーバ	manage server permissions	
use database (on database)	サーバ	manage server permissions	use any database
use database (on sybsecurity)	サーバ	manage security permissions	

- ある権限を持っていれば、より細密な別の権限を暗黙的に持っていることになる場合があります。たとえば、**select any table** 権限を持つユーザは、すべてのユーザ・テーブルに対する **select** パーミッションを持っていることを意味します。暗黙的な関係を持つ権限ペアの完全なリストについては、[表 1-23](#) を参照してください。
- 次のデータベース管理権限を付与する際は、権限ごとに **on database** 句を指定する必要があります。これらは、**checkpoint**、**dump database**、**load database**、**online database**、**own database** です。たとえば、**Smith** に **db1** に対する **dump database** 権限を付与する場合は、次のように使用できます。

```
grant dump database on db1 to smith
```

1 つの **grant** コマンドで、異なるデータベースそれぞれのデータベース管理権限を付与できます。たとえば、**Smith** に **db1** に対する **own database** 権限と **db2** に対する **load database** 権限を付与する場合は、次のように使用できます。

```
grant own database on db1, load database on db2 to smith
```

- パーミッションを付与できるのは、現在のデータベースのオブジェクトに対してだけです。
- **grant** コマンドおよび **revoke** コマンドは、順序を区別します。パーミッションの割り当てが競合した場合に有効になるのは、最後に実行されたコマンドです。
- プロシージャまたはビューが参照するオブジェクトに対するパーミッションがなくても、ユーザにビューまたはストアド・プロシージャに対するパーミッションを付与できます。『セキュリティ管理ガイド』の「ユーザ・パーミッションの管理」を参照してください。

- `declare cursor` 文で参照される基本テーブルまたはビューに定義されているパーミッションとは関係なく、**Adaptive Server** は、すべてのユーザにカーソルを宣言するパーミッションを付与します。カーソルは **Adaptive Server** オブジェクト (テーブルなど) として定義されていないため、カーソルに対するパーミッションは付与できません。ユーザがカーソルをオープンすると、**Adaptive Server** は、そのカーソルの結果セットを定義するオブジェクトに対する `select` パーミッションがユーザにあるかどうかを確認します。**Adaptive Server** は、カーソルがオープンされるたびにパーミッションを確認します。

カーソルによって定義されたオブジェクトへのアクセス・パーミッションをユーザが持っている場合、**Adaptive Server** はカーソルをオープンします。ユーザはそのカーソルを介してローのデータに `fetch` を実行できます。**Adaptive Server** は、各 `fetch` のパーミッション検査は行いません。ただし、ユーザがカーソルを介して `delete` または `update` を実行する場合は、カーソル結果セットで参照されるオブジェクトのデータを削除、更新するための通常のパーミッション検査が行われます。

- `grant` 文は、パーミッションを受け取るユーザ、グループ、または役割ごとに、`sysprotects` システム・テーブルにローを 1 つ追加します。その後ユーザまたはグループからパーミッションを取り消す (`revoke`) と、`sysprotects` システム・テーブルからそのローが削除されます。パーミッションを与えられたグループ全体からではなく、特定のグループ・メンバからだけパーミッションを取り消す場合、**Adaptive Server** は元のローを保持し、その取り消されたパーミッションに対して新しいローを追加します。
- 複数の付与者がユーザ、グループ、役割に同じ権限やパーミッションを付与できます。この状況では、`sysprotects` で同一の権限やパーミッションの複数の付与に対応する複数のローが存在することになります。その後、1 つ以上の付与が取り消されても、取り消されていない付与が残っていれば、ユーザ、グループ、役割が所有する権限またはパーミッションは保持されます。

- グループのメンバであるという理由でユーザが特定のパーミッションを継承し、その後と同じパーミッションがユーザに明示的に割り当てられた場合、**sysprotects** にはローが追加されません。たとえば、**"public"** にすでに **authors** テーブルの **phone** カラムに対する **select** パーミッションが付与されていて、**"public"** のメンバである **John** に **authors** テーブルのすべてのカラムに対する **select** パーミッションが付与されているとします。この場合、**John** に対する **grant** の結果として **sysprotects** に追加されたローには、**authors** テーブルのカラムすべてに対するリファレンスが含まれます。ただし、すでにパーミッションが付与された **phone** カラムは除きます。
- デフォルトでは **create trigger** コマンドを発行するパーミッションがユーザに付与されます。トリガを作成するユーザのパーミッションを取り消すと、そのユーザの **sysprotects** テーブルに取り消しを示すローが追加されます。**create trigger** コマンドを発行するパーミッションをそのユーザに付与するには、**grant** コマンドを2つ発行する必要があります。最初のコマンドで **sysprotects** から取り消しローを削除し、2番目のコマンドで付与ローを挿入します。トリガを作成するパーミッションが取り消されると、そのユーザは自分のテーブルでもトリガを作成できなくなります。ユーザに対するトリガ作成パーミッションが取り消されるのは、取り消しコマンドを発行したデータベース内だけです。
- 次のシステム・プロシージャを使用してパーミッションに関する情報を表示します。
  - **sp\_helprotect** — データベース・オブジェクト、ユーザ、グループ、役割に対するパーミッションの情報をレポートする。
  - **sp\_column\_privileges** — テーブルまたはビューの1つまたは複数のカラムに対するパーミッションの情報をレポートする。
  - **sp\_table\_privileges** — テーブルまたはビューのすべてのカラムのパーミッションの情報をレポートする。
  - **sp\_activeroles** — Adaptive Server の現在のログイン・セッションでアクティブな役割(およびそれらに含まれる役割)をすべて表示する。
  - **sp\_displayroles** — 別の役割またはユーザに与えられている役割をすべて表示する。または役割の階層ツリー全体をテーブル・フォーマットで表示する。
- パーミッションを表示するには、次のように **sp\_helprotect** コマンドを使用します。

```
1> use pubs2
```

```

2> go
1> sp_helprotect
2> go

grantor  grantee  type    action  object          column          grantable
-----  -
dbo      public   Grant   Select  sysalternates  All             FALSE
...
dbo      Walter   Grant   DBCC    DBCC            dbcc checkdb   FALSE

(1 row affected)
(return status = 0)

```

- `grant dbcc` では `grant with grant` オプションを使用できません。

#### 役割へのパーミッションの付与

それぞれのユーザまたはグループに付与されたパーミッションは、役割に付与されたパーミッションによって上書きされます。たとえば、John にシステム・セキュリティ担当者の役割が付与され、`sso_role` に `sales` テーブルのパーミッションが付与されているとします。この場合、`sales` に対する John 個人のパーミッションが取り消されても、役割のパーミッションは個人のパーミッションより優先されるため、John は引き続き `sales` にアクセスできます。

ただし、`grant execute` パーミッションは、特定の役割を持たないユーザにもストアド・プロシージャの実行パーミッションを個別に割り当てます。たとえば、ストアド・プロシージャの実行パーミッションをシステム・セキュリティ担当者だけに確実に割り当てするには、ストアド・プロシージャの中でシステム関数 `proc_role` を使用します。この関数は、呼び出しを行うユーザに正しい役割があるかどうかを検査します。「`proc_role`」を参照してください。

#### `grant all` オブジェクト作成権限

- データベース内のオブジェクト名の指定がない `grant all` では `create encryption key` が付与されません。オブジェクト名なしの `grant all` は、細密なパーミッションが無効化されている場合のみ使用できます。
- オブジェクト名なしで使用した場合に、`grant all` で割り当てられるパーミッションは次のとおりです。これらは、`create database`、`create default`、`create procedure`、`create rule`、`create table`、`create function`、`create view` です。`create database` パーミッションは、システム管理者が `master` データベース内からのみ付与できます。

- オブジェクト名なしで **grant all** 構文を使用して、ユーザまたはグループに **create** コマンド・パーミッションを付与できるのは、データベース所有者とシステム管理者だけです。ただし、データベース所有者が **grant all** コマンドを使用すると、**create database** パーミッションを付与できるのはシステム管理者に限られるという内容の情報メッセージが出力されます。それ以外の上述のパーミッションはすべて付与されます。
- すべてのオブジェクト所有者は、**grant all** をオブジェクト名とともに使用して、自分が所有するオブジェクトに対するパーミッションを付与できます。**grant all** で、テーブル名やビュー名とともにユーザ名またはグループ名を使用すると、そのテーブルに対する **delete**、**delete statistics**、**insert**、**select**、**truncate table**、**update**、**update statistics** パーミッションが付与されます。

#### *grant with grant option* の規則

- “**public**” またはグループや役割に、**with grant option** を使用してパーミッションを割り当てることはできません。
- 細密なパーミッションが無効の場合、パーミッションの付与ではシステム管理者がオブジェクト所有者とみなされます。システム管理者が、別のユーザのオブジェクトに対するパーミッションを付与すると、**sysprotects** と **sp\_helprotect** の出力では、所有者の名前が付与者として表示されます。細密なパーミッションが有効の場合は、付与者の名前が **sysobjects** および **sp\_helprotect** の出力に付与者として表示されます。
- **grant option** パラメータを使用してシステム権限を付与することはできません。
- 各 **grant** コマンドの情報は、次の例外を除いて、**sysprotects** システム・テーブルに保持されます。
  - 同じユーザから、あるユーザに 2 回以上特定のパーミッションを付与しようとする、**Adaptive Server** は情報メッセージを表示する。最初の **grant** レコードだけが有効になる。
  - 2 つのまったく同じ **grant** コマンドが **with grant option** と、その他の方法で実行された場合、**grant with grant option** の方が有効になる。
  - 同じ付与者から 2 つの **grant** 文で特定のテーブルの同じパーミッションを特定のユーザに付与し、対象のカラムが異なっている場合、**Adaptive Server** では、この 2 つの付与が 1 つの文であるとして処理される。たとえば、次の **grant** 文はそれぞれ同じ意味になる。



```
grant select on titles (price, contract)
to keiko
grant select on titles (advance) to keiko
grant select on titles (price, contract,
advance)
to keiko
```

#### granted by の使用

- 述部付き権限の付与で **granted by** を使用することはできません。
- **grantor** が **grant** コマンドを実行するパーミッションを所有している必要はありません。
- 付与者がコマンド実行者でない場合は、**sysprotects.grantor** にリストされます。
- **granted by** パラメータの使用時に **enable granular permissions** を有効化する必要はありません。
- **with grant** オプションによって特定のオブジェクトに対する **grant** パーミッションを受け取るユーザは、**granted by** パラメータを発行できません。その他のユーザはすべて **granted by** パラメータを発行できます。

たとえば、**Mary** が自身のテーブル **books** の **select** パーミッションを **John** に付与した場合、**John** が 2 番目の **grant** コマンドを発行しようとするエラーが発生します。

**Mary:**

```
grant select on mary.books
to john with grant option
```

**John:**

```
grant select on mary.books
to joe granted by smith
```

#### ユーザおよびユーザ・グループ

- ユーザ・グループを使用すると、1 つの文で複数のユーザに対してパーミッションを付与 (**grant**) または取り消す (**revoke**) ことができます。各ユーザがメンバになることができるグループは 1 つだけです。また、各ユーザは常に "**public**" のメンバとなります。

- `sp_adduser` を使用して新規ユーザを追加し、`sp_addgroup` を使用してグループを作成できます。Adaptive Server へのログインを持つユーザに限定的な権限でデータベースの使用を許可するには、`sp_adduser` を使用して "guest" ユーザを追加し、制限付のパーミッションを "guest" に割り当てます。ログインを持つすべてのユーザが "guest" としてデータベースにアクセスできます。
- ユーザを削除するには、`sp_dropuser` を使用します。グループを削除するには、`sp_dropgroup` を使用します。

"public" 以外のグループに新しいユーザを追加するには、`sp_adduser` を使用します。設定したユーザのグループを変更するには、`sp_changegroup` を使用します。

グループのメンバを表示するには、`sp_helpgroup` を使用します。

- `sp_changegroup` を実行してグループ・メンバシップを変更すると、次のコマンドを実行することによってインメモリ保護キャッシュがクリアされます。

```
grant all to null
```

その結果、`sysprotects` テーブルからの更新情報によってキャッシュをリフレッシュできます。`sysprotects` を直接修正するには、Sybase 製品の保守契約を結んでいるサポート・センタに連絡してください。

#### システム・テーブルのデフォルト・パーミッションの付与

任意のデータベースからコマンドを発行してデフォルト・パーミッションの付与や取り消しができるシステム・テーブルは、次のとおりです。

- |                               |                              |                                |
|-------------------------------|------------------------------|--------------------------------|
| • <code>sysalternates</code>  | • <code>syslogs</code>       | • <code>syssegments</code>     |
| • <code>sysattributes</code>  | • <code>sysobjects</code>    | • <code>sysstatistics</code>   |
| • <code>syscolumns</code>     | • <code>syspartitions</code> | • <code>systabstats</code>     |
| • <code>syscomments</code>    | • <code>sysprocedures</code> | • <code>systhresholds</code>   |
| • <code>sysconstraints</code> | • <code>sysprotects</code>   | • <code>systypes</code>        |
| • <code>sysdepends</code>     | • <code>sysqueryplans</code> | • <code>sysusermessages</code> |
| • <code>sysindexes</code>     | • <code>sysreferences</code> | • <code>sysusers</code>        |
| • <code>sysjars</code>        | • <code>sysroles</code>      | • <code>sysxtypes</code>       |
| • <code>syskeys</code>        |                              |                                |

このコマンドでは次の変更も行われています。

- `public` から `syscolumns (encrkyid)` パーミッションと `syscolumns (encrkydb)` パーミッションを取り消す。

- `public` から `syscolumns (encrkydb)` パーミッションと `syscolumns (encrkyid)` パーミッションを取り消す。
- `public` から `sysobjects(audflags)` パーミッションを取り消す。
- `sysobjects` に対するパーミッションを `sso_role` に付与する。
- `public` から `sysencryptkeys` のすべてのカラムに対する `select` を取り消す。
- `sysencryptkeys` のすべてのカラムに対する `select` を `sso_role` に付与する。
- `syscolumns` に対するパーミッションを `sso_role` に付与する。

`master` データベースからコマンドを発行してデフォルト・パーミッションの付与や取り消しができるシステム・テーブルは、次のとおりです。

- |                                |                              |                                  |
|--------------------------------|------------------------------|----------------------------------|
| • <code>sysdatabases</code>    | • <code>sysusages</code>     | • <code>systimeranges</code>     |
| • <code>sysdevices</code>      | • <code>sysconfigures</code> | • <code>sysresourcelimits</code> |
| • <code>syslocks</code>        | • <code>syscurconfigs</code> | • <code>syslogins</code>         |
| • <code>sysmessages</code>     | • <code>syslanguages</code>  | • <code>sysremotelogins</code>   |
| • <code>sysprocesses</code>    | • <code>syscharsets</code>   | • <code>syssessions</code>       |
| • <code>systransactions</code> | • <code>syssservers</code>   |                                  |

さらに、このコマンドは以下を行います。

- `public` から `sysdatabases(audflags)` の `select` を取り消す。
- `public` から `sysdatabases(deftabaud)` の `select` を取り消す。
- `public` から `sysdatabases(defvwaud)` の `select` を取り消す。
- `public` から `sysdatabases(defpraud)` の `select` を取り消す。
- `public` から `sysdatabases(audflags2)` の `select` を取り消す。
- `sso_role` に `sysdatabases` の `select` を付与する。
- `public` から `syslogins(password)` の `select` を取り消す。
- `public` から `syslogins(audflags)` の `select` を取り消す。
- `public` から `syslogins(lpuid)` の `select` を取り消す。
- `syslogins` に対する `select` を `sso_role` に付与する。
- `public` から `syslisteners(net_type)` の `select` を取り消す。
- `public` から `syslisteners(address_info)` の `select` を取り消す。

- `syslisteners` に対する `select` を `sso_role` に付与する。
- `public` から `sysssrvroles(srid)` の `select` を取り消す。
- `public` から `sysssrvroles(name)` の `select` を取り消す。
- `public` から `sysssrvroles(password)` の `select` を取り消す。
- `public` から `sysssrvroles(pwdate)` の `select` を取り消す。
- `public` から `sysssrvroles(status)` の `select` を取り消す。
- `public` から `sysssrvroles(logincount)` の `select` を取り消す。
- `sysssrvroles` に対する `select` を `public` に付与する。
- `public` から `sysloginroles(suid)` の `select` を取り消す。
- `public` から `sysloginroles(srid)` の `select` を取り消す。
- `public` から `sysloginroles(status)` の `select` を取り消す。
- `sysloginroles` に対する `select` を `sso_role` に付与する。
- `public` から `sysinstances(hostname)` の `select` を取り消す。
- `sysinstances` に対する `select` を `sso_role` に付与する。

#### *update statistics*、*delete statistics*、*truncate table* のパーミッションの付与

Adaptive Server では、`update statistics`、`delete statistics`、`truncate table` の各コマンドのパーミッションをユーザ、役割およびグループに付与できます。テーブル所有者も、暗黙の `grant` によってパーミッションを付与できます。具体的には、`update statistics`、`delete statistics`、`truncate table` をストアド・プロシージャに追加してから、そのストアド・プロシージャの実行パーミッションをユーザまたは役割に付与します。

`update statistics` のパーミッションをカラム・レベルで付与できません。`sysroles`、`sysssrvroles`、`sysloginroles` の各セキュリティ・テーブルに対して `update statistics` または `delete statistics` を実行するには、`sso_role` が必要です。

デフォルトでは、`sa_role` を持つユーザは、`sysroles`、`sysssrvroles`、`sysloginroles` 以外のシステム・テーブルに対して `update statistics` と `delete statistics` を実行するパーミッションがあり、この権限を他のユーザに渡すこともできます。

`grant all` を発行して、`update statistics`、`delete statistics`、`truncate table` のパーミッションを付与することもできます。

---

**注意** `update statistics` を実行するパーミッションをユーザに付与すると、付与されたユーザはコマンドのバリエーション (`update all statistics`、`update partition statistics`、`update index statistics`、`update table statistics` など) を実行するパーミッションも付与されています。たとえば、次の例は、`authors` テーブルに対して `update statistics` のすべてのバリエーションを実行するパーミッションを `Billy` に付与します。

```
grant update statistics on authors to billy
```

`update statistics` を実行するパーミッションをユーザから取り消すと、そのコマンドのバリエーションを実行するパーミッションも取り消すこととなります。

---

`update statistics` のバリエーション (`update index statistics` など) のパーミッションを個別に付与することはできません。つまり、次のようなコマンドは発行できません。

```
grant update all statistics to harry
```

ただし、ストアド・プロシージャを作成して、これらのコマンドをどのユーザが実行するかを制御することができます。たとえば、次の例は、`authors` テーブルに対して `update index statistics` を実行するパーミッションを `Billy` に付与します。

```
create proc sp_ups as
update index statistics on authors
go
revoke update statistics on authors from billy
go
grant execute on sp_ups to billy
```

`delete statistics` のパーミッションをカラム・レベルで付与または取り消すことはできません。

Adaptive Server は、その他のグローバルな監査として `truncate table` を監査しますが、`update statistics` の監査は行いません。`truncate table` と `update statistics` の両方について明確な監査証跡を保持するためには、上記のように実行パーミッションをユーザに付与するストアド・プロシージャに両方のコマンドを含めることをおすすめします。

### 代理権限とセッション権限の付与

- `set proxy` や `set session authorization` を実行するパーミッションが付与されたユーザは、Adaptive Server 内の別のログインを同一化することができます。`set proxy` と `set session authorization` の機能は同じですが、`set session authorization` は SQL 標準で、`set proxy` は Transact-SQL 拡張機能である点が異なります。
- `set proxy` または `set session authorization` のパーミッションは、master データベース内から付与する必要があります。
- `grant set proxy` コマンドで指定する名前は、データベースで有効なユーザ名にしてください。つまり、そのデータベースの `sysusers` テーブルに登録されているユーザ名を指定します。
- `grant all` には、`set proxy` パーミッションまたは `set session authorization` パーミッションは含まれていません。
- `grant set proxy` を使用すると、役割の制限を段階的に拡大できます。たとえば、最初に `sa_role` を制限し、次に `sso_role` を制限することができます。

```
grant set proxy to joe
restrict role sa_role
grant set proxy to joe
restrict role sso_role
```

- 個々の役割の制限を解除することはできません。すべての役割からパーミッションを取り消すには、`set proxy` を取り消す必要があります。

### 共有ディスク・クラスタでのパーミッションの付与

ローカル・テンポラリ・データベースでユーザ定義の役割にパーミッションを付与しようとする、`grant` は失敗します。

### 役割への権限の付与

それぞれのユーザまたはグループに付与されたパーミッションは、役割に付与されたパーミッションによって上書きされます。たとえば、John にシステム・セキュリティ担当者の役割が付与され、`sso_role` に `sales` テーブルのパーミッションが付与されているとします。この場合、`sales` に対する John 個人のパーミッションが取り消されても、役割のパーミッションは個人のパーミッションより優先されるため、John は引き続き `sales` にアクセスできます。

ただし、`grant execute` パーミッションは、特定の役割を持たないユーザにもストアド・プロシージャの実行パーミッションを個別に割り当てます。たとえば、ストアド・プロシージャの実行パーミッションをシステム・セキュリティ担当者だけに確実に割り当てするには、ストアド・プロシージャの中でシステム関数 `proc_role` を使用します。この関数は、呼び出しを行うユーザに正しい役割があるかどうかを検査します。「`proc_role`」を参照してください。

#### public またはグループからの権限の取り消し

"public" やグループが持つ特定のパーミッションを取り消すと、個別にパーミッションが与えられているユーザのパーミッションも取り消されます。述部付き権限の付与と取り消しは例外です。『セキュリティ管理ガイド』の「Adaptive Server における述部付き権限の `sysprotects` への保存の仕組み」を参照してください。

#### 標準規格

ANSI SQL — 準拠レベル：初級レベル。`grant dbcc` は Transact-SQL 拡張機能でもあります。

`grant dbcc` および、グループへのパーミッションの付与機能と `set proxy` を付与する機能は、Transact-SQL 拡張機能です。`set session authorization (set proxy と機能は同じ)` を付与する機能は、ANSI 標準に準拠しています。

#### パーミッション

`grant` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	<p>一般に細密なパーミッションが有効の場合は、付与対象の権限またはパーミッションに応じて、次のいずれかの権限管理権限を持つユーザが <b>grant</b> を実行できる。</p> <p>サーバワイドの権限については、<b>manage serverpermissions</b> 権限または <b>manage security permissions</b> 権限を持つユーザであることが必要。</p> <p>データベースワイドの権限については、<b>manage database permissions</b> 権限を持つユーザであることが必要。</p> <p>オブジェクト権限については、オブジェクト所有者であるか、<b>manage any object permission</b> 権限を持つユーザであることが必要。</p> <p><b>grant default</b> を実行するには、データベース所有者であるか、データベースに対する <b>own database</b> 権限を持つユーザであることが必要。</p> <p>詳細については、表 1-23 の「管理 (細密なパーミッションの有効時)」コラムを参照してください。</p>
細密なパーミッションが無効	<p>細密なパーミッションが無効の場合、付与可能なシステム権限は、<b>create database</b>、<b>create default</b>、<b>create function</b>、<b>create procedure</b>、<b>create rule</b>、<b>create table</b>、<b>create view</b>、<b>connect</b>、<b>set proxy</b>、<b>set tracing</b> に制限される。</p> <p>コマンド実行 - システム管理者のみが <b>master</b> データベースからのみ、<b>create database</b>、<b>connect</b>、<b>set tracing</b> パーミッションを付与できる。<b>create trigger</b> パーミッションを付与できるのはシステム・セキュリティ担当者だけです。</p> <p><b>grant default</b> を実行するには、データベース所有者であるか、<b>sa_role</b> が付与されたユーザであることが必要。</p> <p>データベース一貫性の検査 - システム管理者のみが <b>grant dbcc</b> コマンドを実行できる。</p> <p>データベース・オブジェクトの <b>grant</b> アクセス - データベース・オブジェクトのパーミッションのデフォルトはオブジェクト所有者。オブジェクト所有者は、自分が所有するデータベース・オブジェクトのパーミッションを、他のユーザに与えることができます。</p> <p>関数 - システム管理者のみが組み込み関数に対するパーミッションを付与できる。</p> <p>暗号化コラム - システム・セキュリティ担当者とキー管理者のみが暗号化キー作成の暗黙的パーミッションを所有する。</p> <p>プロキシおよびセッション権限 - システム・セキュリティ担当者のみが <b>master</b> データベースからのみ <b>set proxy</b> または <b>set session authorization</b> を付与できる。<b>set proxy</b> や <b>set session authorization</b> を実行するパーミッションが与えられたユーザは、サーバ内の別のログインを同一化することができます。<b>set proxy</b> と <b>set session authorization</b> の機能は同じですが、<b>set proxy</b> は Transact-SQL 拡張機能で、<b>set session authorization</b> は ANSI92 標準である点が異なります。</p> <p>システム・テーブル - データベース所有者は、システム・テーブルのデフォルト・パーミッションを付与できる。</p>



監査 sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
40	grant	grant	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – grant 文の完全なコマンド・テキスト。</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

#### 参照

カタログ・ストアド・プロシージャ [sp\\_column\\_privileges](#),  
[sp\\_table\\_privileges](#)

コマンド [create role](#), [revoke](#), [setuser](#), [set](#)

関数 [proc\\_role](#)、[show\\_role](#)

システム・プロシージャ [sp\\_addgroup](#), [sp\\_adduser](#), [sp\\_changedbowner](#),  
[sp\\_changegroup](#), [sp\\_dropgroup](#), [sp\\_dropuser](#), [sp\\_helpgroup](#), [sp\\_helpprotect](#),  
[sp\\_helpuser](#), [sp\\_role](#)

## grant role

**説明** 指定されたログイン、ユーザ、システムに役割またはユーザ定義の役割を付与します。

**構文**

```
grant role role_name
      [where pred_expression]
      to {username | rolename | login_profile_name }
```

**パラメータ** *role\_name*  
システム・セキュリティ担当者がユーザや役割に付与する、システム役割名またはユーザ定義役割名です。

**where pred\_expression**

役割のアクティブ化述部とも呼ばれます。これは、指定された役割をアクティブにする際に満たす必要がある SQL 条件です。

*pred\_expression* は、*username* または *login\_profile\_name* に役割を付与する場合のみ使用できます。

役割のアクティブ化時に *pred\_expression* が FALSE と評価されると、**set role** コマンドは失敗し、Adaptive Server からエラー・メッセージが返されます。役割に自動アクティブ化が指定されている場合、またはデフォルトの役割である場合、Adaptive Server ではアクティブ化が暗黙的に失敗しますが、ログイン・プロセスは失敗しません。

**to username | rolename | login\_profile\_name**

役割の付与対象のログインの名前、役割、ログイン・プロファイルを特定します。付与先がログイン・プロファイルの場合、そのログイン・プロファイルが設定されたすべてのユーザに役割が付与されます。

**例** **例 1** "doctor" 役割をユーザ Mary に付与します。

```
grant role doctor_role to mary
```

**例 2** *intern\_role* を *doctor\_role* に付与すると、医師はインターンのすべての権限を継承します。

```
grant role intern_role to doctor_role
```

**例 3** *manage roles* 権限を持つユーザ Smith がユーザ *roleAdmin* を付与者として *nurse\_role* をユーザ John に付与します。

```
grant role nurse_role to john
granted by roleAdmin
```

**例 4** ログイン・プロファイル *lp\_10* に役割 *ldap\_user\_role* を付与します。

```
grant role ldap_user_role where
  get_appcontext(login_authentication) = 'LDAP'
to login_profile lp_10
```

上の例を使用した場合、ログイン・プロファイル `lp_10` が割り当てられたユーザのセッションで `ldap_user_role` が有効化されると、Adaptive Server はセッションが LDAP を使用して接続されたことをチェックします。LDAP 接続が存在する場合、このユーザに `ldap_user_role` があるとみなされ、存在しない場合は `ldap_user_role` が有効化されません。ログイン・プロファイル `lp_10` を変更し、`auto activated roles` 属性で `ldap_user_role` を指定することによって、ログイン時に述部評価が自動実行されるように設定します。このように設定しない場合は、`lp_10` が割り当てられたユーザが `set role` 文を実行したときに役割のアクティブ化の述部の評価が行われます。

#### 使用法

`create login`、`alter login`、`create login profile`、または `alter login profile` で役割の自動アクティブ化が指定されている場合、Adaptive Server はユーザのログイン時に、ログインまたはログイン・プロファイルに付与された役割を (述部の評価後に) 自動的にアクティブ化します。これ以外の場合は、`set role` が実行されたときに役割がアクティブ化されます。従属役割がアクティブな場合は、役割に付与された役割が自動的にアクティブ化されます。

指定した役割がすでに付与されているユーザには、`grant` コマンドを使用してパーミッションを付与できます。この役割は、`sso_role` や `sa_role` のようなシステム役割か、またはユーザ定義役割です。ユーザ定義役割は、`create role` コマンドを使用してシステム・セキュリティ担当者が作成する必要があります。

#### パーミッション

`grant role` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>manage roles</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sso_role</code> が付与されたユーザであることが必要。 <code>sa_role</code> を付与するには、 <code>sa_role</code> が付与されたユーザであることが必要。

#### 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

---

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
85	roles	create role、drop role、alter role、grant role、または revoke role	<ul style="list-style-type: none"><li>• <i>Roles</i> – 現在のアクティブな役割</li><li>• <i>Keywords</i> または <i>options</i> – grant role 文の完全なコマンド・テキスト。</li><li>• <i>Previous value</i> – NULL</li><li>• <i>Current value</i> – NULL</li><li>• <i>Other information</i> – NULL</li><li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li></ul>

## group by 句と having 句

**説明** `select` 文で使用され、テーブルをグループに分割して、`having` 句の条件と一致するグループのみを返します。通常、`group by` は集合と組み合わせで使用され、`select` クエリの非集合カラムのグループ分け方法を指定します。`having` 句はこれらのグループに使用されます。

**構文**

```
Start of select statement
[group by [all] aggregate_free_expression
  [, aggregate_free_expression]...]
[having search_conditions]
End of select statement
```

**パラメータ** `group by`

テーブルを分類するグループを指定し、`select` リストに集合関数が含まれている場合は、各グループの合計値を検出します。合計値は、グループごとに 1 つずつ、結果にカラムとして表示されます。これらの計算カラムは、`having` 句で参照できます。

`group by` の前に、`select` リストの `avg`、`count`、`count_big`、`max`、`min`、`sum` 集合関数を使用できます (通常、式はカラム名)。『リファレンス・マニュアル：ビルディング・ブロック』の「[第 2 章 Transact-SQL 関数](#)」を参照してください。

カラムを任意に組み合わせてテーブルをグループ化できます。つまり、後述の例 2 のように、グループを相互にネストできます。

**all**

`where` 句で除外されたグループも含め、結果にはすべてのグループを含める `Transact-SQL` 拡張機能です。次に例を示します。

```
select type, avg (price)
from titles
where advance > 7000
group by all type

type
-----
UNDECIDED                NULL
business                  2.99
mod_cook                  2.99
popular_comp              20.00
psychology                NULL
trad_cook                 14.99
```

(6 rows affected)

集約カラム内の "NULL" は、where 句で除外されるグループを示します。having 句は、all の意味を無効にします。

#### *aggregate\_free\_expression*

集合関数が含まれていない式です。Transact-SQL 拡張機能を使用すると、集合関数が入っていない式を使用しても、カラム名を使用するのと同じようにグループ分けできます。

カラム見出しまたはエイリアスでグループ分けすることはできません。次に正しい例を示します。

```
select Price=avg (price), Pay=avg (advance),
       Total=price * $1.15
from titles
group by price * $1.15
```

#### having

where 句で select 句の条件を設定するのと同じような方法で、group by 句の条件を設定します。

having 探索条件には、集合関数式を含むことができます。その他の点では、having 探索条件は where 探索条件とまったく同じです。次に、集合関数式を持つ having 句の例を示します。

```
select pub_id, total = sum (total_sales)
from titles
where total_sales is not null
group by pub_id
having count (*)>5
```

Adaptive Server は、クエリを最適化するとき **where** 句と **having** 句内の探索条件を評価して、最適なインデックスとクエリ・プランの選択に使用可能な探索索引数 (SARG) の条件を決定します。すべての探索条件を使用して、ローが限定されます。探索索引数の詳細については、『パフォーマンス&チューニング・シリーズ：最適化と抽象プラン』を参照してください。

例

**例 1** 本のタイプごとに前渡し金の平均と総売上げを計算します。

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

**例 2** 結果をタイプでグループ分けしてから、それぞれのタイプ内で pub\_id によってグループ分けします。

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

```
group by type, pub_id
```

**例 3** すべてのグループの結果を計算し、「p」で始まるタイプのグループだけを表示します。

```
select type, avg (price)
from titles
group by type
having type like 'p%'
```

**例 4** すべてのグループの結果を計算し、**having** 句の複数の条件に一致するグループの結果だけを表示します。

```
select pub_id, sum (advance), avg (price)
from titles
group by pub_id
having sum (advance) > $15000
and avg (price) < $10
and pub_id > "0700"
```

**例 5** **titles** テーブルと **publishers** テーブルをジョインしてから、グループ (**publisher**) ごとの総売上げを計算します。

```
select p.pub_id, sum (t.total_sales)
from publishers p, titles t
where p.pub_id = t.pub_id
group by p.pub_id
```

**例 6** 前渡し金が 1,000 ドル以上で、すべてのタイトルの平均価格よりも高い価格の本のタイトルを表示します。

```
select title_id, advance, price
from titles
where advance > 1000
having price > avg (price)
```

#### 使用法

- **group by** 句の後には、カラム名または式 (カラム見出しやエイリアスを除く) を使用できます。**group by** を使用して結果を計算したり、**select** リストに指定していないカラムまたは式を表示することができます (Transact-SQL 拡張機能については「[group by と having に対する Transact-SQL 拡張機能](#)」(532 ページ) の項を参照してください)。
- **group by** カラム (または式) の最大数は明示的に制限されていません。**group by** の結果の唯一の制限は、**group by** カラムの幅と集約結果の合計が 64K を超えることができないことです。
- **group by** カラムの Null 値は、1 つのグループにまとめられます。

- group by 句と having 句内では、text、unitext、または image カラムは指定できません。
- 更新可能カーソルの select 文で group by 句を使用することはできません。
- 集合関数は、select リストか having 句内だけで使用できます where 句や group by 句内では使用できません。

集合関数には、2つのタイプがあります。テーブル内のすべての修飾ローに適用される集合（各関数でテーブル全体に対する値を1つ生成する）関数は、「スカラ集合関数」と呼ばれます。group by 句を持たない select リストの集合関数は、テーブル全体に適用されます。これは、スカラ集合関数の一例です。

特定のカラムや式のローのグループに適用される（各関数で各グループに対する値を1つ生成する）集合関数は、「ベクトル集合関数」と呼ばれます。どちらのタイプの集合関数の場合も、集合演算の結果は、having 句が参照できる新しいカラムとして表示されます。

ベクトル集合関数は、スカラ集合関数内にネストできます。詳細については、『リファレンス・マニュアル』の「[第2章 Transact-SQL 関数](#)」を参照してください。

### オプティマイザでの group by の動作

Adaptive Server version 15.0 には、group by の処理に使用可能なアルゴリズムが2つ存在します。このアルゴリズムは、GroupHashing および GroupSorted です。オプティマイザがどちらの演算子を使用するかは、必要とされる入力データ・ストリームの条件などの要素に基づいて決定されます。

GroupSorted 演算子を使用するには、集合する入力ローがすでに group by カラムでソートされている必要があります。入力ローはソート済みの必要があるため、オプティマイザは次のいずれかの方法を使用します。

- order by カラムに存在するインデックスで、ソース・テーブルからのローを読み込みます。group by カラムの最大幅は、インデックス・キーの最大幅に制限されますが、この幅はデータベースのサイズによって異なります。



- **GroupSorted** 演算子で **group by** カラムのローを処理する前に、**Asort** 演算子でローを順序付けします。**group by** カラムおよび集合するカラムはワーク・テーブルに収まる大きさでなければならないため、**group by** カラムの最大幅は、データベース・ページの最大ロー・サイズから集合するカラムの幅を引いた大きさに制限されます。**group by** カラムの最大幅は、データベース・ページ・サイズによって制限されます。

オブティマイザは、**group by** カラムの順序付けを行えないか、**GroupSorted** 演算子のロー・サイズ制限を超過している場合に、**GroupHashing** 演算子を使用します。**GroupHashing** 演算子は、**group by** カラムの値にハッシュ関数を適用して、同じ **group by** カラム値を持つローを同じハッシュ・バケットに格納できるようにします。すべての入力ローをハッシュ・バケットに格納した後で、バケット内のローが集合されて **group by** の結果が生成されます。**GroupHashing** 演算子の唯一の制限は、**group by** カラムのサイズと集合結果のサイズが 64KB を超えることができないことです。**group by** カラムの数や集合演算の数には制限はなく、全体のロー幅のみの制限です。

#### **group by** と **having** クエリの集合関数の動作

**where** 句は、その探索条件に合わないローを除外します。その機能は、グループ分けされたクエリまたはグループ分けされていないクエリに対しても変わりません。

**group by** 句は残りのローを収集して、**group by** 式のユニークな値ごとに 1 つのグループにします。**group by** 句を省略すると、テーブル全体に対して 1 つのグループが作成されます。

**select** リストで指定された集合関数は各グループの合計値を算出します。スカラ集合関数では、テーブルに対する値は 1 つしかありません。ベクトル集合関数の場合は、個々のグループに対して値を計算します。

**having** 句は、その探索条件に合わないグループを結果から除外します。**having** がローをテストするだけであっても、次のように、**group by** 句の有無によってグループに対する操作が決まります。

- クエリに **group by** 句がある場合、**having** 句は結果のグループ・ローを除外します。このため、**having** 句はグループに作用するよう見えます。
- クエリに **group by** 句がない場合、**having** 句は (1 つのグループの) テーブルから結果ローを除外します。このため、**having** 句はローに作用するよう見えます (結果は **where** 句と同じです)。

### 標準の group by クエリと having クエリ

例の項で紹介されているすべての group by クエリと having クエリは、SQL 標準に従います。group by、having、およびベクトル集合関数を使用するクエリは、次のガイドラインを使用して、グループごとに 1 つのローと合計値を生成するように指示します。

- select リストのカラムは、group by 式内でも指定されているか、または集合関数の引数となっている必要があります。
- group by 式には、select リストにあるカラム名しか指定できません。ただし、select リストの集合関数の引数としてだけ使用されるカラムは、このなかには含まれません。
- having 式内のカラムは、集合関数の引数のように、単一の値でなければならず、select リストか group by 句内に含まれている必要があります。select リスト集合関数と having 句を使用するクエリには、group by 句が必要です。select リスト集合関数を使用しないクエリに対して group by 句を省略すると、where 句で除外されないローはすべて、1 つのグループとみなされます。

グループ分けされていないクエリでは、「where 句はローを除外する」が原則です。グループ分けされたクエリの場合、その原則は、「where 句は group by 句の前にローを除外し、その後、having 句が表示結果からローを除外する」になります。

SQL 標準では、2 つ以上のテーブルをジョインするクエリは、上記のガイドラインに従っていれば、group by と having を使用できます。ジョインまたはその他の複雑なクエリを指定する場合は、Transact-SQL 拡張機能による両方の句への影響を完全に理解するまでは、標準の group by と having 構文を使用することをおすすめします。

Transact-SQL 拡張機能による問題を避けるために、Adaptive Server では、クエリで Transact-SQL 拡張機能が指定されるたびに致命的ではない警告を出す fipsflagger オプションを set コマンドに対して用意しています。詳細については、「set」を参照してください。

### group by と having に対する Transact-SQL 拡張機能

標準の SQL に対する Transact-SQL 拡張機能は、次のように、グループの作成または合計値の計算に使用されていないカラムと式を参照できるようにすることで、柔軟性のあるデータ表示を可能にしています。

- 集合関数を含む select リストには、集合関数の引数ではなく、group by 句にない拡張カラムを含めることができます。拡張カラムは、追加のローが表示された後の最終結果の表示に影響を与えません。

- `group by` 句には、`select` リストにないカラムまたは式を指定できません。
- `group by all` 句は、`where` 句で計算から除外されたものも含めてすべてのグループを表示します。`all` キーワードの例については、前述の「パラメータ」の項を参照してください。
- `having` 句には、`select` リストにも `group by` 句にもないカラムまたは式を指定できます。

Transact-SQL 拡張機能によって表示にローとカラムが追加された場合、または `group by` 句が省略された場合は、クエリ結果の解釈が困難になる可能性があります。次に示す例は、Transact-SQL 拡張機能がクエリ結果にどのように影響するかを理解するのに役立ちます。

次の例は、標準の `group by` 句と `having` 句を使用するクエリと、Transact-SQL 拡張機能を使用するクエリとの相違点を示しています。

#### 1 標準のグループ分けクエリの例

```
select type, avg (price)
from titles
group by type

type
-----
UNDECIDED          NULL
business           13.73
mod_cook            11.49
popular_comp       21.48
psychology          13.50
trad_cook           15.96
```

(6 rows affected)

- 標準の `group by` 句がグループごとに 1 つのローを生成する場合でも、Transact-SQL の拡張カラムである `price` (`select` リスト内にあり、集合ではなく、`group by` 句内にもない) によって、条件を満たすすべてのローが条件を満たすそれぞれのグループ内に表示されます。また、`group by` 句はベクトル集合関数に影響し、それぞれのグループのそれぞれのローに表示されるグループごとの平均価格を計算します (例 a で計算された値と同じです)。

```
select type, price, avg (price)
from titles
group by type

type          price
-----
```

business	19.99	13.73
business	11.95	13.73
business	2.99	13.73
business	19.99	13.73
mod_cook	19.99	11.49
mod_cook	2.99	11.49
UNDECIDED	NULL	NULL
popular_comp	22.95	21.48
popular_comp	20.00	21.48
popular_comp	NULL	21.48
psychology	21.59	13.50
psychology	10.95	13.50
psychology	7.00	13.50
psychology	19.99	13.50
psychology	7.99	13.50
trad_cook	20.95	15.96
trad_cook	11.95	15.96
trad_cook	14.99	15.96

(18 rows affected)

- 3 Transact-SQL の拡張カラムの処理によって、クエリが **where** 句を無視しているかのように見えます。このクエリは、**where** 句の条件を満たすローだけを使用して平均価格を計算するだけでなく、**where** 句の条件に合わないローも表示します。

Adaptive Server は、まず、**where** 句を使用して、タイプと集合関数値だけを含むワークテーブルを構築します。このワークテーブルは、グループ化した **type** カラム内の **titles** テーブルにジョインされ、**price** カラムを結果に組み込みますが、このジョインに **where** 句は使用されません。

結果にはない **titles** の唯一のローは、**type** が "UNDECIDED" で **price** が **null** のロー、すなわち、ワークテーブルにその結果がなかったローです。表示結果から価格が 10.00 ドル未満のローを削除する場合は、例 4 に示されているように、**where** 句と同じ内容の **having** 句を追加してください。

```
select type, price, avg (price)
from titles
where price > 10.00
group by type
```

type	price	
business	19.99	17.31
business	11.95	17.31

business	2.99	17.31
business	19.99	17.31
mod_cook	19.99	19.99
mod_cook	2.99	19.99
popular_comp	22.95	21.48
popular_comp	20.00	21.48
popular_comp	NULL	21.48
psychology	21.59	17.51
psychology	10.95	17.51
psychology	7.00	17.51
psychology	19.99	17.51
psychology	7.99	17.51
trad_cook	20.95	15.96
trad_cook	11.95	15.96
trad_cook	14.99	15.96

(17 rows affected)

- 4 **having** 句で集合関数のような追加条件を指定する場合は、必ず、**where** 句で指定した条件もすべて含めます。Adaptive Server は、**having** 句に指定されていない **where** 句の条件をすべて無視します。

```
select type, price, avg (price)
from titles
where price > 10.00
group by type
having price > 10.00
```

type	price	
business	19.99	17.31
business	11.95	17.31
business	19.99	17.31
mod_cook	19.99	19.99
popular_comp	22.95	21.48
popular_comp	20.00	21.48
psychology	21.59	17.51
psychology	10.95	17.51
psychology	19.99	17.51
trad_cook	20.95	15.96
trad_cook	11.95	15.96
trad_cook	14.99	15.96

(12 rows affected)

- 5 これは、2つのテーブル間のジョインを使用してクエリをグループ分けする標準的な例です。この例では、`pub_id` でグループ分けしてから、`type` でグループ分けし、各ローのベクトル集合関数を計算します。

```
select p.pub_id, t.type, sum (t.total_sales)
from publishers p, titles t
where p.pub_id = t.pub_id
group by p.pub_id, t.type
```

```
pub_id  type
-----  -----
0736    business      18722
0736    psychology     9564
0877    UNDECIDED      NULL
0877    mod_cook       24278
0877    psychology      375
0877    trad_cook     19566
1389    business      12066
1389    popular_comp  12875
```

(8 rows affected)

次の構文を見ると、`pub_id` カラムと `type` カラムを `group by` に指定して結果を生成し、拡張カラムを追加することだけが必要な手順のように見えます。

```
select p.pub_id, p.pub_name, t.type,
       sum (t.total_sales)
from publishers p, titles t
where p.pub_id = t.pub_id
group by p.pub_id, t.type
```

しかし、上記のクエリの結果は、この例の最初のクエリの結果とはかなり異なります。Adaptive Server は、ワークテーブルにあるベクトル集合関数を調べるために2つのテーブルをジョインした後で、最終結果に使用される拡張カラムを持つテーブル (`publishers`) にワークテーブルをジョインします。別々のテーブルからの拡張カラムはそれぞれ、追加のジョインが作成されます。

前述の説明からもわかるように、テーブルをジョインするクエリで拡張カラムによる拡張機能を使用すると、理解が困難な結果が生成されてしまいます。テーブルをジョインするクエリには、ほとんどの場合、標準の `group by` 句を使用します。

- 6 この例では、`select` リストにないカラムを含めるために、`group by` 句に **Transact-SQL** 拡張機能を使用しています。`pub_id` カラムと `type` カラムは、両方ともベクトル集合関数の結果をグループ分けするために使用されます。ただし最終結果では、それぞれの `publishers` の項目に `type` は入っていません。この場合、知ることができる内容は、それぞれの `publishers` に対して別個の `type` がどれだけ販売されたかということだけです。

```
select p.pub_id, sum (t.total_sales)
from publishers p, titles t
where p.pub_id = t.pub_id
group by p.pub_id, t.type
```

```
pub_id
-----
0736      18722
0736       9564
0877       NULL
0877      24278
0877        375
0877      19566
1389      12066
1389      12875
```

(8 rows affected)

- 7 この例では、2つの **Transact-SQL** 拡張機能を結合しています。1つは、`group by` 句を省略して、`select` リストに集合関数を指定しています。もう1つは、拡張カラムを指定しています。`group by` 句を省略すると、次のようになります。

- テーブルは1つのグループになり、スカラ集合関数は、条件を満たす3つのローをカウントする。
- `pub_id` は、`group by` 句には指定されていないため、**Transact-SQL** の拡張カラムになる。`having` 句がないため、グループ内のローがすべて表示される。

```
select pub_id, count (pub_id)
from publishers
```

```
pub_id
-----
0736      3
0877      3
1389      3
```

(3 rows affected)

- 8 **where** 句は、1つのグループから 1000 以上の **pub\_id** を除外します。したがって、スカラ集合関数は条件を満たす 2 つのローをカウントします。拡張カラム **pub\_id** は、**publishers** テーブルから条件を満たしたすべてのローを表示します。

```
select pub_id, count (pub_id)
from publishers
where pub_id < "1000"

pub_id
-----
0736          2
0877          2
1389          2

(3 rows affected)
```

- 9 この例は、**group by** 句を指定しないで **having** 句を使用した場合の結果を示しています。

- テーブルは 1 つのグループとみなされます。ローを除外する **where** 句が指定されていないため、そのグループ (テーブル) のローはすべてカウントされます。
- この単一グループのテーブルのローは、**having** 句によってテストされます。
- 上記の 2 つの結果から、条件を満たす 2 つのローが表示されます。

```
select pub_id, count (pub_id)
from publishers
having pub_id < "1000"

pub_id
-----
0736          3
0877          3

(2 rows affected)
```

- 10 この例では、**select** リスト内にも **group by** 句内にもカラムまたは式を指定できない **having** 句に対して、拡張機能を使用しています。ここでは、それぞれの **type** の平均価格を調べますが、**sum** 集合関数が結果に表示されていなくても、総売上げが 10,000 ドル以下の **type** は除外されます。

```
select type, avg (price)
from titles
group by type
```



```
having sum (total_sales) > 10000
```

```
type
-----
business          13.73
mod_cook           11.49
popular_comp      21.48
trad_cook          15.96
```

```
(4 rows affected)
```

### *group by* と *having*、およびソート順

大文字／小文字を区別しないソート順がサーバに設定されている場合、*group by* は、グループ化カラムの大文字／小文字を無視します。たとえば、大文字／小文字を区別しないサーバに、次のデータがある とします。

```
select lname, amount
from groupdemo
lname          amount
-----
Smith          10.00
smith          5.00
SMITH          7.00
Levi           9.00
Levi           20.00
```

*lname* でグループ分けすると、次のような結果が生成されます。

```
select lname, sum (amount)
from groupdemo

lname
-----
Levi          9.00
Levi          20.00
Smith         22.00
```

大文字／小文字およびアクセントの有無を区別しないサーバで上記と同じクエリを実行すると、次のような結果が生成されます。

```
lname
-----
Levi          29.00
Smith         22.00
```

### 標準規格

ANSI SQL – 準拠レベル：初級レベル

group by リストになく、集合関数の指定がない select リスト内でのカラムの使用は Transact-SQL 拡張機能です。

all キーワードの使用は Transact-SQL 拡張機能です。

### 参照

コマンド [compute 句](#), [declare](#), [select](#), [where 句](#)

マニュアル 『リファレンス・マニュアル：ビルディング・ブロック』の「[第 2 章 Transact-SQL 関数](#)」

## if...else

説明	SQL 文の実行に条件を設定します。
構文	<pre> if <i>logical_expression</i> [plan "<i>abstract plan</i>"]     <i>statements</i>  [else     [<i>logical_expression</i>] [plan "<i>abstract plan</i>"]     <i>statement</i>] </pre>
パラメータ	<p><i>logical_expression</i></p> <p>TRUE、FALSE、または NULL を返す式 (カラム名、定数、算術演算子またはビット処理演算子で連結されたカラム名と定数の任意の組合せ、またはサブクエリ) です。論理式に <code>select</code> 文がある場合、<code>select</code> 文をカッコで囲む必要があります。</p> <p>plan "<i>abstract plan</i>"</p> <p>クエリを最適化するために使用する抽象プランを指定します。抽象プラン言語で指定された完全プランまたは部分プランを指定できます。プランは、最適化可能な SQL 文、つまりテーブルにアクセスする <code>select</code> クエリにのみ指定できます。『パフォーマンス&amp;チューニング・シリーズ：オプティマイザと抽象プラン』の「抽象プランの作成と使用」を参照してください。</p> <p><i>statements</i></p> <p>単一の SQL 文、または <code>begin</code> と <code>end</code> で区切られた SQL 文のブロックのどちらかです。</p>
例	<p><b>例 1</b> 3 が 2 より大きいとき "yes" を出力します。</p> <pre> if 3 &gt; 2     print "yes" </pre> <p><b>例 2</b> <code>if...else</code> 条件は、郵便番号が 94705 の著者がいるかどうかを調べて、結果セットの "Berkeley author" を出力します。</p> <pre> if exists (select postalcode from authors           where postalcode = "94705")     print "Berkeley author" </pre> <p><b>例 3</b> <code>if...else</code> 条件は、データベース内にユーザ作成のオブジェクト (ID 番号が 100 より大きいオブジェクトすべて) があるかどうかを調べます。ユーザ・テーブルが存在する場合、<code>else</code> 句はメッセージを出力し、テーブルの名前、タイプ、ID 番号を選択します。</p> <pre> if (select max (id) from sysobjects) &lt; 100     print "No user-created objects in this database" else </pre>

```
begin
  print "These are the user-created objects"
  select name, type, id
  from sysobjects
  where id > 100
end
```

**例 4** titles テーブル内の PC9999 の総売上高が NULL であるため、このクエリは FALSE を返します。if 部分が FALSE または null を返すと、クエリの else 部分が実行されます。真理値と論理式の詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の「[第 4 章 式、識別子、およびワイルドカード文字](#)」にある「[式](#)」を参照してください。

```
if (select total_sales
     from titles
     where title_id = "PC9999") > 100
select "true"
else
select "false"
```

#### 使用法

- if キーワードに続く文とその条件は、条件が満たされた場合（論理式が TRUE を返したとき）に実行されます。オプションの else キーワードは、if 条件が満たされない場合（論理式が FALSE を返したとき）に実行される代替 SQL 文を導入します。
- 複数の SQL 文を、begin キーワードと end キーワードで囲んで 1 つのブロックにグループ分けしない限り、if または else 条件文は、単一の SQL 文のみに作用します（例 3 を参照）。

文の句を、execute ストアド・プロシージャ・コマンド、または他の有効な SQL 文や文ブロックにすることができます。

- ブール式の一部として select 文を使用する場合、その文は単一の値を返す必要があります。
- if...else 構文は、ストアド・プロシージャ（パラメータの存在をテストするために使用される場合）、またはアドホック・クエリで使用できます（例 1 と 2 を参照）。

- if テストは、他の if 内、または else の後のどちらかにネストできます。ネストできる if テストの最大数は、それぞれの if...else 構成で指定する select 文 (または他の言語構成体) の複雑さによって異なります。

---

**注意** if...else ブロック内で alter table、create table、または create view コマンドが実行された場合、条件が真かどうか判定される前にテーブルまたはビュー用のスキーマが作成されます。テーブルまたはビューがすでに存在する場合は、エラーが発生する可能性があります。

---

- varchar、nvarchar、univarchar、または varbinary の各カラムを使用して、定義された幅の合計が最大許容ロー・サイズより大きいテーブルを作成すると、警告メッセージが表示されますが、テーブルは作成されます。このようなローに最大バイト数を超過して挿入を行うか、ローに update を実行してロー・サイズの合計が最大長を超えるようにすると、Adaptive Server によってエラー・メッセージが表示され、コマンドは正常に動作しません。

---

**注意** if...else ブロックまたは while ループ内で create table コマンドを実行すると、条件が真かどうか判定される前に、そのテーブルのスキーマが作成されます。このため、そのテーブルがすでに存在する場合は、エラーが発生することがあります。このようなエラーが発生しないようにするには、データベースに同じ名前のビューがないことを確認するか、または execute 文を次のように使用してください。

```
if not exists
    (select * from sysobjects where name="my table")
begin
    execute ("create table mytable (x int)")
end
```

---

標準規格

ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

パーミッション

if...else の使用にパーミッションは必要ありません。

参照

コマンド [begin...end](#), [create procedure](#)

## insert

**説明** テーブルまたはビューに新しいローを追加します。

**構文**

```
insert [into] [database.[owner.]]{table_name|view_name}
    [(column_list)]
    {values (expression [, expression]...)}
    [select_statement [plan "abstract plan"]]
```

**パラメータ**

**into**

省略可能なオプションです。

**table\_name | view\_name**

ローを追加するテーブルまたはビューの名前です。テーブルまたはビューが別のデータベース内にある場合は、データベース名を指定します。データベース内に同じ名前のテーブルまたはビューが複数ある場合は、所有者の名前を指定します。**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。

**column\_list**

データが追加される1つまたは複数のカラムのリストです。リストはカッコで囲んでください。複数のカラムはどんな順序でもリストできますが、受信データは (**values** 句内であれ **select** 句内であれ)、同じ順番でリストする必要があります。**IDENTITY** プロパティを持つカラムの場合は、実際のカラム名の代わりに **syb\_identity** キーワードを使用できます。

テーブル内にある複数のカラムの一部がデータを受信するときに、カラム・リストが必要になります。カラム・リストが指定されていない場合、Adaptive Server は、**insert** が受信テーブル内のすべてのカラムに (**create table** の順序で) 作用しているものとみなします。

詳細については、「[カラム・リスト](#)」(546 ページ) を参照してください。

**values**

式のリストを導入します。

**expression**

指示されたカラムに挿入する定数式、変数、パラメータ、または **null** 値を指定します。定数は、一重引用符または二重引用符で囲んでください。

サブクエリは **expression** として使用できません。

値リストは次の条件を満たす必要があります。

- カッコで囲む

- 暗黙的または明示的なカラム・リストと対応している
- 「デフォルト」値を使用できる

データ入力規則の詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の「[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。

#### *select\_statement*

挿入する値の検索に使用する標準の **select** 文です。

#### plan "*abstract plan*"

クエリを最適化するために使用する抽象プランを指定します。抽象プラン言語で指定された完全プランまたは部分プランを指定できません。プランは **insert...select** 文にのみ指定できます。詳細については、『パフォーマンス&チューニング・シリーズ：オプティマイザと抽象プラン』の「抽象プランの作成と使用」を参照してください。

例

#### 例 1

```
insert titles
values ("BU2222", "Faster!", "business", "1389",
       null, null, null, "ok", "06/17/87", 0)
```

#### 例 2

```
insert titles
(title_id, title, type, pub_id, notes, pubdate,
 contract)
values ('BU1237', 'Get Going!', 'business',
       '1389', 'great', '06/18/86', 1)
```

#### 例 3

```
insert newauthors
select *
from authors
where city = "San Francisco"
```

#### 例 4

```
insert test
select *
from test
where city = "San Francisco"
```

#### 例 5

```
insert table1 (col1, col2, col3, col4)
values (10, 4, default, 34)
```

## 使用法

- 新しいローを追加するだけの場合は、`insert` を使用してください。すでに挿入されているローのカラム値を変更する場合は、`update` を使用してください。

## カラム・リスト

複数の値を入力する順序を指定します。たとえば、`newpublishers` というテーブルがあり、その構造と内容が `pubs2` 内の `publishers` テーブルと同一であると仮定します。`newpublishers` テーブルのカラム・リスト内にあるカラムが、`publishers` テーブル内の `select` リストのカラムと一致している例を次に示します。

```
insert newpublishers (pub_id, pub_name)
select pub_id, pub_name
from publishers
where pub_name="New Age Data"
```

"New Age Data" の `pub_id` と `pub_name` はそれぞれ、`newpublishers` の `pub_id` カラムと `pub_name` カラムに保管されます。

次の例では、`newpublishers` テーブルのカラム・リスト内にあるカラムの順序が `publishers` テーブルの `select` リストのカラム順序と一致していません。

```
insert newpublishers (pub_id, pub_name)
select pub_name, pub_id
from publishers
where pub_name="New Age Data"
```

結果的に、"New Age Data" の `pub_id` は `newpublishers` テーブルの `pub_name` カラム内に保管され、"New Age Data" の `pub_name` は、`newpublishers` テーブルの `pub_id` カラムに保管されます。

省略されたカラムが `null` 値を許容していれば、カラム・リストおよび値リストから項目を省略できます (例 2 を参照)。

## カラム値の妥当性の検証

- `insert` は、`create index` コマンドによって設定される `ignore_dup_key`、`ignore_dup_row`、`allow_dup_row` の各オプションと相互に作用します。詳細については、「`create index`」を参照してください。
- ルールまたは `check` 制約 (検査制約) によって、カラムに入力できる有効値の領域を限定できます。ルールは `create rule` コマンドで作成し、`sp_bindrule` でバインドします。`check` 制約は `create table` 文で宣言します。



- ユーザが明示的に入力を行わない場合は、デフォルトによって値が提供されます。デフォルトは `create default` コマンドで作成し、`sp_bindefault` でバインドするか、または `create table` 文で宣言します。
- `insert` 文がドメインまたは整合性ルール (`create rule` と `create trigger` を参照) に違反している場合、またはデータ型が正しくない場合 (`create table`、および『リファレンス・マニュアル：ビルディング・ブロック』の「第 1 章 システム・データ型とユーザ定義データ型」を参照)、その文は失敗し、Adaptive Server にエラー・メッセージが表示されます。

#### 空文字列の処理

- 空文字列 ("") を可変長の文字型または `text` カラムに挿入すると、シングル・スペースが挿入されます。`char` カラムには、定義した長さにスペースが埋め込まれます。
- 文字列にスペースだけが含まれる場合を除いて、`varchar` カラムと `univarchar` カラムに挿入されるデータの後続スペースは削除されます。スペースだけからなる文字列は、トランケートされて 1 つのスペースになります。`char`、`nchar`、`unichar`、`univarchar`、`varchar`、または `nvarchar` カラムの指定された長さよりも長い文字列は、`string_rtruncation` オプションを `on` に設定しない限り暗黙的にトランケートされます。

#### `text`、`unitext`、`image` カラムへの挿入

`text`、`ortext`、または `image` カラムに `NULL` を `insert` (挿入) しても、領域がテキスト・ポインタに割り当てられるだけです。`update` を使用して、有効なテキスト・ポインタをカラムに割り当ててください。

#### `insert` トリガ

特定のテーブルで `insert` コマンドを実行すると、特定の動作を行うトリガを定義できます。

#### コンポーネント統合サービスが有効な場合の `insert` の使用

`insert` は言語イベントとして、またはパラメータ化した動的文としてリモート・サーバに転送できます。

#### 別のテーブルから選択したローの挿入

1 つの文を使用して、テーブルからローを選択して同じテーブルに挿入できます (例 4 を参照)。

`select` を使用して、`null` 値のフィールドがあるテーブルから、`null` 値を許可しないテーブルにデータを挿入するには、元のテーブルの `null` エントリに値を代入する必要があります。たとえば次の例では、`null` 値を許可しない `advances` テーブルにデータを挿入するために、`null` フィールドに `0` を代入しています。

```
insert advances
select pub_id, isnull (advance, 0) from titles
```

`isnull` 関数を使用しない場合、このコマンドは `null` 以外の値を持つすべてのローを `advances` テーブルに挿入します。ただし、`titles` テーブルの `advance` カラムに `null` が含まれている場合は、すべてのローに対してエラー・メッセージを生成します。

このようなデータ値の置き換えを行わないと、`null` 値を含むデータを `not null` 指定のカラムに挿入できません。

2つのテーブルの構造を同じにして、一部のフィールドで `null` 値を使用できるかどうかを別々に設定することができます。テーブル内のカラムの `null` タイプを表示するには、`sp_help` を使用してください。

#### トランザクションと `insert`

連鎖トランザクション・モードを設定すると、現在アクティブ状態のトランザクションがない場合は、`insert` 文を使用してトランザクションが暗黙的に開始されます。挿入を完了させるには、トランザクションをコミットするか、または変更内容をロールバックする必要があります。次に例を示します。

```
insert stores (stor_id, stor_name, city, state)
values ('999', 'Books-R-Us', 'Fremont', 'AZ')
if exists (select t1.city
from stores t1, stores t2
where t1.city = t2.city
and t1.state = t2.state
and t1.stor_id < t2.stor_id)
rollback transaction
else
commit transaction
```

連鎖トランザクション・モードでは、上記のバッチによってトランザクションが開始され、`stores` テーブルに新しいローが挿入されます。テーブル内の別の店と同じ都市および州の情報を持つローが挿入されると、変更内容は `stores` テーブルにロールバックされ、トランザクションが終了します。それ以外の場合は、挿入がコミットされて、トランザクションが終了します。連鎖トランザクション・モードの詳細については、『*Transact-SQL ユーザーズ・ガイド*』を参照してください。

## IDENTITY カラムへの値の挿入

- テーブルにローを挿入する場合、カラム・リストに **IDENTITY** カラムの名前を入れたり、値リストに **IDENTITY** カラムに対する値を入れたりしないでください。テーブルが **IDENTITY** カラム 1 つで構成されている場合、カラム・リストを省略し、次のように値リストを空にします。

```
insert id_table values ()
```

- テーブルに初めてローを挿入するとき、Adaptive Server は **IDENTITY** カラムに 1 の値を割り当てます。新しく挿入するローには、直前の値よりも 1 大きいカラム値が設定されます。この値は、**create table** 文または **alter table** 文でカラムに宣言されたデフォルトや、**sp\_bindefault** によってカラムにバインドされたデフォルトよりも優先されます。

サーバの障害によって、**IDENTITY** カラムの値にギャップが生じることがあります。誤差の最大サイズは、**identity burning set factor** 設定パラメータの設定によって異なります。誤差は、手動で **IDENTITY** カラムにデータを挿入した場合、ローを削除した場合、およびトランザクションのロールバックが原因で発生することもあります。

- **IDENTITY** カラムの基本テーブルに **identity\_insert table\_name on** を設定した後は、テーブル所有者またはテーブルに対する **insert** パーミッションを持つユーザだけが、**IDENTITY** カラムに明示的に値を挿入できます。テーブル所有者がテーブルに対する **identity\_insert** が設定されたユーザである場合のみ、そのテーブルの **identity\_insert table\_name on** を設定できます。ユーザは、一度にデータベース内の 1 つのテーブルに対して **identity\_insert table\_name on** を設定できます。**identity\_insert** が **on** に設定されている場合は、各 **insert** 文にカラム・リストを入れて、**IDENTITY** カラムに明示的な値を指定する必要があります。

**IDENTITY** カラムに値を挿入すると、カラムへの初期値の指定や、エラー時に削除されたローのリストアを実行できます。

**IDENTITY** カラムでユニーク・インデックスを作成していないかぎり、Adaptive Server は値の一意性を検証しません。したがって、正の整数であれば何でも挿入することができます。

明示的な値を **IDENTITY** カラムに挿入するには、テーブル所有者、データベース所有者、またはシステム管理者は、値を挿入するビューではなく、カラムのベース・テーブルに **identity\_insert table\_name on** を設定する必要があります。

- **IDENTITY** カラムに挿入できる最大値は、数値の場合は **10 precision - 1** です。整数の識別子の場合、この値はデータ型で使用できる最大値（たとえば **tinyint** では **255**、**smallint** では **32767**）です。**IDENTITY** カラムがこの値に達すると、後続の **insert** 文は、現在のトランザクションをアボートするエラーを返します。

その場合は、**create table** 文を使用して、古いテーブルと同じテーブルを作成し、**IDENTITY** カラムの精度をより大きくします。新しいテーブルの作成後、**insert** 文または **bcp** ユーティリティのどちらかを使用して、古いテーブルから新しいテーブルにデータをコピーします。

- **IDENTITY** カラムに最後に挿入した値を取得するには、**@@identity** グローバル変数を使用します。**IDENTITY** カラムのないテーブルが、最後に実行された **insert** 文や **select into** 文の影響を受けた場合、**@@identity** は **0** を返します。
- 選択、および結果テーブルへの挿入が行われる **IDENTITY** カラムは、**IDENTITY** プロパティの継承に関して次の規則に従います。
  - **IDENTITY** カラムは、複数回選択されると、新しいテーブル内で **not null** として定義される。**IDENTITY** プロパティは継承されない。
  - 式の一部として **IDENTITY** カラムが選択される場合、結果カラムは **IDENTITY** プロパティを継承しない。**null** を指定できるカラムが式に含まれている場合には、結果カラムは **null** として作成される。それ以外の場合は **not null** として作成される。
  - **select** 文に **group by** 句または集合関数がある場合、結果カラムは **IDENTITY** プロパティを継承しない。**IDENTITY** カラムの集約が含まれているカラムは **null** として作成され、その他のカラムは **not null** として作成される。
  - **union** または **ジョイン** を使用してテーブルに選択された **IDENTITY** カラムでは、**IDENTITY** プロパティが保持されない。テーブル内に **IDENTITY** カラムや **null** カラムの **union** がある場合、新しいカラムは **null** として定義される。それ以外は **not null** として定義される。

#### ビューを介したデータの挿入

- **with check option** 付きでビューが作成される場合、ビューを介して挿入される各ローは、ビューの選択基準を満たしている必要があります。

たとえば、`stores_cal` ビューには、`state` の値が “CA” である `stores` テーブルのすべてのローが含まれます。

```
create view stores_cal
as select * from stores
where state = "CA"
with check option
```

`with check option` 句は、各 `insert` 文がビューの選択基準を満たしているかどうかを検査します。`state` に “CA” 以外の値を持つローは拒否されます。

- `with check option` 付きでビューを作成する場合は、*base* ビューをもとに作成したすべてのビューが、基になるビューの選択基準を満たしている必要があります。派生したビューを介して挿入される新しいローは、基になるビューから参照できなければなりません。

`stores_cal` から抽出されたビュー `stores_cal30` を考えてみます。新しいビューには、“Net 30” という支払期限が指定されているカリフォルニア州にある支店の情報が含まれます。

```
create view stores_cal30
as select * from stores_cal
where payterms = "Net 30"
```

`stores_cal` は `with check option` を指定して作成されたため、`stores_cal30` を介して挿入または更新されたすべてのローは、`stores_cal` を通して参照できる必要があります。`state` 値が “CA” 以外であるローは拒否されます。

`stores_cal30` には、それ自体の `with check option` 句がありません。これは、“Net 30” 以外の `payterms` 値を持つローを、`stores_cal30` を介して挿入または更新できることを示します。次の `update` 文は、`stores_cal30` を通してローを参照することはできなくなりますが、正常に実行されます。

```
update stores_cal30
set payterms = "Net 60"
where stor_id = "7067"
```

- `with check option` で作成されたジョイン・ビューでは `insert` 文を使用できません。
- ジョイン・ビューからローを挿入または更新した場合、影響を受けるすべてのカラムは同一のベース・テーブルに属する必要があります。

### 挿入パフォーマンス向上のためにテーブルを分割する

クラスタード・インデックスを持たず、分割されていないテーブルは1つの双方向データベース・ページ・チェーンで構成されているため、テーブルへの挿入はそのチェーンの最終ページが使用されます。Adaptive Server は、他の同時トランザクションがデータをテーブルに挿入できないようにブロックするため、ローの挿入中は最終ページに排他ロックをかけます。

`alter table` コマンドの `partition` 句を使用してテーブルを分割すると、追加ページ・チェーンが作成されます。それぞれのチェーンには、同時挿入オペレーションに使用できる最終ページがあります。これによってページの競合が少なくなり、挿入パフォーマンスが向上します。また、テーブルが複数の物理デバイスに分割されている場合も、パーティションに分割することで、サーバがキャッシュからディスクへデータをフラッシュするときの I/O 競合が減り、挿入操作のパフォーマンスが向上します。テーブルを分割して `insert` のパフォーマンスを強化する方法の詳細については、『パフォーマンス & チューニング・ガイド：基本』の「データの物理的配置の制御」を参照してください。

### 標準規格

ANSI SQL — 準拠レベル：初級レベル

Transact-SQL 拡張機能は次のとおりです。

- `insert` 文の `select` 部に `union` 演算子を指定。
- データベース名によってテーブル名またはカラム名を修飾する。
- ジョインを含むビューによってデータを挿入する。

---

**注意** FIPS フラガは、ジョインを含んでいるビューからは挿入を検出しません。

---

### パーミッション

`insert` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であるか <code>insert</code> パーミッションを持つユーザであることが必要。テーブル所有者または <code>insert</code> および <code>identity_insert</code> パーミッションを持つユーザのみがテーブルの <code>IDENTITY</code> カラムの挿入を実行できる。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者であるか <code>sa_role</code> が付与されたユーザであることが必要。  <code>insert</code> パーミッションは、デフォルトではデータベース所有者にあります。テーブルの <code>IDENTITY</code> カラムに対する <code>insert</code> のパーミッションは、テーブル所有者、データベース所有者、およびシステム管理者に限定されます。

監査 sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
41	insert	テーブルへの insert	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> <ul style="list-style-type: none"> <li>• insert の場合 – INSERT</li> <li>• select into の場合 – INSERT INTO の後ろに完全修飾されたオブジェクト名が続く</li> </ul> </li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>
42	insert	ビューへの insert	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – INSERT</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

#### 参照

コマンド [alter table](#), [create default](#), [create index](#), [create rule](#), [create table](#), [create trigger](#), [dbcc](#), [delete](#), [select](#), [update](#)

データ型 『リファレンス・マニュアル：ビルディング・ブロック』の「第 1 章 システム・データ型とユーザ定義データ型」

システム・プロシージャ [sp\\_bindefault](#), [sp\\_bindrule](#), [sp\\_help](#), [sp\\_helppartition](#), [sp\\_unbindefault](#), [sp\\_unbindrule](#)

ユーティリティ [bcp](#)

## kill

**説明** プロセスを強制終了します。

**構文** `kill spid with statusonly`

**パラメータ** *spid*

強制終了するプロセスの ID 番号です。*spid* は定数である必要があります。ストアド・プロシージャへのパラメータとして渡したり、ローカル変数として使用することはできません。プロセスのリストおよびその他の情報を参照するには、`sp_who` を使用してください。

**with statusonly**

ロールバック・ステータスであるサーバ・プロセス ID (*spid*) の進捗状況についてレポートします。指定した *spid* は強制終了されません。`statusonly` レポートには、ロールバックの完了率と完了までにかかる推定時間 (秒単位) が表示されます。

**例** **例 1** プロセス番号 1378 を強制終了します。

```
kill 1378
```

**例 2** たとえば、次の例は *spid* 番号 13 のロールバック・プロセスについてレポートします。

```
kill 13 with statusonly
spid:13 Transaction rollback in progress.Estimated rollback completion:17%
Estimated time left:13 seconds
```

ロールバックの進捗状況を追跡するには、`kill...with statusonly` を複数回実行する必要があります。`kill...statusonly` を発行したときに、指定した *spid* のロールバックがすでに完了している場合、または指定した *spid* がロールバックされていない場合は、`kill...statusonly` から次のメッセージが返されます。

```
Status report cannot be obtained.KILL spid:nn is not
in progress.
```

**使用法**

- 次のような現在のプロセスに関するレポートを取得するには、`sp_who` を実行してください。

```
fid spid status      loginame  origname  hostname  blk dbname  cmd
-----
  0  1  recv sleep  bird      bird      jazzy     0  master  AWAITING CO
MMAND
  0  2  sleeping  NULL      NULL               0  master  NETWORK HAN
DLER
  0  3  sleeping  null      null               0  master  MIRROR HAND
LER
  0  4  sleeping  null      null               0  master  AUDIT PROCE
```



```

SS
 0 5 sleeping null null 0 master CHECKPOINT
SLEEP
 0 6 recv sleep rose rose petal 0 master AWAITING CO
MMAND
 0 7 running robert sa helos 0 master SELECT
 0 8 send sleep daisy daisy chain 0 pubs2 SELECT
 0 9 alarm sleep lily lily pond 0 master WAITFOR
 0 10 lock sleep viola viola cello 7 pubs2 SELECT

```

spid カラムには、Transact-SQL kill コマンドで使用されるプロセス ID 番号が格納されます。ブロックしているプロセスがあれば、blk 列に、ブロックしているプロセスのプロセス ID が表示されます。ブロックしている側のプロセス (排他ロックを持つことができるプロセス) には、他のプロセスで必要なリソースが保持されています。この例では、プロセス 10 (テーブル上の select) が、プロセス 7 (同じテーブル上の begin transaction の後の insert) によってブロックされています。

status カラムはコマンドのステータスをレポートします。表 1-24 は、ステータス値と sp\_who の影響を示します。

表 1-24:sp\_who でレポートされるステータスの値

ステータス	説明	kill コマンドの影響
recv sleep	ネットワーク読み込みの待機中。	即時。
send sleep	ネットワーク送信の待機中。	即時。
alarm sleep	waitfor delay "10:00" などの警告の待機中。	即時。
lock sleep	ロック取得の待機中。	即時。
sleeping	ディスク I/O、またはほかのリソースの待機中。ほとんどの場合、これは、実行されてはいるが大容量のディスク I/O が行われていることを示す。	通常、プロセスは「ウェイクアップ」するとただちに強制終了される。スリープしているプロセスの中にはウェイクアップしないものがあり、Adaptive Server を再起動してクリアする必要がある。
runnable	実行可能なプロセスのキュー内にある。	即時。
running	サーバ・エンジンのいずれかで実行中。	即時。
infected	Adaptive Server が重大なエラー条件を検出した。発生することはほとんどない。	kill コマンドの実行はすすめられない。Adaptive Server は、クリアする必要があると思われるプロセスを再起動する。
background	ユーザ・プロセスによってではなく Adaptive Server によって実行される、スレッシュホールド・プロシージャなどのプロセス。	即時。細心の注意を払って kill を実行すること。バックグラウンド・プロセスを強制終了する前に、sysprocesses を十分に確認することをすすめる。

ステータス	説明	kill コマンドの影響
log suspend	ログでラストチャンス・スレッシュホルドに達したために中断されているプロセス。	即時。

- 現在のロックと、ロックを保持しているプロセスの *spid* に関するレポートを取得するには、`sp_lock` を使用します。
- クラスタ環境で、権限のある Kerberos ユーザは `kill` コマンドを使用して、リモート・インスタンスの DBMS タスクの *spid* を停止できます。

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

**パーミッション** `kill` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、自身のプロセスを強制終了するには、 <code>kill</code> 権限が必要。別のユーザのプロセスを強制終了するには <code>kill any process</code> 権限が必要。
---------------	---

細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> が付与されたユーザであることが必要。 <code>kill</code> 権限は譲渡できません。
---------------	--

**参照** コマンド [shutdown](#)

システム・プロシージャ [sp\\_lock](#), [sp\\_who](#)

## load database

### 説明

`dump database` で作成されたトランザクション・ログなどのユーザ・データベースのバックアップ・コピーをロードし、データベース・ダンプとともにロードされたアーカイブ・データベースをマテリアライズします。

`load database` オペレーションのターゲット・プラットフォームは、`dump database` オペレーションを実行するソース・プラットフォームと同じプラットフォームである必要はありません。`dump database` と `load database` をビッグ・エンディアン・プラットフォームからリトル・エンディアン・プラットフォームに対して (またはリトル・エンディアン・プラットフォームからビッグ・エンディアン・プラットフォームに対して) 実行できます。

サイトで Tivoli Storage Manager がライセンスされている場合の `load database` 構文については、『IBM Tivoli Storage Manager と Backup Server の使用』を参照してください。

### 構文

定期的なデータベースのロードを実行します。

```
load database database_name
  from stripe_device
    [at backup_server_name]
    [density = density_value,
    blocksize = number_bytes,
    dumpvolume = volume_name,
    file = file_name]
    with verify only [= header | full]
  [stripe on stripe_device
    [at backup_server_name]
    [density = density_value,
    blocksize = number_bytes,
    dumpvolume = volume_name,
    file = file_name]
  [[stripe on stripe_device
    [at backup_server_name]
    [density = density_value,
    blocksize = number_bytes,
    dumpvolume = volume_name,
    file = file_name],...]
  [with {
    listonly=load_sql | create_sql,
    density = density_value,
    blocksize = number_bytes,
    dumpvolume = volume_name,
    file = file_name,
    [dismount | nodismount],
    [nounload | unload],
    passwd = password,
```

```
until_time = datetime,  
notify = {client | operator_console},  
[override]]]
```

バックアップをロードしないでヘッダ情報またはファイル情報を調べます。

```
load database database_name  
from [compress::stripe_device  
[at backup_server_name  
[density = density_value,  
blocksize = number_bytes,  
dumpvolume = volume_name,  
file = file_name]  
[stripe on [compress::stripe_device  
[at backup_server_name  
[density = density_value,  
blocksize = number_bytes,  
dumpvolume = volume_name,  
file = file_name]  
[[stripe on [compress::stripe_device  
[at backup_server_name  
[density = density_value,  
blocksize = number_bytes,  
dumpvolume = volume_name,  
file = file_name]]...]  
[with {  
density = density_value,  
blocksize = number_bytes,  
dumpvolume = volume_name,  
file = file_name,  
[dismount | nodismount],  
[nounload | unload],  
passwd = password,  
listonly [= full],  
headeronly,  
notify = {client | operator_console}  
}]
```

アーカイブ・データベースをマテリアライズします。

```
load database database_name  
from dump_device  
[ [stripe on stripe_device] ...]  
[with [norecovery,][passwd=password]
```

データベースを指定した時点までリストアする load database SQL 文のシーケンスを生成します。

```
load database database_name  
[from stripe_device]  
with listonly=[load_sql | create_sql | volume]
```

サイトで Tivoli Storage Manager がライセンスされている場合はデータベースのコピーをロードします。

```
load database database_name
  from syb_tsm::[[-S source_sever_name][[-D source_database_name]
    ::]object_name [blocksize = number_bytes]
  [stripe on syb_tsm::[[-S source_sever_name]
    [-D source_database_name::]object_name
    [blocksize = number_bytes]]
  [[stripe on syb_tsm::[[-S source_sever_name]
    [-D source_database_name::]object_name
    [blocksize = number_bytes]]...]
  [with {
    blocksize = number_bytes,
    passwd = password,
    listonly [= full],
    headeronly,
    notify = {client | operator_console},
    [[verifyonly | verify] [= header | full]]
  }]
```

#### パラメータ

##### *database\_name*

バックアップ・コピーを受け取るデータベースの名前。for load オプションを使用して作成されたデータベースか、既存のデータベースのいずれかにすることができます。ダンプされたデータを既存のデータベースにロードすると、すべての既存のデータが上書きされます。受け取り側のデータベースは、少なくとも、ダンプされるデータベースと同等の大きさでなければなりません。データベース名は、リテラル、ローカル変数、またはストアド・プロシージャのパラメータとして指定できます。

アーカイブ・データベースの場合、*database\_name* はロード先のアーカイブ・データベースの名前です。

##### compress::

アーカイブされたデータベースの圧縮を解除します。compress オプションの詳細については、『システム管理ガイド』の「ユーザ・データベースのバックアップとリストア」を参照してください。

---

**注意** ネイティブの "compression = *compress\_level*" オプションを使用し、古い "compress::*compression\_level*" オプションは使用しないことを推奨します。dump database でネイティブのオプションを使用する場合は、データベースをロードするときに "compress::*compression\_level*" を使用する必要はありません。

---

**from dump\_device**

ダンプのロード元のディスク・データベース・ダンプの名前を指定します。

**from stripe\_device**

データのロード元のデバイス名です。ダンプ・デバイスの指定時に使用する形式については、「[ダンプ・デバイスの指定](#)」(588 ページ)を参照してください。サポートされているダンプ・デバイスのリストについては、Adaptive Server の『インストール・ガイド』および『設定ガイド』を参照してください。

**at backup\_server\_name**

ダンプ・デバイスが接続されているマシン上で稼働しているリモート Backup Server の名前です。interfaces ファイルを使用するプラットフォームでは、*backup\_server\_name* が interfaces ファイル内に表示されます。

**listonly = [load\_sql | create\_sql | volume]**

次のように生成します。

- **load\_sql** — 指定された時点のリストアを実行する load database または load transaction SQL コマンドのシーケンス。
- **create\_sql** — バックアップ履歴から取得された最新のダンプ・イメージの disk init/sp\_cacheconfig、disk init、create/alter database および create/alter database のシーケンス。

ストライプ・デバイスからのデータベースのロード時に with listonly=create\_sql を使用すると、このオプションによってダンプ・イメージから disk init/sp\_cacheconfig、disk init、create または alter database のシーケンスを表示。

- **volume** — ダンプ・イメージからボリューム情報を表示。

**density = density\_value**

このオプションは無視されます。詳細については、[dump database](#) コマンドを参照してください。

**blocksize = number\_bytes**

ダンプ・デバイスのデフォルトのブロック・サイズを上書きします。UNIX システムでブロック・サイズを指定する場合、ダンプの作成に使用したブロック・サイズと同じでなければなりません。詳細については、[dump database](#) コマンドを参照してください。

`dumpvolume = volume_name`

ANSI テープ・ラベルのボリューム名フィールドです。`load database` は、テープがオープンされるとこのラベルを検査し、誤ったボリュームがロードされている場合はエラー・メッセージを生成します。

---

**注意** `load database` の使用時に、`file=filename` オプションで指定されたファイル名が正しくなくても、`dumpvolume` オプションはエラー・メッセージを出力しません。`Backup Server` は、マウントされているテープが間違っている場合でも、テープ全体を検索してそのファイルを探します。

---

`file = file_name`

テープ・ボリューム上での特定のデータベース・ダンプの名前です。ダンプの作成時にダンプ・ファイル名を記録しなかった場合は、`listonly` を使用して、すべてのダンプ・ファイルの情報を表示してください。

`stripe on stripe_device`

追加のダンプ・デバイスです。`to stripe_device` 句で指定したデバイスを含め、32 個までのデバイスを使用できます。`Backup Server` は、すべてのデバイスから同時にデータをロードします。これによって、必要なボリューム変更の時間と回数が減少されます。詳細については、「[ダンプ・デバイスの指定](#)」(588 ページ)を参照してください。

`dismount | nodismount`

(論理的なマウント解除をサポートするプラットフォームで)テープをマウントしたままにするかどうかを指定します。デフォルトでは、ロードの完了時に、ロードに使用したすべてのテープのマウントが解除されます。追加のロードまたはダンプに引き続きテープを使用できるようにするには、`nodismount` を使用します。

`nounload | unload`

ロードの完了後にテープを巻き戻すかどうかを指定します。デフォルトではテープは巻き戻されません。したがって、同じテープ・ボリュームからロードを追加できます。マルチダンプ・ボリュームからロードされる最後のダンプ・ファイルには、`unload` を指定します。これによって、ロードの完了時にテープが巻き戻されアンロードされます。

`with [norecovery,]`

`load database` コマンドによりリカバリを実行せず、`load database` コマンドが完了した後データベースを自動的にオンラインにするアーカイブ・データベースをいつマテリアライズするのかを指定します。

**passwd = password**

不正なユーザからダンプ・ファイルを保護するために指定するパスワードです。パスワードの長さは、6～30文字にする必要があります。パスワードに変数を使用できません。パスワードの規則については、『システム管理ガイド 第1巻』の「Adaptive Server のログイン、データベース・ユーザ、クライアント接続の管理」を参照してください。

**until\_time = datetime**

(指定時点までの)ロードに使用されるロード・シーケンスを生成します。

**listonly [= full]**

テープ・ボリューム上のすべてのダンプ・ファイルの情報を表示しますが、データベースはロードしません。**listonly** は、データベースとデバイス、ダンプが行われた日付と時刻、上書きできる日付と時刻を識別します。**listonly = full** は、ダンプの補足情報を提供します。どちらのレポートも ANSI テープ・ラベル別にソートされます。

Backup Server は、ボリューム上でファイルのリストを作成した後、ボリューム変更要求を送信します。オペレータは、別のテープ・ボリュームをマウントするか、またはすべてのダンプ・デバイスのリスト・オペレーションを終了できます。

現在の実装では、**listonly** オプションは **headeronly** オプションを上書きします。

---

**警告！** 1/4 インチ・カートリッジ・テープでは、load database with **listonly** を使用しないでください。

---

**with verify[only][=header | full]**

アーカイブにコピーするデータ・ページに対して最小限のヘッダ検査またはロー構造検査を実行します。ただし、データベースのロードは実行しません。この時点では、**gam**、**oam**、**allocation pages**、**indexes**、**text**、**log** ページに構造検査は実行されません。ページ番号がページ・ヘッダと一致するページに対して、他の検査だけが実行されます。



**headeronly**

単一のダンプ・ファイルのヘッダ情報を表示しますが、データベースはロードしません。file = *file\_name* オプションを使用して別のファイル名を指定しないかぎり、**headeronly** はテーブル上の最初のファイルについての情報を表示します。ダンプ・ヘッダは、以下の情報を提供します。

- ダンプのタイプ (データベースまたはトランザクション・ログ)
- データベース ID
- ファイル名
- ダンプが実行された日付
- 文字セット
- ソート順
- ページ数
- 次のオブジェクト ID

**notify = {client | operator\_console}**

デフォルトのメッセージ送信先を上書きします。

- オペレータ端末機能を提供するオペレーティング・システムでは、ボリューム交換メッセージは、**Backup Server** が実行されているマシンのオペレータ端末に常に送信されます。**dump database** を開始した端末セッションに **Backup Server** の他のメッセージを送信するには、**client** を使用します。
- オペレータ端末機能が備わっていないオペレーティング・システム (UNIX など) では、**dump database** を開始したクライアントにメッセージが送信されます。**Backup Server** が実行されている端末にメッセージを送信するには、**operator\_console** を使用してください。

**override**

他のデータベースのカラムを暗号化する暗号化キーを含んでいるデータベースを正常にロードするには、**with override** を使用する必要があります。

**syb\_tsm::object\_name**

**Backup Server** と **TSM** 間の通信を有効にする **libsyb\_tsm.so** モジュールを起動するキーワードです。

**-S *source\_server\_name***

送信元の Adaptive Server がターゲットの Adaptive Server と同じではない場合に送信元 Adaptive Server の名前を指定します。このパラメータは、ロード操作対象のターゲットであるサーバが、ダンプ操作で使用される送信元のサーバと異なるときに必要になります。

**-D *source\_database\_name***

送信元のデータベースがターゲットのデータベースと同じではない場合に送信元データベースの名前を指定します。このパラメータは、ロード操作対象のターゲットであるデータベースが、ダンプ操作で使用される送信元のデータベースと異なるときに必要になります。

**例 1** テープ・デバイスから pubs2 データベースを再ロードします。

```
load database pubs2
from "/dev/nrmt0"
```

**例 2** Backup Server REMOTE\_BKP\_SERVER を使用して、pubs2 データベースをロードします。このコマンドでは 3 つのデバイスを指名します。

```
load database pubs2
from "/dev/nrmt4" at REMOTE_BKP_SERVER
stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
stripe on "/dev/nrmt0" at REMOTE_BKP_SERVER
```

**例 3** pubs2 データベースを、*/opt/bin/Sybase/dumps* に配置されている *dmp090100.dmp* という名前の圧縮ダンプ・ファイルからロードします。

```
load database pubs2 from
"compress::/opt/bin/Sybase/dumps/dmp090100.dmp"
```

**例 4** key\_db データベースをロードします。このデータベースは暗号化キーが含まれています。key\_db の暗号化キーを使用して他のデータベースのカラムを暗号化した場合は、with override を使用する必要があります。

```
load database key_db from "/tmp/key_db.dat" with
override
```

**例 5** “syb\_tsm::obj1.2” から testdb データベースをロードします。関連する dump コマンドについては、[「dump database」\(424 ページ\)](#) を参照してください。

```
load database testdb from "syb_tsm::obj1.2"
stripe on "syb_tsm::obj1.2"
stripe on "syb_tsm::obj1.2"
stripe on "syb_tsm::obj1.2"
stripe on "syb_tsm::obj1.2"
```

**例 6** 関連する dump コマンドの送信元データベース (testdb) が load コマンドのターゲット・データベース (pubs2) と異なる場合に、TSM バックアップ・オブジェクト "obj1.1" から pubs2 データベースをロードします。

```
load database pubs2 from "syb_tsm::-D testdb::obj1.1"
```

**例 7** testdata1、testdata2、testdata3、testlog4 および testlog5 と命名された 5 つのデータベース・デバイス上にデータベース testdb が作成されます。

```
disk init name = 'testdata1', physname='/tmp/t_dat1',size='10M'
go
disk init name='testdata2',physname='/tmp/t_dat2',size='10M'
go
disk init name='testdata3',physname='/tmp/t_dat3',size='10M'
go
disk init name='testlog4',physname='/tmp/t_log4',size='10M'
go
disk init name='testlog5',physname='/tmp/t_log5',size='10M'
go
create database testdb on testdata1='10M', testdata2='8M', testdata3='5M'
    log on testlog4='6M',testlog5 with override
go
alter database testdb on testdata3 = '5M'
go
alter database testdb log on testlog4 = '2M'
go
```

データベース testdb のダンプが dump database を使用して取得されます。これによって補足のデータベース・デバイス情報がダンプ・ヘッダに書き込まれます。

```
dump database testdb to "test.dmp"
go
```

with headeronly オプションを指定し、ダンプ・イメージ test.dmp を使用してデータベース testdb をロードすると、ダンプ・ヘッダの内容が表示されます。これによって、データベース・デバイスに関する補足情報が表示されます。

```
1> load database testdb from "test.dmp" with headeronly
2> go
Backup Server:6.28.1.1:Dumpfile name 'test1025109FD6 ' section number 1
mounted on disk file '/punedbaccess3_dev3/kelkara/backupserver/test.dmp'
...
dbdevinfo:vdevno=1 devname=testdata1 path=/tmp/test1.dat db_size=10485760
device_size=20967424
```

```

dbdevinfo:vdevno=2 devname=testdata2 path=/tmp/test2.dat db_size=8388608
device_size=20967424
dbdevinfo:vdevno=3 devname=testdata3 path=/tmp/test3.dat db_size=10485760
device_size=20967424
dbdevinfo:vdevno=4 devname=testlog4 path=/tmp/test4.dat db_size=8388608
device_size=20967424
dbdevinfo:vdevno=5 devname=testlog5 path=/tmp/test5.dat db_size=6291456
device_size=20967424
...

```

データベース・デバイス情報は、vdevno、devname、path、db\_size および device\_size で構成されます。device\_size は、disk init コマンドの実行時点で割り振られているデバイスの合計サイズです。db\_size は、データベース testdb が使用するデバイスのサイズです。create\_sqlgenddonly オプションを指定し、ダンプ・イメージ test.dmp を使用してデータベース testdb をロードすると、create/alter database コマンドのシーケンスが表示されます。このシーケンスを使用すると、dump コマンド実行時点のソース・データベースと同じ data/log セグメント・レイアウトでターゲット・データベースを作成できます。この出力をファイルに転送すると、ターゲットのデータベースを作成する isql コマンド・スクリプトを生成できます。

```

1> load database test from "test.dmp" with listonly=create_sql
2> go
DISK INIT
      name = 'testdata1'
      , physname = '/tmp/t_dat1'
      , size = '10M'
go
DISK INIT
      name = 'testdata2'
      , physname = '/tmp/t_dat2'
      , size = '10M'
go
DISK INIT
      name = 'testdata3'
      , physname = '/tmp/t_dat3'
      , size = '10M'
go
DISK INIT
      name = 'testlog4'
      , physname = '/tmp/t_log4'
      , size = '10M'
go
DISK INIT
      name = 'testlog5'

```

```

        , physname = '/tmp/t_log5'
        , size = '10M'
go

CREATE DATABASE testdb
ON testdata1 = '10M'
, testdata2 = '8M'
, testdata3 = '5M'
LOG ON testlog4 = '6M'
, testlog5 = '6M'
go
ALTER DATABASE testdb
ON testdata3 = '5M'
LOG ON testlog4 = '2M'
go

```

**例 8** 使用可能な最新のダンプを使用した特定データベースのリストアに必要なロード・コマンド・シーケンスを表示します。ダンプ履歴ファイルのダンプ・レコードが読み込まれ、ロード・シーケンスが準備されます。

```

1> load database testdb with listonly=load_sql
2>go
LOAD DATABASE testdb FROM '/dumpdir/testdb_DB_1.1.dmp'
STRIPE ON '/dumpdir/testdb_DB_1.2.dmp'
STRIPE ON '/dumpdir/testdb_DB_1.3.dmp'
go
LOAD TRANSACTION testdb FROM '/dumpdir/testdb_XACT_2.dmp'
go
LOAD TRANSACTION testdb FROM '/dumpdir/testdb_XACT_3.dmp'
go
LOAD TRANSACTION testdb FROM '/dumpdir/testdb_XACT_4.1.dmp'
STRIPE ON '/dumpdir/testdb_XACT_4.2.dmp'
go

```

#### 使用法

- `sp_hidetext` の後で、プラットフォーム間の `dump` と `load` を使用した場合は、すべての隠しオブジェクトを手動で削除して再作成する必要があります。
- `listonly` オプションと `headeronly` オプションを指定すると、ダンプ・ファイルがロードされずにその情報が表示されます。
- ダンプとロードは Backup Server を介して実行されます。

- データベースを正しく同期させて、再ロードしたプライマリ・データベースの内容に適したスキーマがすべてのプロキシ・テーブルで定義されるようにするには、プロキシ・データベースをホストするサーバ上で `alter database dbname for proxy_update` コマンドを実行する必要があります。
- [表 1-25](#) は、バックアップからのデータベースのリストアに使用されるコマンドとシステム・プロシージャを示しています。

表 1-25: ダンプからのデータベースのリストアに使用するコマンド

使用するコマンド	目的
<code>create database for load</code>	ダンプをロードするためのデータベースを作成する。
<code>load database</code>	データベースをダンプからリストアする。
<code>load transaction</code>	リストアしたデータベースに前回のトランザクションを適用する。
<code>online database</code>	通常のロード処理の後、または、データベースを現在のバージョンの Adaptive Server にアップグレードした後で、データベースを誰でも使用できるようにする。
<code>load {database   transaction} with {headeronly   listonly}</code>	テープ上のダンプ・ファイルを識別する。
<code>sp_volchanged</code>	Backup Server のボリューム変更のメッセージに応答する。

- 暗号化されたカラムのあるデータベースのロードの詳細については、「[暗号化カラムと dump database](#)」(445 ページ)の「[dump database](#)」のセクションを参照してください。

#### 制限事項

- コンポーネント統合サービスに限り、データベース内のプロキシ・テーブルはデータベース保存セットに含まれます。プロキシ・テーブルの内容は保存されません。ポインタのみが保存およびリストアされます。
- 異なるプラットフォームで作成されたダンプはロードできません。
- バージョン 12.5.4 より前のサーバで生成されたダンプはロードできません。
- データベースにデータベース間の参照整合性制約がある場合、`sysreferences` システム・テーブルは外部データベースの ID 番号ではなく名前を保存します。Adaptive Server では、`load database` を使用してデータベース名を変更したり、別のサーバにデータベースをロードしたりすると、参照整合性は保証されません。

- データベース間の制約を追加または削除した場合や、データベース間の制約を含むテーブルを削除した場合は、その都度、影響を受けたデータベースを両方ともダンプしてください。

---

**警告！** これらのデータベースの古いダンプをロードすると、データベースが破壊される可能性があります。データベースを別の名前でロードするか、またはデータベースを別の **Adaptive Server** に移動するためにデータベースをダンプするには、事前に **alter table** を使用して外部参照整合性制約をすべて削除してください。

---

- **load database** は、ロードされたデータベースに関連する疑わしいページのエントリを **master.sysattributes** からクリアします。
- **load database** はデータベースの既存データを上書きします。
- データベース・ダンプのロード後、データベースをオンラインにするまでに2つのプロセスでさらに時間がかかることがあります。
  - **Backup Server** は、ソース・データベースの領域マップにある未割り付けページに **0** を書き込む。この **0** の書き込みは物理的なロードの一部として組み込まれ、データベースをロードするときに実行される。

ターゲット・データベースがソース・データベースより大きい場合、**Backup Server** がロードを完了した後で、**Adaptive Server** はソース・データベースの領域マップの最上部より上の領域に **0** を書き込む。
  - **dump database** によって操作の開始時に書き込まれたチェックポイントの前で完了したトランザクションは、リカバリでは無視される。トランザクション・ログのアクティブな部分で完了したトランザクションは、リカバリによってロールフォワードされる。ロード・シーケンスの最後の段階である **online database** で、未完了のトランザクションのロールバックが実行される。
- 受信側データベースのサイズは、ロードされるデータベース以上でなければなりません。受信側データベースのサイズが小さすぎると、**Adaptive Server** は必要なサイズを伝えるエラー・メッセージを表示します。
- **null** デバイスからロードすることはできません (UNIX の場合は **/dev/null**)。
- ユーザ定義トランザクションでは、**load database** を使用できません。

- データベースをロードすると、Adaptive Server はダンプ・ファイルのエンディアン・タイプを自動的に調べ、load database コマンドと online database コマンドの実行中に必要なすべての変換を実行します。

Adaptive Server がインデックス・ローを変換すると、インデックス・ローの順序が正しくなくなることがあります。Adaptive Server は、online database の実行中にユーザ・テーブルの次のインデックスをサスペクト (疑わしい) インデックスとしてマークします。

- 全ページ・ロック (APL) テーブルのノンクラスタード・インデックス
- データオンリーロック (DOL) テーブルのクラスタード・インデックス
- DOL テーブルのノンクラスタード・インデックス

プラットフォーム間のダンプ操作とロード操作時、疑わしいパーティションは次のように処理されます。

- 初めての online database コマンドの実行中、エンディアン・タイプの異なる 2 つのプラットフォーム間で load database を実行すると、ハッシュ・パーティションに suspect のマークが付けられます。
- unichar または univarchar パーティション・キーで内部生成されたパーティション条件を持つラウンド・ロビン分割のグローバル・クラスタード・インデックスは、suspect のマークが付けられます。
- データベースをオンラインにした後に、sp\_post\_xpload を使用して疑わしいパーティションとインデックスを修正します。

---

**注意** sp\_post\_xpload ストアド・プロシージャを使用してユーザ・テーブルのインデックスを検査および再構築する方法の詳細については、『リファレンス・マニュアル：プロシージャ』の「システム・プロシージャ」を参照してください。

---

- dump transaction と load transaction をプラットフォーム間で実行することはできません。
- リモート backupserver に対する dump database と load database をプラットフォーム間で実行することはできません。



- パスワードで保護されたダンプ・ファイルをプラットフォーム間でロードすることはできません。
- 解析済み XML オブジェクトに対して `dump database` と `load database` を実行する場合は、`load database` コマンドの実行後にテキストを再度解析する必要があります。
- `dump database` と `load database` はバージョン 11.9 より前のバージョンの Adaptive Servers のプラットフォーム間では実行できません。
- Adaptive Server は、`binary`、`varbinary`、または `image` カラムとして格納されている埋め込みデータ構造を変換することはできません。
- プラットフォーム間で `master` データベースに対する `load database` を実行することはできません。
- ストアド・プロシージャやその他のコンパイル済みオブジェクトは、`load database` の実行後最初に実行されるとき `syscomments` 内の SQL テキストから再コンパイルされます。

テキストから再コンパイルするパーミッションがない場合は、パーミッションを持つ人が `dbcc upgrade_object` でテキストから再コンパイルしてオブジェクトをアップグレードする必要があります。

#### ロード時のユーザのロックアウト

- ロード中にデータベースを使用することはできません。 `load database` によってデータベースのステータスが "オフライン" に設定されます。ステータスが "offline" のデータベースは誰も使用できません。ロード・シーケンスの実行中は、"offline" ステータスにすることにより、ユーザはデータベースにアクセスしたり、変更することができなくなります。
- `online database` が発行されるまで、`load database` によってロードされたデータベースにはアクセスできません。

#### データベース・ダンプとトランザクション・ログ・ダンプのアップグレード

- バージョン 11.9 以降のサーバから現在のバージョンの Adaptive Server にユーザ・データベース・ダンプをリストアおよびアップグレードするには、次の手順に従います。
  - a 最新のデータベース・ダンプをロードします。
  - b 前回のデータベース・ダンプ以降に行ったすべてのトランザクション・ログ・ダンプを順序どおりにロードします。

Adaptive Server は、ダンプが正しいデータベースに正しいシーケンスで確実にロードされるように、それぞれのダンプのタイムスタンプを検査します。

- c **online database** コマンドを発行して、データベースをアップグレードしてパブリックで使用できるようにします。
- d アップグレード直後に、新しくアップグレードされたデータベースをダンプして、**Adaptive Server** の現在のバージョンと一貫性のあるダンプを作成します。

#### ダンプ・デバイスの指定

- ダンプ・デバイスは、リテラル、ローカル変数、またはストアド・プロシージャへのパラメータとして指定できます。
- ローカル・デバイスは次のように指定できます。
  - **sysdevices** システム・テーブルの論理デバイス名
  - 絶対パス名
  - 相対パス名

**Backup Server** は、**Adaptive Server** の現在の作業ディレクトリを使用して相対パス名を解決します。

- ネットワーク間でロードを行う場合は、ダンプ・デバイスの絶対パス名を指定します。パス名は、**Backup Server** が起動しているマシンで有効なものでなければなりません。パス名の中で文字、数字、またはアンダースコア ( \_ ) 以外の文字を使用している場合は、パス名全体を引用符で囲みます。
- ダンプ・デバイスの所有権とパーミッションの問題によって、**load** コマンドが使用できないことがあります。
- それぞれのロードが別の物理デバイスを使用していれば、複数のロード ( またはダンプ ) を同時に実行できます。

#### Backup Server

- **Adaptive Server** と同じマシン上で稼動する **Backup Server** が必要で  
す。また、**Backup Server** は **master..syssservers** テーブルにリスト  
されている必要があります。このエントリはインストール時または  
アップグレード時に作成されます。削除しないでください。
- バックアップ・デバイスが別のマシン上にあり、ネットワーク間  
でロードする場合は、**Backup Server** をリモート・マシン上にもイ  
ンストールしておく必要があります。

#### ボリューム名

- ダンプ・ボリュームは、ANSI テープ・ラベル標準に準拠してラ  
ベル付けされます。ラベルには、論理ボリューム番号とストライ  
プ・セット内のデバイスの位置が含まれています。

- Backup Server はロード中にテープ・ラベルを使用して、ボリュームが正しい順番でマウントされているかどうかを確認します。これにより、ダンプ時に使用したデバイス数より少ないデバイスからロードできます。

---

**注意** ネットワーク間でダンプおよびロードを実行する場合は、各オペレーションに同じ数のストライプ・デバイスを指定しなければなりません。

---

### ダンプ・ボリュームの変更

Backup Server は、現在マウントされているボリュームで問題を検出すると、クライアントまたはそのオペレータ・コンソールにメッセージを送信して、ボリューム交換を要求します。別のボリュームをマウントした後、オペレータは、Backup Server に通知するために、Backup Server と通信可能な Adaptive Server 上で `sp_volchanged` を実行します。

### システム・データベースのリストア

- master データベースのダンプは、master データベースまたはアーカイブ・データベースにしかロードできません。
- システム・データベースをダンプからリストアする手順については、『システム管理ガイド』を参照してください。

### ディスク・ミラーリング

- ロードの初期の段階で、Adaptive Server は、各論理データベースとログ・デバイスのプライマリ・デバイス名を Backup Server に渡します。プライマリ・デバイスのミラーリングが解除されている場合、Adaptive Server は代わりにセカンダリ・デバイス名を渡します。Backup Server によるデータ転送が完了する前に、指名されたデバイスが失敗すると、Adaptive Server はロードをアボートします。
- load database の進行中に、指名されているデバイスのミラーリングを解除しようとする、メッセージが表示されます。disk unmirror を実行しているユーザは、ロードが完了するまで、ロードをアボートするか、または disk unmirror を遅延させることができます。
- Backup Server がプライマリ・デバイスにデータをロードすると、load database がそのデータをセカンダリ・デバイスにコピーします。データベース・デバイスがミラーリングされている場合は、load database コマンドの完了に時間がかかります。

### アーカイブ・データベースのマテリアライズ

アーカイブ・データベースは、データベース・ダンプとともにロードされたときにだけ有用なプレースホルダです。ロード処理では、実際にはページはコピーされませんが、ページ・マッピングを使用してデータベースがマテリアライズされます。

---

**注意** データベース・ダンプをアーカイブ・データベースにロードするときに、Backup Server が実行されている必要はありません。

---

### load database with norecovery の使用

load database コマンドの with norecovery オプションを使用すると、リカバリを行わずにデータベース・ダンプをアーカイブ・データベースにロードすることで、ロードの所要時間を短縮できます。多くのデータベース・ページは、リカバリ中に変更や割り付けを行い、変更済みページ・セクションに格納することができます。そのため、リカバ리를省略すると、消費される変更済みページ・セクションの領域が最小限に抑えられます。with norecovery オプションを使用すると、アーカイブ・データベースをすばやく表示できます。

with norecovery を使用した場合、データベースは自動的にオンラインになります。

ただし、リカバリが必要なデータベースに load database with norecovery を使用すると、トランザクションの一貫性と物理的な一貫性が保たれないままになる可能性があります。物理的な一貫性がないデータベースで dbcc 検査を実行すると、エラーが発生することがあります。

with norecovery を使用してアーカイブ・データベースをロードした場合、アーカイブ・データベースを使用するには sa\_role が付与されているか、データベース所有者の権限が必要です。

### アーカイブ・データベースでの論理デバイスの使用

sp\_addumpdevice を使用すると、アーカイブ・データベースのロードが可能な論理デバイスを作成できます。

```
sp_addumpdevice 'archive database' , 'logical_name' ,  
                'physical_name'
```

このコマンドを実行すると、`load database` コマンドの `dump_device` または `stripe_device` として、`physical_name` の代わりに `logical_name` を使用できるようになります。

---

**注意** 従来のデータベースへのロードのデバイス指定として、または従来のデータベースをダンプする際に、アーカイブ・データベース論理デバイスは使用できません。

---

#### アーカイブ・データベースでの `load database` の制限

`load database` をアーカイブ・データベースに対して使用する場合は、次の制限があります。

- アーカイブ・データベースのデータベース・ダンプは、ローカル・マシンにマウントされたファイル・システムのディスク・ダンプでなければならない。これは、ローカル記憶領域でも NFS 記憶領域でもよい。`load database ... at <remote server>` 構文はサポートされず、テープ上のデータベース・ダンプもサポートされない。
- 異なるアーキテクチャ間のロードはサポートされない。バイト順序を合わせるために、データベース・ダンプおよび `load database` コマンドは同じアーキテクチャで実行する必要がある。
- ダンプしたデータベースのページ・サイズは、アーカイブ・データベースをホストするサーバが使用するページ・サイズと同じにする。
- ダンプが行われたサーバのメジャー・バージョンは、アーカイブ・データベースをホストするサーバのメジャー・バージョン以前のバージョンでなければならない。
- データベース・ダンプが取得されたサーバの文字セットおよびソート順は、アーカイブ・データベースをホストするサーバの文字セットおよびソート順と同じでなければならない。

#### データベースと暗号化カラムのロード

データベース内でカラムを暗号化するキーとは異なるキーをデータベースに保存する場合は、同時に実行されたダンプから両方のデータベースをロードして、ロード後に暗号化カラムが損失しないようにする必要があります。

キーやデータのためにデータベースをロードしたら、両方のデータベースを同時にオンラインにしてください。

メタデータの依存関係は暗号化カラムとそのキー間に存在するため、キー・データベースを異なる名前を持つデータベースにロードする必要はありません。キー・データベースの名前を変更する必要がある場合は、次の手順を実行します。

- 1 暗号化カラムを含むデータベースをダンプする前に、**alter table** を使用してデータを復号化します。
- 2 キーおよび暗号化カラムを含むデータベースをダンプします。
- 3 データベースをロードした後で、**alter table** を使用して、名前を変更したデータベースのキーでデータを再暗号化します。

#### 圧縮データのロード

- 1つのプラットフォームで作成した圧縮テーブルのダンプを、別のプラットフォームにロードすることはできません。
- あらゆる形式の圧縮ローまたは非圧縮ローを含む圧縮テーブルに対する **create index** コマンドは **load transaction** 時に完全にリカバリされます。

#### 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

#### パーミッション

**load database** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、データベース所有者であるか、データベースに対する <b>load database</b> 権限または <b>own database</b> 権限を持つユーザーであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、次の役割が付与されたユーザーであることが必要。 <ul style="list-style-type: none"> <li>• <b>sa_role</b>、または</li> <li>• <b>replication_role</b>、または</li> <li>• <b>oper_role</b></li> </ul>

#### 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
43	load	load database	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

マニュアル 『システム管理ガイド』の「ユーザ・データベースのバックアップとリストア」

コマンド [alter database](#), [dbcc](#), [dump database](#), [dump transaction](#), [load transaction](#), [online database](#)

システム・プロシージャ [sp\\_helpdb](#), [sp\\_helpdevice](#), [sp\\_hidetext](#), [sp\\_volchanged](#)

## load transaction

**説明** dump transaction によって作成されたトランザクション・ログのバックアップ・コピーをロードします。

**構文** 定期的なログのロードを実行します。

```
load tran[saction] database_name
  from [compress::stripe_device
        [at backup_server_name]
        [density = density_value,
         blocksize = number_bytes,
         dumpvolume = volume_name,
         file = file_name]
  [stripe on [compress::stripe_device
             [at backup_server_name]
             [density = density_value,
              blocksize = number_bytes,
              dumpvolume = volume_name,
              file = file_name]
  [[stripe on [compress::stripe_device
             [at backup_server_name]
             [density = density_value,
              blocksize = number_bytes,
              dumpvolume = volume_name,
              file = file_name]]...]]
  [with {
    density = density_value,
    blocksize = number_bytes,
    compression,
    dumpvolume = volume_name,
    file = file_name,
    [dismount | nodismount],
    [nounload | unload],
    notify = {client | operator_console}
  }]]
```

バックアップ・ログをロードしないでヘッダ情報またはファイル情報を調べます。

```
load tran[saction] database_name
  from [compress::stripe_device
        [at backup_server_name]
        [density = density_value,
         blocksize = number_bytes,
         dumpvolume = volume_name,
         file = file_name]
  [stripe on [compress::stripe_device
             [at backup_server_name]
             [density = density_value,
              blocksize = number_bytes,
              dumpvolume = volume_name,
              file = file_name]]...]]
```



```

file = file_name]
[[stripe on [compress::stripe_device
[at backup_server_name]
[density = density_value,
blocksize = number_bytes,
dumpvolume = volume_name,
file = file_name]]...]
[with {
density = density_value,
blocksize = number_bytes,
compression,
dumpvolume = volume_name,
file = file_name,
[dismount | nodismount],
[nounload | unload],
listonly [= full],
headeronly,
notify = {client | operator_console}
until_time = datetime]}]

```

アーカイブ・データベースにトランザクション・ログをロードします。

```

load tran[saction] database_name
from dump_device
[[stripe on stripe_device] ...]

```

(Tivoli Storage Manager のみ) サイトで Tivoli Storage Manager がライセンスされている場合はトランザクション・ログのコピーをロードします。

```

load transaction database_name
from syb_tsm::[[-S source_sever_name][-D source_database_name]
::object_name [blocksize = number_bytes]
[stripe on syb_tsm::[[-S source_sever_name]
[-D source_database_name]]::object_name
[blocksize = number_bytes]
[[stripe on syb_tsm::[[-S source_sever_name]
[-D source_database_name]]::object_name
[blocksize = number_bytes]]...]
[with {
blocksize = number_bytes,
passwd = password,
listonly [= full],
headeronly,
notify = {client | operator_console},
until_time = datetime
}]

```

## パラメータ

**database\_name**

トランザクション・ログのダンプされたバックアップ・コピーからデータを受け取るデータベースの名前です。受け取り側のデータベースのログ・セグメントのサイズは、少なくともダンプされたデータベースのログ・セグメント以上でなければなりません。データベース名は、リテラルやローカル変数として指定できます。または、ストアド・プロシージャに対するパラメータとしても指定できます。アーカイブ・データベースの場合、*database\_name* はトランザクション・ログをロードするアーカイブ・データベースです。

**compress::**

アーカイブされたトランザクション・ログを圧縮解除します。**compress** オプションの詳細については、『システム管理ガイド』の「ユーザ・データベースのバックアップリケーションとリストア」を参照してください。

---

**注意** ネイティブの "**compression = compress\_level**" オプションを使用し、古い "**compress::compression\_level**" オプションは使用しないことを推奨します。dump database でネイティブのオプションを使用する場合は、データベースをロードするときに "**compress::compression\_level**" を使用する必要はありません。

---

**from stripe\_device**

トランザクション・ログのロード元のダンプ・デバイス名です。ダンプ・デバイスの提示に使用する形式については、「[ダンプ・デバイスの指定](#)」(588 ページ) を参照してください。サポートされているダンプ・デバイスのリストについては、Adaptive Server の『インストール・ガイド』および『設定ガイド』を参照してください。

**at backup\_server\_name**

ダンプ・デバイスが接続されているマシン上で稼働しているリモート Backup Server の名前です。interfaces ファイルを使用するプラットフォームでは、interfaces ファイルに *backup\_server\_name* が示されている必要があります。

**from dump\_device**

ローカル・ディスク・トランザクション・ログ・ダンプです。

**density = density\_value**

テープ・デバイスのデフォルトの記録密度を上書きします。このオプションは無視されます。

**blocksize = *number\_bytes***

ダンプ・デバイスのデフォルトのブロック・サイズを上書きします。UNIX システムでブロック・サイズを指定する場合、ダンプの作成に使用したブロック・サイズと同じでなければなりません。

**dumpvolume = *volume\_name***

ANSI テープ・ラベルのボリューム名フィールドです。load transaction コマンドは、テープがオープンされるとこのラベルを検査し、誤ったボリュームがロードされている場合はエラー・メッセージを生成します。

**file = *file\_name***

テープ・ボリューム上での特定のデータベース・ダンプの名前です。ダンプの作成時にダンプ・ファイル名を記録しなかった場合は、listonly を使用して、すべてのダンプ・ファイルの情報を表示してください。

**stripe on *stripe\_device***

追加のダンプ・デバイスです。to *stripe\_device* 句で指定したデバイスを含め、32 個までのデバイスを使用できます。Backup Server は、すべてのデバイスから同時にデータをロードします。これによって、必要なボリューム変更の時間と回数が減少されます。ダンプ・デバイスの指定方法の詳細については、「[ダンプ・デバイスの指定](#) (588 ページ) を参照してください。

**compression**

ロードするログが、リモート・サーバ上でファイルに圧縮されていたことを示します。load transaction で圧縮レベルを指定する必要はありません。

with compression オプションは、圧縮されたログをローカル・ファイルからロードする compress オプションとは異なります。

---

**注意** ネイティブの "compression = *compress\_level*" オプションを使用し、古い "compress::*compression\_level*" オプションは使用しないことを推奨します。dump database でネイティブのオプションを使用する場合は、データベースをロードするときに "compress::*compression\_level*" を使用する必要はありません。

---

**dismount | nodismount**

(論理的なマウント解除をサポートするプラットフォームで)テープをマウントしたままにするかどうかを指定します。デフォルトでは、ロードの完了時に、ロードに使用したすべてのテープのマウントが解除されます。追加のロードまたはダンプに引き続きテープを使用できるようにするには、**nodismount** を使用します。

**nounload | unload**

ロードの完了後にテープを巻き戻すかどうかを指定します。デフォルトではテープは巻き戻されません。したがって、同じテープ・ボリュームからロードを追加できます。マルチダンプ・ボリュームからロードされる最後のダンプ・ファイルには、**unload** を指定します。これによって、ロードの完了時にテープが巻き戻されアンロードされます。

**listonly [= full]**

テープ・ボリューム上のすべてのダンプ・ファイルの情報を表示しますが、トランザクション・ログはロードしません。**listonly** は、データベースとデバイス、ダンプが行われた日付と時刻、上書きできる日付と時刻を識別します。**listonly = full** は、ダンプについての補足情報を提供します。どちらのレポートも ANSI テープ・ラベル別にソートされます。

**Backup Server** は、ボリューム上でファイルのリストを作成した後、ボリューム変更要求を送信します。オペレータは、別のテープ・ボリュームをマウントするか、またはすべてのダンプ・デバイスのリスト・オペレーションを終了できます。

現在の実装では、**listonly** は **headeronly** を上書きします。

---

**警告！** 1/4 インチ・カートリッジ・テープでは、**load transaction with listonly** を使用しないでください。

---

**headeronly**

単一のダンプ・ファイルのヘッダ情報を表示しますが、データベースはロードしません。file = *file\_name* オプションを使用して別のファイル名を指定しないかぎり、headeronly はテーブル上の最初のファイルについての情報を表示します。ダンプ・ヘッダは、以下の情報を提供します。

- ダンプのタイプ (データベースまたはトランザクション・ログ)
- データベース ID
- ファイル名
- ダンプが実行された日付
- 文字セット
- ソート順
- ページ数
- 次のオブジェクト ID
- ログ内のチェックポイント・ロケーション
- 最も古い begin transaction レコードのロケーション
- 新旧のシーケンス日付

**notify = {client | operator\_console}**

デフォルトのメッセージ送信先を上書きします。

- オペレータ端末機能を提供するオペレーティング・システムでは、ボリューム交換メッセージは、Backup Server が実行されているマシンのオペレータ端末に常に送信されます。dump database を開始した端末セッションに Backup Server の他のメッセージを送信するには、client を使用します。
- オペレータ端末機能が備わっていないオペレーティング・システム (UNIX など) では、dump database を開始したクライアントにメッセージが送信されます。Backup Server が実行されている端末にメッセージを送信するには、operator\_console を使用してください。

**until\_time**

トランザクション・ログの指定時間まで、トランザクション・ログをロードします。指定された時間の前にコミットされたトランザクションのみがデータベースに保存されます。

**syb\_tsm**

Backup Server と TSM 間の通信を有効にする `libsyb_tsm.so` モジュールを起動するキーワードです。

**object\_name**

TSM サーバのバックアップ・オブジェクトの名前です。

**-S source\_server\_name**

送信元の Adaptive Server がターゲットの Adaptive Server と同じではない場合に送信元 Adaptive Server の名前を指定します。このパラメータは、ロード操作対象のターゲットであるサーバが、ダンプ操作で使用される送信元のサーバと異なるときに必要になります。

**-D source\_database\_name**

送信元のデータベースがターゲットのデータベースと同じではない場合に送信元データベースの名前を指定します。このパラメータは、ロード操作対象のターゲットであるデータベースが、ダンプ操作で使用される送信元のデータベースと異なるときに必要になります。

**例**

**例 1** pubs2 データベースのテープのトランザクション・ログをロードします。

```
load transaction pubs2
from "/dev/nrmt0"
```

**例 2** Backup Server の `REMOTE_BKP_SERVER` を使用して、pubs2 データベースのトランザクション・ログをロードします。

```
load transaction pubs2
from "/dev/nrmt4" at REMOTE_BKP_SERVER
stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
stripe on "/dev/nrmt0" at REMOTE_BKP_SERVER
```

**例 3** pubs2 のトランザクション・ログを、2008 年 3 月 20 日午前 10 時 51 分 43 秒 866 までロードします。

```
load transaction pubs2
from "/dev/ntmt0"
with until_time = "mar 20, 2008 10:51:43:866am"
```

**例 4** TSM バックアップ・オブジェクト “demo2.1” のトランザクションを `testdb` データベースにロードします。送信元のデータベースとターゲットのデータベースは同じです。詳細については、「[dump transaction](#)」(448 ページ)を参照してください。

```
load transaction testdb from "syb_tsm::demo2.1"
```

**例 5** ターゲット・データベース (pubs2) と送信元データベース (testdb) が異なる場合に、TSM バックアップ・オブジェクト "obj1.1" のトランザクションをロードします。

```
load transaction pubs2 from "syb_tsm::
-D testdb::obj1.1"
```

#### 使用法

- sp\_hidetext の後で、プラットフォーム間の dump と load を使用した場合は、すべての隠しオブジェクトを手動で削除して再作成する必要があります。
- listonly オプションと headeronly オプションを指定すると、ダンプ・ファイルがロードされずにその情報が表示されます。
- ダンプとロードは Backup Server を介して実行されます。
- 表 1-26 は、バックアップからのデータベースのリストアに使用されるコマンドとシステム・プロシージャを示しています。

表 1-26: データベースのリストアに使用するコマンド

使用するコマンド	目的
create database for load	ダンプをロードするためのデータベースを作成する。
load database	データベースをダンプからリストアする。
load transaction	リストアしたデータベースに前回のトランザクションを適用する。
online database	通常のロード処理の後、または、データベースを現在のバージョンの Adaptive Server にアップグレードした後で、データベースを誰でも使用できるようにする。
load {database   transaction} with {headeronly   listonly}	テープ上のダンプ・ファイルを識別する。
sp_volchanged	Backup Server のボリューム変更メッセージに応答する。

#### 制限事項

- バージョン 11.9 より前のサーバで生成されたダンプはロードできません。
- データベースとトランザクション・ログは、同じリリース・レベルのものでなければなりません。
- トランザクション・ログを生成順にロードします。
- null デバイスからロードすることはできません (UNIX の場合は /dev/null)。
- アップグレードを実行する online database コマンドの後で load transaction を使用することはできません。データベースのアップグレードの正しいシーケンスは、load database、load transaction、online database です。

- すべてのトランザクション・ログがロードされるまで、**online database** コマンドを発行しないでください。コマンドの順序は次のとおりです。

a **load database**

b **load transaction** ( 必要なだけ繰り返し )

c **online database**

ただし、データベースへの読み込み専用アクセスを保持したままトランザクション・ログをさらにロードするには ( 典型的な「ウォーム・バックアップ」状況 )、**dump tran for standby\_access** オプションを使用してトランザクション・ダンプを生成します。その後で、読み込み専用アクセス用に **online database for standby\_access** を発行できます。

- ユーザ定義トランザクション内では **load transaction** コマンドを使用できません。

#### データベースのリストア

- データベースをリストアするには、次の手順に従います。
  - 最新のデータベース・ダンプをロードします。
  - 前回のデータベース・ダンプ以降に作成されたすべてのトランザクション・ログ・ダンプを順序どおりにロードします。
  - **online database** を発行して、データベースを誰でも使用できるようにします。
- データベース間の制約を追加または削除した場合や、データベース間の制約を含むテーブルを削除した場合は、影響を受けたデータベースを両方ともダンプしてください。

---

**警告！** これらのデータベースの古いダンプをロードすると、データベースが破壊される可能性があります。

---

- **Adaptive Server** データベースのバックアップとリカバリの詳細については、『システム管理ガイド』を参照してください。

#### 指定時刻までのデータベースのリカバリ

- ロードまたはダンプが可能なほとんどのデータベースで **until\_time** オプションを使用できます。ただし **until\_time** は、データとログが同じデバイス上にある **master** などのデータベースには適用できません。このオプションは、**tempdb** など、前回の **dump database** 以降にログがトランケートされたデータベースでも使用できません。



- `until_time` オプションが有益な理由は、次のとおりです。
  - 特定の時間までデータベースの整合性を保つことができる。たとえば、意思決定支援システム (DSS) データベースとオンライン・トランザクション処理 (OLTP) データベースが構築されている環境では、システム管理者は DSS データベースを指定時間までさかのぼって現在のバージョンと古いバージョンを比較できる。
  - ユーザが間違っって重要なテーブルを削除したためにデータが破壊された場合などは、`until_time` オプションを使用して、データベースが破壊された直前の時点でデータベースをロールフォワードすることによって、その誤ったコマンドを取り消すことができる。
- データが破壊された後に `until_time` オプションを効果的に使用するためには、エラーが発生した正確な時刻を特定する必要があります。エラー発生直後に `select getdate ()` コマンドを実行すると、その時刻を調べることができます。ミリ秒単位のさらに正確な時刻が必要なときは、次のように `convert` 関数を使います。

```
select convert (char (26), getdate (), 109)
-----
Feb 26 1997 12:45:59:650PM
```

- `until_time` を使用してトランザクション・ログをロードすると、その時点からデータベースのログ・シーケンスが再び始まります。したがって、`until_time` を使用して `load transaction` を実行した後は、データベースを再度ダンプするまでは後続のトランザクション・ログをロードすることはできません。データベースをダンプしてから別のトランザクション・ログをダンプしてください。
- データベースには、指定時間の前にコミットされたトランザクションだけが保存されます。ただし、`until_time` 指定直後にコミットされたトランザクションがデータベース・データに適用される場合もあります。これは複数のトランザクションが同時にコミットされた場合に発生します。トランザクションの順序は、時間順にトランザクション・ログに書き込まれないことがあります。その場合、時間シーケンス外のトランザクションは、リカバリされたデータに反映されます。この時間は 1 秒未満です。
- 特定の時刻までデータベースをリカバリする方法については、『システム管理ガイド』を参照してください。

### ロード時のユーザのロックアウト

- ロード中のデータベースは使用できません。データベースをロードしているときは使用不可です。load transaction は load database とは異なり、データベースのオフラインとオンラインのステータスを変更しません。load transaction の実行時にデータベースのステータスは検出時と同じに保持されます。一方 load database ではデータベースのステータスが "offline" に設定されます。"offline" のデータベースは誰も使用できません。ロード・シーケンスの実行中は、"offline" ステータスにすることにより、ユーザはデータベースにアクセスしたり、変更することができなくなります。
- online database が発行されるまで、load database によってロードされたデータベースにはアクセスできません。

**データベース・ダンプとトランザクション・ログ・ダンプのアップグレード**  
バージョン 11.9 以降のサーバから現在のバージョンの Adaptive Server にユーザ・データベース・ダンプをリストアおよびアップグレードするには、次の手順に従います。

- 1 最新のデータベース・ダンプをロードします。
- 2 前回のデータベース・ダンプ後に生成されたすべてのトランザクション・ログを順序どおりにロードします。
- 3 online database を使用してアップグレードを実行します。
- 4 アップグレード直後に、新しくアップグレードされたデータベースをダンプして、現在のバージョンの Adaptive Server と一貫性のあるダンプを作成します。

### ダンプ・デバイスの指定

- ダンプ・デバイスは、リテラル、ローカル変数、またはストアド・プロシージャへのパラメータとして指定できます。
- ローカル・デバイスからロードするときは、次のように指定できます。
  - 絶対パス名
  - 相対パス名
  - sysdevices システム・テーブルの論理デバイス名

Backup Server は、Adaptive Server の現在の作業ディレクトリを使用して相対パス名を解決します。

- ネットワーク間でロードを行う場合は、ダンプ・デバイスの絶対パス名を指定します(相対パス名または `sysdevices` システム・テーブルからの論理デバイス名は使用できません)。パス名は、**Backup Server** が起動しているマシンで有効なものでなければなりません。名前に文字、数字、またはアンダースコア ( `_` ) 以外の文字が含まれる場合は、引用符で囲んでください。
- ダンプ・デバイスでの所有権とパーミッションの問題によって、`load` コマンドが使用できないことがあります。`sp_addumpdevice` によってデバイスをシステム・テーブルに追加できますが、そのデバイスからロードできるか、またはダンプ・デバイスとしてファイルを作成できるかどうかは保証されていません。
- それぞれのロードが別の物理デバイスを使用していれば、複数のロード(またはダンプ)を同時に実行できます。

#### Backup Server

- **Backup Server** は、**Adaptive Server** が稼働しているマシン上で実行する必要があります。また、**Backup Server** は `master.syssservers` テーブルにリストされている必要があります。このエントリはインストールまたはアップグレード中に作成されます。削除しないでください。
- バックアップ・デバイスが別のマシン上にあり、ネットワーク間でロードする場合、**Backup Server** をリモート・マシン上にもインストールしておかなければなりません。

#### ボリューム名

- ダンプ・ボリュームは、ANSI テープ・ラベル標準に準拠してラベル付けされます。ラベルには、論理ボリューム番号とストライプ・セット内のデバイスの位置が含まれています。
- **Backup Server** はロード中にテープ・ラベルを使用して、ボリュームが正しい順番でマウントされているかどうかを確認します。これにより、ダンプ時に使用したデバイス数より少ないデバイスからロードできます。

---

**注意** ネットワーク間でダンプおよびロードを実行する場合は、各オペレーションに同じ数のストライプ・デバイスを指定しなければなりません。

---

### ダンプ・ボリュームの変更

Backup Server は、現在マウントされているボリュームで問題を検出すると、クライアントまたはクライアントのオペレータ・コンソールにメッセージを送信して、ボリュームの変更を要求します。別のボリュームをマウントした後、オペレータは Backup Server に通知するために、Backup Server と通信できる Adaptive Server 上で、`sp_volchanged` システム・プロシージャを実行します。

### システム・データベースのリストア

システム・データベースをダンプからリストアする手順については、『システム管理ガイド』を参照してください。

### ディスク・ミラーリング

- ロードの初期の段階で、Adaptive Server は、各論理データベースとログ・デバイスのプライマリ・デバイス名を Backup Server に渡します。プライマリ・デバイスのミラーリングが解除されている場合、Adaptive Server は代わりにセカンダリ・デバイス名を渡します。Backup Server によるデータ転送が完了する前に、指名されたデバイスが失敗すると、Adaptive Server はダンプをアボートします。
- `load transaction` の進行中に、指定されたデバイスのいずれかのミラーリングを解除しようとする、メッセージが表示されます。`disk unmirror` を実行しているユーザは、ロードが完了するまで、ロードをアボートするか、または `disk unmirror` を遅延させることができます。
- Backup Server がプライマリ・デバイスにデータをロードすると、`load transaction` がそのデータをセカンダリ・デバイスにコピーします。データベース・デバイスがミラーリングされている場合は、`load transaction` コマンドの完了に時間がかかります。

### アーカイブ・データベースへのトランザクション・ログのロード

アーカイブ・データベースにトランザクション・ログをロードするときに、`load tran` はリカバリ再実行パスを実行します。変更された、または新規のデータベース・ページが永続の変更セグメントに書き込まれます。これらの変更に対応するために、変更済みページ・セクションに十分な領域が必要です。必要に応じて、`alter database` を使用して、変更済みページ・セクションの領域を増やし、アーカイブ・データベースに割り付けられた通常のデータベース領域を拡大します。

従来のデータベースとは異なり、アーカイブ・データベースは、ロード・シーケンスをブレイクしないでロード・シーケンス中にオンラインにできます。従来のデータベースでは、ロードしてから `for standby_access` 句を使用せずにオンラインにすると、ロード・シーケンスで次のトランザクション・ログをロードできなくなりますが、アーカイブ・データベースでは、`for standby_access` 句以降を使用せずにオンラインにし、ロード・シーケンスで次のトランザクション・ログを使用してロードできます。これによって、ロード・シーケンス中にいつでも一貫性チェックの実行といった読み込み専用のオペレーションを実行できます。これが可能なのは、トランザクション・ログをアーカイブ・データベースにロードする際に、Adaptive Server が処分可能な変更セグメントを変更済みページ・セクションから自動的に削除するためです。こうすることで、効率的にアーカイブ・データベースを前回のロードが実行された後の状態に戻し、シーケンスの次のトランザクション・ログをロードできるようにします。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

パーミッション

`load transaction` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、データベース所有者であるか、データベースに対する <code>load database</code> 権限または <code>own database</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、次の役割が付与されたユーザであることが必要。 <ul style="list-style-type: none"> <li>• <code>sa_role</code>、または</li> <li>• <code>replication_role</code>、または</li> <li>• <code>oper_role</code></li> </ul>

監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
44	load	load transaction	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照

マニュアル 『システム管理ガイド』の「ユーザ・データベースのバックアップとリストア」

コマンド [disk unmirror](#), [dump database](#), [dump transaction](#), [load database](#), [online database](#)

システム・プロシージャ [sp\\_dboption](#), [sp\\_helpdb](#), [sp\\_helpdevice](#), [sp\\_hidetext](#), [sp\\_volchanged](#)

## lock table

説明	トランザクション内でテーブルを明示的にロックします。
構文	lock table <i>table_name</i> in [partition <i>data_partition_name</i> ] in {share   exclusive} mode [wait [ <i>numsecs</i> ]   nowait]
パラメータ	<p><i>table_name</i>          ロックされるテーブルの名前です。</p> <p>partition <i>data_partition</i>          データ・パーティションをロックしていることを示します。</p> <p>share   exclusive          テーブルまたはパーティションに適用されるロックのタイプ (共有または排他) を指定します。</p> <p>wait <i>numsecs</i>          ロックがすぐに取得されない場合に、待機する秒数を指定します。  <i>numsecs</i> が省略されている場合は、ロックが付与されるまで lock table コマンドが待機するように指定します。</p> <p>nowait          ロックがすぐに取得されない場合は、コマンドが失敗します。</p>
例	<p><b>例 1 titles</b> テーブル上の共有テーブル・ロックの取得を試みます。セッション・レベルの待機時間を set lock wait で設定している場合、lock table コマンドはその指定時間に従って待機します。セッション・レベルの待機時間が設定されていない場合は、サーバ・レベルの待機時間が使用されます。</p> <pre>begin transaction lock table titles in share mode</pre> <p><b>例 2 authors</b> テーブル上の排他テーブル・ロックの取得を試みます。ロックが 5 秒以内に取得されない場合は、コマンドが情報メッセージを返します。トランザクション内の後続のコマンドは、lock table が指定されていない場合と同様に引き続き実行されます。</p> <pre>begin transaction lock table authors in exclusive mode wait 5</pre> <p><b>例 3</b> テーブル・ロックが 5 秒以内に取得されない場合は、プロシージャがユーザの役割を確認します。ユーザが sa_role を使用してプロシージャを実行すると、プロシージャは警告メッセージを出力し、テーブル・ロックなしで実行を続けます。ユーザが sa_role を持っていない場合、トランザクションはロールバックされます。</p>

```
create procedure bigbatch
as
begin transaction
lock table titles in share mode wait 5
if @@error = 12207
begin
  /*
  ** Allow SA to run without the table lock
  ** Other users get an error message
  */
  if (proc_role ("sa_role") = 0)
  begin
    print "You cannot run this procedure at
          this time, please try again later"
    rollback transaction
    return 100
  end
else
  begin
    print "Couldn't obtain table lock,
          proceeding with default locking."
  end
end
/* more SQL here */
commit transaction
```

#### 使用法

- アーカイブ・データベースで、**lock table** を使用できます。
- **set chained on** コマンド後の最初の文として **lock table** を使用すると、新しいトランザクションが作成されます。
- **lock table** はトランザクション内でのみ使用できます。テーブル・ロックは、トランザクションの間はずっと保持されます。
- **lock table** の動作は、コマンドで指定されている待機時間のオプションによって異なります。または、そのオプションがセッション・レベルまたはサーバ・レベルのどちらでアクティブになっているかによっても異なります。
- **wait** オプションと **nowait** オプションが指定されていない場合、**lock table** は、セッション・レベルの待機時間またはサーバ・レベルの待機時間のどちらかが使用されます。セッション・レベルの待機時間が **set lock wait** によって設定されている場合は、セッション・レベルの待機時間が使用されます。設定されていない場合は、サーバ・レベルの待機時間が使用されます。



- 制限時間を指定している場合に、テーブル・ロックを制限時間内に取得できない場合は、`lock table` コマンドが 12207 のメッセージを返します。トランザクションはロールバックされません。トランザクション内の後続のコマンドは、`lock table` コマンドが指定されていない場合と同様に引き続き実行されます。
- システム・テーブルまたはテンポラリ・テーブルでは、`lock table` コマンドを使用できません。
- 同じトランザクション内で、複数の `lock table` コマンドを発行できます。

**標準規格**

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

**パーミッション**

テーブルの所有者であることが必要です。`lock table in share mode` を使用するには、テーブルに対する `select` アクセス・パーミッションが必要です。`lock table in exclusive mode` を使用するには、テーブルに対する `delete`、`insert`、または `update` のいずれかのアクセス・パーミッションが必要です。

**参照**コマンド `set`

## merge

### 説明

送信元テーブルからターゲット・テーブルにローを転送します。

- 送信元に存在し、ターゲットに一致するキー・カラムのないローを挿入します。
- ターゲット内に既に存在するキー・カラムのあるローを、送信元のローの値で更新します。

### 構文

merge

```
into [[database.]owner.]identifier [as table_alias]
using [[database.]owner.]identifier [as table_alias]
    | (select_query) as alias_name [column_list]
on merge_search_condition
[ when matched [and search_conditions ]
  then {update set {col_name = expression} | delete} ]
[ when not matched [and search_conditions ]
  then insert [(column_list) values (value_list)
```

### パラメータ

into [[database.]owner.]identifier [as table\_alias]

ターゲット・オブジェクトをテーブルとして指定するか、更新可能なビューをテーブルのエイリアスとして指定します。ターゲットは、データベース名および所有者名を含まない完全に修飾された名前識別子または短い名前識別子とすることができます。この場合には、Adaptive Server により現在のデータベースとユーザまたはデータベース所有者が使用されます。

ターゲット・テーブルを参照する代わりに、テーブルのエイリアスを指定することもできます。

using [[database.]owner.]identifier [as table\_alias] | (select\_query) as alias\_name [column\_list]

ソース・オブジェクトをテーブル、ビュー、または抽出テーブルとして指定します。ソース・オブジェクトに応じて以下のようにします。

- テーブルまたはビュー — [[database.]owner.]identifier を使用します。
- 抽出テーブル — エイリアス名および抽出テーブルを定義するオプションのカラム・リストの形式で、select クエリにより参照します。

merge\_search\_conditions

送信元テーブルのローがターゲット・テーブルのローに一致し、“col\_name = col\_name”などの述部リストから構成されるかどうかを確認します。

*search\_conditions*

matched/not matched 句で使用される正しい形式のブール式です。

*update set {col\_name = expression} | delete*

これらのオプションは常に matched 句に使用されます。update は新しい値を一致するローに割り当て、delete は現在の一致するローを削除します。

*insert [(column\_list)] values (value\_list)*

not matched 句に常に使用され、ターゲット・テーブルに一致しないローを挿入します。

例

**例 1** DailySales テーブルを GlobalSales にマージします。

```
merge into GlobalSales
  (Item_number, Description, Quantity)as G
using DailySales as D
ON D.Item_number = G.Item_number
when not matched
  then
    insert (Item_number, Description, Quantity )
  values (D.Item_number, D.Description, D.Quantity)
when matched
  then update set
    G.Quantity = G.Quantity + D.Quantity
```

**例 2** 動的パラメータ・マーカを持つ送信元テーブルとして抽出テーブルを使用します。

```
merge into GlobalSales
  (Item_number, Description, Quantity)as G
using select (?, ?, ?) as
D (Item_number, Description, Quantity)
ON D.Item_number = G.Item_number
when not matched
  then
    insert (Item_number, Description, Quantity )
  values (D.Item_number, D.Description, D.Quantity)
when matched
  then update set
    G.Quantity = G.Quantity + D.Quantity
```

使用法

- ターゲット・テーブルを参照整合性制約に含めることはできません。
- (1=0) や (1=1) などの定数ブール式を参照する on 句の merge クエリには特定の最適化がありません。

- on 句で参照されるターゲット・カラムは、update アクションの set 句に含めることはできません。
- ストアド・プロシージャ内で merge 文を呼び出すことができますが、update 文および insert 文はスカラ SQL 関数内で使用できません。
- ターゲット・テーブルを instead of トリガで更新可能なビューとすることはできません。
- merge 文はキャッシュすることができ、update アクションの set 句、および insert アクションの insert 値リストにあるリテラルがリテラルのパラメータ化プロセスのターゲットになります。
- プロキシ・テーブルをターゲット・テーブルにすることはできません。
- 送信元テーブルの各ローは、以下のようになります。
  - ターゲット・テーブルに一致するローがあり、検索条件が true と評価されるロー — ターゲット・テーブル、またはターゲット・テーブル内の対応するローに update を実行します。
  - ターゲット・テーブルに一致するローがなく、検索条件が true と評価されるロー — ターゲット・テーブルにローを挿入します。
  - ターゲット・テーブルに一致するローが複数存在する場合 — エラーが発生します。これは SQL-2003 標準の動作です。

merge 文には、検索条件の異なる when matched 句と when not matched 句を複数組み込むことができます。検索条件が一致した when 句の最初の when が対応するアクションを実行し、残りは無視されます。

## パーミッション

ソース・オブジェクトに対する select パーミッションと、ターゲット・オブジェクトに対する insert、update、または delete パーミッションを持つすべてのユーザが merge を使用できます。

## mount

### 説明

データベースを移送先またはセカンダリの Adaptive Server に接続します。

`mount` は、**manifest file** 内の情報を復号化して一連のデータベースを使用可能にします。`mount` は、データベース・デバイスの追加、必要に応じたアクティブ化、新規データベースのカatalog・エントリの作成とリカバリなど、必要なすべてのサポート動作が実行される点が、`bcp` バルク・コピー・ユーティリティなどの他のコピー・プロシージャとは異なります。

データベースのマウント時に移送先 Adaptive Server で異なるデバイス名を使用している場合は、`mount with listonly` を使用し、移送先サーバのデバイス・パス名を変更してから、`mount` を使用してデータベースを実際にマウントします。

---

**注意** 元の Adaptive Server 上のデータベースへのアクセスが許可されたログインはすべて、移送先 Adaptive Server で同一の `suid` に対応するログインを設定しておくと便利です。これによって、ユーザ ID 調整問題を回避できます。

パーミッションを変えないようにするには、送信先 Adaptive Server 上のログイン・マップを送信元 Adaptive Server 上のログイン・マップと同一にします。ログイン・マップの詳細については、『システム管理ガイド 第 1 巻』の「リモートサーバの管理」を参照してください。

---

### 構文

```
mount database all | database_mapping [, database_mapping, ...]
    from "manifest_file"
    [using device_mapping [, device_mapping...]
      [with listonly]
    database_mapping:
        origdbname as newdbname
        | newdbname = origdbname
        | origdbname
        | newdbname
    device_mapping
        logical_device_name as new_physical_name
        | new_physical_name = logical_device_name
        | original_physical_name
        | new_physical_name
```

## パラメータ

*manifest\_file*

マニフェスト・ファイルは、一連のデータベース・デバイスに存在するデータベースを記述するバイナリ・ファイルです。

ファイルの内容の文字変換を伴うオペレーション (**ftp** など) をバイナリ・モード以外で実行すると、マニフェスト・ファイルは破損します。

**例 1** 送信元の Adaptive Server のマニフェスト・ファイルにリストされているパス名を検索します。

```
mount database all from "/data/sybase2/mfile1" with listonly
go

[database]
  mydb
[device]
  "/data/sybase1/d0.dbs" = "1dev1"
  "/data/sybase2/d14.dbs" = "1dev13"
```

送信元とは異なるパス名を使用する場合は、それらを検証または変更して、送信先 Adaptive Server での基準に合わせるすることができます。

**例 2** データベース・デバイスをセカンダリ Adaptive Server にロードしたら、データベースをマウントします。

```
mount database all from "/data/sybase2/mfile1" using
  "/data/sybase2/d0.dbs" = "1dev1",
  "/data/sybase2/d14.dbs" = "1dev13"
```

**mount** プロセスが完了しても、データベースはオフラインのままです。**online database** コマンドを使用してデータベースをオンラインにします。サーバを再起動する必要はありません。

**例 3** 移送先サーバは送信元サーバと同一でも構いません。この場合は、データベース名を異なる名前でもマップする必要があります。論理デバイス名は、内部で名前が変更されます。

1 同一サーバでデータベース **mydb** の完全なコピーを作成します。

```
1> quiesce database mydb_tag hold mydb for external dump to
  "/data/mydb.manifest"
2> go
```

2 OS ファイルをコピーします。

```
$ cp /data/sybase2/mydb.dbs /data/sybase2/mydb_copy.dbs
```

3 これで、コピーとしてデータベースをマウントできます。

```
1> quiesce database mydb_tag release
```

```

2> go
1> mount database mydb as mydb_copy
2> from "/data/mydb.manifest"
3> using mydb_dev as "/data/sybase2/mydb_copy.dbs"
3> go

```

物理デバイス `//data/sybase2/mydb_copy.dbs/` には、`Cccc$<mydb_dev>` の形式でマシンが生成する論理名が割り当てられます。ここでは、

- `C` – [A-Z] です。
- `c` – [A-Z, 0-9] で、コード化された論理デバイス番号を参照します。
- `mydb_dev` – 古い論理デバイス名から最大 26 文字が含まれます。

移送するデータベースのデータベース ID が、送信先の Adaptive Server に存在している必要はありません。データベースは同じサーバ上にマウントされているため、データベース ID は変更しなければなりません。マウントされるデバイスのアロケーション・ページには元のデータベース ID が保持され、この情報は `disk refit` コマンドによって使用されます。`mount database` の実行後に `dbid` を調整してマウントされたデバイスで `disk refit` が機能するようにするには、`dbcc checkalloc` コマンドを使用します。そのデータベースが一時使用のためにマウントされているのではない場合は、`checkalloc` を実行します。

#### 使用法

- `using` 句を使用すると、`=` 記号または `as` 句によってマッピングを定義できます。
- 複数のデバイスが存在する場合のマッピングは、一方で `=` を使用し、もう一方で `as` を使用できます。
- デバイスは、データベースとデバイスの両方で論理と物理の両方を指定して、名前によって、また、順序によってマッピングすることができます。データベースが名前でマッピングされている場合は、すべてのデータベースが名前でマッピングされなければなりません。順序でのマッピングも同様です。デバイスについても同じです。
- 移送先 Adaptive Server で、移送されたデータベースのセットの一部に対して `mount` を実行することはできません。マニフェスト・ファイル内のすべてのデータベースとそのデバイスを同時にマウントする必要があります。
- 移送先 Adaptive Server で `mount` コマンドを実行するときは、以下の規則に従ってください。
  - 移送先 Adaptive Server のページ・サイズは、移送元 Adaptive Server のページ・サイズと同一でなければならない。

- マウントするデータベースに属するすべてのデバイスを正常に追加するのに十分なデバイスが、移送先 Adaptive Server において設定されている必要がある。
- 送信元 Adaptive Server からマウントする論理デバイスの名前が、移送先 Adaptive Server 上の論理デバイスの名前と同じ場合、mount コマンドでエイリアスを含めていなければこれらのデバイス名は自動的に変更される。

物理デバイスの名前がすでに移送先 Adaptive Server に存在する場合は、オペレーティング・システム・レベルで送信元 Adaptive Server 上の物理デバイス名を変更し、mount コマンドで新しい物理デバイス名を変更する必要がある。

- 移送先と移送元の Adaptive Server のログ・バージョンは同一でなければならない。
- 移送先のメジャー・バージョン番号よりも大きいメジャー・バージョン番号の Adaptive Server からデータベースをマウントすることはできない。たとえば、12.5.x バージョンの Adaptive Server に 15.0 バージョンのデータベースをマウントすることはできない。
- 移送先と移送元の Adaptive Server のプラットフォームは同一でなければならない。
- ソート順や文字セットの相違は、load database のときと同じルールによって解決される。異なる文字セットを使用するデータベースをマウントできるのは、ソート順がバイナリの場合のみ。

#### Cluster Edition

mount database と unmount database は Cluster Edition でサポートされません。このどちらかのコマンドが進行中にインスタンスでエラーが発生した場合は、コマンドがアボートされる可能性があります。この場合、インスタンスのフェールオーバー・リカバリが完了したら、mount database または unmount database を再発行する必要があります。

#### 移送先での変更

データベースを送信先 Adaptive Server にマウントすると、次に示すように、マウントされたデータベースの特定の設定がクリアされます。

- 複写がオフになる。
- 監査設定がクリアされてオフになる。



- コンポーネント統合サービスオプション、デフォルト・リモート・ロケーション、タイプがクリアされる。
- マウントされたデータベースとそのオブジェクトの両方でキャッシュ・バインドが削除される。
- マウントされたデータベースのリカバリ順が削除され、デフォルトの dbid 順になる。

#### システムの考慮事項

- トランザクション内では `mount` コマンドを使用できません。
- 高可用性が設定されたサーバ上のデータベースに対して `mount` を実行することはできません。

#### パフォーマンスの考慮事項

Adaptive Server のデータベースに `mount` を実行する場合、マウントするデータベースの `dbid` を変更すると、すべてのプロシージャがデータベースで再コンパイルされることを示すマークが付けられます。これにより、送信先でデータベースのリカバリに要する時間が長くなり、プロシージャの最初の実行が遅れます。

#### デバイス名の変更

マニフェスト・ファイルには、マニフェスト・ファイルを作成した送信元 Adaptive Server が認識しているデバイス・パスが記述されています。移送先 Adaptive Server がアクセスするデバイスが異なるパスにある場合は、`mount` コマンドで新しいパスを指定します。

- 1 以下のように、`listonly` オプションを指定して `mount` コマンドを使用し、古いパスを表示します。

```
mount database all from "/work2/Mpubs_file" with listonly
go
```

```
[database]
```

```
mydb
```

```
[device]
```

```
"/work2/Devices/pubsdats.dat" = "pubs2dat"
```

- 2 デバイス `pubs2dat` の新しいパスが `/work2/Devices/pubdevice.dat` (Windows のデバイス・パス) である場合、`mount` コマンドで新しいデバイスを次のように指定します。

```
mount database all from "/work2/Mpubs_file" using
"/work2/datadevices/pubdevice.dat" = "pubs2dat"
```

論理デバイスの名前が移送先サーバに存在すると、自動的に生成される一意の名前によって名前が変更されます。

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

**パーミッション** mount のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>mount any database</code> 権限を持つユーザーであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> または <code>oper_role</code> を持つユーザーであることが必要。

**監査** sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
101	mount	mount database	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

**参照** コマンド [unmount](#), [quiesce database](#)

マニュアル 『システム管理ガイド 第2巻』の「データベースのマウントとマウント解除」

## online database

説明	<p>通常のロード・シーケンスの後でデータベースを誰でも使用できるように指定し、必要に応じて、ロードされたデータベースを現在のバージョンの Adaptive Server にアップグレードします。また、<code>for standby_access</code> オプションによってダンプされたトランザクション・ログのロード後に、データベースをオンラインにします。また、<code>online database</code> を使用すると、アーカイブ・データベースをオンラインにすることもできます。</p>
構文	<pre>online database <i>database_name</i> [for standby_access]</pre>
パラメータ	<p><i>database_name</i></p> <p>オンラインにするデータベースの名前を指定します。</p> <p>for standby_access</p> <p>オープン状態にあるトランザクションがデータベース内にはないという想定のもとに、データベースをオンラインにします。</p>
例	<p><b>例 1</b> ロード・シーケンスの完了後に、<code>pubs2</code> データベースを誰でも使用できるようにします。</p> <pre>online database pubs2</pre> <p><b>例 2</b> <code>inventory_db</code> データベースをオンラインにします。<code>dump tran...with standby_access</code> を通じて取得されたトランザクション・ログのダンプを使って <code>inventory_db</code> をロードしてから、このデータベースを使用します。</p> <pre>online database inventory_db for standby_access</pre>
使用法	<ul style="list-style-type: none"> <li>• <code>online database</code> コマンドは、データベースやトランザクション・ログの通常のロード・シーケンス後に誰でも使用できるように、データベースをオンラインにします。</li> <li>• <code>load database</code> を発行すると、データベースのステータスは "offline" に設定されます。オフライン・ステータスは、<code>sysdatabases</code> システム・テーブルで設定され、<code>online database</code> が完了するまで、設定された状態が保持されます。</li> <li>• すべてのトランザクション・ログがロードされるまでは、<code>online database</code> コマンドを発行しないでください。コマンドの順序は次のとおりです。 <ul style="list-style-type: none"> <li>• <code>load database</code></li> <li>• <code>load transaction</code> (<code>load transaction</code> を複数回実行することも可能)</li> <li>• <code>online database</code></li> </ul> </li> </ul>

- 現在オンラインであるデータベースに対して `online database` を実行すると、処理は行われずエラー・メッセージも表示されません。
- `online database...for standby_access` は、`dump transaction...with standby_access` を使用してダンプしたトランザクション・ログのみ使用できます。`dump transaction...with standby access` を使わずにダンプしたトランザクション・ログをロードした後に `online database...for standby_access` を使用すると、`online database` コマンドはエラー・メッセージを生成して失敗します。
- `sp_helpdb` を使用すると、データベースが現在オンライン状態なのか、スタンバイ・アクセス用のオンライン状態なのか、またはオフラインなのかを調べることができます。

#### データベースのアップグレード

- `online database` は、ロードされたデータベースとトランザクション・ログ・ダンプのアップグレードを必要に応じて開始し、データベースと現在のバージョンの `Adaptive Server` との互換性を確保します。アップグレードが完了すると、データベースは誰でも使用できるようになります。処理中にエラーが発生した場合、データベースのステータスはオフラインのままになります。
- `online database` が必要なのは、データベースまたはトランザクション・ログのロード・シーケンスの後だけです。新しくインストールするときやアップグレードするときには必要ありません。`Adaptive Server` を新しいバージョンにアップグレードすると、そのサーバに関連するすべてのデータベースが自動的にアップグレードされます。
- `online database` は、バージョン 12.5.4 以降のユーザ・データベースのみアップグレードします。
- `online database` を使用してデータベースをアップグレードした後は、新しくアップグレードされたデータベースをダンプして、現在のバージョンの `Adaptive Server` と一貫性のあるダンプを作成します。アップグレードしたデータベースをダンプしてから `dump transaction` コマンドを発行してください。

#### アーカイブ・データベース

`online database database_name` コマンドは、変更済みページおよび割り付けられたページを変更済みページ・セクションに再マップ可能なときに、リカバリの取り消しを実行します。

データベースをロードすると、リカバリの取り消しパスを実行しなくても自動的にオンラインになるため、**with norecovery** を使用してデータベースをロードした場合はデータベースをオンラインにする必要はありません。

標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

パーミッション

**create database** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、データベース所有者であるか、データベースに対する <b>online database</b> 権限または <b>own database</b> 権限を持つユーザーであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、次の役割が付与されたユーザであることが必要。 <ul style="list-style-type: none"> <li>• <b>sa_role</b>、または</li> <li>• <b>replication_role</b>、または</li> <li>• <b>oper_role</b></li> </ul>

監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
83	security	online database	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <b>set proxy</b> が有効な場合は元のログイン名</li> </ul>

参照

コマンド [dump database](#), [dump transaction](#), [load database](#), [load transaction](#)  
 システム・プロシージャ [sp\\_helpdb](#)

## open

説明	処理に使用するカーソルをオープンします。
構文	<code>open cursor_name</code>
パラメータ	<code>cursor_name</code> オープンするカーソルの名前です。
例	<code>authors_crsr</code> という名前のカーソルをオープンします。  <code>open authors_crsr</code>
使用法	<ul style="list-style-type: none"><li>• <code>open</code> によって、カーソルを開きます。カーソルを使用すると、ローを個別に修正したり削除したりできます。<code>fetch</code> 文、<code>update</code> 文、<code>delete</code> 文を使用するには、まずカーソルを開く必要があります。カーソルの詳細については、『<i>Transact-SQL ユーザーズ・ガイド</i>』を参照してください。</li><li>• カーソルがすでにオープンしている場合や、カーソルが <code>declare cursor</code> 文によって作成されていない場合は、Adaptive Server がエラー・メッセージを返します。</li><li>• カーソルをオープンすると、Adaptive Server は (<code>declare cursor</code> 文で指定されている) カーソルを定義した <code>select</code> 文を評価します。次に、カーソル結果セットを処理できるようにします。</li><li>• カーソルは最初に開かれると、カーソル結果セットの最初のローの前に配置されます。</li><li>• アーカイブ・データベースで、<code>open</code> を使用できます。</li><li>• 連鎖トランザクション・モードを設定すると、現在アクティブであるトランザクションがない場合、Adaptive Server は <code>open</code> 文で暗黙的にトランザクションを開始します。</li></ul>
標準規格	ANSI SQL – 準拠レベル : Transact-SQL 拡張機能
パーミッション	<code>open</code> のパーミッションは、すべてのユーザにデフォルトで設定されています。
参照	コマンド <a href="#">close</a> , <a href="#">declare cursor</a> , <a href="#">fetch</a>

## order by 句

**説明** クエリ結果を、指定されたカラムのソート順で返します。

**構文**

```
[Start of select statement]
[order by
    {[table_name.| view_name.]
     column_name | select_list_number | expression}
    [asc | desc]
    [, {[table_name.| view_name.]
     column_name | select_list_number | expression}
    [asc | desc]}...]
[End of select statement]
```

**パラメータ**

**order by**  
カラムに基づいて結果をソートします。

**asc**  
実行結果を昇順でソートします。**asc** または **desc** を指定しない場合は、**asc** が採用されます。

**desc**  
実行結果を降順でソートします。

**例** **例 1** 価格 (**price**) が 19.99 ドルを超える書籍のタイトルを選び、タイトルのアルファベット順にリストします。

```
select title, type, price
from titles
where price > $19.99
order by title
```

```
title
type          price
-----
But Is It User Friendly?
    popular_comp          22.95
Computer Phobic and Non-Phobic Individuals:Behavior Variations
    psychology            21.59
Onions, Leeks, and Garlic:Cooking Secrets of the Mediterranean
    trad_cook             20.95
Secrets of Silicon Valley
    popular_comp          20.00
```

**例 2** **titles** テーブルから書籍を選び出して、種類をアルファベットの降順でリストし、種類ごとに平均価格と印税の前払いを計算します。

```
select type, price, advance
```

```

from titles
order by type desc
compute avg (price), avg (advance) by type

```

**例 3** 印税 (advance) の前払いを総売り上げ (total\_sales) で割った値とともに、合計計算額の最低から最高までの順序でタイトル ID を titles テーブルからリストします。

```

select title_id, advance/total_sales
from titles
order by advance/total_sales

```

```

title_id
-----
MC3026          NULL
PC9999          NULL
MC2222          0.00
TC4203          0.26
PS3333          0.49
BU2075          0.54
MC3021          0.67
PC1035          0.80
PS2091          1.11
PS7777          1.20
BU1032          1.22
BU7832          1.22
BU1111          1.29
PC8888          1.95
TC7777          1.95
PS1372          18.67
TC3218          18.67
PS2106          54.05

```

**例 4** 本のタイトル (title) と種類 (type) を種類別にリストし、出力内のカラムの名前を変更します。

```

select title as BookName, type as Type
from titles
order by Type

```

#### 使用法

- order by は、クエリ結果をソート順に従って指定カラムに戻します。order by は select コマンドの一部です。
- Transact-SQL では、order by を使用して、select リストに表示されない項目をソートできます。ソートは、select リスト内で指定されている場合は、カラム見出し、カラム名、式、エイリアス名を基準にして実行できます。また、select リスト内の項目の位置を表す番号 (select\_list\_number) を基準にしてソートすることもできます。



- `select_list_number` でソートする場合は、`order by` 句が参照するカラムを `select` リストに指定してください。 `select` リストは \* (アスタリスク) では指定できません。
- `order by` を使用して、有意義な順序でクエリ結果を表示します。 `order by` 句を使用しない場合は、Adaptive Server が結果を返す順序を制御できません。

#### 同名のテーブル・カラムおよびカラム・エイリアス名が存在する場合の `order by` の動作

次の 3 つの条件が存在する場合、Adaptive Server は `order by` 句でカラム名をエイリアス名として解釈します。

- `order by` 句に修飾されたカラム名 (つまり、`order by table.column`) に対する参照が含まれている。
- テーブル・カラム名とエイリアス名の両方が存在している。
- テーブル・カラムとエイリアスの両方が `order by` 句のカラム名と一致する。

次の例では、`order by` 句が同じであっても、`order by` 句が参照するカラムが異なり、2 つのクエリで結果が異なります。

```
create table t (A int, B char(3))
insert into t (A, B) values(1, 'az')
insert into t (A, B) values(2, 'bb')
go
/* t.B refers to the table column B */
select A, reverse(B) as C from t order by t.B
go
/* t.B refers to the alias column B */
select A, reverse(B) as B from t order by t.B
go

A C
-----
1 za
2 bb

(2 rows affected)

A B
-----
2 bb
1 za

(2 rows affected)
```

Adaptive Server では、order by 句によるテーブル名で修飾されたエイリアス・カラム名の参照が可能であるために、このような動作が発生します。基本テーブル・カラムもエイリアス・カラムと同じカラム名で存在する場合、Adaptive Server ではエイリアス・カラムが優先されます。

#### 制限事項

- order by 句で指定できるカラム数は最大 31 です。
- order by を、text、unitext、image データ型のカラムで使用することはできません。
- サブクエリおよびビュー定義には、order by 句 (または compute 句や into キーワード) を含めることができません。言い換えれば、order by リスト内ではサブクエリを使用できません。
- サーバまたは言語タイプのカーソルの結果セットは、カーソルの select 文に order by 句が含まれている場合には更新できません。更新可能なカーソルに適用される制限の詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。
- compute by 句を使用する場合は、order by 句も使用する必要があります。compute by 句の後にリストされている式は、order by 句の後にリストされている式と等しいか、またはそのサブセットでなければなりません。また、左から右に同じ順序で、同じ式で開始される必要があります。どの式も省略しないでください。たとえば、次のような order by 句があるとします。

```
order by a, b, c
```

compute by 句には、次のいずれか (またはすべて) を指定できます。

```
compute by a, b, c
compute by a, b
compute by a
```

compute キーワードは、by を付けずに使用して、総計や総数などを生成することもできます。この場合、order by 句は省略可能です。

#### 照合順

- order by 句を使用すると、他のすべての値よりも前に null 値が置かれます。
- Adaptive Server のソート順 (照合順) によって、データをソートする方法が決定されます。ソート順を決定する選択肢には、バイナリ順、辞書順、大文字／小文字の区別なし、大文字／小文字の区別なしで大文字優先、大文字／小文字とアクセントを区別しない、があります。また各国言語に固有のソート順も用意されていることもあります。

表 1-27: ソート順選択による影響

Adaptive Server のソート順	order by 句の結果による影響
バイナリ順	文字セットの各文字のバイト-数値に従って、すべてのデータをソートする。バイナリ順では、すべての大文字をソートしてから小文字をソートする。バイナリ・ソート順は、マルチバイト文字セットの唯一のオプション。
辞書順	先に大文字、次にその小文字の順にソートする (大文字/小文字を区別する)。辞書順は、各種のアクセント記号付きの形式の文字を認識し、アクセント記号なしの形式、アクセント記号付きの形式の順にソートする。
辞書順、大文字と小文字を区別しない	データを辞書順でソートするが、大文字と小文字は区別しない。大文字と小文字は同等に扱われ、次の「ソートの規則」の説明どおりにソートされる。
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	大文字を優先してソートし、大文字ソート分を小文字ソート分の前に置く。ただし、比較を (where 句内などで) 実行する場合は、大文字と小文字を区別しない。
辞書順、大文字/小文字とアクセントを区別しない	データを辞書順にソートするが、大文字と小文字を区別しない。また、アクセント記号付きの文字を、アクセント記号なしの元の文字と等しく扱う。このソート順により、ソートの結果では、アクセント記号付きとアクセント記号なしの文字の並べ方に、特定の順序はない。

- `sp_helpsort` は、Adaptive Server にインストールされているソート順をレポートします。

#### ソートの規則

2つのローが Adaptive Server のソート順で同じ値になるとき、以下の規則に従って、それらのローの優先順位を決定します。

- `order by` 句で指定されたカラム内の値が比較されます。
- 2つのローに等しいカラム値がある場合は、ロー全体のバイナリ値がバイトごとに比較されます。この比較は、カラムが内部に保管された順でローに対して行われ、クエリ内またはテーブル作成時の `create table` 句内で指定されたカラムの順では比較されません。つまり、データはすべての固定長カラムを順番に並べ、その後すべての可変長カラムを順番に並べて保管されます。
- ローが等しい場合は、ローの ID が比較されます。

次のテーブルを作成するとします。

```
create table sortdemo (lname varchar (20),
                      init char (1) not null)
```

次のようなデータを使用します。

```
lname      init
```

```

-----
Smith      B
SMITH     C
smith     A

```

*lname* でソートすると、結果は次のようになります。

```

lname      init
-----
smith     A
Smith     B
SMITH     C

```

固定長の `char` データ (`init` カラム) が最初に内部で保管されるため、`order by` 句は、バイナリ値 “Asmith”、“BSmith”、“CSMITH” を基準にしてこれらのローをソートします。

ただし、`init` が `varchar` 型である場合は、最初に *lname* カラム、次に `init` カラムが保管されます。バイナリ値 “SMITHC”、“SmithB”、“smithA” で比較が行われ、ローがこの順序で返されます。

#### 降順スキャン

- `order by` 句の `desc` キーワードを使用すると、クエリ・オプティマイザはワークテーブルとソート手順を必要としない方式を選択し、降順で結果を返すことができます。この最適化は、各インデックス・ページの前ページ・ポインタに続いて、逆の順序でインデックスのページ・チェーンをスキャンします。

この最適化を使用するには、`order by` 句内のカラムとインデックス順が一致している必要があります。カラムとしてはキーのサブセットが使えますが、必ずプレフィクス・サブセットであるものとします。つまり先頭キーを持っていなければなりません。`order by` 句で指名したカラムがインデックス・キーのスーパーセットである場合、降順スキャンの最適化は使用できません。

クエリにジョインがある場合、キーのプレフィクス・サブセットが一致すれば、すべてのテーブルをキーの降順でスキャンできます。降順スキャンの最適化は、ジョイン内の1つ以上のテーブルにも使えます。このとき、他のテーブルは昇順でスキャンされます。

- 他のユーザ・プロセスが更新または削除を実行するために順方向にスキャンしているときに、降順スキャンを実行するとデッドロックが生じます。デッドロックは、ページ分割とページ縮小のときにも生じることがあります。`sp_sysmon` を使用してサーバのデッドロックをトラックするか、`print deadlock information` 設定パラメータを使用してデッドロック情報をエラー・ログに送信できます。

- アプリケーションが降順で結果を返さなければならないのに、降順スキャン最適化でデッドロックの問題が生じた場合は、次の対処方法が有効です。
  - 降順スキャンには、`set transaction isolation level 0` スキャンを使用する。独立性レベル 0 読み取りの影響の詳細については、`set` コマンドの説明と『パフォーマンス&チューニング・ガイド：ロック』の「ロック・コマンドの使用」を参照してください。
  - 設定パラメータ `allow backward scans` を使用して、降順スキャンの最適化を無効にする。こうすることで `desc` を使用しているすべてのクエリが昇順でテーブルをスキャンでき、結果セットを降順にソートできるようになる。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。
  - 問題のある降順スキャンを2つの手順に分ける。最初の手順で必要なローを昇順で選択してテンポラリー・テーブル内に入れ、2番目の手順でテンポラリー・テーブルから降順で選択する。
- 重複するキー値があるため、オーバーフロー・ページのあるクラスタード・インデックスを後方スキャンで使用する場合、昇順スキャンが返した結果セットは、降順スキャンが返した結果セットの正反対の順序にはならないことがあります。指定のキー値は順序どおりに返されますが、オーバーフロー・ページ上の同一キーのローの順序は異なることがあります。クラスタード・インデックスのオーバーフロー・ページを格納する方法については、『パフォーマンス&チューニング・ガイド：基本』の「インデックス」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

`union` 演算子を使用するときに、`select` 文の `order by` 句内に新しいカラム見出しを指定する機能は、Transact-SQL 拡張機能です。

同名のテーブル・カラムおよびカラム・エイリアス名が存在する場合の `order by` の動作は、ANSI SQL 標準のベンダ固有の拡張です。

## 参照

コマンド `compute` 句, `declare`, `group by` 句と `having` 句, `select`, `where` 句  
システム・プロシージャ `sp_configure`, `sp_helpsort`, `sp_lock`, `sp_sysmon`

## prepare transaction

説明	2 フェーズ・コミット・アプリケーションで、サーバがトランザクションをコミットする準備ができているかどうかを確認するために、DB-Library によって使用されます。
構文	prepare tran[saction]
使用法	<ul style="list-style-type: none"><li>『Open Client DB-Library リファレンス・マニュアル』を参照してください。</li></ul>
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
参照	コマンド <a href="#">begin transaction</a> , <a href="#">begin transaction</a> , <a href="#">rollback</a> , <a href="#">save transaction</a>

# print

説明

ユーザ定義のメッセージをユーザの画面に表示します。

構文

```
print
    {format_string | @local_variable |
    @@global_variable}
    [, arg_list]
```

パラメータ

*format\_string*

変数または文字列のどちらでも可能です。*format\_string* の最大長は 1023 バイトです。

フォーマット文字列には、順不同で最大 20 個までのユニークなプレースホルダを入れることができます。プレースホルダは、メッセージのテキストがクライアントに送信される時に、*format\_string* に続く引数のフォーマット済みの内容に置き換えられます。

フォーマット文字列が異なる文法構造の言語に翻訳されるときに、引数の再編成ができるように、プレースホルダに番号が付けられます。1 つの引数のプレースホルダは、“%*nm*!” パーセント記号 (%) の後に 1 ~ 20 の整数を指定し、感嘆符 (!) を付けます。引数リスト内の文字列の整数は、引数の番号を表します。“%1!” “%1!” は元のバージョンでの最初の引数、“%2!” は 2 番目の引数となります (以降同様)。

この方法で引数の位置を指示すると、目的の言語に現れる引数の順序が、元の言語での順序と異なっても、引数は正しく変換されます。

たとえば、次は英語で書かれたメッセージです。

```
%1! is not allowed in %2!.
```

次は、このメッセージをドイツ語で書いたものです。

```
%1! ist in %2! nicht zula}ssig.
```

次は、このメッセージを日本語で書いたものです。

```
%2! の中で %1! は許されません。
```

この例で使用されている 3 つの言語では、“%1!” と “%2!” は同じ引数を表します。この例は、翻訳文で必要な引数の再編成を示しています。

**@local\_variable**

char 型、nchar 型、varchar 型、または nvarchar 型でなければなりません。また、使用されるバッチまたはプロシージャ内で宣言する必要があります。

**@@global\_variable**

char 型または varchar 型でなければなりません。あるいは、これらの型に自動的に変換される @@version などである必要があります。現在、@@version は、唯一の文字型グローバル変数です。

**arg\_list**

カンマで区切った一連の変数または定数です。arg\_list は、"%nn !" 形式のプレースホルダを含むフォーマット文字列が提供されないかぎり、省略可能です。この場合、arg\_list は、少なくともプレースホルダの最大番号と同じ数の引数を持っていなければなりません。引数は、text または image 以外の任意のデータ型を指定できます。これは文字データ型に変換されてから、最終のメッセージに組み込まれます。

**例** **例 1** authors テーブルに、郵便番号が 94705 の地域 (Berkeley) に在住している著者がいれば、"Berkeley author" を出力します。

```
if exists (select postalcode from authors
where postalcode = '94705')
print "Berkeley author"
```

**例 2** 変数を宣言し、値を変数に割り当て、値を出力します。

```
declare @msg char (50)
select @msg = "What's up, doc?"
print @msg

What's up, doc?
```

**例 3** メッセージ内の変数とプレースホルダの使い方の例です。

```
declare @tablename varchar (30)
select @tablename = "titles"

declare @username varchar (30)
select @username = "ezekiel"

print "The table '%1!' is not owned by the user '%2!'.",
@tablename, @username

The table 'titles' is not owned
by the user 'ezekiel.'
```

使用法

- 置き換えられたすべての引数と format\_string を加えた出力文字列の長さは、最大 1023 バイトです。



- フォーマット文字列でプレースホルダを使用する場合は、次の点に注意してください。文字列内の各  $n$  プレースホルダでは、 $1 \sim n-1$  のプレースホルダが同じ文字列内に存在しなければなりません。ただし、番号順である必要はありません。たとえば、フォーマット文字列にプレースホルダ 2 がなくて、プレースホルダ 1 と 3 を持つことはできません。フォーマット文字列の番号を省略すると、`print` の実行時にエラー・メッセージが生成されます。
- `arg_list` には、`format_string` 内の各プレースホルダに対して、1 つの引数が含まれていなければなりません。そうでないと、トランザクションがアボートされます。プレースホルダよりも多くの引数を使用することができます。
- エラー・メッセージの一部としてリテラルのパーセント記号を使用する場合は `format_string` 内にパーセント記号を 2 つ ( ‘%%’ ) 使用してください。`format_string` にプレースホルダとして使用しないパーセント記号を 1 つ ( ‘%’ ) 使用すると、エラー・メッセージが返されます。
- 引数が `null` であると評価された場合、ゼロの長さの文字列に変換されます。ゼロの長さの文字列を出力しないようにするには、`isnull` 関数を使用してください。たとえば、`@arg` が `null` である場合、次の文では I think we have nothing here が出力されます。

```
declare @arg varchar (30)
select @arg = isnull (coll, "nothing") from
table_a where ...
print "I think we have %1! here", @arg
```

- ユーザ定義メッセージは、どのアプリケーションも使用できるように `sysusermessages` システム・テーブルに追加できます。メッセージを `sysusermessages` に追加するには、`sp_addmessage` を使用します。`print` と `raiserror` 用のメッセージを検索するには `sp_getmessage` を使用します。
- ユーザ定義のエラー・メッセージを出力して、エラー番号を `@@error` に保管するには、`print` ではなく、`raiserror` を使用します。

## 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

## パーミッション

`print` の使用にパーミッションは必要ありません。

## 参照

コマンド `declare`, `raiserror`システム・プロシージャ `sp_addmessage`, `sp_getmessage`

## quiesce database

**説明** 指定されたデータベースのリストに対する更新処理を中断および再開します。

**構文** `quiesce database tag_name hold database_list [for external dump]  
[to manifest_file [with override]]`

または

`quiesce database tag_name release`

**パラメータ**

*tag\_name*

保留 (hold) または解放 (release) するデータベースのリストを表す、ユーザ定義の名前です。 *tag\_name* は、識別子の規則に従っていなければなりません。

**hold**

`to manifest_file` 句を指定すると、データベースが保留状態になり、マニフェスト・ファイルが作成されます。

---

**警告！** マニフェスト・ファイルはバイナリ・ファイルであるため、ファイルの内容の文字変換が可能な操作 (ftp など) は、バイナリ・モードで実行しなければファイルが破損します。

---

*database\_list*

`quiesce database hold` コマンドの対象に含まれるデータベースのリストです。

**for external dump**

リスト内のデータベースへの更新が中断されている間に、影響を受けるすべてのデータベース・デバイスを、**Adaptive Server** 外部の機能を使用して物理的にコピーすることを指定します。コピー操作は、`dump database` と `load database` の組み合わせに代わるものとして機能します。

*manifest\_file*

一連のデータベース・デバイスに存在するデータベースを記述するバイナリ・ファイルです。該当のデバイスを占有する一連のデータベースが、切り離されていて、デバイス上で独立している場合にのみ作成できます。

マニフェスト・ファイルはバイナリ・ファイルであるため、ファイルの内容の文字変換が可能な操作 (ftp など) は、バイナリ・モードで実行しないと、ファイルが破損します。

with override

データベースへの `quiesce database` の正常な実行を妨げるすべての制限を無効にします。

例

**例 1** `salesdb` と `ordersdb` 上の更新アクティビティを中断します。

```
quiesce database report_dbs hold salesdb, ordersdb
```

**例 2** `report_dbs` とラベル付けされたデータベース上の更新アクティビティを再開します。

```
quiesce database report_dbs release
```

**例 3** `pubs2` データベースへの更新アクティビティを中断し、このデータベースの外部コピーを作成する意図を表します。

```
quiesce database pubs_tag hold pubs2 for external dump
```

**例 4** データベースを保留状態にし、別の `Adaptive Server` にコピーするためのデータベースのマニフェスト・ファイルを作成します。

```
quiesce database pubs_tag hold pubs2 for external dump to
"/work2/sybase1/mpubs_file with override
```

コマンドが完了すると、制御がユーザに返されます。

**例 5** データベース・デバイスをコピーし、`mount database with listonly` を使用して、コピー対象のすべてのデバイスを一覧表示します。

```
1> mount database all from "/data/sybase2/mfile1" with listonly
2> go
```

```
"/data/sybase1/d0.dbs" = "1dev1"
```

クワイース ( 静止 ) 状態の一連のデータベースのセットに、そのセットに含まれないデータベースへの参照が含まれている場合は、マニフェスト・ファイルを作成することはできません。この制約を回避するには、`with override` オプションを使用します。

```
quiesce database pubs2_tag release for external dump to Mpubs_file
```

**例 6** `col_db` のカラムの暗号化に使用された暗号化キーが `key_db` に含まれる場合、次のコマンドは正常に終了します。

```
quiesce database key_tag hold key_db for external
dump to "/tmp/keydb.dat"
```

```
quiesce database encr_tag hold col_db for external dump
to "/tmp/col.dat" with override
```

```
quiesce database col_tag hold key_db, col_db for
external dump to "/tmp/col.dat"
```

## 使用法

- `quiesce database` を `hold` キーワードと一緒に使用すると、指定されているデータベースへの更新すべてが中断されます。トランザクションは、中断しているデータベース内のデータを更新できません。また、チェックポイント・プロセスやハウスキーパ・プロセスのようなバックグラウンド・タスクは、中断状態にあるすべてのデータベースを省略します。
  - `quiesce database` を `release` キーワードと一緒に使用すると、前に中断していたデータベース上で、更新作業が再開します。
  - `quiesce database` を `for external dump` 句と一緒に使用すると、データベースの外部コピーを作成する意図を示すことができます。
  - 次の場合は、このデータベースのリカバリに失敗したり、データベースの一貫性が失われたりする可能性があります。
    - 持続性が低いデータベース (`tempdb` を含む) をクワイース状態にする。
    - デバイスをコピーする。
    - このデータベースを他の `Adaptive Server` にマウントする。
- `alter database` を使用してデータベースの持続性を変更してから、`quiesce database` コマンドを発行してください。
- `quiesce database hold` コマンドと `release` コマンドは、同じユーザ・セッションから実行する必要はありません。
  - `quiesce database hold` コマンドで指定されたデータベース内に分散データベースまたはマルチ・データベースのトランザクションが準備状態で格納されていると、タイムアウト期間まで `Adaptive Server` はこれらのトランザクションの完了を待ちます。このタイムアウト時間内にトランザクションが完了しなかった場合は、`quiesce database hold` が失敗します。
  - `Adaptive Server` が、`quiesce database hold` 内で指定されているデータベース上で `dump database` コマンドまたは `dump transaction` コマンドを実行すると、データベースは、`dump` コマンドが完了した後のみ中断されます。
  - データベースへの更新が中断している間に、そのデータベース上で `dump database` コマンドまたは `dump transaction` コマンドを実行すると、`Adaptive Server` はこれらのコマンドをブロックします。`quiesce database release` コマンドによってデータベースが解放されるまで、このブロックは継続します。

- クワイスされたデータベースに対してクエリを実行しようとする  
と、Adaptive Server は次のようなエラー・メッセージ 880 を返し  
ます。

```
Your query is blocked because it tried to write and
database '%.*s' is in quiesce state. Your query will
proceed after the DBA performs QUIESCE DATABASE
RELEASE
```

データベースがクワイス状態でなくなると、クエリが実行されます。

- データベースを複写またはコピーするには、**quiesce database** の拡張機能を使用してマニフェスト・ファイルを作成します。**quiesce database** によって、データベースへの書き込みがブロックされて **quiesce hold** の状態になり、マニフェスト・ファイルが作成されません。その後、データベースの制御はユーザに返されます。これで、ユーティリティを使用して、データベースを別の Adaptive Server にコピーできます。コピー操作を行うときは、**quiesce database hold** の次のルールに従ってください。
  - quiesce database hold** プロセスが完了するまで、コピー操作は開始できません。
  - quiesce database** コマンドで指定したすべてのデータベースのすべてのデバイスをコピーする必要があります。
  - quiesce database release** を呼び出すには、コピー・プロセスが完了している必要があります。

#### 暗号化カラムと *quiesce database*

- データベースに暗号化キーが含まれるときに **quiesce database** を使用できます。
- 別のデータベースに格納されているキーで暗号化されたカラムを含むデータベースに対して **quiesce** を実行するときは、**with override** を使用してください。
- key\_db* が暗号化キーを含むデータベースで、*col\_db* が、*key\_db* にあるキーを使用して暗号化されたカラムを含むテーブルを格納するデータベースである場合は、**quiesce database key\_db, col\_db** を使用できます。

#### クラスタ環境での *quiesce database*

- shutdown instance** または **shutdown cluster** を発行すると、クラスタではすべての **quiesce database** コマンドがアボートされます。

- `shutdown instance` コマンドまたは `shutdown cluster` コマンドが進行中の場合、クラスタでは、ユーザによって発行されたすべての `quiescedb` コマンドが拒否されます。
- インスタンスのフェール・オーバー・リカバリが進行中の場合、クラスタではすべての `quiesce database` コマンドがアボートされます。
- インスタンスのフェール・オーバー・リカバリが進行中の場合、クラスタでは、ユーザによって発行されたすべての `quiesce database` コマンドが拒否されます。
- `master` データベースが進行中の `quiesce database hold` コマンドの要素になっている間は、クラスタに新しいインスタンスを追加できません。

パーミッション

`quiesce database` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>quiesce any database</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> が付与されたユーザであることが必要。

監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
96	quiesce	quiesce database	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照

コマンド [dump database](#)、[dump transaction](#)、`mount`、`unmount`  
 システム・プロシージャ [sp\\_helpdb](#)、[sp\\_who](#)

## raiserror

**説明** ユーザ定義のエラー・メッセージをユーザの画面に表示し、エラー条件が発生したことを記録するシステム・フラグを設定します。

**構文** `raiserror error_number`  
`[{format_string | @local_variable}] [, arg_list]`  
`[with errordata restricted_select_list]`

**パラメータ** `error_number`

`error_number` が 17,000 ~ 19,999 で、`format_string` がないか、または空 ("") である場合、Adaptive Server は、エラー・メッセージ・テキストの検索を、master データベース内の `sysmessages` テーブルから実行します。これらエラー・メッセージは、主にシステム・プロシージャによって使用されます。

`error_number` が 20,000 以上で、`format_string` がないか空である場合、`raiserror` は、クエリまたはストアド・プロシージャが発生したデータベース内の `sysusermessages` テーブルからメッセージ・テキストを取得します。Adaptive Server は、`@@langid` の現在の設定によって定義された言語の `sysmessages` または `sysusermessages` のいずれかからメッセージを取得しようとします。

`format_string`

最大長が 1024 バイトの文字列です。任意選択として、ローカル変数で `format_string` を宣言し、そのローカル変数を `raiserror` とともに使用できます (`@local_variable` を参照)。

`raiserror` は、表示される文字列でプレースホルダを認識します。フォーマット文字列には、20 個までのユニークなプレースホルダを順不同で含めることができます。これらのプレースホルダは、メッセージ・テキストがクライアントに送信されるときに、`format_string` の後に続く引数の中のフォーマットされた内容に置き換えられます。

フォーマット文字列が異なる文法構造の言語に変換される場合に、引数の順序を変更させるためプレースホルダには番号が付いています。1 つの引数のプレースホルダは、"%nn!" パーセント記号 (%) の後に 1 ~ 20 の整数を指定し、感嘆符 (!) を付けます。引数リスト内の文字列の整数は、引数の番号を表します。“%1!” は元のバージョンでの最初の引数、“%2!” は 2 番目の引数となります (以降同様)。

この方法で引数の位置を示すと、目的の言語に現れる引数の順序が、元の言語での順序と異なっても、引数は正しく変換されます。

たとえば、次は英語で書かれたメッセージです。

```
%1! is not allowed in %2!.
```

次は、このメッセージをドイツ語で書いたものです。

```
%1! ist in %2! nicht zulässig.
```

次は、このメッセージを日本語で書いたものです。

```
%2! の中で %1! は許されません。
```

この例で使用されている3つの言語では、"%1!"と"%2!"は同じ引数を表します。この例は、翻訳文に必要な引数の再編成を示しています。

#### @local\_variable

*format\_string* 値を含むローカル変数です。char 型または varchar 型でなければなりません。また、使用されるバッチまたはプロシージャ内で宣言する必要があります。

#### arg\_list

カンマで区切った一連の変数または定数です。arg\_list は、"%nn!"形式のプレースホルダを含むフォーマット文字列が提供されないかぎり、省略可能です。引数は、text 型または image 型以外であれば、どのようなデータ型でもかまいません。この引数は、char データ型に変換されてから最後の文字列に組み込まれます。

引数が NULL であると評価されると、この引数は Adaptive Server によってゼロの長さの char 文字列に変換されます。

#### with errordata

Client-Library™ プログラムの拡張エラー・データを提供します。

#### restricted\_select\_list

1つ以上の次の項目から構成されます。

- create table で指定した順序ですべてのカラムを表す "\*"。
- 参照する順序で指定したカラム名のリスト。既存の IDENTITY カラムを選択する場合は、実際のカラム名の代わりに、必要に応じて、テーブル名で修飾した syb\_identity キーワードを使用できます。
- 結果テーブルへの新しい IDENTITY カラムの追加の指定。

```
column_name = identity (precision)
```

- 次のいずれかの形式によるデフォルトのカラム見出し(カラム名)の置き換え。

```
column_heading = column_name
```



```
column_name column_heading
column_name as column_heading
```

これらの形式のカラム見出しは、引用符で囲むこともできます。カラム見出しが有効な識別子ではない場合（カラム見出しが予約語である場合、特殊文字で始まる場合、またはスペースや句読表記記号が入っている場合）は、その見出しを引用符で囲んでください。

- 式（カラム名、定数、関数、それらを算術演算子かビット処理演算子で接続して任意に組み合わせたもの、またはサブクエリ）。
- 組み込み関数または集合関数。
- 以上の項目の組み合わせ。

また、*restricted\_select\_list* は、次の形式で変数の割り当ても実行できます。

```
@variable = expression
[, @variable = expression ...]
```

次に、*restricted\_select\_list* に対する制限事項を示します。

- 変数の割り当てをその他のどの *restricted\_select\_list* オプションとも一緒に使用することはできません。
- *restricted\_select\_list* で *from*、*where*、またはその他の *select* 句を使用することはできません。
- “\*” を使用して、*restricted\_select\_list* のすべてのカラムを表すことはできません。

『Transact-SQL ユーザーズ・ガイド』を参照してください。

例

**例 1** このストアド・プロシージャの例では、ストアド・プロシージャが *@tablename* パラメータで指定されたテーブルを検出しない場合に、エラーが返されます。

```
create procedure showtable_sp @tablename varchar (18)
as
if not exists (select name from sysobjects
where name = @tablename)
begin
raiserror 99999 "Table %1! not found.",
@tablename
end
else
begin
select sysobjects.name, type, crdate, indid
```

```

        from sysindexes, sysobjects
        where sysobjects.name = @tabname
        and sysobjects.id = sysindexes.id
    end
end

```

**例 2** この例では、メッセージを `sysusermessages` に追加し、その後 `raiserror` で代わりの引数を提供してそのメッセージをテストします。

```

sp_addmessage 25001,
    "There is already a remote user named '%!'"
for remote server '%2!'.

```

```
raiserror 25001, jane, myserver
```

**例 3** この例では、`with errordata` オプションを使用して拡張エラー・データ `column` および `server` をクライアント・アプリケーションに返して、関連したカラムと使用されたサーバを示します。

```

raiserror 20100 "Login must be at least 5
    characters long" with errordata "column" =
    "login", "server" = @@servername

```

## 使用法

- ユーザ定義メッセージは、上記の例 1 と例 3 のように臨機応変に生成することができます。また、例 2 のようにどのアプリケーションでも使用できるようにシステム・テーブル `sysusermessages` に追加することもできます。メッセージを `sysusermessages` に追加するには、`sp_addmessage` を使用します。`print` および `raiserror` で使用されるメッセージを検索するには、`sp_getmessage` を使用します。
- ユーザ- 定義エラー・メッセージ用のエラー番号は、20,000 より大きくしてください。最大値は 2,147,483,647 ( $2^{31}-1$ ) です。
- ユーザ定義エラー・メッセージの重大度のレベルは、すべて 16 です。このレベルは、ユーザが致命的でないエラーを発生させたことを示します。
- `format_string` と変換後のすべての引数の最大出力文字列長は、1024 バイトです。
- フォーマット文字列でプレースホルダを使用する場合は、次の点に注意してください。文字列内の各  $n$  プレースホルダでは、 $1 \sim n-1$  のプレースホルダが同じ文字列内に存在しなければなりません。ただし、番号順である必要はありません。たとえば、フォーマット文字列にプレースホルダ 2 がなくて、プレースホルダ 1 と 3 を持つことはできません。フォーマット文字列の番号を省略すると、`raiserror` の実行時にエラー・メッセージが生成されます。

- *format\_string* 内のプレースホルダの数に対して引数の数が極端に少ない場合は、Adaptive Server がエラー・メッセージを表示し、現在実行中の文はアボートされますが、オープン・トランザクションはアボートされません。ただし、このエラーが保管されたプロシージャ内で発生した場合、Adaptive Server は `raiserror` を呼び出した行で次の文を継続し、オープン・トランザクションはそのままオープンとなります。ただし、このエラーが SQL コードのバッチ内で発生した場合は、Adaptive Server はそのバッチをアボートし、オープン・トランザクションはそのままオープンとなります。
- エラー・メッセージの一部としてリテラルのパーセント記号を使用する場合は *format\_string* 内にパーセント記号を2つ(‘%%’)使用してください。*format\_string* にプレースホルダとして使用しないパーセント記号を1つ(‘%’)使用すると、エラー・メッセージが返されます。
- 引数が NULL であると評価された場合、ゼロの長さの char 文字列に変換されます。ゼロの長さの文字列を出力しないようにするには、`isnull` 関数を使用してください。
- `raiserror` が実行されると、エラー番号は `@@error` グローバル変数に保管されます。このグローバル変数は、システムによって生成された最新のエラー番号を保管します。
- `@@error` にエラー番号を保管する場合は、`print` ではなく `raiserror` を使用してください。
- `raiserror` に *arg\_list* を含めるには、最初の引数の前にある *error\_number* または *format\_string* の後にカンマを入れてください。拡張エラー・データを指定するには、最初の *extended\_value* を、(カンマではなく)スペースを使用して、*error\_number*、*format\_string*、または *arg\_list* と区切ってください。

標準規格

パーミッション

参照

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

`raiserror` の使用にパーミッションは必要ありません。コマンド `declare`、`print`システム・プロシージャ `sp_addmessage`、`sp_getmessage`

## readtext

**説明** 指定のオフセット位置から開始して、指定のバイト数または文字数の `text` 値、`unitext` 値、`image` 値を読み込みます。

**構文**

```
readtext [[database.]owner.]table_name.column_name
        text_pointer offset size
        [holdlock | noholdlock] [readpast]
        [using {bytes | chars | characters}]
        [at isolation {
            [read uncommitted | 0] |
            [read committed | 1] |
            [repeatable read | 2] |
            [serializable | 3 ]}]
```

**パラメータ**

*table\_name.column\_name*

`text`、`unitext`、または `image` カラムの名前です。テーブル名を入れてください。テーブルが別のデータベース内にある場合はデータベース名を指定し、データベース内にその名前のテーブルが複数ある場合は所有者名を指定します。`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。

*text\_pointer*

`text`、`unitext`、または `image` データへのポインタを格納する `varbinary(16)` 値です。この値を調べるには、`textptr` 関数を使用します。`text`、`unitext`、`image` データは、他のテーブル・カラムと同じリンク・ページ・セットには格納されません。このデータは、リンク・ページの個別のセットに格納されます。実際のロケーションへのポインタはデータとともに格納されます。`textptr` はこのポインタを返します。

*offset*

`text`、`unitext`、または `image` データを読み取る前に、省略するバイト数または文字数を指定します。

*size*

読み取るデータのバイト数または文字数を指定します。

*holdlock*

トランザクションが終了するまで、テキスト値を読み取り用にロックします。他のユーザはこのテキスト値を参照することはできませんが、変更することはできません。

**noholdlock**

現在有効であるトランザクションの独立性レベルに関係なく、この文の実行中に取得したロックをサーバが保持することのないようにします。**holdlock** オプションと **noholdlock** オプションを同時に 1 つのクエリで指定することはできません。

**readpast**

**readtext** が排他ロックを保持するローを、待機したりメッセージを生成したりすることなく、暗黙的に省略するように指定します。

**using**

**readtext** が **offset** パラメータと **size** パラメータを、バイト数 (bytes) として解釈するか、または **textptr** 文字数 (chars や characters は同義語) として解釈するかどうかを指定します。このオプションは、1 バイト文字セットまたは、**image** 値と一緒に使用すると効果がありません (**readtext** は、**image** 値をバイト単位で読み込みます)。using オプションが指定されていない場合、**readtext** は **size** 引数と **offset** 引数をバイトとして解釈します。

**at isolation**

クエリの独立性レベル (0、1、または 3) を指定します。この句を省略すると、クエリでは、それが実行されるセッションの独立性レベル (デフォルトでは独立性レベル 1) が使用されます。**holdlock** を、**at isolation read uncommitted** も指定しているクエリ内で指定すると、Adaptive Server は警告を出し、**at isolation** 句を無視します。他の独立性レベルの場合、**holdlock** は **at isolation** 句よりも優先されます。

**read uncommitted**

クエリに対して独立性レベル 0 を指定します。**read uncommitted** の代わりに、**at isolation** 句で 0 を指定できます。

**read committed**

クエリに対して独立性レベル 1 を指定します。**read committed** の代わりに、**at isolation** 句で 1 を指定できます。

**repeatable read**

クエリに対して独立性レベル 2 を指定します。**serializable** の代わりに、**at isolation** 句で 2 を指定できます。

**serializable**

クエリに対して独立性レベル 3 を指定します。**serializable** の代わりに、**at isolation** 句で 3 を指定できます。

## 例

**例 1** copy カラムの 2 ~ 6 番目の文字が選択されます。

```
declare @val varbinary (16)
select @val = textptr (copy) from blurbs
```

```
where au_id = "648-92-1872"  
readtext blurbs.copy @val 1 5 using chars
```

## 例 2

```
declare @val varbinary (16)  
select @val = textptr (copy) from blurbs readpast  
where au_id = "648-92-1872"  
readtext blurbs.copy @val 1 5 readpast using chars
```

## 使用法

- ローが複数返される場合、`textptr` 関数は 16 バイトのバイナリ文字列 (テキスト・ポインタ) を返します。この文字列が返される先は、指定されたロー内にある `text`、`unitext`、`image` カラム、あるいはクエリによって返された最後のローにある `text`、`unitext`、`image` カラムです。テキスト・ポインタを保持するローカル変数を宣言し、その変数を `readtext` で使用してください。
- グローバル変数内の値は、返されるデータのバイト数の上限です。`readtext` に指定されたサイズよりこの値が小さい場合は、`@@textsize` の値が適用されます。`@@textsize` の値を変更するには、`set textsize` を使用します。
- オフセットおよびサイズとしてバイトを使用すると、返される `text` データの始めと終わりでの不完全な文字が、`Adaptive Server` によって検出されることがあります。検出した場合、文字セット変換がオンになっていると、サーバは不完全な文字を疑問符 (?) に置き換えてから、テキストをクライアントに返します。
- `Adaptive Server` は、`readtext` コマンドに応答してクライアントに送信するバイト数を決定しなければなりません。`offset` と `size` がバイト単位である場合には、返されるテキスト内のバイト数は、簡単に決定できます。オフセットとサイズが文字で指定されている場合、サーバはクライアントに返されるバイト数を計算する必要があります。この結果、文字を `offset` や `size` として使用すると、パフォーマンスが低下することがあります。`using characters` オプションは、`Adaptive Server` がマルチバイト文字セットを使用している場合に便利です。このオプションを指定すると、`readtext` によって不完全な文字が返されることはありません。
- ビューの `text`、`unitext`、または `image` カラムでは、`readtext` を使用できません。
- マルチバイトの文字セットに変更した後で `text` 値で `readtext` を使用し、`dbcc fix_text` を実行しなかった場合、このコマンドは失敗します。そのテーブルで `dbcc fix_text` を実行するように指示するエラー・メッセージが表示されます。

### *unitext* カラムでの *readtext* の使用

*unitext* データ型用に定義したカラムで *readtext* を発行するときは、*unitext* データを読み込む前にスキップするバイト数または Unicode 値を *readtext offset* パラメータで指定します。*readtext size* パラメータで、読み込むバイト数または 16 ビットの Unicode 値を指定します。*using bytes* (デフォルト) を指定すると、*offset* 値と *size* 値は必要に応じて Unicode 文字境界の開始と終了になるように調整されます。

*enable surrogate processing* を有効にすると、*readtext* はサロゲート境界でのみトランケートされ、開始位置と終了位置も適切に調整されて Unicode 文字の全体が返されます。この理由から、*unitext* 用に定義したカラムに対して *readtext* を発行すると、指定より少ないバイト数が返されることがあります。

次の例では、*unitext* カラム *ut* に文字列 U+0101U+0041U+0042U+0043 が含まれます。

```
declare @val varbinary (16)
select @val = textptr (ut) from unitable
where i = 1
readtext foo.ut @val 1 5
```

このクエリは値 U+0041U+0042 を返します。

*offset* 位置は 2 に調整されます。これは、*readtext* が Unicode 文字の 2 番目のバイトから開始できないためです。Unicode 文字は、必ず偶数のバイト数から構成されます。2 番目のバイトから開始 (または奇数のバイト数で終了) すると、結果が 1 バイト分ずれて、結果セットが不正確になります。

上の例では、*size* 値は 4 に調整されます。これは、*readtext* が 4 番目の文字 U+0043 のバイトの一部を読むことができないためです。

上のクエリでは、*enable surrogate processing* が有効で、*ut* カラムには文字列 U+d800dc00U+00c2U+dbffdeffU+d800dc00 が入っています。

```
declare @val varbinary (16)
select @val = textptr (ut) from unitable
where i = 2
readtext foo.ut @val 1 8
```

このクエリは値 `U+00c2U+dbffdeff` を返します。開始位置は 2 にリセットされ、実際の結果サイズは 8 バイトではなく 6 バイトになります。これは、`readtext` がサロゲート・ペアの中で分割を行わないためです。サロゲート・ペア (この例では、範囲 `d800..dbff` にある最初の値と範囲 `dc00..dfff` にある 2 番目の値) は 4 バイト境界を必要としますが、UTF-16 準拠の Unicode のルールではこれらの 4 バイト文字の分割はできません。

#### *readpast* オプションの使用

- `readpast` オプションは、データオンリー・ロック・テーブルにのみ適用されます。全ページ・ロック・テーブルに指定されている `readpast` は無視されます。
- `readpast` は、`holdlock` オプションと同時に使用できません。両方のオプションを 1 つのコマンド内で指定すると、エラーが生成されて、コマンドは終了します。
- `readtext` が `at isolation read uncommitted` を指定すると、`readpast` が警告を生成します。コマンドは終了しません。
- 文の独立性レベルが 3 に設定されると、`readpast` はエラー・メッセージを生成し、コマンドを終了します。
- セッション-全体の独立性レベルが 3 の場合、`readpast` は無視されます。
- セッションワイドの独立性レベルが 0 の場合、`readpast` は警告を生成しますが、コマンドの強制終了は行いません。

#### 標準規格

ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

#### パーミッション

`readtext` には、テーブルに対する `select` パーミッションが必要です。

#### 参照

コマンド [set](#)、[writetext](#)

システム・プロシージャ [text](#)、[image](#)、[unitext](#) データ型



## reconfigure

説明	現在、 <code>reconfigure</code> コマンドは無効です。このコマンドは既存のスクリプトを変更せずに実行できるようにすることを目的として存在します。
構文	<code>reconfigure</code>
使用法	<code>reconfigure</code> コマンドを含むスクリプトがある場合は、できるだけ早く変更してください。 <code>reconfigure</code> コマンドは、今回のバージョンには含まれていますが、次のバージョンではサポートされなくなる可能性があります。
標準規格	ANSI SQL – 準拠レベル : Transact-SQL 拡張機能
パーミッション	<code>reconfigure</code> パーミッションは、デフォルトではシステム管理者に付与されており、他のユーザに譲渡することはできません。
参照	システム・プロシージャ <a href="#">sp_configure</a>

## refresh

説明	指定された事前計算済み結果セットをリフレッシュします。
構文	<code>refresh precomputed result set <i>prs_name</i></code>
パラメータ	<i>prs_name</i> 事前計算済み結果セットの名前。完全修飾された <i>prs_name</i> にサーバまたはデータベースの名前を含めることはできません。
例	事前計算済み結果セット <code>au_prs</code> をリフレッシュします。 <pre>refresh precomputed result set au_prs</pre>
使用法	<ul style="list-style-type: none"><li>• <code>refresh precomputed result set</code> パーミッションは、デフォルトでは事前計算済み結果セットの所有者に与えられています。このパーミッションは、他のユーザへの譲渡が可能です。</li></ul>
標準規格	<code>refresh precomputed result set</code> コマンドは、Transact-SQL 拡張機能で、SQL 標準には含まれていません。
パーミッション	事前計算済み結果セットのリフレッシュには、 <code>create table</code> 権限が必要です。
監査	事前計算済み結果セットのリフレッシュは監査対象ではありません。

## remove java

**説明** Java クラスがデータベース内にインストールされている場合に、1 つ以上の Java-SQL クラス、パッケージ、JAR をデータベースから削除します。

**構文**

```
remove java
    class class_name[, class_name]...
        | package package_name[, package_name]...
        | jar jar_name[, jar_name]...[retain classes]
```

**パラメータ**

**class *class\_name***  
データベースから削除される Java クラス (1 つまたは複数) の名前です。クラスは現在のデータベースにインストールされていなければなりません。

**package *package\_name***  
削除される Java パッケージ (1 つまたは複数) の名前です。パッケージは現在のデータベースに保管されていなければなりません。

**jar *jar\_name***  
SQL 識別子または文字列値 (有効な SQL 識別子を含む最大 30 バイトの文字列) のどちらかです。各 *jar\_name* は、現在のデータベース内に保持されている JAR の名前と同じでなければなりません。

**retain classes**  
指名された JAR がデータベース内に保持されないように指定します。また、保持されているクラスが対応する JAR を持たないように指定します。

- 使用法**
- `remove java` 文がストアド・プロシージャ内に含まれていると、プロシージャが作成される時点で使用されているデータベースがカレント・データベースになります。プロシージャがコールされた時点で使用されているデータベースではありません。
  - `remove java` 文がストアド・プロシージャ内に含まれていない場合、`remove` 文が実行されるときの時点で使用されているデータベースが、カレント・データベースになります。
  - `class` または `package` が指定されたときに、対応する JAR を削除クラスが持っている場合、例外が発生します。
  - ストアド・プロシージャ、テーブル、ビューに、削除クラスへの参照 (カラムのデータ型、変数、またはパラメータ) が含まれていると、例外が発生します。
  - すべての削除クラスは、次のように処理されます。
    - 現在のデータベースから削除される。

- 現在の接続の **Java VM (Java 仮想マシン)** からアンロードされる。削除クラスは、他の接続の **Java VM** からはアンロードされない。
- **remove java** の実行中に例外が発生すると、**remove java** のすべての動作がキャンセルされます。
- **Java-SQL** クラスが **SQLJ** のストアド・プロシージャまたは関数によって直接参照されている場合は、このクラスを削除することはできません。
- **Java-SQL** クラスをデータベースから削除するには、次の手順に従います。
  - a クラスを直接参照している **SQLJ** ストアド・プロシージャと関数すべてを削除します。削除には、**drop procedure** および **drop function** を使用します。
  - b **remove java** を使用して、**Java-SQL** クラスをデータベースから削除します。

#### ロック

- **remove java** を使用すると、排他的テーブル・ロックが **sysxtypes** 上に配置されます。
- **jar** を指定すると、排他的テーブル・ロックが **sysjars** 上に配置されます。

#### パーミッション

**remove java** を使用できるのは、システム管理者またはデータベース所有者だけです。

**remove jar class** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <b>manage database</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であることが必要。

#### 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
94	remove	remove java	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

マニュアル 『Adaptive Server Enterprise における Java』

システム・プロシージャ `sp_helpjava`

システム・テーブル `sysjars`、`sysxtypes`

ユーティリティ `extractjava`、`installjava`

## reorg

**説明** ページ上の未使用領域を再利用したり、ロー転送を取り除きます。またはテーブル内のローすべてを新しいページに書き込み直します(使用しているオプションによって動作が異なります)。

**構文**

```
reorg compact table_name [partition partition_name]  
    [with {resume, time = no_of_minutes, compress}]  
reorg forwarded_rows table_name [partition partition_name]  
    [with {resume, time = no_of_minutes, compress}]  
reorg rebuild table_name [index_name [partition index_partition_name]  
    [with online]]]  
reorg reclaim_space table_name [index_name] [partition partition_name]  
    [with {resume, time = no_of_minutes, compress}]
```

**パラメータ**

**compact**

reorg reclaim\_space と reorg forwarded\_rows の関数を組み合わせて、領域の再利用とロー転送の取り消しの両方を、一度で実行します。

**forwarded\_rows**

ロー転送を取り除きます。

圧縮用に設定されたテーブルでは、データ・ローの非転送中や再挿入中、テーブルの圧縮レベルに合わせてローが圧縮されます。

**index\_partition\_name**

reorg を実行しているインデックス・パーティションの名前です。update statistics は、*index\_partition\_name* がインデックス・パーティションであることを確認するために検査を実行します。インデックス・パーティションを指定すると、そのインデックス・パーティションのみが再構築されます。

**indexname**

再編成されるインデックスの名前を指定します。

**partition\_name**

reorg を実行しているパーティションの名前です。

**tablename**

再編成されるテーブルの名前を指定します。*indexname* が指定されている場合、インデックスのみが再編成されます。

**rebuild**

テーブル名が指定されている場合は、テーブルのすべてのローを新しいページに書き込み直し、すべてのページを現在の領域管理設定に準拠させます。また、転送されたローおよびページ上のローの間にギャップがないようにし、クラスタド・インデックスがあればそれに応じてテーブルが編成されるようにします。テーブルにインデックスがある場合、すべてのインデックスは削除され、再作成されます。インデックス名が指定されると、**reorg** は、読み込みと更新のアクティビティ用にテーブルを利用できるようにしたまま、そのインデックスを再構築します。

新しいローは、元のテーブルやパーティションのデータ圧縮レベルに関わらず、このパーティションやテーブルの圧縮レベルに従います。

---

**注意** **reorg rebuild** は、システム・カタログではサポートされていません。

---

**with online**

**reorg rebuild** 実行中のテーブルへの同時アクセスを可能にします。

**reclaim\_space**

削除や更新によって残された未使用の領域を再利用します。テーブル内の各データ・ページについて、コミットされた削除やローを短縮する更新の結果として残された使用していない領域がある場合、**reorg reclaim\_space** は現在のローを連続して書き込み直し、使用していない領域すべてをページの最後に残します。ページにまったくローがなければ、ページの割り付けは解除されます。

テーブルに圧縮が指定されている場合は、**reclaim\_space** によってデータが圧縮されます。

---

**注意** **reorg reclaim\_space** は、可変長のローのあるテーブルのみに影響し、ページ内の領域のみを解放します。使用するページ数を削減するには、**reorg rebuild** コマンドを使用します。

---

**with resume**

前の **reorg** コマンドが終了したポイントから再編成を開始します。前回実行した **reorg** コマンドが制限時間 (**with time = no\_of\_minutes**) を指定していた場合に使用されます。

**with time = no\_of\_minutes**

**reorg** コマンドが実行される時間の長さ (分単位) を指定します。

with compress

reorg オペレーションに影響されるローを圧縮できるようにします。

例

**例 1 titles** テーブル内の未使用ページ領域を再利用します。

```
reorg reclaim_space titles
```

**例 2 titleind** インデックスの未使用ページ領域を再利用します。

```
reorg reclaim_space titles titleind
```

**例 3 titles** テーブル上で **reorg compact** を開始します。reorg は、そのテーブルの初めから開始され、120 分間実行されます。reorg が制限時間内で完了すると、このコマンドはテーブルの初めに戻り、制限時間全体が経過するまで実行を続けます。

```
reorg compact titles with time = 120
```

**例 4** 前の **reorg compact** が停止したポイントで **reorg compact** を開始し、30 分間続行します。

```
reorg compact titles with resume, time = 30
```

**例 5 titles** テーブルの **smallsales** パーティションで **reorg forwarded\_rows** を実行します。

```
reorg forwarded_rows titles partition smallsales
```

**例 6 authors** テーブルで **reorg forwarded\_rows** を実行します。

```
reorg forwarded_rows authors
```

**例 7 titles** の **bigsales** パーティションで、**reorg reclaim\_space** を実行します。

```
reorg reclaim_space titles partition bigsales
```

**例 8 titles** の **bigsales** パーティションで **reorg compact** を実行します。

```
reorg compact titles partition bigsales
```

**例 9 titles** テーブルで **reorg compact** を実行し、影響を受けるローを圧縮します。

```
reorg compact titles with compress
```

**例 10** テーブル **sales** のインデックス **local\_idx** のインデックス・パーティション **idx\_p2** で **reorg rebuild** を実行します。

```
reorg rebuild sales local_idx partition idx_p2
```

使用法

- **reorg rebuild** を除き、**reorg** で指定されたテーブルには、データロー・ロック・スキームまたはデータページ・ロック・スキームが必要です。



- `reorg` を実行した後は、インデックス・スキャンの速度が向上します。
- テーブルに対して `reorg` を実行すると、同時クエリのパフォーマンスに悪影響を与える可能性があります。
- インデックス名またはパーティション名を指定しないと、テーブル全体が再構築されます。
- テーブルのインデックスを再構築した後も、そのテーブルに対して `dump tran` を実行できます。ただし、テーブル全体を再構築した場合は、`dump tran` を実行することはできません。
- 配置インデックスに対してはオンラインでのインデックスの再構築が可能ですが、再構築されるのはインデックス・ページだけです。データ・ページは変更されません。したがって、データローはソートされず、新しいページへの再書き込みも行われません。データ・ページを再構築するには、配置インデックスを削除して再作成します。
- `systabstats` のインデックスは再構築できますが、このテーブル自体に `reorg rebuild` を実行することはできません。
- 15.0 以前のバージョンの Adaptive Server では、全ページ・ロック・テーブルでの `reorg rebuild` の使用が制限されていました。バージョン 15.0 以降の Adaptive Server では全ページ・ロックを使用するすべてのテーブルで `reorg rebuild` が使用できます。`reorg rebuild` は全テーブルを再構築し、データを新しいページ・セットにコピーし、すべてのインデックスを再構築します。
- 全ページロック・テーブルでは、`reorg rebuild` サブ・コマンド (`compact`、`reclaim_space`、`forwarded_rows` など) は使用できません。
- 全ページロック・テーブルでは、`reorg rebuild table_name index_name` を使用することはできません。
- `reorg rebuild table_name` を実行すると、すべての先行インデックス・カラムの統計が更新されます。ただし、`reorg rebuild table_name index_name` を実行した場合、統計の自動更新は行われません。代わりに、プランの選択やパフォーマンスに影響する大幅なデータ変更が更新に含まれる場合に `reorg rebuild index_name` を実行すると、インデックス統計が自動的に更新されます。
- `writetext` は、`online` パラメータと同時に実行できます。
- `reorg` は、`text` または `image` カラムに割り当てられた領域には影響しません。

- トランザクション内では、`reorg` は発行できません。
- `reorg rebuild` は、データベース・オプションの `select into/bulkcopy/pllsort` を `true` に設定し、データベース内で `checkpoint` を実行するように要求します。
- `reorg rebuild` には、テーブルとテーブルのインデックスと同じサイズの追加ディスク領域が必要です。テーブルが現在どれだけの領域を占有しているかを調べるには、`sp_spaceused` を使用します。また、`sp_helpsegment` を使用すると、使用可能領域の合計を調べることができます。
- `reorg rebuild` を実行した後、データベースをダンプしてからトランザクション・ログをダンプしてください。
- インデックスに対して `reorg rebuild` を使用するための条件は、テーブルの場合ほど厳密ではありません。次のような規則に従います。
  - インデックスの再構築には、`select into` を設定する必要はない。
  - テーブルの再構築には、テーブルを完全にコピーするための領域が別に必要となる。インデックスの再構築は、小さなトランザクションで実行でき、コピーされたページは割り付けが解除される。したがって、個々のトランザクションでコピーされるページの分の領域があれば十分である。
  - テーブルのインデックスを再構築できるのは、トランザクション・レベルのスキャン(ダーティ・リード)がアクティブのときである。

`reclaim_space`、`forwarded_rows`、`compact` の各パラメータには、次の特徴があります。

- 実行時間が非常に短い小さなトランザクションを複数実行して、他のアクティビティとの干渉を最低限に抑えます。個々のトランザクションは、8 ページ分の `reorg` 処理を行うように制限されます。
- 1 つのパーティションの領域に書き込み直します。
- `resume` オプションと `time` オプションを指定すると、`reorg` の実行の制限時間を設定できます。また、前回実行の `reorg` が停止した時点から `reorg` を再開することもできます。これにより、たとえば、オフピーク時に部分的な再編成を連続して実行することにより、大きなテーブルで `reorg` コマンドを実行できます。

### ガーベジ・コレクションとロック

- データロー テーブル – Adaptive Server はラッチを使用してガーベジ・コレクションを行い、次のページに進む前にページのラッチをリリースします。ガーベジ・コレクションが転送されたローに遭遇するまではロックは取得されません。この時点で排他テーブル・ロックが必要になり、そのトランザクションが終了するまで保持されます。それに続くトランザクションは転送ローに遭遇するまでラッチを使用します。
- データページ テーブル – Adaptive Server はページ・ロックを使用してガーベジ・コレクションを行います。次のページに進む前にページ・ロックをリリースします。ガーベジ・コレクションが転送されたローに遭遇すると排他テーブル・ロックが取得され、そのトランザクションが終了するまで保持されます。それに続くトランザクションは他の転送ローに遭遇するまでページ・ロックを使用します。
- ガーベジ・コレクションがオブジェクトに割り当てられた OAM ページに遭遇する場合は `reorg compact` を使用します。ただし、割り当ては照合しません (`reorg compact` を実行するには共有テーブル・ロックが必要です)。

### resume パラメータと time パラメータの使用

resume パラメータと time パラメータを使用するときは、次の点に注意してください。

- `resume` オプションのみを指定した場合には、`reorg` は前回の `reorg` の停止位置から が開始され、テーブルの最後まで続行します。
- `time` オプションのみを指定した場合は、`reorg` はテーブルの先頭から開始され、指定した時間 (分単位) だけ実行されます。
- 両方のオプションを指定すると、`reorg` は前回の `reorg` の停止位置から開始され、指定した時間だけ実行されます。

### 圧縮テーブルでの reorg の実行

- `reorg rebuild` – 配置インデックス (存在する場合) に従ってローをソートし、現在有効の領域管理設定に従って新しいデータ・ページにローを書き込みます。新しいローは、元のテーブルやパーティションのデータ圧縮レベルに関わらず、個々のパーティションの圧縮レベルに従って圧縮および圧縮解除されます。
- `reorg reclaim_space` – テーブルが圧縮用に指定されていれば、スペースを節約するためにデータを圧縮します。

標準規格

ANSI SQL – 準拠レベル: Transact-SQL 拡張機能

---

パーミッション	reorg のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であるか reorg any table 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者であるか sa_role が付与されたユーザであることが必要。
参照	マニュアル 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・プロシージャ sp_chgattribute

## return

説明	バッチまたはプロシージャを無条件に終了し、オプションでリターン・ステータスを提供します。return に続く文は実行されません。
構文	return [ <i>integer_expression</i> ] [plan " <i>abstract_plan</i> "]
パラメータ	<p><i>integer_expression</i></p> <p>プロシージャによって返された整数値です。ストアド・プロシージャは、呼び出しているプロシージャまたはアプリケーション・プログラムに整数値を返すことができます。</p> <p>plan "<i>abstract_plan</i>"</p> <p>クエリを最適化するために使用する抽象プランを指定します。抽象プランは、抽象プラン言語で指定された完全なものまたは部分的なものです。プランは、最適化可能な SQL 文、つまりテーブルにアクセスするクエリにのみ指定できます。詳細については、『パフォーマンス&amp;チューニング・シリーズ：オプティマイザと抽象プラン』の「抽象プランの作成と使用」を参照してください。</p>
例	<p><b>例 1</b> パラメータにユーザ名が指定されていない場合、return コマンドは、ユーザの画面にメッセージを送信した後、プロシージャを終了します。ユーザ名が指定されている場合は、現在のデータベース内のそのユーザによって作成されたルール名が、適切なシステム・テーブルから検索されます。</p> <pre> create procedure findrules @nm varchar (30) = null as if @nm is null begin     print "You must give a user name"     return end else begin     select sysobjects.name, sysobjects.id,            sysobjects.uid     from sysobjects, master..syslogins     where master..syslogins.name = @nm     and sysobjects.uid = master..syslogins.suid     and sysobjects.type = "R" end </pre> <p><b>例 2</b> 更新によってタイトルの平均価格が 15 ドルを超える場合、return コマンドは titles で他の更新が行われる前に、このバッチ処理を終了します。</p> <pre> print "Begin update batch" update titles </pre>

```

        set price = price + $3
        where title_id = 'BU2075'
update titles
        set price = price + $3
        where title_id = 'BU1111'
if (select avg (price) from titles
    where title_id like 'BU%') > $15
begin
    print "Batch stopped; average price over $15"
    return
end
update titles
        set price = price + $2
        where title_id = 'BU1032'

```

**例 3** このプロシージャでは、ユーザ定義のステータス・コードが2つ作成されます。contract カラムに1があるときには値1が、その他の状態（たとえば、contract の値が0、または title\_id がローと一致しない状態）では、値2が返されます。

```

create proc checkcontract @param varchar (11)
as
declare @status int
if (select contract from titles where title_id = @param)
= 1
    return 1
else
    return 2

```

## 使用法

- リターン・ステータス値は、現在のプロシージャを実行したバッチやプロシージャの後の文で使用できますが、次のフォーマットで指定する必要があります。

```
execute @retval = procedure_name
```

詳細については [execute](#) を参照してください。

- Adaptive Server では、値が正しく返されたことを示す0が予約されています。また、負の値の-1～-99は、それぞれ異なる障害の理由を示します。ユーザ定義の値が与えられない場合、Adaptive Server の値が使用されます。ユーザ定義のリターン・ステータス値は、Adaptive Server が予約している値と競合しないようにしてください。次の表は、現在使われている0および-1～-14の値を示します。

値	意味
0	プロシージャが正常に実行された
-1	オブジェクトがない

値	意味
-2	データ型のエラー
-3	プロセスがデッドロックの被害対象として選択された
-4	パーミッション・エラー
-5	構文エラー
-6	その他のさまざまなユーザ・エラー
-7	領域不足などのリソース・エラー
-8	致命的ではない内部の問題
-9	システムの限界に達した
-10	致命的な内部不整合
-11	致命的な内部不整合
-12	テーブルまたはインデックスが破壊されている
-13	データベースが破壊されている
-14	ハードウェア・エラー

-15 ~ -99 の値は、Adaptive Server で将来使用するために予約されています。

- プロシージャ実行中に複数のエラーが発生した場合は、最も大きな絶対値を持つステータスが返されます。ユーザ定義の戻り値は、Adaptive Server が提供する戻り値に常に優先します。
- バッチまたはプロシージャを終了するどの時点でも `return` コマンドを実行できます。完全な戻り値がすぐに返されます。`return` の後の文は実行されません。
- ストアド・プロシージャは、`null` リターン・ステータスを返すことはできません。たとえば、`@status` が `null` の場合に `return @status` を使用して `null` 値を返そうとすると、警告のメッセージが生成され、`0` ~ `-14` の値のいずれかが返されます。

標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

パーミッション

`return` の使用にパーミッションは必要ありません。

参照

コマンド [begin...end](#), [execute](#), [if...else](#), [while](#)

## revoke

### 説明

ユーザ、役割、グループに付与されたパーミッションを取り消します。

### 構文

データベース・オブジェクトにアクセスするパーミッションを取り消す場合：

```
revoke [grant option for]
      {all [privileges] | permission_list}
      on {table_name [(column_list)]
         | view_name [(column_list)]
         | stored_procedure_name | function_name
         | keyname}
      [with { pred_name | {all |no} predicates}]
      from {public | name_list | role_list}
      [cascade]
      [granted by grantor]
```

組み込み関数を選択するパーミッションを取り消す場合：

```
revoke select
      on [builtin] builtin
      from {name_list | role_list}
      [granted by grantor]
```

システム権限を取り消す場合：

```
revoke {all [privileges] | privilege_list}
      from {public | name_list | role_list}
      [granted by grantor]
```

set proxy の実行パーミッションを取り消す場合：

```
revoke set proxy
      from {public | name_list | role_list}
      [granted by grantor]
```

dbcc 権限を取り消す場合：

```
revoke {dbcc_privilege [on database]
       [, dbcc_privilege [on database ], ...]}
      from {user_list | role_list}
      [granted by grantor]
```

public からデフォルトのパーミッションを取り消す場合：

```
revoke default permissions on system tables
```



## パラメータ

## all

データベース・オブジェクトへのアクセス・パーミッションの割り当てに使用すると(「データベース・オブジェクトにアクセスするパーミッションを取り消す場合」の構文参照)、**all**では、指定されたオブジェクトに適用される **decrypt** 以外のすべてのパーミッションが取り消されます。オブジェクト所有者はすべて、オブジェクト名を指定した **revoke all** を使用して、所有するオブジェクトに対するパーミッションを取り消すことができます。**decrypt** パーミッションは別途取り消す必要があります。

細密なパーミッションが有効でない場合、システム管理者またはデータベース所有者は **revoke all** を使用して、データベース・オブジェクト作成の権限を取り消すことができます(「システム権限を取り消す場合」の構文を参照)。システム管理者が使用すると、**revoke all** では、すべての **create** 権限 (**create database**、**create default**、**create procedure**、**create rule**、**create table**、**create function**、**create view**) を取り消すことができます。データベース所有者が **revoke all** を使用するか、または **master** データベース以外で **revoke all** を実行すると、**Adaptive Server** によって **create database** 以外のすべての **create** 権限が取り消され、情報メッセージが出力されます。

細密なパーミッションが有効化されている場合、**revoke all** を使用した **create** 権限の取り消しのすべてを実行できることにはなりません。詳細については、『セキュリティ管理ガイド』の「細密なパーミッションの使用」を参照してください。

**where** 句が含まれる場合は、**revoke** 文で **all** を使用できません。

*permission\_list*

取り消すパーミッションのリストです。複数のパーミッションをリストするにはカンマで区切ってください。次の表は各タイプのオブジェクトに付与したり取り消したりできるアクセス・パーミッションを示します。

オブジェクト	<b>permission_list</b> に指定できるパーミッション
テーブル	<b>select</b> 、 <b>insert</b> 、 <b>delete</b> 、 <b>update references</b> 、 <b>update statistics</b> 、 <b>delete statistics</b> 、 <b>truncate table</b> 、 <b>decrypt</b> 、 <b>identity_insert *</b> 、 <b>identity_update *</b>
ビュー	<b>select</b> 、 <b>insert</b> 、 <b>delete</b> 、 <b>update</b> 、 <b>decrypt</b> 、 <b>identity_insert *</b> 、 <b>identity_update *</b>
カラム	<b>select</b> 、 <b>update</b> 、 <b>references</b> 、 <b>decrypt</b> カラム名は、 <b>permission_list</b> または <b>column_list</b> のどちらでも指定できる。
ストアド・プロシージャ	<b>execute</b>

オブジェクト	permission_list に指定できるパーミッション
暗号化キー	select
関数	execute*

**注意** アスタリスク (\*) が付けられたパーミッションは、細密なパーミッションが有効化されている場合のみ付与できます。

#### *builtin*

組み込み関数です。組み込み関数を指定すると、同じ名前のテーブルと取り消し可能な組み込み関数とを区別できます。これらの関数は、`set_appcontext`、`get_appcontext`、`list_appcontext`、`authmech`、`rm_appcontext` です。

#### *privilege\_list*

取り消し可能なシステム権限のリストです。システム権限にはサーバワイド、およびデータベースワイドの権限が含まれます。取り消し可能なシステム権限のリストについては、[表 1-21](#) と [表 1-22](#) を参照してください。また、システム権限の取り消し方法の詳細については、「[使用法](#)」のセクションも参照してください。複数のコマンドは、カンマで区切って使用します。

#### *table\_name*

パーミッションを取り消すテーブルの名前です。テーブルは、現在のデータベースになければなりません。

#### *column\_list*

権限を適用するカラムのリストです。リストはカンマで区切りません。カラムが指定されている場合は、`select`、`reference`、`decrypt`、`update` のパーミッションのみを取り消すことができます。

#### *view\_name*

パーミッションを取り消すビューの名前です。ビューは、現在のデータベースになければなりません。

#### *stored\_procedure\_name*

パーミッションを取り消すストアド・プロシージャの名前です。ストアド・プロシージャは、現在のデータベースになければなりません。

#### *]function\_name*

パーミッションを取り消す関数の名前です。関数は、現在のデータベースになければなりません。

**keyname**

パーミッションを取り消すキーの名前です。暗号化キーは、現在のデータベースになければなりません。revoke 文ごとにリストできるオブジェクトは 1 つだけです。キーから取り消すことができるのは、select パーミッションだけです。

**[with {pred\_name | {all | no} predicates}]**

指定述部、ダブルキーワード **all predicates**、またはダブルキーワード **no predicates** を続けることができます。

- **pred\_name** — 取り消す述部付き付与の名前。sp\_helprotect を使用するとロー・フィルタリング付与を識別する指定述部が表示されます。
- **no predicates** — 指定された付与対象者から述部なしの付与のみを削除するように Adaptive Server に指示します。
- **all predicates** - 指定した付与者から特定のアクセスのすべての述部付き付与を削除するように、Adaptive Server に指示します。述部なしの付与はすべて保持されます。

with 句を省略すると、述部付きと述部なしの両方の grant が取り消されます (デフォルトの動作)。

**public**

すべてのユーザです。オブジェクトのアクセス・パーミッションについては、public にオブジェクト所有者は含まれていません。オブジェクト作成パーミッションまたは set proxy 権限については、public にデータベース所有者は含まれていません。"public" や他のグループまたは役割に with grant option を使用してパーミッションを割り当てる (grant) ことはできません。

**name\_list**

ユーザ名とグループ名をカンマで区切ったリストです。

**set proxy**

ユーザから、他のユーザと同一化する権限を取り消します。

**grant option for**

**with grant option** パーミッションを取り消し、*name\_list* に指定されたユーザが、特定のパーミッションを他のユーザに付与できないようにします。リスト内に指定されているユーザが他のユーザに付与している場合は、**cascade** オプションを使用して、それらのユーザからもパーミッションを取り消す必要があります。*name\_list* に指定されているユーザは、オブジェクトにアクセスするためのパーミッションは保持しますが、他のユーザにアクセス権を付与することはできなくなります。**grant option for** は、オブジェクト・アクセス・パーミッションだけに適用され、オブジェクト作成パーミッションには適用されません。

**cascade**

パーミッションが取り消されたユーザがパーミッションを付与していたすべてのユーザから、特定のオブジェクトのアクセス・パーミッションを取り消します。オブジェクトのアクセス・パーミッションにだけ適用され、オブジェクトの作成パーミッションには適用されません (**revoke** を **grant option for** なしで使用すると、パーミッションが取り消されたユーザから他のユーザに付与されていたパーミッションも取り消されるため、自動的にカスケードが発生します)。

**granted by grantor**

**revoke** コマンドを実行しているユーザになり代わる *grantor* によって付与されたパーミッションまたは権限の取り消しを指定します。

**grantor**

現在のデータベース内の有効ユーザ名です。

**dbcc\_privilege**

取り消す **dbcc** 権限の名前です。変数は使用できません。表 1-21 と表 1-22 は **dbcc** 権限のリストを示しています。

**database**

パーミッションを取り消すデータベースの名前です。ターゲット・データベースのみを対象としてパーミッションを取り消す場合に、データベース固有の **dbcc** 権限とともに使用します。パーミッションが取り消されるユーザは、ターゲット・データベース内の有効なユーザでなければなりません。*database* は識別子の規則に従っている必要があります。また、変数を使用することはできません。

1つのコマンドで複数のアクションのパーミッションを取り消す場合、*database* はユニークでなければなりません。

**user\_list**

パーミッションを取り消すユーザのリストです。変数は使用できません。

*role\_list*

パーミッションを取り消すシステム定義またはユーザ定義の役割のリストです。変数は使用できません。

---

**注意** `public` またはグループに対して `dbcc` 権限を付与したり、取り消したりすることはできません。

---

## all [privileges]

all または all privileges を使用して付与または拒否されたすべての権限を取り消します。

*column\_list*

with *pred\_name* とともに使用すると、指定カラムに対する述部付きのロー・アクセスが取り消されます。このローレベル権限に関連して参照される他のカラムが存在する場合、縮小後のカラム・リストに対する権限とそれに関連する指定述部は `sysprotects` に保持されません。

## default permissions on system tables

「システム・テーブルのデフォルト・パーミッションの取り消し」(662 ページ) にリストしたシステム・テーブルのデフォルトのパーミッションを取り消します。

例

**例 1** Mary および "sales" グループから、titles テーブル上の insert パーミッションと delete パーミッションを取り消します。

```
revoke insert, delete
on titles
from mary, sales
```

**例 2** `get_appcontext` 関数の "public" (すべてのユーザを含む) から select パーミッションを取り消します。

```
revoke select on builtin get_appcontext from public
```

これを次のコマンドと比較してください。次の例では、`get_appcontext` というテーブル (存在する場合) に対する select パーミッションが取り消されます。

```
revoke select on get_appcontext from public
```

**例 3** 2 通りの方法で、“public” から、titles テーブルの price および advance カラムの update パーミッションを取り消します。

```
revoke update
on titles (price, advance)
from public
```

または

```
revoke update (price, advance)
on titles
from public
```

**例 4** Mary と John から、`create database` コマンドと `create table` コマンドを使用するパーミッションを取り消します。`create database` パーミッションが取り消されているため、このコマンドは `master` データベース内から実行する必要があります。Mary と John の `create table` パーミッションは、`master` データベース内でだけ取り消されます。

```
revoke create database, create table from mary, john
```

**例 5** Harry と Billy から、サーバ内の他のユーザと同一化するための `set proxy` または `set session authorization` を実行するパーミッションを取り消します。

```
revoke set proxy from harry, billy
```

**例 6** `sso_role` を持つユーザから、`set proxy` か `set session authorization` を実行するパーミッションを取り消します。

```
revoke set session authorization from sso_role
```

**例 7** 現在のデータベースで、Mary から、`create encryption key` と `create trigger` を除くすべてのオブジェクトの作成パーミッションを取り消します。

```
revoke all from mary
```

**例 8** Mary から、`decrypt` パーミッションを除く、`titles` テーブルのオブジェクトのアクセス・パーミッションをすべて取り消します。

```
revoke all on titles from mary
```

**例 9** `titles` テーブルにある `price` カラムと `advance` カラムを参照する他のテーブルに対して参照整合性の制約を作成するパーミッションを、2 とおりの方法で、Tom から取り消します。

```
revoke references
on titles (price, advance)
from tom
```

または

```
revoke references (price, advance)
on titles
from tom
```

**例 10** 「オペレータ」の役割を付与されたすべてのユーザから、`new_sproc` の実行パーミッションを取り消します。

```
revoke execute on new_sproc from oper_role
```

**例 11** authors テーブルでの insert、update、delete パーミッションを他のユーザに付与する John のパーミッションを取り消します。また、John が他のユーザに付与したパーミッションも取り消します。

```
revoke grant option for
insert, update, delete
on authors
from john
cascade
```

**例 12** ユーザ Frank の dbcc の権限を取り消します。

```
revoke dbcc checkdb on all from frank
```

**例 13** Harry の authors テーブルに対する truncate table 権限と update statistics 権限を取り消します。

```
revoke truncate table on authors from harry
revoke update statistics on authors from harry
```

ユーザ Billy と Harry は、これらのコマンドを authors に対して実行できなくなります。

**例 14** oper\_role を持つすべてのユーザの truncate table、update statistics、delete statistics の各権限を取り消します。

```
revoke truncate table on authors from oper_role
revoke update statistics on authors from oper_role
revoke delete statistics on authors from oper_role
```

**例 15** public から decrypt パーミッションを取り消します。

```
revoke decrypt on customer from public
```

**例 16** ユーザ joe から create encryption key パーミッションを取り消します。

```
revoke create encryption key from joe
```

**例 17** データベース所有者から ssn\_key の select on パーミッションを取り消します。

```
revoke select on ssn_key from dbo
```

**例 18** 以下の例は、user1 によって、テーブル t1 からの選択を行う以下の付与が実行されていることを前提としています。

- t1.col1 と t1.col4 のすべての行を表示できる無条件付与。

```
grant select on t1 (col1, col4) to user1
```

- `t1.col2` または `t1.col3` を選択するときに適用されるロー・フィルタリング付与。

```
grant select on t1 (col2, col3)
  where col1 = 1 as pred1
  to user1
```

`pred1` 付きの `t1.col2` に対する `select` パーミッションを削除します。

```
revoke select on t1 (col2) with pred1
  from user1
```

---

**注意** ただし、この取り消し後も `user1` が `t1.col3` を選択すると、引き続き `pred1` が適用されます。

---

付与者が以下のいずれかを発行した場合、`pred1` を使用した `t1` のすべてのパーミッションが `user1` によって取り消されます。

```
revoke select on t1 (col2, col3) with pred1
  from user1
```

または

```
revoke select on t1 with pred1
```

**例 19** 前述の例で示された付与の後に、以下を使用して、`t1.col2` と `t1.col3` に対する述部 `pred1` 付きの付与を取り消します (`all predicates` によって、特定のアクセスおよび付与対象者に対するロー・フィルタリングの述部付き付与のすべてが取り消されます)。

```
revoke select on t1 with all predicates
  from user1
```

**例 20** `user1` から `t1` に対するすべての `select` アクセスを削除します。つまり、`t1` に対するすべての述部付きおよび述部なしの付与で、`t1` に対する `select` アクセスが `user1` に付与されていました。

```
revoke select on t1 from user1
```

**例 21** 述部なしの付与にのみ適用。

```
revoke select on t1 with no predicates
```

**例 22** `Mary` によって付与されていた `John` のテーブル `mary.books` に対する `select` パーミッションを取り消す場合：

```
revoke select on mary.books from john granted by mary
```

**例 23** ユーザ `Smith` のシステム権限 `manage any login` を取り消す場合：

```
use master
revoke manage any login from smith
```



## 使用法

- パーミッションの詳細については、「grant」コマンドを参照してください。
- オブジェクトのパーミッションは、現在のデータベース上でだけ取り消すことができます。
- 自身が **granted by grantor** オプションを使用しないで付与したパーミッションと権限のみを取り消すことができます。 **granted by grantor** オプションを指定すると、他のユーザによって付与されたパーミッションまたは権限を取り消すことができます。
- **grant** コマンドおよび **revoke** コマンドは、順序に影響されます。競合があると、最後に実行されるコマンドが有効になります。述部付き権限の付与と取り消しは例外です。『セキュリティ管理ガイド』の「Adaptive Server における述部付き権限の sysprotects への保存の仕組み」を参照してください。
- **revoke** 構文の **from** の代わりに **to** を使用できます。
- **revoke** 文内の **grant option for** を指定しなければ、**with grant option** パーミッションが、指定されたオブジェクトのアクセス・パーミッションと同時に、ユーザから取り消されます。また、そのユーザが特定のパーミッションを他のユーザに与えている場合は、これらのパーミッションはすべて取り消されます。つまり、**revoke** がカスケードします。
- 複数の付与者がユーザ、グループ、役割に同じ権限やパーミッションを付与できます。この状況では、**sysprotects** で、同一の権限やパーミッションの複数の付与に対応する複数のローが存在することになります。その後、1つ以上の付与が取り消されても、取り消されていない付与が1つ以上残っていれば、ユーザ、グループ、役割が所有する権限またはパーミッションは保持されます。
- **grant** 文は、パーミッションを受け取るユーザ、グループ、または役割ごとに、**sysprotects** システム・テーブルにローを1つ追加します。その後ユーザまたはグループからパーミッションを取り消す (**revoke**) と、**sysprotects** システム・テーブルからそのローが削除されます。パーミッションを与えられたグループ全体からではなく、特定のグループ・メンバからだけパーミッションを取り消す場合は、Adaptive Server はもとのローを保持し、その取り消しに対して新しいローを追加します。

- `create trigger` を発行するパーミッションは、デフォルトでユーザに付与されます。トリガを作成するユーザのパーミッションを取り消すと、そのユーザの `sysprotects` テーブルに取り消しを示すローが追加されます。`create trigger` を発行するパーミッションを付与するには、2つの `grant` コマンドを発行します。最初のコマンドで `sysprotects` から取り消しローを削除し、2番目のコマンドで付与ローを挿入します。トリガを作成するパーミッションが取り消されると、そのユーザは自分のテーブルでもトリガを作成できなくなります。ユーザに対するトリガ作成パーミッションが取り消されるのは、取り消しコマンドを発行したデータベース内だけです。

#### `cascade` オプションの使用

`revoke grant option for` は、特定のパーミッションを他のユーザに付与する権利をユーザから取り消しますが、そのユーザ自身のパーミッションは取り消しません。ユーザがそのパーミッションを他のユーザに与えた場合は、`cascade` オプションを使用してください。使用しなければ、エラー・メッセージが表示され、`revoke` が失敗します。

たとえば、`titles` のユーザ `Bob` の `with grant option` パーミッションを取り消す場合は、次の文を使用します。

```
revoke grant option for select
on titles
from bob
cascade
```

- `Bob` が他のユーザにこのパーミッションを付与していない場合は、このコマンドによって `Bob` が他のユーザにこのパーミッションを与える権利が取り消されます。しかし、`Bob` は依然として `titles` テーブルの `select` パーミッションを所有します。
- `Bob` がこのパーミッションを他のユーザに付与している場合は、`cascade` オプションを使用してください。使用しなければ、エラー・メッセージを受信して `revoke` が失敗します。`cascade` は、この `select` パーミッションを、`Bob` が付与したすべてのユーザから取り消します。同時に、このパーミッションを他のユーザに付与するパーミッションも取り消します。

`revoke` を `cascade` オプションとともに使用しても、テーブル所有者から与えられた権限を取り消すことはできません。たとえば、次の場合などで、テーブル所有者 (`UserA`) は別のユーザ (`UserB`) に権限を与えることができます。

```
create table T1 (... )
grant select on T1 to UserB
```

ただし、システム管理者は、次のように `cascade` オプションとともに権限の `revoke` コマンドを使用しても、`UserB` の権限を取り消すことはできません。

```
revoke select on T1 from UserA cascade
```

この文によってテーブル所有者の `select` 権限は取り消されますが、`UserB` のそれらの権限は取り消されません。

デフォルトでは、テーブル所有者以外のユーザに対して暗黙的にすべてのデータ操作言語 (DML) オペレーションが取り消されます (`restricted decrypt permission` が有効化されている場合の `decrypt` パーミッションは除きます。『暗号化カラム・ユーザズ・ガイド』を参照してください)。これは、`sysprotects` テーブルにはテーブル所有者に権限が与えられてから取り消されたことを示す記録がなく、`cascade` オプションが適用されないからです。

`UserB` から明示的に `select` 権限を取り消す必要があります。

#### granted by の使用

- 述部付き権限の取り消しで `granted by` を使用することはできません。
- `grantor` が `grant` コマンドを実行するパーミッションを所有している必要はありません。
- コマンドの実行者ではなく、付与者によって付与されたパーミッション (`sysprotects.grantor` によって示される) が取り消されます。
- `granted by` パラメータの使用時に `enable granular permissions` を有効化する必要はありません。
- `with grant` オプションによって特定のオブジェクトに対する `grant` パーミッションを受け取るユーザは、`granted by` パラメータを発行できません。その他のユーザはすべて `granted by` パラメータを発行できます。

たとえば、`John` が `with grant` オプションの指定によって `Mary` のテーブルに対するパーミッションを取得したとします。この状況で `granted by` オプションを使用して `revoke` 文を発行しようとする とエラーが返されます。

Mary:

```
grant select on mary.books to john
with grant option
```

Mary:

```
grant select on mary.books to john
```

John:

```
revoke select on mary.books from smith
granted by mary
```

### *set proxy* と *set session authorization* の取り消し

- *set proxy* または *set session authorization* は、master データベース内から取り消す必要があります。
- *set proxy* または *set session authorization* SQL 標準のコマンドと構文しか使わない場合は、*set session authorization* を使用します。
- *revoke all* には、*set proxy* パーミッションや *set session authorization* パーミッションは含まれません。

### データベース・ユーザ・グループ

- データベース・ユーザ・グループを使用すると、一度で複数のパーミッションを付与 (*grant*) または取り消す (*revoke*) ことができます。ユーザは常に、デフォルト・グループ "public" のメンバであり、それ以外にメンバとして所属できるグループは1つだけです。Adaptive Server のインストール・スクリプトでは、"public" に一連のパーミッションが割り当てられます。

グループは、*sp\_addgroup* で作成し、*sp\_dropgroup* で削除します。*sp\_adduser* でグループに新しいユーザを追加し、*sp\_changegroup* でユーザのグループ・メンバシップを変更します。グループのメンバを表示するには、*sp\_helpgroup* を使用します。

### システム・テーブルのデフォルト・パーミッションの取り消し

"public" の、システム・テーブルのデフォルト・パーミッションの取り消し

以下は、任意のデータベースからコマンドを発行してデフォルト・パーミッションを取り消しできるシステム・テーブルです。

- |                  |                 |                   |
|------------------|-----------------|-------------------|
| • sysalternates  | • syslogs       | • syssegments     |
| • sysattributes  | • sysobjects    | • sysstatistics   |
| • syscolumns     | • syspartitions | • systabstats     |
| • syscomments    | • sysprocedures | • systhresholds   |
| • sysconstraints | • sysprotects   | • systypes        |
| • sysdepends     | • sysqueryplans | • sysusermessages |
| • sysindexes     | • sysreferences | • sysusers        |
| • sysjars        | • sysroles      | • sysxtypes       |
| • syskeys        |                 |                   |

以下は、コマンドを master データベースから発行してデフォルト・パーミッションを取り消しできるシステム・テーブルです。

- sysdatabases
- sysdevices
- syslocks
- sysmessages
- sysprocesses
- systransactions
- sysusages
- sysconfigures
- syscurconfigs
- syslanguages
- syscharsets
- syssservers
- systimeranges
- sysresourcelimits
- syslogins
- sysremotelogins

*update statistics*、*delete statistics*、*truncate table* のパーミッションの取り消し

Adaptive Server では、*update statistics*、*delete statistics*、*truncate table* の各コマンドに対する、ユーザ、役割、グループのパーミッションを取り消せます。テーブル所有者も、暗黙の *grant* によってパーミッションを付与できます。具体的には、*update statistics*、*delete statistics*、*truncate table* をストアド・プロシージャに追加してから、そのストアド・プロシージャの実行パーミッションをユーザまたは役割に付与します。

*update statistics* のパーミッションをカラム・レベルで取り消すことはできません。*sysroles*、*sysssrroles*、*sysloginroles* の各セキュリティ・テーブルに対して *update statistics* または *delete statistics* を実行するには、*sso\_role* が必要です。

デフォルトでは、データベース所有者は、*sysroles*、*sysssrroles*、*sysloginroles* 以外のシステム・テーブルに対して *update statistics* と *delete statistics* を実行するパーミッションがあり、この権限を他のユーザに譲渡することもできます。

*grant all* を発行して、*update statistics*、*delete statistics*、*truncate table* のパーミッションを付与することもできます。

---

**注意** *update statistics* を実行するパーミッションをユーザから取り消すと、そのユーザはコマンドのバリエーション (*update all statistics*、*update partition statistics*、*update index statistics*、*update statistics table* など) を実行するパーミッションも失います。たとえば、次の例は、*authors* テーブルに対して *update statistics* のすべてのバリエーションを実行するパーミッションを *Billy* から取り消します。

```
revoke update statistics on authors to billy
```

*update statistics* を実行するパーミッションをユーザから取り消すと、そのコマンドのバリエーションを実行するパーミッションも取り消すことになります。

---

`update statistics` のバリエーション (`update index statistics` など) のパーミッションを個別に取り消すことはできません。つまり、次のようなコマンドは発行できません。

```
revoke update all statistics from harry
```

`delete statistics` のパーミッションをカラム・レベルで付与または取り消すことはできません。「[grant](#)」(486 ページ) の「使用法」の項を参照してください。

#### クラスタ環境での revoke

ローカル・テンポラリ・データベースでユーザ定義の役割からパーミッションを取り消そうとすると、`revoke` は失敗します。

#### 標準規格

ANSI SQL – 準拠レベル : Transact-SQL 拡張機能

#### パーミッション

`revoke` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

一般に細密なパーミッションが有効の場合は、取り消し対象の権限またはパーミッションに応じて、次のいずれかの権限管理権限を持つユーザが `revoke` を実行できる。

サーバワイドの権限については、`manage serverpermissions` 権限または `manage security permissions` 権限を持つユーザであることが必要。

データベースワイドの権限については、`manage database permissions` 権限を持つユーザであることが必要。

オブジェクト権限については、オブジェクト所有者であるか、`manage any object permission` 権限を持つユーザであることが必要。

`revoke default` を実行するには、データベース所有者であるか、データベースに対する `own database` 権限を持つユーザであることが必要。

詳細については、[表 1-23](#) の「管理 (細密なパーミッションの有効時)」カラムを参照してください。

<p>細密なパーミッションが無効</p>	<p>細密なパーミッションが有効化されている場合は、次のようになる。</p> <p>コマンドの実行 - <code>create database</code> パーミッションを取り消せるのはシステム管理者だけである。また、<code>master</code> データベースからのみ取り消すことができる。<code>create encryption key</code> パーミッションと <code>create trigger</code> パーミッションを取り消すことができるのは、システム・セキュリティ担当者だけである。</p> <p>データベース一貫性の検査 - システム管理者のみが <code>revoke dbcc</code> コマンドを実行できる。データベース所有者は <code>revoke dbcc</code> コマンドを実行できません。</p> <p>データベース・オブジェクト・アクセス - データベース・オブジェクトのデフォルト・パーミッションをオブジェクト所有者にする。オブジェクト所有者は、自分が所有するデータベース・オブジェクトのパーミッションを、他のユーザから取り消すことができます。</p> <p>関数 - システム管理者のみが、組み込み関数のパーミッションを取り消すことができる。</p> <p>プロキシおよびセッション権限 - システム・セキュリティ担当者のみが <code>master</code> データベースからのみ <code>set proxy</code> または <code>set session authorization</code> を取り消すことができる。</p> <p>役割 - <code>master</code> データベースからのみ取り消すことができる。<code>sso_role</code>、またはユーザ定義役割をユーザや役割から取り消すことができるのは、システム・セキュリティ担当者のみである。ユーザや役割から <code>oper_role</code> <code>sa_role</code> を取り消すことができるのは、システム管理者のみである。<code>sa_role</code> が含まれる役割を取り消すことができるのは、<code>sa_role</code> と <code>sso_role</code> の両方を持っているユーザだけである。</p> <p>テーブル - データベース所有者は、システム・テーブルのデフォルト・パーミッションを取り消すことができる。テーブル所有者とシステム・セキュリティ担当者は、テーブルまたはテーブル内のカラム・リストの <code>decrypt</code> パーミッションを取り消すことができる。</p> <p>デフォルト - データベース所有者と <code>sa_role</code> が付与されたログインは、デフォルトを取り消すことができる。</p>
----------------------	--

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
47	revoke	revoke	<ul style="list-style-type: none"> <li>• <i>Roles</i> - 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> - <code>revoke</code> 文の完全なコマンド・テキスト。</li> <li>• <i>Previous value</i> - NULL</li> <li>• <i>Current value</i> - NULL</li> <li>• <i>Other information</i> - NULL</li> <li>• <i>Proxy information</i> - <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照

コマンド [grant](#), [setuser](#), [set](#)

関数 [proc\\_role](#)

システム・プロシージャ [sp\\_activeroles](#), [sp\\_adduser](#), [sp\\_changedbowner](#),  
[sp\\_changegroup](#), [sp\\_displaylogin](#), [sp\\_displayroles](#), [sp\\_dropgroup](#), [sp\\_dropuser](#),  
[sp\\_helpgroup](#), [sp\\_helprotect](#), [sp\\_helpuser](#), [sp\\_modifylogin](#), [sp\\_role](#)



## revoke role

**説明** グループ、ログイン、ログイン・プロファイルまたは役割から役割を取り消します。

**構文**

```
revoke role {role_name [, role_list ...]} from
           {grantee [, grantee ...]}
```

**パラメータ**

**role**

システム定義またはユーザ定義の役割の名前です。revoke role を使用すると、付与された役割を、役割、ログイン、ログイン・プロファイルから取り消すことができます。

**role\_name**

システム定義またはユーザ定義の役割の名前です。付与対象者から役割を取り消すと、その付与対象者が役割のメンバシップによって所有していたすべてのパーミッションも取り消されます。

**grantee**

役割を取り消すシステム役割名、ユーザ定義役割名、ログイン・プロファイルまたはユーザのログイン名です。

**role\_list**

パーミッションを取り消すシステム役割またはユーザ定義の役割のリストです。変数は使用できません。

**例**

**例 1** "specialist\_role" から "doctor\_role" を取り消します。

```
revoke role doctor_role from specialist_role
```

**例 2** "specialist\_role" と "intern\_role"、およびユーザ Mary と Tom から、"doctor\_role" と "surgeon\_role" を取り消します。

```
revoke role doctor_role, surgeon_role from
specialist_role, intern_role, mary, tom
```

**例 3** manage roles 権限を持つユーザ Smith が、doctor\_role から nurse\_role を取り消します。この役割は roleAdmin によって付与されていました。

```
revoke role nurse_role from doctor_role
granted by roleAdmin
```

---

**注意** manage roles 権限は、細密なパーミッションが有効の場合のみ使用できます。

---

**例 4** システム・オペレータに予定されるログイン・プロファイルからシステム役割 oper\_role を取り消します。

```
revoke role oper_role from lp_operator
```

- 使用法**
- ユーザがログインしている間にそのユーザから役割を取り消すことができます。Adaptive Server は、アクセス検査を実行する前にユーザのアクティブな役割を確認します。
  - ログイン・プロファイルから役割を取り消す場合、Adaptive Server は、現在 Adaptive Server にログイン中のユーザも含めて、そのプロファイルに割り当てられたすべてのユーザから役割を取り消します。

**標準規格** ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

**パーミッション** revoke role のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <b>manage roles</b> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが有効化されている場合は、次のようになる。 役割 - master データベースからのみ取り消すことができる。sso_role、oper_role、またはユーザ定義役割をユーザや役割から取り消すことができるのは、システム・セキュリティ担当者だけです。ユーザや役割から sa_role を取り消すことができるのは、システム管理者だけです。sa_role が含まれる役割を取り消すことができるのは、sa_role と sso_role の両方を持っているユーザだけです。

**監査** sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
85	role	create role、drop role、alter role、grant role、または revoke role	<ul style="list-style-type: none"> <li>• Roles — 現在のアクティブな役割</li> <li>• Keywords または options — revoke role 文の完全なコマンド・テキスト。</li> <li>• Previous value — NULL</li> <li>• Current value — NULL</li> <li>• Other information — NULL</li> <li>• Proxy information — set proxy が有効な場合は元のログイン名</li> </ul>

**参照** コマンド [grant](#)、[setuser](#)、[set](#)

関数 [proc\\_role](#)

システム・プロシージャ [sp\\_activeroles](#)、[sp\\_adduser](#)、[sp\\_changedbowner](#)、[sp\\_changegroup](#)、[sp\\_displaylogin](#)、[sp\\_displayroles](#)、[sp\\_dropgroup](#)、[sp\\_dropuser](#)、[sp\\_helpgroup](#)、[sp\\_helprotect](#)、[sp\\_helpuser](#)、[sp\\_modifylogin](#)、[sp\\_role](#)

## rollback

説明	ユーザ定義トランザクションを、トランザクション内の指定したセーブポイントまで、またはトランザクションの開始点までロールバックします。
構文	rollback [tran   transaction   work] [ <i>transaction_name</i>   <i>savepoint_name</i> ]
パラメータ	tran   transaction   work トランザクションまたは作業をロールバックすることを指定します。tran、transaction、または work を指定する場合は、 <i>transaction_name</i> または <i>savepoint_name</i> も指定できます。  <i>transaction_name</i> 一番外側のトランザクションに割り当てられる名前です。トランザクション名は識別子の規則に従います。  <i>savepoint_name</i> save transaction 文のセーブポイントに割り当てられる名前です。名前は、識別子の規則に従っている必要があります。
例	トランザクションをロールバックします。  begin transaction delete from publishers where pub_id = "9906" rollback transaction
使用法	<ul style="list-style-type: none"> <li>• <i>transaction_name</i> または <i>savepoint_name</i> がない rollback transaction は、ユーザ定義のトランザクションを一番外側のトランザクションの先頭までロールバックします。</li> <li>• rollback transaction <i>transaction_name</i> は、指定されたトランザクションの先頭までユーザ定義トランザクションをロールバックします。トランザクションをネストできますが、ロールバックできるのは一番外側のトランザクションだけです。</li> <li>• rollback transaction <i>savepoint_name</i> は、一致する save transaction <i>savepoint_name</i> までユーザ定義トランザクションをロールバックします。</li> </ul> <p><b>制限事項</b></p> <ul style="list-style-type: none"> <li>• 現在アクティブなトランザクションがない場合は、commit または rollback 文は効果がありません。</li> <li>• rollback コマンドはトランザクション内で指定してください。commit の入力後は、トランザクションをロールバックできません。</li> </ul>

### トランザクション全体のロールバック

- セーブポイント名を指定しないで `rollback` を実行すると、トランザクション全体が取り消されます。トランザクションのすべての文またはプロシージャが取り消されます。
- `rollback` コマンドに `savepoint_name` または `transaction_name` が指定されていない場合、トランザクションはバッチ内で最初の `begin transaction` までロールバックします。これには、連鎖トランザクション・モードを使用した暗黙の `begin transaction` によって開始されたトランザクションも含まれます。

### セーブポイントへのロールバック

トランザクションの一部を取り消すには、`savepoint_name` を指定して `rollback` を使用します。セーブポイントとは、ユーザが `save transaction` コマンドを使用してトランザクション内に設定したマーカです。セーブポイントと `rollback` の間のすべての文とプロシージャは取り消されます。

トランザクションがセーブポイントまでロールバックしたら、次に `commit` を使用してトランザクションの完了まで続行する (rollback 以降の SQL 文の実行する) か、セーブポイントを指定しない `rollback` を使用してトランザクション全体を取り消すことができます。トランザクション内のセーブポイントの数に制限はありません。

### トリガとストアド・プロシージャでのロールバック

- トリガまたはストアド・プロシージャでは、トランザクションやセーブポイント名が指定されていない `rollback` 文を実行すると、プロシージャを呼び出したバッチ、またはトリガを起動したバッチにある、最初の明示的または暗黙の `begin transaction` までのすべての文がロールバックされます。
- セーブポイント名が指定されていない `rollback` コマンドがトリガに含まれている場合、ロールバックによってバッチ全体がアボートされます。バッチ内のロールバック以降の文は実行されません。
- リモート・プロシージャ・コール (RPC) は、このコールが含まれているトランザクションから独立して実行されます。通常のトランザクション (Open Client™ DB-Library の 2 フェーズ・コミットを使用しないトランザクション) の場合、`rollback` コマンドではリモート・サーバが RPC を使用して実行するコマンドがロールバックされず、`commit` コマンドの実行にも依存しません。
- トランザクションを管理する文の使用法と、ストアド・プロシージャ、トリガ、バッチに対する `rollback` の効果の詳細については、『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：初級レベル Transact-SQL 拡張機能 <code>rollback transaction</code> および <code>rollback tran</code> の文の形式とトランザクション名の使用を含みます。
パーミッション	<code>rollback</code> の使用にパーミッションは必要ありません。
参照	コマンド <a href="#">begin transaction</a> , <a href="#">commit</a> , <a href="#">create trigger</a> , <a href="#">save transaction</a>

## rollback trigger

説明	トリガを起動したデータ変更処理を含めて、トリガによって実行された処理をロールバックします。また、オプションで <code>raiserror</code> 文を実行します。
構文	<code>rollback trigger</code> <code>[with raiserror_statement]</code>
パラメータ	<code>with raiserror_statement</code> <code>raiserror</code> 文を指定します。この文は、ユーザ定義のエラー・メッセージを出力し、エラー条件が発生したことを記録するシステム・フラグを設定します。この文は、エラー状態のトランザクションのステータスがロールバックを反映するように、 <code>rollback trigger</code> 文の実行時にクライアントにエラーを表示します。 <code>raiserror_statement</code> を定義する構文と規則の詳細については、 <code>raiserror</code> コマンドを参照してください。
例	トリガをロールバックして、ユーザ定義のエラー・メッセージ 25002 を表示します。 <pre>rollback trigger with raiserror 25002     "title_id does not exist in titles table."</pre>
使用法	<ul style="list-style-type: none"><li>• <code>rollback trigger</code> が実行されると、Adaptive Server は現在実行中のコマンドをアボートし、トリガの残りの部分の実行を停止します。</li><li>• <code>rollback trigger</code> を発行するトリガが他のトリガ内にネストされている場合、Adaptive Server は最初のトリガを起動した更新処理と、これらのトリガによってそれまでに実行されたすべての処理をロールバックします。</li><li>• Adaptive Server では、トリガの外側で実行された <code>rollback trigger</code> 文は無視され、この文に関連付けられている <code>raiserror</code> は実行されません。ただし、トリガの外側だがトランザクションの内側の位置で <code>rollback trigger</code> 文が実行されると、トランザクションがロールバックされ、現在の文バッチがアボートされます。</li></ul>
標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>rollback trigger</code> の使用にパーミッションは必要ありません。
参照	コマンド <a href="#">create trigger</a> , <a href="#">raiserror</a> , <a href="#">rollback</a>

## save transaction

説明	トランザクション内にセーブポイントを設定します。
構文	<code>save transaction <i>savepoint_name</i></code>
パラメータ	<i>savepoint_name</i> セーブポイントに割り当てられる名前です。トランザクション名は識別子の規則に従います。
例	2 人の著者の <code>royaltypers</code> エントリを更新した後で、セーブポイント <code>percentchanged</code> を挿入し、本の価格が 10% 増加すると、著者の印税収入にどの程度の影響が出るかを確認します。トランザクションは、 <code>rollback transaction</code> によってセーブポイントまでロールバックされます。

```
begin transaction royalty_change

update titleauthor
set royaltypers = 65
from titleauthor, titles
where royaltypers = 75
and titleauthor.title_id = titles.title_id
and title = "The Gourmet Microwave"

update titleauthor
set royaltypers = 35
from titleauthor, titles
where royaltypers = 25
and titleauthor.title_id = titles.title_id
and title = "The Gourmet Microwave"

save transaction percentchanged

update titles
set price = price * 1.1
where title = "The Gourmet Microwave"

select (price * total_sales) * royaltypers
from titles, titleauthor
where title = "The Gourmet Microwave"
and titles.title_id = titleauthor.title_id

rollback transaction percentchanged

commit transaction
```

### 使用法

- セーブポイントとは、トランザクションを部分的にロールバックできるように、トランザクション内にユーザが定義するマークです。 `rollback savepoint_name` は、指定されたセーブポイントまでをロールバックします。セーブポイントと `rollback` の間のすべての文とプロシージャが取り消されます。

セーブポイントよりも前にある文は取り消されませんが、コミットも実行されません。トランザクションはセーブポイントまでロールバックしてから、文の実行を続けます。セーブポイントが指定されていない `rollback` は、トランザクション全体を取り消します。 `commit` によって、トランザクションは処理を完了させます。

- トランザクションをネストしていると、 `save transaction` は、一番外側のトランザクションにだけセーブポイントを作成します。
- トランザクション内のセーブポイントの数に制限はありません。
- `rollback` コマンドに `savepoint_name`、`transaction_name` のいずれも指定しない場合、バッチの最初の `begin transaction` までのすべての文がロールバックされ、トランザクション全体が取り消されます。

### 標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

### パーミッション

`save transaction` の使用にパーミッションは必要ありません。

### 参照

コマンド [begin transaction](#)、[commit](#)、[rollback](#)

マニュアル トランザクション文の使用方法については、『Transact-SQL ユーザーズ・ガイド』を参照してください。



## select

説明

データベース・オブジェクトからローを取得します。

構文

```
select ::=
    select [all | distinct]
    [top unsigned_integer]
    select_list
    [into_clause]
    [from_clause]
    [where_clause]
    [group_by_clause]
    [having_clause]
    [order_by_clause]
    [compute_clause]
    [read_only_clause]
    [isolation_clause]
    [browse_clause]
    [plan_clause]
    [for_xml_clause]
```

```
select_list ::=
```

---

**注意** *select\_list* の詳細については、「パラメータ」の項を参照してください。

---

```
into_clause ::=
```

```
into [[database.] owner.] table_name
    [(colname encrypt [with [database.]owner.]keyname] [,
        colname encrypt_clause ...])]
    | [compressed = compression_level | not compressed]
    [in row [(length)] | off row ]
    [{{external table at
        'server_name.[database.]owner.object_name'
        | external directory at 'pathname'
        | external file at 'pathname' [column delimiter 'string'
    ]}}]]

    [on segment_name]
    dml_logging = (full | minimal)
    [partition_clause]
    [lock {datarows | datapages | allpages}]
    [with [, into_option [, into_option] ...]]]

| into existing table table_name

partition_clause ::=
    partition by range (column_name [, column_name]...)
        ([partition_name] values <= ({constant | MAX}
            [, {constant | MAX}] ...) [on segment_name])
```

```

        [compression_clause] [on segment_name]
    [, [partition_name] values <= ({constant | MAX}
    [, {constant | MAX}] ...) [on segment_name]]...)
    [compression_clause] [on segment_name]

| partition by hash (column_name[, column_name]...)
  { (partition_name [on segment_name]
    [compression_clause] [on segment_name]
    [, partition_name [on segment_name]]...)
    [compression_clause] [on segment_name]
  | number_of_partitions
    [on (segment_name[, segment_name] ...)]}

| partition by list (column_name)
  ([partition_name] values (constant[, constant] ...)
  [compression_clause] [on segment_name]
  [, [partition_name] values (constant[, constant] ...)
  [compression_clause] [on segment_name]

| partition by roundrobin
  { (partition_name [on segment_name]
    [, partition_name [on segment_name]]...)
    [compression_clause] [on segment_name]
  | number_of_partitions
    [on (segment_name [, segment_name]...)]}

into_option ::=
  | max_rows_per_page = num_rows
  | exp_row_size = num_bytes
  | reservepagegap = num_pages
  | identity_gap = gap
  | compression = {none | page | row}
  | lob_compression = off | compression_level

from_clause ::=
  from table_reference [, table_reference]...

table_reference ::=
  table_view_name | ANSL_join

table_view_name ::=
  [[database.]owner.]{table_name | view_name}
  [as] [correlation_name]
  [(index {index_name | table_name})]
  [parallel [degree_of_parallelism]]
  [prefetch size][lru | mru]
  [holdlock | noholdlock]
  [readpast]
  [shared]

ANSL_join ::=
  table_reference join_type join table_reference

```

```

        join_conditions
        join_type ::= inner | left [outer] | right [outer]
        join_conditions ::= on search_conditions

compression_clause ::=
    with compression = {none | page | row}

where_clause ::=
    where search_conditions
    for update [of column_list]

group_by_clause ::=
    group by [all] aggregate_free_expression
        [, aggregate_free_expression]...

having_clause ::=
    having search_conditions

order_by_clause ::=
    order by sort_clause [, sort_clause]...

    sort_clause ::=
        {[[[database.]owner.]{table_name.|view_name.}]column_name
        | select_list_number
        | expression }
        [asc | desc]

compute_clause ::=
    compute row_aggregate (column_name)
        [, row_aggregate (column_name)]...
    [by column_name [, column_name]...]

read_only_clause ::=
    for {read only | update [of column_name_list]}

isolation_clause ::=
    at isolation
        {read uncommitted | 0}
        | {read committed | 1}
        | {repeatable read | 2}
        | {serializable | 3}

browse_clause ::=
    for browse

plan_clause ::=
    plan "abstract plan"

```

---

**注意** select...for\_xml\_clause の構文、サンプルおよび使用方法については、『XML サービス』を参照してください。

---

パラメータ

all

すべてのローが結果に含まれます。all がデフォルトです。

**distinct**

ユニークなローだけが結果に含まれます。**distinct** は **select** リストの最初のワードとして指定してください。ブラウザ・モードでは、**distinct** は無視されます。

**null** 値は、**distinct** キーワードについてはすべて等しいものとしてみなされます。検出される **null** の数に関係なく、1つの **null** 値だけが選択されます。

**top unsigned\_integer**

**select...into** 文と組み合わせて使用することにより、ターゲット・テーブルに挿入されるローの数を制限します。これは、**select...into** で無視される **set rowcount** とは異なります。

- **delete**、**update**、またはビューで使用するとき、順序を指定することはできません。テーブルにクラスタード・インデックスに由来する暗黙の順序がある場合はその順序が適用され、そうでない場合は順序が決まらず予測不能になります。
- $n$  は、 $0 \sim 2^{32}-1$  (4GB-1 すなわち 4,294,967,295) の間の符号なし 32 ビット値です。ゼロは、ローがないことを意味します。
- カーソルとともに使用すると、**top n** は結果セットの全体のサイズを制限します。**set cursor rowcount** を指定すると、単一のフェッチの結果が制限されます。
- ビューの定義の中に **select top n** があり、クエリの **where** 句でそれを使用しているときは、矛盾する結果が生じることがあります。

**select\_list**

1つ以上の次の項目から構成されます。

- **create table** で指定した順序ですべてのカラムを表す "\*"。
- 参照する順序で指定したカラム名のリスト。既存の **IDENTITY** カラムを選択する場合は、実際のカラム名の代わりに、必要に応じて、テーブル名で修飾した **syb\_identity** キーワードを使用できます。
- 結果テーブルへの新しい **IDENTITY** カラムの追加の指定。

**column\_name** = **identity** (**int** | **smallint** | **tinyint** | **precision**)

**int**、**smallint**、または **tinyint** を指定すると、結果のカラムは整数になります。**precision** を指定すると、結果は **numeric** データ型になります。

- 次のいずれかの形式によるデフォルトのカラム見出し (カラム名) の置き換え。

**column\_heading** = **column\_name**

```
column_name column_heading
column_name as column_heading
```

上記のどの形式でも、カラム見出しが疑問符で囲まれていることがあります。カラム見出しが有効な識別子ではない場合(カラム見出しが予約語である場合、特殊文字で始まる場合、またはスペースや句読表記記号が入っている場合)は、その見出しを引用符で囲んでください。

- 式(カラム名、定数、関数、それらを算術演算子かビット処理演算子で接続して任意に組み合わせたもの、またはサブクエリ)。
- 組み込み関数または集合関数。
- 以上の項目の組み合わせ。

`select_list` でも、次の構文を使用して変数に値を割り当てることができます。

```
@variable = expression
[, @variable = expression ...]
```

変数の割り当てを、他の `select_list` オプションと同時に指定することはできません。

#### into

`existing table` を使用する場合を除き、`select` リストで指定されたカラムと `where` 句で選択されたローに基づいて、新しいテーブルが作成されます。「[select into の使用](#)」(700 ページ)を参照。

#### colname encrypt

ターゲット・テーブルの `colname` に暗号化を指定します。デフォルトでは、Adaptive Server はソース・テーブルから選択されたデータを復号化します。データの暗号化を維持する場合や、ソース・データベースで暗号化されていないカラムをターゲット・データベースで暗号化する場合は、`encrypt` キーワードを使用してください。

#### compression = compression\_level | not compressed

ロー内のラージ・オブジェクト (LOB) データが圧縮されるかどうか、およびその場合の圧縮レベルを示します。

**compression\_level | not compressed**

ローの圧縮レベルを示します。

- 0 – ローは圧縮されません。
- 1 ~ 9 – Adaptive Server は ZLib 圧縮を使用します。通常、圧縮の数値が大きいくほど、Adaptive Server での LOB データの圧縮レベルが高くなり、圧縮データと非圧縮データの比率も大きくなります (つまり、非圧縮データと比べて、圧縮データによる記憶領域の節約量 (バイト) が多くなります)。

ただし、圧縮の量は LOB の内容によって異なります。圧縮レベルが高いほど、圧縮に使用される CPU の量が多くなります。そのためレベル 9 を使用すると、圧縮率が最高になりますが、CPU の使用量も最高になります。

- 100 – Adaptive Server は FastLZ 圧縮を使用します。CPU 使用量が最も少ない圧縮率です。通常、短いデータに使用します。
- 101 – Adaptive Server は FastLZ 圧縮を使用します。値に 101 を指定すると、100 を指定した場合よりも、CPU の使用量は若干多くなりますが、圧縮率は良くなります。

圧縮アルゴリズムでは、LOB データを使用しないローは無視されます。

**column\_list**

複数のカラムをカンマで区切ったリストです。

**with database...key**

ソース・データで使用されたキーまたは異なるキーを指定します。

**in row [(length)]**

ターゲット・テーブルで、LOB カラムのロー内特性を設定または変更します。length を指定しない場合、Adaptive Server は設定されているデフォルトのロー内の長さを使用します。

デフォルトでは、ターゲット・テーブルの LOB カラムは select リストの対応する LOB カラムの保存プロパティを継承します。ターゲット・テーブルの LOB カラムが convert(text, column) built-in 関数などの式から生成された場合、カラムは自動的にロー外の記憶領域を使用します。ただし、in row [(length)] を指定して設定を変更した場合はこの限りではありません。

**off row**

カラムの記憶フォーマットをロー内からロー外に変更します。

`external [[table] | directory | file]`

外部オブジェクトのタイプがテーブル、ディレクトリ、またはファイルであることを示します。ファイル、ディレクトリ、またはテーブルを指定しないと、`select into` ではテーブルを使用するものと想定されます。

---

**注意** `partition_clause` のいずれかの部分を使用する場合は、外部ロケーションを指定できません。パーティションは、現在のサーバとデータベースにあるテーブルにのみ作成できます。

---

`'server_name.[database].[owner].object_name'`

選択したカラムをリモート `server_name` 上のテーブルまたはビューに挿入することを示します。

`dml_logging`

`insert`、`update`、`delete` オペレーションといくつかの形式のバルク挿入に対するロギングの量を決定します。次のいずれかになります。

- `full` — すべてのトランザクションのログが取られます。
- `minimal` — ローまたはページの変更のログは取られません。

`at 'path_name'`

選択したカラムを挿入する外部ファイルまたはディレクトリのオペレーティング・システム固有のフルパス名を示します。`path_name` に含まれるすべてのディレクトリは、`Adaptive Server` からアクセス可能である必要があります。

`column delimiter 'string'`

カラムのデータを文字列フォーマットに変換した後で、カラムを区切るデリミタを示します。`string` には、16 文字まで指定できます。デリミタを指定しないと、`select into` ではタブ文字が使用されます。

`existing table table_name`

選択したデータがプロキシ・テーブルに挿入されることを示します。プロキシ以外のタイプのテーブルでは `select into` を実行できません。`select` リストのカラム・リストが、プロキシ・テーブルのタイプ、長さ、数と一致している必要があります。

**on segment\_name**

指定したセグメントにテーブルを作成するように指定します。on **segment\_name** オプションを使用するには、事前に **disk init** を使用してデバイスを初期化する必要があります。また、**sp\_addsegment** を使用して、セグメントをデータベースに追加しておくことも必要です。データベースで使用できるセグメント名のリストについては、システム管理者に問い合わせるか、**sp\_helpsegment** を使用して参照してください。

**partition by range**

分割するレコードを、分割カラムの値に基づいて指定します。分割カラムの値がユーザ提供の上限および下限と比較されて、パーティションの割り当てが決定されます。

**column\_name**

**partition\_clause** で使用する場合にパーティション・キー・カラムを指定します。

**partition\_name**

テーブル・レコードを保管する新しいパーティションの名前を指定します。テーブル内またはインデックス内にすでに存在するパーティションと同じ名前は指定できません。**set quoted\_identifier** オプションを設定した場合、パーティション名に区切り識別子を使用できます。それ以外の場合、パーティション名は有効な識別子でなければなりません。

**partition\_name** を省略すると、**table\_name\_partition\_id** という形式の名前が作成されます。最大長を超える長さのパーティション名は、トランケートされます。

**values <= constant | MAX**

指定したパーティションの上限値を指定します。パーティションの上限に定数値 (**constant**) を指定すると、テーブルに暗黙的な一意性制約が適用されます。キーワード **MAX** を使用すると、特定のデータ型の最大値が指定されます。

**on segment\_name**

**partition\_clause** で使用するときには、パーティションを配置するセグメントの名前を指定します。on **segment\_name** を使用する場合は、**create database** または **alter database** を使用して論理デバイスが事前にデータベースに割り当てられている必要があります。また、**sp\_addsegment** を使用してセグメントがデータベースに作成されていることも必要です。データベースで使用できるセグメント名のリストについては、システム管理者に問い合わせるか、**sp\_helpsegment** を使用して参照してください。



**partition by hash**

レコードの分割に、システムから提供されるハッシュ関数を使用します。ハッシュ関数は、レコードが割り当てられるパーティションを指定するパーティション・キー値を計算します。

**partition by list**

分割するレコードを、指定したカラム内のリテラル値に基づいて指定します。パーティション・キーのカラムは 1 つだけです。リスト・パーティションには、最大で 250 個の定数をパーティション値としてリストできます。

**partition by roundrobin**

レコードの分割を順次処理で行うことを指定します。ラウンドロビン分割テーブルに分割キーはありません。ユーザおよびオブティマイザは、特定のレコードがどのパーティションに配置されるかを確認することはできません。

**lock datarows | datapages | allpages**

**select into** コマンドによって作成されたテーブルに使用されるロック・スキームを指定します。デフォルトは、サーバ全体に対する設定パラメータ **lock scheme** の設定です。

**max\_rows\_per\_page**

**select into** によって作成されたテーブルのデータ・ページ上のロー数を制限します。**fillfactor** とは異なり、**max\_rows\_per\_page** 値はデータが挿入または削除されるまで保持されます。**max\_rows\_per\_page** は、データオンリーロック・テーブルでは使用できません。

**exp\_row\_size = num\_bytes**

**select into** コマンドによって作成されたテーブルに予期されるロー・サイズを指定します。データローおよびデータページ・ロック・スキームと可変長ローを持つテーブルにのみ有効です。有効な値は 0、1、およびローの最小長より大きくテーブルのローの最大長より小さい任意の値です。デフォルト値は 0 です。これは、サーバワイドなデフォルトが使用されることを示します。

**reservepagegap = num\_pages**

埋められたページと、**select into** がデータ格納のためのエクステントを割り付けるときに残される空ページとの比率を指定します。このオプションは、**select into** コマンドでのみ有効です。指定した各 **num\_pages** につき 1 つの空ページが、今後のテーブル拡張のために残されます。有効な値は 0 ~ 255 です。デフォルト値は 0 です。

**readpast**

クエリが、待機もメッセージの生成も行わずに、排他ロックが設定されたローを暗黙的に省略するように指定します。

**identity\_gap**

テーブルの **identity** ギャップを指定します。この値は、このテーブルのシステムの **identity** ギャップ設定だけを上書きします。

**compression =**

テーブルまたはパーティションに適用する圧縮レベルを示します。新しい圧縮レベルは、新しく挿入または更新されたデータに適用されます。

- **none** — このテーブルまたはパーティションのデータは圧縮されません。パーティションの場合、**none** は、テーブル圧縮が **row** 圧縮または **page** 圧縮に変更されても、このパーティションのデータが圧縮されないままであることを意味します。
- **row** — 個々のロー内の 1 つ以上のデータ項目を圧縮します。Adaptive Server は、非圧縮形式に比べ、圧縮形式でのデータの格納により記憶領域を節約できる場合にのみ、**row** 圧縮形式でデータを格納します。パーティション・レベルまたはテーブル・レベルで **row** 圧縮を設定します。
- **page** — ページがいっぱいになると、ロー圧縮された既存のデータ・ローが、ページ・レベルの圧縮を使用して圧縮され、ページ・レベルの辞書、インデックス、文字コードのエントリが作成されます。パーティション・レベルまたはテーブル・レベルで **page** 圧縮を設定します。

Adaptive Server では、データがロー・レベルで圧縮されてからしかページ・レベルで圧縮されないため、圧縮を **page** に設定することは、**page** と **row** の両方が圧縮されることを意味します。

**lob\_compression = *compression\_level***

テーブルの圧縮レベルを決定します。

**value**

**identity** ギャップの値です。

固有の **identity** ギャップが設定されているテーブルから、**select into** 文でテーブルを作成している場合、新しいテーブルは親テーブルの **identity** ギャップの設定を継承しません。新しいテーブルでは **identity burning set factor** の設定が使用されます。新しいテーブルに固有の **identity\_gap** 設定を指定するには、**select into** 文で **identity** ギャップを指定します。新しいテーブルには、親テーブルと同じ **identity** ギャップも、異なる **identity** ギャップも指定できます。

from

select 文で使用されるテーブルとビューを示します。select リストにカラム名が含まれていない場合 (select リストに定数と算術演算子だけが含まれている場合) を除いて、from を指定する必要があります

```
select 5 x, 2 y, "the product is", 5*2 Result
```

x	y	Result
5	2 the product is	10

クエリでは最大で 50 個のテーブルと 14 個のワーク・テーブル (集合関数によって作成されたテーブルなど) を参照できます。50 個のテーブル制限には次のものが含まれます。

- from 句にリストされるテーブル (またはテーブルのビュー)
- 同じテーブルに対する複数の参照 (セルフジョイン) の各インスタンス
- サブクエリで参照されるテーブル
- into で作成されるテーブル
- from 句にリストされるビューによって参照される基本テーブル

view\_name, table\_name

select 文で使用されるテーブルとビューをリストします。テーブルまたはビューが別のデータベース内にある場合は、データベース名を指定します。データベース内に同じ名前のテーブルまたはビューが複数ある場合は、所有者の名前を指定します。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。

複数のテーブルまたはビューを指定する場合は、カンマを使用してテーブル名およびビュー名を区切ります。キーワード from の後に指定するテーブルとビューの順序は、コマンド実行結果には影響しません。

1 つの文で、異なるデータベースにある複数のテーブルを問い合わせることができます。

セルフジョインまたはサブクエリでのテーブルまたはビューの複数の役割を区別するため、テーブル名とビュー名に対して相関名 (エイリアス) を指定できます。相関名を割り当てるには、次に示すように、テーブル名またはビュー名、スペース、相関名の順序で指定します。

```
select pub_name, title_id
       from publishers pu, titles t
       where t.pub_id = pu.pub_id
```

そのテーブルまたはビューへの他のすべての参照（たとえば **where** 句内などの）も、この相関名を使用する必要があります。相関名は数字では開始できません。

#### index *index\_name*

*table\_name* へアクセスするときに使用するインデックスを指定します。ビューから選択する場合はこのオプションを使用できませんが、**create view** 文の **select** 句には使用できます。

#### parallel

Adaptive Server の設定で並列処理が有効な場合は、並列パーティションまたはインデックス・スキャンを指定します。

#### degree\_of\_parallelism

テーブルまたはインデックスのスキャンを並列実行するワーカー・プロセスの数を指定します。1 に設定すると、クエリが逐次実行されます。

#### prefetch size

大容量 I/O が設定されているキャッシュにバインドされたテーブルの I/O サイズをキロバイト単位で指定します。ビューから選択する場合はこのオプションを使用できませんが、**create view** 文の **select** 句ではこのオプションを使用できます。**sp\_helpcache** は、オブジェクトがバインドされるキャッシュまたはデフォルトのキャッシュの有効サイズを表示します。データ・キャッシュ・サイズを設定するには、**sp\_cacheconfigure** を使用します。

**prefetch** を使用してプリフェッチ・サイズ (*size*) を指定するとき、最小値は 2K、および 16K までの各論理ページ・サイズに基づいた 2 の累乗になります。キロバイト単位の **prefetch** サイズ・オプションは、次のとおりです。

論理ページ・サイズ	プリフェッチ・サイズ・オプション
2	2, 4, 8, 16
4	4, 8, 16, 32
8	8, 16, 32, 64
16	16, 32, 64, 128

クエリで指定される `prefetch` サイズは、1つの候補にすぎません。サイズ設定を有効にするには、そのサイズでデータ・キャッシュを設定してください。データ・キャッシュを特定のサイズに設定していない場合、デフォルトの `prefetch` サイズが使用されます。

コンポーネント統合サービスが有効な場合は、リモート・サーバに対して `prefetch` を使用できません。

#### `lru` | `mru`

テーブルに対して使用するバッファ置換方式を指定します。

**MRU/LRU** (最も最近に使用された/最も長い間使用されていない) チェーン上のキャッシュへのテーブルの読み込みをオプティマイザに強制的に実行させるには、`lru` を使用してください。キャッシュからバッファを解放し、解放したバッファをテーブルの次のバッファに置き換えるには、`mru` を使用してください。ビューから選択する場合はこのオプションを使用できませんが、`create view` 文の `select` 句には使用できます。

#### `holdlock`

(トランザクションの完了状態に関係なく、要求されたデータ・ページが必要でなくなった時点で共有ロックを解放するのではなく) トランザクションが完了するまで共有ロック保持することで、指定のテーブルやビューの共有ロックをさらに制限します。

`holdlock` オプションは、このオプションが使用される文によって定義されたトランザクションの間だけ、このオプションが指定されたテーブルまたはビューにのみ適用されます。`set` コマンドの `transaction isolation level 3` オプションを設定すると、`holdlock` は、トランザクション内の各 `select` 文に暗黙的に適用されます。キーワード `holdlock` は、`for browse` オプションが指定された `select` 文では使用できません。`holdlock` オプションと `noholdlock` オプションを同時に1つのクエリで指定することはできません。

コンポーネント統合サービスが有効な場合は、リモート・サーバに対して `holdlock` を使用できません。

#### `noholdlock`

現在有効なトランザクションの独立性レベルに関係なく、サーバでこの `select` 文の実行中に取得されたロックが保持されないようにします。`holdlock` オプションと `noholdlock` オプションを同時に1つのクエリで指定することはできません。

### shared

Adaptive Server に対し、指定のテーブルまたはビューで (更新ロックではなく) 共有ロックを使用するように指示します。このキーワードを指定すると、他のクライアントがテーブルまたはビューの更新ロックを取得できません。shared キーワードは、declare cursor 文の一部である select 句でのみ使用できます。次に例を示します。

```
declare shared_crsr cursor
for select title, title_id
from titles shared
where title_id like "BU%"
```

holdlock キーワードは、各テーブルまたはビュー名の後に、shared とともに使用できますが、shared より前に holdlock を記述する必要があります。

### ANSI join

ANSI 構文を使用する内部または外部ジョインです。from 句は、ジョインするテーブルを指定します。

### inner

on 句の条件に適合する内部および外部テーブルのローだけを含みます。内部ジョインを含むクエリの結果セットには、on 句の条件を満たさない外部テーブルのローに null が指定されたローは含まれません。

### outer

on 句の条件に適用するかどうかに関係なく、外部テーブルのローすべてを含みます。ローが on 句の条件を満たさない場合、内部テーブルからの値が null 値としてジョイン・テーブルに格納されます。ANSI 外部ジョインの where 句によってクエリ結果に含まれるローが限定されます。

### left

左ジョインは、join 句の左側にリストされたテーブル参照のローをすべて保持します。左テーブル参照は、外部テーブルまたはロー維持テーブルと呼ばれます。

以下のクエリで、T1 は外部テーブルを、T2 は内部テーブルを表します。

```
T1 left join T2
T2 right join T1
```

### right

右ジョインは、join 句の右側にあるテーブル参照のローをすべて保持します (上記の例を参照)。

**search\_conditions**

ローの探索条件を設定するために使用します。探索条件には、カラム名、式、算術演算子、比較演算子、not、like、is null、and、or、between、in、exists、any、all の各キーワード、サブクエリ、case 式またはこれらの組み合わせを使用できます。詳細については「[where 句](#)」(839 ページ)を参照してください。

**group by**

各グループの値を検索します。検索結果には、これらの値は、新しいローではなく新しいカラムとして表示されます。

標準 SQL で group by を使用する場合は、select リスト内の各項目の固定値が、グループの各ローに格納されているか、またはこれらの項目が、グループごとに単一値を生成する集合関数で使用される必要があります。Transact-SQL では、select リストの項目についてこのような制約はありません。Transact-SQL では、任意の式 (カラムのエイリアスを除く) を使用してグループ化できます。標準 SQL では、カラムでのみグループ化できます。

[表 1-28](#) に示す集合は group by とともに使用できます (*expression* はカラム名である場合がほとんどです)。

表 1-28:group by で集合を使用した結果

集合関数	結果
sum ([all   distinct] expression)	数値カラムの値の合計
avg ([all   distinct] expression)	数値カラムの値の平均
count ([all   distinct] expression)	integer として返された、カラム内の (重複しない) null 以外の値の数
count_big ([all   distinct] expression)	bigint として返された、カラム中の (重複しない) null 以外の値の数
count(*)	integer として返された、選択したローの数
count_big (*)	bigint として返された、選択したローの数
max (expression)	カラム内の最大値
min (expression)	カラム内の最小値

詳細については「[group by 句と having 句](#)」(527 ページ)を参照してください。

カラムを任意に組み合わせてテーブルをグループ化できます。グループを相互にネストできます。カラム見出しによるグループ分けはできません。カラム名、式、または select リスト内の項目の位置を表す数字を使用する必要があります。

**group by all**

探索条件に一致するローがない場合でも、すべてのグループを実行結果に組み込みます。例については、「[group by 句と having 句](#)」(527 ページ)を参照してください。

**aggregate\_free\_expression**

集合関数が含まれていない式です。

**having**

**where** 句が **select** 句の条件を設定する場合と同様の方法で、**group by** 句の条件を設定します。含めることができる条件の数に制限はありません。

**group by** 句なしで **having** 句を使用することもできます。

**select** リスト内のカラムに対して集合関数が適用されておらず、クエリの **group by** 句内にもこれらのカラムが含まれない(標準 SQL では無効)場合、**having** および **where** の意味はやや異なります。

この場合、**where** 句は、集合関数の処理対象となるローを制限しますが、クエリにより返されるローは制限しません。逆に、**having** 句は、クエリによって返されるローを制限しますが、集合関数の処理対象となるローは制限しません。例については、「[group by 句と having 句](#)」(527 ページ)を参照してください。

**order by**

カラムに基づいて結果をソートします。Transact-SQL では、**select** リストにない項目に **order by** を使用できます。カラム名、カラム見出し(またはエイリアス)、式、または **select** リスト内の項目の位置を表す番号 (*select\_list\_number*) によってソートできます。

*select\_list\_number* でソートする場合は、**order by** 句が参照するカラムを **select** リストに指定します。**select** リストを \* (アスタリスク) で指定することはできません。

**select max** を **order by** とともに使用すると、結果セットで複数のローが返される場合があります。

**asc**

実行結果を昇順でソートします(デフォルト)。

**desc**

実行結果を降順でソートします。



**compute**

ローの集合関数 (sum、avg、min、max、count、count\_big) で制御ブレイク計算値を生成するために使用されます。計算値は、クエリ結果で追加ローとして表示されるため、1 つの文に関するディテール・ローと計算ローを参照できます。

**compute** を **select into** 句とともに使用することはできません。

**compute by** 句を使用する場合は、**order by** 句も使用する必要があります。**compute by** 句の後にリストする各カラムは、**order by** 句の後にリストする各カラムと一致するか、それらのサブセットである必要があります。また、左から右に同じ順序で同じ式から始まっている必要があります。式を省略することはできません。

たとえば、**order by** 句が **order by a, b, c** である場合、**compute by** 句には、次のいずれか (またはすべて) を指定できます。

```
compute by a, b, c
compute by a, b
compute by a
```

合計、合計カウントなどを生成するために、キーワード **compute by** を付けずに使用できます。**compute** に **by** を付けない場合、**order by** 句はオプションになります。詳細については、「[compute 句](#)」(107 ページ) を参照してください。

コンポーネント統合サービスが有効な場合は、リモート・サーバに対して **compute** は転送されません。

**for {read only | update}**

カーソル結果セットが読み込み専用または更新可能であることを指定します。

**Adaptive Server** バージョン 15.7 以前では、このオプションはストアード・プロシージャ内のみで、プロシージャがカーソルへのクエリを定義するときだけ使用できます。この場合、**select** は、プロシージャ内で使用できる唯一の文です。この文は、**for read only** または **for update** オプションを (**declare cursor** 文の代わりに) 定義します。この方法でカーソルを宣言すると、ローをフェッチしながら、ページレベル・ロックを設定することができます。

**Adaptive Server** のバージョン 15.7 以前ではカーソルを定義するときにはストアード・プロシージャ内の **select** 文を使用しないと、**Adaptive Server** が **for read only | update** オプションを無視します。ストアード・プロシージャの使用の詳細については、**Embedded SQL™** のマニュアルを参照してください。

Adaptive Server 15.7 以降では `select for update` が設定されている場合、ストアド・プロシージャ以外の言語レベルのトランザクション内で `for update` オプションを使用することができます。このような `select` はカーソルを参照する必要はありません。`select for update` をデータローがロックされたテーブルで使用すると、選択されたローは、トランザクション中排他的にロックされます。

読み込み専用または更新可能カーソルについては、『*Transact-SQL ユーザーズ・ガイド*』を参照してください。

#### `of column_name_list`

`for update` オプションで更新可能として定義されたカーソル結果セットからのカラムのリストです。

#### `at isolation`

クエリの独立性レベル (0、1、2、または 3) を指定します。この句を省略すると、クエリでは、それが実行されるセッションの独立性レベル (デフォルトでは独立性レベル 1) が使用されます。`at isolation` 句は、単一クエリに対して指定されている場合、または `declare cursor` 文の中で使用されている場合に有効です。次のようなクエリで `at isolation` を使用した場合、Adaptive Server は構文エラーを返します。

- `into` 句を使用したクエリ
- サブクエリ内
- `create view` 文の中のクエリ
- `insert` 文の中のクエリ
- `for browse` 句を使用したクエリ

クエリ内に `union` 演算子がある場合は、最後の `select` の後に `at isolation` 句を指定する必要があります。`holdlock`、`noholdlock`、または `shared` を、`at isolation read uncommitted` が指定されているクエリに指定すると、Adaptive Server から警告が発行され、`at isolation` 句は無視されます。他の独立性レベルの場合、`holdlock` は `at isolation` 句よりも優先されます。独立性レベルの詳細については、『*Transact-SQL ユーザーズ・ガイド*』を参照してください。

コンポーネント統合サービスが有効な場合でも、リモート・サーバに対して `at isolation` を使用できません。

#### `read uncommitted | 0`

クエリに対して独立性レベル 0 を指定します。

**read committed | 1**

クエリに対して独立性レベル 1 を指定します。

**repeatable read | 2**

クエリに対してトランザクション独立性レベル 2 を指定します。

**serializable | 3**

クエリに対して独立性レベル 3 を指定します。

**for browse**

DB-Library ブラウズ・アプリケーションで Adaptive Server に送信された SQL 文の終わりに追加されます。詳細については、『Open Client DB-Library リファレンス・マニュアル』を参照してください。

**plan "abstract plan"**

クエリを最適化するために使用する抽象プランを指定します。抽象プラン言語で指定された完全プランまたは部分プランを指定できません。詳細については、『パフォーマンス&チューニング・ガイド』の「抽象プランの作成と使用」を参照してください。

**例**

**例 1** publishers テーブルのすべてのローとカラムを選択します。

```
select * from publishers
```

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

**例 2** publishers テーブルの特定のカラムからすべてのローを選択します。

```
select pub_id, pub_name, city, state from publishers
```

**例 3** publishers テーブルの特定のカラムからすべてのローを選択し、カラム名を 1 つ代入して文字列を出力に追加します。

```
select "The publisher's name is",
       Publisher = pub_name, pub_id
from publishers
```

	Publisher	pub_id
The publisher's name is	New Age Books	0736
The publisher's name is	Binnet & Hardley	0877
The publisher's name is	Algodata Infosystems	1389

**例 4** titles テーブルの特定の列からすべてのローを選択し、列名を代入します。

```
select type as Type, price as Price
from titles
```

**例 5** select into にロック・スキームとページ・ギャップの予約を指定します。

```
select title_id, title, price
into bus_titles
lock datarows with reservepagegap = 10
from titles
where type = "business"
```

**例 6** 選択して bigspenders テーブルに挿入するときに creditcard 列を暗号化します。

```
select creditcard, custid, sum(amount)
into #bigspenders (creditcard
encrypt with cust.database.new_cc_key) from daily_xacts
group by creditcard having sum(amount) > $5000
```

**例 7** 排他ロックされていないローだけを選択します。他のユーザが修飾ローに排他ロックを設定している場合、そのローは返されません。

```
select title, price
from titles readpast
where type = "news"
and price between $20 and $30
```

**例 8** 特定の列とローを選択し、結果をテンポラリー・テーブル #advance\_rpt に格納します。

```
select pub_id, total = sum (total_sales)
into #advance_rpt
from titles
where advance < $10000
and total_sales is not null
group by pub_id
having count (*) > 1
```

**例 9** authors テーブルの au\_lname から上の 3 つのローを選択します。

```
select top 3 au_lname from authors
```

**例 10** 2 つの列を連結し、結果をテンポラリー・テーブル #tempnames に格納します。

```
select "Author_name" = au_fname + " " + au_lname
into #tempnames
from authors
```

**例 11** 特定のカラムとローを選択し、結果を最上位から最下位の順に並べ替え、概要情報を計算します。

```
select type, price, advance from titles
order by type desc
compute avg (price), sum (advance) by type
compute sum (price), sum (advance)
```

**例 12** 特定のカラムとローを選択し、price カラムと advance カラムの合計を計算します。

```
select type, price, advance from titles compute sum (price), sum (advance)
```

**例 13** coffeetabletitles テーブルを作成します。このテーブルは価格が 20 ドルより高い本だけで構成された titles テーブルのコピーです。

```
select * into coffeetabletitles from titles
where price > $20
```

**例 14** newtitles テーブルを作成します。このテーブルは titles テーブルの空のコピーです。

```
select * into newtitles from titles
where 1 = 0
```

**例 15** オプティマイザ・ヒントを提供します。

```
select title_id, title
from titles (index title_id_ind prefetch 16)
where title_id like "BU%"
```

**例 16** sales\_east テーブルと sales\_west テーブルから、syb\_identity キーワードを使用して IDENTITY カラムを選択します。

```
select sales_east.syb_identity,
sales_west.syb_identity
from sales_east, sales_west
```

**例 17** newtitles テーブルを作成します。このテーブルは、IDENTITY カラム付きの titles テーブルのコピーです。

```
select *, row_id = identity (10)
into newtitles from titles
```

**例 18** クエリのトランザクション独立性レベルを指定します。

```
select pub_id, pub_name
from publishers
at isolation read uncommitted
```

**例 19** `repeatable read` 独立性レベルを使用して、`titles` から選択します。他のユーザは、トランザクションが完了するまで、影響を受けるローの値を変更または削除できません。

```
begin tran
select type, avg (price)
  from titles
  group by type
at isolation repeatable read
```

**例 20** クエリに、並列度に関するオブティマイザ・ヒントを提供します。

```
select ord_num from salesdetail
  (index salesdetail parallel 3)
```

**例 21** `titleauthor` テーブルと `titles` テーブルを、これらのテーブルの `title_id` カラムでジョインします。結果セットには、15 を超える `price` を含むローだけが含まれます。

```
select au_id, titles.title_id, title, price
from titleauthor inner join titles
on titleauthor.title_id = titles.title_id
and price > 15
```

**例 22** 結果セットには `authors` テーブルのすべての作家が含まれます。出版社と同じ都市には住んでいない作家は `pub_name` カラムに `null` 値が生成されます。出版社と同じ都市に住んでいる作家の `Cheryl Carson` と `Abraham Bennet` のみ、`pub_name` カラムに `null` 以外の値が生成されます。

```
select au_fname, au_lname, pub_name
from authors left join publishers
on authors.city = publishers.city
```

**例 23** `identity` ギャップを使用して、既存のテーブル (`oldtable`) から新しいテーブル (`newtable`) を作成します。`select into` 文でこの処理を記述します。

```
select identity into newtable
with identity_gap = 20
from oldtable
```

`identity` ギャップの詳細については、『`Transact-SQL ユーザーズ・ガイド`』の「データベースおよびテーブルの作成」の「テーブルの `identity` ギャップの管理」を参照してください。

**例 24** 新しいテーブル、`bay_area_authors` を低レベル圧縮で作成し、そのテーブルにサンフランシスコ・ベイ・エリアに住む著者の情報を移植します。

```
select * into bay_area_authors
with compression = row
from authors
where postalcode like "94%"
```

**例 25** title と advance 以外のカラムをすべて圧縮する新しいテーブルを titles\_2 という名前で作成します。

```
select * into titles_2
(title not compressed,
advance not compressed)
with compression = page
from titles
```

**例 26** select for update 設定パラメータを設定して、select for update を実行し、同じトランザクション内で update を行います。

```
sp_configure 'select for update', 1
Parameter Name          Default  Memory Used  Config Value  Run Value
Unit                    Type
-----
```

```
select for update      0          0           1             1
not applicable        dynamic
(1 row affected)
```

Resulting configuration value and memory use have not changed from previous values:new configuration value 1, previous value 1.  
(return status = 0)

```
begin tran
select c_int, c_bigdatetime from basetb11 where c_int > 90 for update of
c_bigint
```

```
c_int          c_bigdatetime
-----
91             Sep 14 2009  9:00:00.000000PM
92             Sep 15 2009  9:00:00.000000PM
93             Sep 16 2009  9:00:00.000000PM
94             Sep 17 2009  9:00:00.000000PM
95             Sep 18 2009  9:00:00.000000PM
96             Sep 19 2009  9:00:00.000000PM
97             Sep 20 2009  9:00:00.000000PM
98             Sep 21 2009  9:00:00.000000PM
99             Sep 22 2009  9:00:00.000000PM
100            Sep 23 2009  9:00:00.000000PM
(10 rows affected)
```

```
update basetb11 set c_bigint = 5000 where c_int > 90
```

```
(10 rows affected)
commit tran
go
```

**例 27** 既存のテーブル `sales_detail` から新しいテーブル `sales_report` を作成します。新しいテーブルは、`qty` カラムで範囲分割されます。

```
select * into sales_report partition by range (qty)
  (smallorder values <= (500) on seg1,
  bigorder values <= (5000) on seg2)
from sales_detail
```

**例 28** 次のクエリを使用して、チューニングの候補となる過大な I/O を発生させる文を検出します。

```
select lio_avg, qtext from sysquerymetrics order by
lio_avg
```

**例 29** `titles` テーブルを選択して `pubs3` データベースに挿入します。

```
select title_id, title, price
into bus_titles
with dml_logging = minimal
from titles
```

## 使用法

- 他のすべての SQL 文同様、`select` 文のキーワードは、構文に示されている順序で使用してください。
- `select` 文には最大 4096 個の式を指定できます。
- 他の SQL 実装との互換性を実現するため、`select` の後にキーワード `all` を指定します。`all` はデフォルトです。このコンテキストでの `all` は、`distinct` とは対照的なキーワードです。結果には、重複しているローを含め、取得されたローがすべて含まれています。
- `create table`、`create view`、`select into` 文以外の文では、カラム見出しを引用符で囲むことによって、ブランクや `Adaptive Server` キーワードなど、任意の文字を含めることができます。見出しを引用符で囲まない場合には、識別子の規則に従って見出しを指定します。
- `like` が示す文字列は 255 バイト以下である必要があります。
- 255 を超える数のカラムが含まれているテーブルには `select...for browse` オプションを使用できません。
- `create table`、`create view`、`select into` 文では、カラム見出しは、テーブルのエイリアスと同様に、識別子の規則に従っている必要があります。



- `select` を使用して、`null` 値を持つフィールドが含まれるテーブルから、`null` 値を許可しないテーブルヘデータを挿入するには、元のテーブルの `NULL` エントリに値を代入する必要があります。たとえば、次の例は、`NULL` フィールドに “0” を代入して、`null` 値を許可しない `advances` テーブルにデータを挿入します。

```
insert advances
select pub_id, isnull (advance, 0) from titles
```

`isnull` 関数を指定しないと、このコマンドは `null` 以外の値を持つすべてのローを `advances` テーブルに挿入し、`titles` テーブル内で `advance` カラムに `null` 値が含まれているすべてのローに対してエラー・メッセージを生成します。

このような置き換えがユーザのデータに対して行われなかった場合、`null` 値を持つデータを `NOT NULL` 指定のカラムに挿入することはできません。

2 つのテーブルの構造を同じにして、一部のフィールドで `null` 値を使用できるかどうかを別々に設定することができます。テーブル内のカラムの `null` タイプを表示するには、`sp_help` を使用してください。

- `select` 文で返される `text`、`unitext`、または `image` データのデフォルトの長さは 32K です。この値を変更するには、`set textsize` を使用してください。現在のセッションのサイズは、グローバル変数 `@@textsize` に格納されています。一部のクライアント・ソフトウェアでは、`Adaptive Server` へのログイン時に `set textsize` コマンドが発行される場合があります。
- リモート `Adaptive Server` のデータを取得するには、リモート・プロシージャ・コールを使用します。詳細については、「`create procedure`」および「`execute`」を参照してください。
- (`declare cursor` での) カーソル定義で使用される `select` 文には、`from` 句を指定する必要があります。ただし、この文には `compute`、`for browse`、または `into` 句は指定できません。`select` 文に次のいずれかの構成要素が指定されている場合には、カーソルは更新可能ではなく読み込み専用とみなされます。
  - `distinct` オプション
  - `group by` 句
  - 集合関数
  - `union` 演算子

ストアド・プロシージャ内で、`order by` 句を指定した `select` 文を使用してカーソルを宣言すると、このカーソルは読み込み専用とみなされます。複数のテーブルのジョインが指定されている `select` 文で定義されたカーソルが、更新可能とみなされる場合でも、このカーソルを使用してローを削除することはできません。詳細については、「`declare cursor`」を参照してください。

- 値を変数に割り当てる `select` 文によって複数のローが返される場合は、最後の戻り値が変数に割り当てられます。次に例を示します。

```
declare @x varchar (40)
select @x = pub_name from publishers
print @x
(3 rows affected)
Algodata Infosystems
```

#### ANSI ジョイン構文の使用

- ANSI 内部および外部ジョイン構文を使用してクエリを記述するには、まず『`Transact-SQL ユーザーズ・ガイド`』の「ジョイン：複数テーブルからのデータの検索」の「外部ジョイン」を必ず参照してください。

#### `select into` の使用

- `select into` は2段階の操作からなる文です。第1段階では新しいテーブルを作成し、第2段階では指定したローをその新しいテーブルへ挿入します。

---

**注意** コンポーネント統合サービスの既存のテーブルに対して `select into` を実行できます。

---

`select into` オペレーションによって挿入されたローはログに記録されないため、`ddl in tran` データベース・オプションが `true` に設定されている場合でも、ユーザ定義トランザクション内で `select into` コマンドを発行することはできません。`select into` オペレーション中のページ割り付けはログに取られるため、大規模な `select into` オペレーションの場合は、トランザクション・ログが満杯になることがあります。

新しいテーブルの作成後に `select into` 文が失敗した場合、Adaptive Server は、テーブルを自動的に削除したり、最初のデータ・ページの割り付けを自動的に解除したりしません。これは、エラーが発生する前に最初のページに挿入されたローがそのページに残ることを意味します。`select into` 文の後で `@@error` グローバル変数の値を調べて、エラーが発生していないことを確認してください。`drop table` 文を使用して新しいテーブルを削除してから、`select into` 文を再発行します。

- 新しいテーブルの名前はデータベース内でユニークであり、識別子の規則に従っている必要があります。テンポラリー・テーブルに対して `select into` を実行することもできます(例 7、8、および 11 を参照)。
- ベース・テーブルに関連付けられているルール、制約、デフォルトは、新しいテーブルへは継承されません。新しいテーブルにルールまたはデフォルトをバインドするには、`sp_bindrule` および `sp_bindefault` を使用します。
- `select into` は、ベース・テーブルの `max_rows_per_page` 値を継承せずに、値 0 の `max_rows_per_page` を使用して新しいテーブルを作成します。`sp_chgattribute` を使用して、`max_rows_per_page` 値を設定してください。
- 永久テーブルに対して `select into` を実行するため、(`sp_dboption` を実行して) `select into/bulkcopy/pllsort` オプションを `true` に設定してください。テンポラリー・データベースがリカバリされることはないため、テンポラリー・テーブルに `select into` を実行するために `select into/bulkcopy/pllsort` オプションを `true` に設定する必要はありません。

データベースで `select into` を使用した後は、完全データベース・ダンプを実行します。これで、`dump transaction` コマンドを実行できます。`select into` オペレーションでは、ページ割り付けだけがログに取られ、データ・ローへの変更はログに取られません。したがって、トランザクション・ログから変更をリカバリできません。この状況で `dump transaction` 文を実行すると、この文の代わりに `dump database` を使用するよう指示するエラー・メッセージが生成されます。

新しく作成されたデータベースでは、`select into/bulkcopy/pllsort` オプションはデフォルトで `false` に設定されています。このデフォルト設定を変更するには、`model` データベースでこのオプションを `true` に設定します。

- `select into` は、アーカイブ・データベースで使用できます。

- `dump database` の実行中は、`select into` は実行速度が遅くなります。
- `select into` を使用して `where` 句に `false` の条件を指定することで、データの無い複製テーブルを作成できます (例 12 を参照)。
- 集合関数または式が含まれている `select` リストのカラムには、カラム見出しを指定してください。 `select` リストで、定数、算術式か文字式、組み込み関数、または連結を使用する場合には、これらの影響を受ける項目に対してカラム見出しを指定する必要があります。カラム見出しに有効な識別子を使用するか、またはカラム見出しを引用符で囲んでください (例 7 および 8 を参照)。
- 次のように `select into` を使用すると、データ型と `null` 入力可否性はリテラル値に暗黙的に割り当てられます。

```
select x = getdate () into mytable
```

これにより `allow nulls by default` がオンかオフかにかかわらず、カラムは `null` 入力不可となります。これは、`select` コマンドの使用方法和、その構文でともに使用される他のコマンドに依存します。

`convert` 構文により、対象カラムのデータ型や `null` 入力可否性をデフォルトではなく、明示的に指定することができます。

たとえば次のように、結果が `null` 値になる関数で `getdate` をラップします。

```
select x = nullif (getdate (), "1/1/1900") into mytable
```

または、`convert` 構文を使用します。

```
select x = convert (datetime null, getdate ()) into mytable
```

- ユーザ定義トランザクション内、または `compute` 句が指定されている文の中で `select into` を使用することはできません。
- 結果テーブルに `IDENTITY` カラムを入れるには、`select` 文の `column_list` にカラム名 (または `syb_identity` キーワード) を指定してください。新しいカラムは、次の規則に従います。
  - `IDENTITY` カラムは、複数回選択されると、新しいテーブル内で `NOT NULL` として定義されます。`IDENTITY` プロパティは継承されません。

- 式の一部として **IDENTITY** カラムが選択される場合、結果カラムには **IDENTITY** プロパティは継承されません。null を指定できるカラムが式に含まれている場合には、結果カラムは null として作成されます。それ以外の場合は **NOT null** として作成されます。
- **select** 文に **group by** 句または集合関数が含まれている場合、結果カラムには **IDENTITY** プロパティは継承されません。**IDENTITY** カラムの集約が含まれているカラムは **null** として作成され、その他のカラムは **NOT null** として作成されます。
- **union** またはジョインを使用してテーブルに選択された **IDENTITY** カラムでは、**IDENTITY** プロパティが保持されません。テーブルに **IDENTITY** カラムの **union** および **NULL** カラムが含まれる場合、新しいカラムは **NULL** として定義されます。それ以外の場合は、新しいカラムは **NOT NULL** として定義されます。
- **select into** では、複数の **IDENTITY** カラムを持つ新しいテーブルを作成できません。**select** 文に既存の **IDENTITY** カラムが含まれており、なおかつ `column_name = identity(precision)` という形式で新しい **IDENTITY** カラムが指定されている場合、この文は失敗します。
- コンポーネント統合サービスが有効で、**Adaptive Server** に **into** テーブルがある場合、**Adaptive Server** ではバルク・コピー・ルーチンを使用して新しいテーブルにデータがコピーされます。**select into/bulkcopy** データベース・オプションを **true** に設定してから、リモート・テーブルに対して **select into** を実行します。
- **Embedded SQL** コマンドの **select into host\_var\_list** の詳細については、『**Open Client Embedded SQL** リファレンス・マニュアル』を参照してください。

#### **select...into** を使用したターゲット・カラムの **NULL** プロパティの変換

- **convert** コマンドを使用して、データを選択して挿入するターゲット・カラムの **null** 入力可能性を変更できます。たとえば次の例では、**titles** テーブルのデータを選択してターゲット・テーブル **temp\_titles** に挿入し、**total\_sales** カラムを **null** から **not null** に変換します。

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

### 圧縮レベルの指定

- `select into` で作成したターゲット・テーブルはソーステーブルからの設定はどれも継承しません。つまり、データの取り込み元テーブルがローレベルで圧縮されている場合、`select into` から結果を得るテーブルは圧縮レベルを明確に指定しない限り、圧縮は設定されません。
- ターゲット・テーブルにはカラム、パーティション、テーブルのデータ圧縮レベルを指定できます (パーティションの圧縮レベルはテーブルの圧縮レベルをオーバーライドします)。
- ターゲット・テーブルを圧縮する場合、ソーステーブルの対応するカラムの圧縮レベルに関わらず、**Adaptive Server** はカラムの圧縮レベルを選択します (条件を満たす場合)。
- `select into` コマンドには、ターゲット・テーブルで圧縮されないカラムのリストを指定できます。
- ターゲット・テーブル内のカラムを暗号化圧縮することはできません。

### パラメータとデータ圧縮の相互作用

- `max_rows_per_page` は全ページ・ロック・テーブルのみに適用され、データ・オンリー・ロック・テーブルには使用できません。
- 可変長カラムを持つ全ページ・ロック・テーブルと、データ・オンリー・ロック・テーブルに `exp_row_size` を使用できます。ただし、テーブルはページが全ページ・ロック・テーブルにページ分割を起こしたり、データ・オンリー・ロック・テーブルにロー転送を起こさずにアップデートを拡張できなければなりません。

`exp_row_size` は圧縮されたテーブルと圧縮されていないテーブルに同等のスペース・リミットを計算します。ページ内の圧縮結果は大きなフリー・スペースをもたらすので、`exp_row_size` に確保したスペースは、`exp_row_size` を設定しなかった場合と比べてはるかに少ないロー数になる可能性があります。

変換中に **Adaptive Server** が固定長カラムのいくつかを可変長カラムに変更しても、固定長カラムを持つテーブルに `exp_row_size` を使用することはできません。

- `fillfactor` は、圧縮後のデータページで使用されるスペース決定にパラメータを適用します。これは、クラスタード・インデックスの作成時、データページのソートを必要とする操作のみに関連します。

### *select for update* の使用

- Adaptive Server 15.7 以降では、設定パラメータ *select for update* が 1 に設定されていると、トランザクションのコンテキストや連鎖モード内でデータ・ロー・ロック・テーブルに対しコマンドが実施された場合、*select for update* で選択されるローは排他的に固定されます。*select for update* がカーソル・コンテキストで実行される場合、カーソル *open* と *fetch* 文はトランザクションのコンテキスト内でなければなりません。
- *select for update* で選択されるローは、カーソル・コンテキストの中でも外でも、トランザクションが完了するまで排他的ロックを維持します。
- *select for update* の制限事項は次のとおりです。
  - *select for update* はサブクエリ・ブロックには無効である。
  - *select for update* は *select* がワークテーブルではなくベーステーブルから直接ローを戻すときにのみ適用される。*select for update* は、集合または *group by*、*computed*、*union*、*having*、*distinct* 句を持つクエリと一緒に使用できない。
  - 大量ローの場合は *select for update* よりも *update* を使用した方が実際のトランザクション要件を満たす。更新されたローが最新のセットに現われるかもしれないので、独立性レベル 3 を使用して「幻ロー」を抑止する。
  - *select* 処理時に *select for update* タスクを同時に実行すると、同じロー・セットを異なる順序でロックしようとして、アプリケーションのデッドロックが発生する可能性がある。ただし、いったん *select for update* が終了すると、それに続くこのロー・セットの更新はブロックされず、デッドロックは生じない。
  - 更新可能なカーソルの既存の制限はすべて、カーソル・コンテキスト内外両方の *select for update* に適用される。唯一の違いは、*select for update* では *order by* 句が更新可能なカーソルでサポートされることである。更新可能なカーソルの既存の制限と制約はすべて、言語と実行カーソルの両方に適用される。
  - *select for update* によって参照されるすべてのテーブルは同一データベース上に存在している必要がある。
  - *select for update* は、データ・ロー・ロック・テーブルでのみ使用できる。

### ロー内の LOB カラムで指定

- デフォルトでは、ターゲット・テーブルの LOB カラムは `select` リスト内の対応する LOB カラムの保存プロパティを継承します。ターゲット・テーブルの LOB カラムが `convert(text, column)` 組み込み関数などの式から生成された場合、カラムは自動的にロー外の記憶領域を使用します。

### `select...into` を使用したロック・スキームの指定

- `select...into` で `lock` オプションを使用すると、このコマンドによって作成されるテーブルのロック・スキームを指定できます。ロック・スキームを指定しないと、設定パラメータ `lock scheme` によって設定されたデフォルトのロック・スキームが適用されます。
- `lock` オプションを使用する場合は、記憶領域管理プロパティ `max_rows_per_page`、`exp_row_size`、`reservepagegap` も指定できます。`select into` によって作成されたテーブルの記憶領域管理プロパティを変更するには、`sp_chgattribute` を使用します。

### `select...into` を使用したパーティション方式の指定

- `select...into` で `partitions_clause` を使用すると、このコマンドによって作成されるテーブルのパーティション・プロパティを指定できます (詳細については、[create table](#) を参照してください)。パーティション・タイプを指定しない場合、Adaptive Server は分割されていないテーブルを作成します。ターゲット・テーブルのパーティションの条件を満たすローが、挿入するローの中にない場合、`select...into` は失敗します。

### `index`、`prefetch`、`lru | mru` の使用

- `index`、`prefetch`、`lru | mru` オプションは、クエリ実行のインデックス、キャッシュ、および I/O 方式を指定します。これらのオプションを指定すると、Adaptive Server オプティマイザによって設定された内容が上書きされます。これらのオプションは注意して使用し、`set statistics io on` を使用してパフォーマンスへの影響を常にチェックしてください。これらのオプションの使用の詳細については、『パフォーマンス&チューニング・ガイド』を参照してください。

### 暗号化カラムの使用

- キー名を指定せずに `encrypt` 句を使用すると、Adaptive Server はデータベースのデフォルト・キーを使用してターゲット・カラムのデータを暗号化します。



- ソース・テーブルのカラムが暗号化されている場合、ターゲット・カラムに `encrypt` 句を指定しないと、Adaptive Server はソース・テーブルのデータを復号化し、ターゲット・カラムにプレーン・テキスト・データを挿入します。
- ソース・カラム・データに使用したキーと同じキーを使用してターゲット・カラムに暗号化を指定し、そのキーが初期化ベクトルまたはランダム埋め込みを使用しない場合、Adaptive Server は、中間の復号化や再暗号化を行わずに、データを暗号テキストとしてソース・カラムからターゲット・カラムにコピーします。
- ただし、ソース・カラムとは異なるキーを使用してターゲット・カラムの暗号化を指定した場合、または暗号化中にキーが初期化ベクトルまたは埋め込みを使用する場合、Adaptive Server は、暗号化カラムの選択されたローごとに復号化および暗号化処理を実行します。

#### *parallel* の使用

- `parallel` オプションを使用すると、Adaptive Server オプティマイザが並列処理に使用できるワーカー・スレッドの数が減ります。設定されている `max parallel degree` よりも大きい値を `degree_of_parallelism` に指定することはできません。設定されている `max parallel degree` よりも大きい値を指定すると、オプティマイザは `parallel` オプションを無視します。
- 複数のワーカー・プロセスの結果がマージされる場合、Adaptive Server から返されるローの順序は、実行されるたびに異なります。分割されたテーブルのローを一定の順序で取得するには、`order by` 句を使用するか、またはクエリの `from` 句に `parallel 1` を指定して並列クエリ実行を無効にします。
- 以下の条件のどれかがあてはまる場合は、`parallel` を指定する `from` 句は無視されます。
  - `select` 文が更新または挿入に使用されている。
  - `from` 句がカーソル定義で使用されている。
  - サブクエリにあるいずれかの内部クエリ・ブロック内の `from` 句で `parallel` が使われている。
  - `select` 文によりビューが作成される。
  - テーブルが外部ジョインの内部テーブルである。
  - クエリによりテーブルの `min` または `max` が指定されており、インデックスが指定されている。

- 分割を解除されたクラスタード・インデックスが指定されているか、またはこのインデックスが唯一の `parallel` オプションである。
- クエリによりテーブルに対し `exists` が指定されている。
- 設定パラメータ `max scan parallel degree` の値が 1 であり、クエリでインデックスが指定されている。
- ノンクラスタード・インデックスがカバーされている。インデックス・カバーリングの詳細については、『パフォーマンス & チューニング・ガイド』の「インデックス」を参照してください。
- テーブルがシステム・テーブルか仮想テーブルである。
- クエリが **OR** 方式を使用して処理される。OR 方式の説明については、『パフォーマンス & チューニング・ガイド』を参照してください。
- クエリが多数のローをユーザに返す。

#### *readpast* の使用

- `readpast` オプションを使用すると、他のタスクで設定された非両立ロックによってブロックされることなく、`select` コマンドによって、指定のテーブルにアクセスできます。`readpast` クエリは、データオンリーロック・テーブルに対してのみ実行できます。
- `readpast` オプションを全ページロック・テーブルに指定した場合、`readpast` オプションは無視されます。このコマンドは、コマンドまたはセッションに指定された独立性レベルで動作します。独立性レベルが 0 の場合は、ダーティ・リードが実行されて、コマンドはロックされたローから値を返し、ブロックしません。独立性レベルが 1 または 3 の場合、互換性のないロックを伴うページを読み込む必要があるときにコマンドがブロックします。
- [表 1-29](#) は、`select` コマンド内のテーブルに対するセッション・レベルの独立性レベルと `readpast` の相互作用を示しています。

表 1-29: セッション・レベルの独立性レベルと `readpast` の影響

セッションの独立性レベル	影響
0, read uncommitted (ダーティ・リード)	<code>readpast</code> は無視され、コミットされていないトランザクションを含むローがユーザに返される。警告メッセージが表示される。

セッションの独立性レベル	影響
1、read committed	<p>互換性のないロックが設定されたローまたはページが省略される。読み込まれたローまたはページにロックが設定されない。</p> <p><code>readpast</code> を使用すると重複が生じることがあり、<code>distinct</code> 句を追加してもこの問題は解消しない。</p> <p>この問題を解決するには、<code>readpast</code> を使用するとき、<code>distinct</code> 句に加えて <code>group by</code> 句も使用して重複を回避する。。</p>
2、repeatable read	<p>互換性のないロックが設定されたローまたはページは省略される。文またはトランザクションの最後までに読み込まれたローまたはページすべてに対して共有ロックが設定される。トランザクションの完了までに文によって読み込まれたページすべてにロックが設定される。</p>
3、serializable	<p><code>readpast</code> は無視され、コマンドはレベル 3 で実行される。コマンドは、非両立ロックを伴うすべてのローまたはページをブロックする。</p>

- `readpast` を指定する `select` コマンドに次のいずれかが含まれている場合には、このコマンドは失敗してエラー・メッセージが出力されます。
  - 0 または `read uncommitted` を指定する `at isolation` 句
  - 3 または `serializable` を指定する `at isolation` 句
  - 同じテーブルに対する `holdlock` キーワード
- `at isolation 2` または `at isolation repeatable read` が、`readpast` を指定する `select` クエリに指定されている場合は、文またはトランザクションが完了するまで、`readpast` テーブルに対して共有ロックが設定されます。
- `readpast` オプションが指定された `select` コマンドが、非両立ロックが設定された `text` カラムを検出した場合、`readpast` ロックはローを取得しますが、値が `null` の `text` カラムを返します。この場合、カラムがロックされているため、`null` 値を含む `text` カラムと返される `null` 値は区別されません。

#### `select *` 構文の拡張

ストアド・プロシージャまたはトリガのソース・テキストがシステム・テーブル `syscomments` に格納される場合、`select *` を使用するクエリは、`select *` で参照されるカラム・リストを拡張する `syscomments` に格納されます。

たとえば、カラム `col1` と `col2` が含まれるテーブルの `select *` は、次のように格納されます。

```
select <table>.col1, <table>.col2 from <table>
```

Adaptive Server バージョン 12.5.4 では、識別子 (テーブル名、カラム名など) が識別子のルールに準じているかどうかを確認するよう、カラム・リストの拡張が強化されました。

たとえば、テーブルにカラム col1 と 2col が含まれる場合、2 番目のカラム名は数値で始まります。create table 文ではこの 2 番目のカラム名をカッコに入れる必要があります。

このテーブルから、ストアド・プロシージャまたはトリガで select \* を実行すると、syscomments 内のテキストは次のようになります。

```
select <table>.col1, <table>[2col] from <table>
```

select \* を拡張したテキストで使用されるすべての識別子で、識別子が識別子のルールに従わない場合は角カッコが追加されます。

最新リリースにアップグレードしても Adaptive Server が SQL テキストを使用できるように、識別子の前後に角カッコを追加します。

## 標準規格

ANSI SQL — 準拠レベル：初級レベル

Transact-SQL 拡張機能は次のとおりです。

- 新しいテーブルを作成する select into
- lock 句
- compute 句
- グローバル変数およびローカル変数
- index 句、prefetch、parallel、lru | mru
- holdlock、noholdlock、shared キーワード
- “column\_heading = column\_name”
- 修飾したテーブル名およびカラム名
- for browse 句での select
- group by リストになく、集合関数が指定されていないカラムを select リストで使用
- at isolation repeatable read | 2 オプション

## パーミッション

select のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

細密なパーミッションが有効の場合、テーブルまたはビューの所有者であることが必要。  
テーブルまたはビューに対する select パーミッションを持つユーザであることが必要。

細密なパーミッションが無効	<p>細密なパーミッションが無効の場合、テーブルまたはビューの所有者であるか <code>sa_role</code> が付与されたユーザであることが必要。</p> <p>テーブルまたはビューに対する <code>select</code> パーミッションを持つユーザであることが必要。</p> <p><code>select</code> パーミッションは、デフォルトではテーブルまたはビューの所有者に付与されています。所有者はこのパーミッションを他のユーザに譲渡できます。</p>
---------------	--

監査 `sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
62	select	テーブルからの select	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — select、select into、または readtext</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — set proxy が有効な場合は元のログイン名</li> </ul>
63	select	ビューからの select	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — select、select into、または readtext</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — set proxy が有効な場合は元のログイン名</li> </ul>

参照 コマンド `compute 句`、`create index`、`create trigger`、`delete`、`group by 句` と `having 句`、`insert`、`order by 句`、`set`、`union operator`、`update`、`where 句`

関数 `avg`、`count`、`isnull`、`max`、`min`、`sum`

システム・プロシージャ `sp_cachestrategy`、`sp_chgattribute`、`sp_dboption`

## set

**説明** ユーザの作業セッション中の Adaptive Server クエリ処理オプションを設定します。一部のオプションは、トリガまたはストアド・プロシージャ内部で設定されます。

**構文**

```
set advanced_aggregation on/off
set @variable = expression [, @variable = expression...]
set ansinull { on | off }
set ansi_permissions { on | off }
set arithabort [ arith_overflow | numeric_truncation ] { on | off }
set arithignore [arith_overflow] { on | off }
set bulk array size number
set bulk batch size number
set builtin_date_strings number
set {chained, close on endtran, nocount, noexec, parseonly,
    self_recursion, showplan, sort_resources} {on | off}
set char_convert {off | on [with {error | no_error}] |
    charset [with {error | no_error}]}
set cis_rpc_handling { on | off }
set [clientname client_name | clienthostname host_name
    | clientapplname application_name]
set compression {on | off | default}
set cursor rows number for cursor_name
set {datefirst number, dateformat format, language language}
set delayed_commit {on | off | default}
set deferred_name_resolution { on | off }
set dml_logging {minimal | default}
set encryption passwd 'password_phrase'
    for {key | column} {keyname | column_name}
set export_options [on | off]
set fipsflagger { on | off }
set flushmessage { on | off }
set fmtonly {on | off}
set forceplan { on | off }
set identity_insert [database.owner.]table_name {on | off}
set identity_update table_name {on | off}
```

```
set index_union on | off
set literal_autoparam on | off
set lock {wait [numsecs] | nowait}
set logbulkcopy {on | off }
set materialized_view_optimization {disable | fresh | stale}
set metrics_capture on | off
set mon_stateful_history on | off
set nodata
set offsets {select, from, order, compute, table,
             procedure, statement, param, execute} {on | off}
set option show
set opttimeoutlimit
set parallel_degree number
set plan {dump | load} [group_name] {on | off}
set plan exists check { on | off }
set plan for show
set plan optgoal {allrows_oltp | allrows_mix | allrows_dss |
                 user_defined_goal_identifier}
set plan optlevel value
set plan opttimeoutlimit number
set plan replace { on | off }
set prefetch [on|off]
set print_minlogged_mode_override
set proc_output_params {on | off}
set proc_return_status {on | off}
set process_limit_action { abort | quiet | warning }
set proxy login_name
set quoted_identifier { on | off }
set repartition_degree number
set repthreshold number
set resource_granularity number
set role {"sa_role" | "sso_role" | "oper_role" |
         role_name [with passwd "password"]} {on | off}
set {rowcount number, textsize number}
set scan_parallel_degree number
```

```

set send_locator {on | off }
set session authorization login_name
set switch [serverwide] {on | off} trace_flag , [trace_flag,] [with option [, option]
set show_exec_info [ “on” | “off” ]
set show_permission_source ["on" | "off" ]
set show_permission_source, {on|off}
set show_sqltext {on | off}
set show_transformed_sql, {on|off}
set statement_cache on | off
set statistics {io, subquerycache, time, plancost} {on | off}
set statistics simulate {on | off}
set strict_dtm_enforcement { on | off }
set string_rtruncation { on | off }
set system_view {instance | cluster | clear}
set textsize {number}
set tracefile [filename] [off] [for spid]
set transaction isolation level {
    [read uncommitted | 0] |
    [read committed | 1] |
    [repeatable read | 2] |
    [serializable | 3]}
set transactional_rpc { on | off }

```

## パラメータ

```

set advanced_aggregation
    セッション・レベルで高度集約を有効化または無効化します。

set @variable = expression
    1 つの文に複数の変数を割り当てられます。
    set @variable = expression コマンドは、Transact-SQL の
    select @variable = expression コマンドと同じであり、このコマンドの
    代わりに使用できます。

    expression は、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、またはサブクエリを含みます。

set ansinull { on | off }
    集合と比較の両方の動作に影響します。集合と比較の詳細については、
    「集合動作」(758 ページ)を参照してください。

```



`set ansi_permissions { on | off }`

`delete` 文および `update` 文の ANSI SQL パーミッション条件が確認されるかどうかを決定します。デフォルト値は `off` です。表 1-30 に、パーミッションの条件のまとめを示します。

表 1-30:update と delete に必要なパーミッション

コマンド	set ansi_permissions で必要なパーミッション	
	オフ	On
update	<ul style="list-style-type: none"> <li>値を設定するカラムに対する <code>update</code> パーミッション</li> </ul>	<ul style="list-style-type: none"> <li>値を設定するカラムに対する <code>update</code> パーミッション</li> <li><code>where</code> 句にあるすべてのカラムへの <code>select</code> パーミッション</li> <li><code>set</code> 句の右側のすべてのカラムへの <code>select</code> パーミッション</li> </ul>
delete	<ul style="list-style-type: none"> <li>テーブルへの <code>delete</code> パーミッション</li> </ul>	<ul style="list-style-type: none"> <li>テーブルに対する <code>delete</code> パーミッション</li> <li><code>where</code> 句にあるすべてのカラムへの <code>select</code> パーミッション</li> </ul>

`set arithabort [ arith_overflow | numeric_truncation ] { on | off }`

算術演算エラーが発生したときに Adaptive Server がどのように動作するかを指定します。arithabort arith\_overflow と arithabort numeric\_truncation の2つの arithabort オプションは、それぞれタイプが異なる算術エラーを処理します。各オプションを別々に設定したり、1つの `set arithabort on` または `set arithabort off` 文に両方のオプションを設定したりできます。

- arithabort arith\_overflow** — 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、範囲のオーバフロー、またはドメイン・エラーが発生した後の Adaptive Server の動作を指定します。このタイプのエラーは重大です。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。`arithabort arith_overflow on` が設定されていると、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、エラー発生時まで実行されたバッチ内のコマンドはロールバックされません。ただし Adaptive Server では、バッチ内でエラーを発生させた文よりも後にある文は実行されません。

`arith_overflow` を `on` に設定すると、Adaptive Server に設定された正規化のレベルではなく、実行時を参照します。

`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` — 暗黙的なデータ変換中に真数値型による位取りのロスが発生した後の Adaptive Server 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、エラーが発生させた文をアボートし、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` を設定した場合、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。

#### `set arithignore [arith_overflow] { on | off }`

ゼロによる除算または精度のロスが発生した場合に Adaptive Server によってメッセージが表示され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` に設定されています。この場合、Adaptive Server は数字オーバー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。オーバーフロー・エラーを無視するには、`set arithignore on` を使用してください。オプションの `arith_overflow` キーワードを省略しても何も影響はありません。

#### `set bulk array size number`

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・サーバのメモリでバッファされるローの数を指定します。

このオプションは、コンポーネント統合サービスで `select into` を使用してリモート・サーバにローを転送する場合にのみ使用してください。

現在の設定は、`@@bulkarraysize` グローバル変数を使用して表示します。

`number` — バッファするローの数を示します。転送するローに `text`、`unitext`、`image`、または `java ADTs` が含まれている場合、バルク・コピー・インタフェースは配列サイズの現在の設定を無視して 1 の値を使用します。また、実際に使用される配列サイズは、`@@bulkbatchsize` の値を超えることはありません。`@@bulkbatchsize` が配列サイズより小さい場合は、さらに小さい値が使用されます。

配列サイズの初期値は、新しい接続によって、設定プロパティ `cis bulk insert array size` の現在の設定から継承されます。デフォルトは 50 です。この値を 0 に設定すると、値はデフォルトにリセットされます。

**set bulk batch size number**

バルク・インタフェースを使用したときに、**select into proxy\_table** 経由でリモート・サーバへ転送されるローの数を指定します。バルク・インタフェースは、すべての **Adaptive Server**、および **DirectConnect for Oracle** バージョン 12.5.1 で使用できます。

このオプションは、コンポーネント統合サービスで **select into** を使用してリモート・サーバにローを転送する場合にのみ使用してください。

現在の設定は、**@@bulkbatchsize** グローバル変数を使用して表示します。

バルク・インタフェースを使用すると、指定した数のローの後に **commit** を実行できます。これにより、リモート・サーバでは、バルク転送処理で消費されているログ領域を開放し、トランザクション・ログが満杯になる前に、大量のデータ・セットをサーバからサーバへ転送することができます。

バッチ・サイズの初期値は、新しい接続によって、設定プロパティ **bulk insert batch size** の現在の設定から継承されます。デフォルトは 0 です。0 に設定すると、最後のローが転送されるまで、ローはまったくコミットされません。

**set builtin\_date\_strings number**

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。これはデフォルトの動作で、**builtin\_date\_strings** 値 0 で示されます。

**builtin\_date\_strings** の値を 1 に変更すると、サーバは引数文字列を **bigdatetime** として解釈します。これは、日付順の組み込みの結果に影響します。

**set {chained, close on endtran, nocount, noexec, parseonly, self\_recursion, showplan, sort\_resources} {on | off}**

- **chained** — セッション開始時およびトランザクション終了後に、最初のデータ検索文またはデータ修正文の直前にトランザクションを開始します。連鎖モードの場合、**Adaptive Server** は、次の文の前に **begin transaction** を暗黙的に実行します。**delete**、**fetch**、**insert**、**lock table**、**open**、**select**、**update**。トランザクション内では **set chained** を実行できません。

- **close on endtran** — Adaptive Server は、トランザクションの終了時にそのトランザクション内でオープンされたカーソルすべてをクローズします。トランザクションは、**commit** 文または **rollback** 文のどちらかを使用して終了します。ただし、このオプションを設定するスコープ (ストアド・プロシージャ、トリガなど) 内で宣言されたカーソルだけは影響を受けます。カーソル・スコープの詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

評価済み設定の詳細については、『システム管理ガイド』を参照してください。

- **nocount** — 文の影響を受けるローの表示を制御します。**set nocount on** に設定するとローが表示されなくなります。**set nocount off** に設定すると、ローのカウントが有効になります。
- **noexec** — 各クエリをコンパイルしますが、クエリを実行しません。**noexec** は **showplan** でよく使用されます。**noexec on** を設定すると、**noexec off** を設定するまでは、(その他の **set** コマンドを含め) 後続のコマンドは実行されません。
- **parseonly** — 各クエリの構文をチェックし、エラー・メッセージがある場合は返しますが、クエリのコンパイルと実行は行いません。ストアド・プロシージャまたはトリガの内部で **parseonly** を使用しないでください。
- **self\_recursion** — Adaptive Server で、トリガによってトリガ自体を再起動できるようにするかどうかを決定します (これは **自己再帰** と呼ばれます)。デフォルトでは、Adaptive Server ではトリガの自己再帰はできません。このオプションは、現在のクライアント・セッションの間だけオンにすることができます。その影響は、このオプションを設定するトリガのスコープによって制限されます。たとえば、**self\_recursion on** を設定するトリガが戻るか、または別のトリガを起動すると、このオプションは **off** に復元されます。このオプションは、トリガ内でのみ有効であり、ユーザ・セッションには影響しません。
- **showplan** — クエリの処理プランの説明を生成します。**showplan** の結果は、パフォーマンスの診断に役立ちます。**showplan** がストアド・プロシージャやトリガ内で使用された場合、結果は出力されません。並列クエリでは、実行時の調整されたクエリ・プランがある場合には **showplan** の出力にこのクエリ・プランも含まれます。『パフォーマンス&チューニング・シリーズ』を参照してください。

- `sort_resources` — `create index` 文のソート・プランの説明を生成します。`sort_resources` の結果は、ソート操作が逐次または並列に実行されるかどうかを判断するのに役立ちます。`sort_resources` が `on` の場合、Adaptive Server ではソート・プランが出力されますが、`create index` 文は実行されません。『パフォーマンス&チューニング・ガイド』の「並列ソート」を参照してください。

`set char_convert {off | on [with {error | no_error}] charset [with {error | no_error}]}`

Adaptive Server /クライアント間での文字セット変換を有効または無効にします。クライアントで Open Client DB-Library リリース 4.6 以降が使用されており、クライアントとサーバで使用されている文字セットが異なる場合には、ログイン・プロセス中に変換がオンになり、クライアントで使用されている文字セットに基づいたデフォルトに設定されます。サーバとクライアントの間で文字セットの変換を開始するには、`set char_convert charset` も使用できます。

`charset` には、文字セットの ID または `type` 値が 2000 未満の `sycharsets` の名前のいずれかを設定できます。

`set char_convert off` は、文字が変更されることなく送受信されるように変換をオフにします。`set char_convert on` は、変換がオフに設定されている場合に変換をオンにします。ログイン・プロセス中または `set char_convert` コマンドによって変換がオンに設定されなかった場合は、`set char_convert on` を実行するとエラー・メッセージが生成されます。

`set char_convert charset` を使用して文字セット変換を要求したが、Adaptive Server でこの変換を実行できない場合には、文字セットの変換状態は要求発行前と同じです。たとえば、`set char_convert charset` コマンドの実行前に変換を `off` に設定すると、要求が失敗した場合でも、変換はオフのままです。

`with no_error` オプションが含まれていると、Adaptive Server の文字をクライアントの文字セットに変換できない場合、Adaptive Server からアプリケーションに対して通知は出されません。クライアントが Adaptive Server に接続する時点では、エラー・レポートが最初オンになっています。エラー・レポートが不要な場合は、`set char_convert {on | charset} with no_error` によって、各セッションでこの機能をオフにする必要があります。セッション内でエラー・レポートを再びオンにするには、`set char_convert {on | charset} with error` を使用します。

エラー・レポートが設定されているかどうかにかかわらず、変換できないバイトは、ASCII の疑問符 (?) に置き換えられます。

文字セット変換におけるエラー処理の詳細については、『システム管理ガイド』を参照してください。

**set cis\_rpc\_handling { on | off }**

クラスタ環境で CIS RPC 処理のデフォルト・メカニズムにします。

**set [clientname *client\_name* | clienthostname *host\_name* | clientappliance *application\_name*]**

クライアントに名前を割り当てます。

- **clientname *client\_name*** — クライアントに個別の名前を割り当てます。複数のクライアントが同じクライアント名を使用して Adaptive Server に接続するシステムで、クライアントを区別するのに役立ちます。ユーザに割り当てた新しい名前は、その新しい名前でも **sysprocesses** テーブルに表示されます。

*client\_name* は、ユーザに割り当てる新しい名前です。

- **clienthostname *host\_name*** — ホストに個別の名前を割り当てます。複数のクライアントが同じホスト名を使用して Adaptive Server に接続するシステムで、クライアントを区別するのに役立ちます。ホストに割り当てた新しい名前は、その新しい名前でも **sysprocesses** テーブルに表示されます。

*host\_name* は、ホストに割り当てる新しい名前です。

- **clientappliance *application\_name*** — アプリケーションに個別の名前を割り当てます。複数のクライアントが同じアプリケーション名を使用して Adaptive Server に接続するシステムで、クライアントを区別するのに役立ちます。アプリケーションに割り当てた新しい名前は、その新しい名前でも **sysprocesses** テーブルに表示されます。

*application\_name* は、アプリケーションに割り当てる新しい名前です。

**set compression {on | off | default}**

セッションの圧縮を有効または無効にします。

- **on** — 新しいデータのデータ圧縮または圧縮が設定されるパーティションやテーブルを有効にします。Adaptive Server は条件を満たすすべてのローを圧縮します。
- **off** — すべての新しいデータを非圧縮として挿入、更新します。それを挿入することで、ページ圧縮は非圧縮ローを無視します。更新されたローは非圧縮のままになります。update の実行中に非圧縮されたロー (**uncompressed** として挿入されたロー) は、明示的に圧縮を指定しない限り、非圧縮のままになります。

- **default** — 圧縮レベルをデフォルト設定にリセットします (挿入や更新はテーブルまたはパーティション設定に従って圧縮されます)。

---

**注意** ほとんどの **set** パラメータと異なり、**set export\_options** を実行してからネストプロシージャで **set compression** を発行すると、**Adaptive Server** は圧縮レベルと親プロシージャのコンテキストにエクスポートしません。

---

#### **set cursor rows number for cursor\_name**

**Adaptive Server** に、クライアント・アプリケーションからの各カーソル **fetch** 要求に対し、ローの **number** を返させます。この **number** は、小数点のない数値リテラル、または **integer** 型のローカル変数です。この **number** がゼロ以下の場合、値は 1 に設定されます。オープンされていてもクローズされていても、カーソルには **cursor rows** オプションを設定できます。ただし、このオプションは、**into** 句を含む **fetch** 要求には影響しません。**cursor\_name** は、返されるローの数を設定するカーソルを指定します。

#### **set {datefirst number, dateformat format, language language}**

以下の設定を指定します。

- **datefirst number** — 数値設定を使用して週の最初の曜日を指定します。**us\_english** 言語のデフォルトは日曜日です。週の最初の曜日を設定するには、次の設定を使用します。

週の最初の曜日	使用する設定値
月曜日	1
火曜日	2
水曜日	3
木曜日	4
金曜日	5
土曜日	6
日曜日 ( <b>us_english</b> 言語のデフォルト)	7

---

**注意** 週の最初にどの曜日を設定しても、その最初の曜日の値は 1 になります。この値は、`set datefirst n` で使用する数値設定とは異なります。たとえば、日曜日を週の最初の曜日に設定すると、日曜日の値が 1 になります。月曜日を週の最初の曜日に設定すると、月曜日の値が 1 になります。水曜日を週の最初の曜日に設定すると、水曜日の値が 1 になり、その他も同様になります。

---

- `dateformat format` — `datetime`、`smalldatetime`、`date`、または `time` のデータを入力する場合の日付要素 `month/day/year` の順序です。有効な引数は `mdy`、`dmy`、`ymd`、`ydm`、`myd`、`dym` です。us\_english 言語のデフォルト順序は `mdy` です。
- `language language` — システム・メッセージを表示する言語の公式名を示します。この言語が、Adaptive Server にインストールされている必要があります。デフォルトは `us_english` です。

#### `set deferred_name_resolution`

現在のセッションのみに対して `deferred name resolution on` を設定します。

#### `set delayed_commit {on | off | default}`

ログ・レコードをいつディスクに書き込むかを決定します。

`delayed_commit` パラメータを `true` に設定すると、ログ・レコードはディスクに非同期に書き込まれ、I/O の完了を待たずに、制御はクライアントに戻ります。

セッション・レベルの設定は、既存のデータベース・レベルの設定に優先します。`delayed_commit` をデフォルト値に変更すると、データベースレベルの設定に戻ります。

---

**注意** `delayed_commit` を使用する前に、アプリケーションへの影響について十分に考慮してください。

---

#### `set dml_logging {minimal | default}`

`insert`、`update`、`delete` (DML) オペレーションのロギングの量を決定します。有効な値は次のとおりです。

- `minimal` — Adaptive Server は DML 文の変更についてログを取りません。多くの場合、Adaptive Server は `syslogs` へのロギングをほとんど生成しません。



- **default** — Adaptive Server は、セッション固有の最低限のロギングを無効にし、テーブル固有およびデータベースワイドのロギング・レベルに基づいて個々のテーブルの有効なロギング・モードを使用します。

ロギングの変更は、該当する場合、このセッションのユーザが所有するオブジェクトにのみ適用されます。さらに、セッション所有者が所有するテーブルのみが **set dml\_logging** の影響を受けます。

- すべてのユーザは **minimal** ロギングに **set dml\_logging** を実行し、**default** のロギング・モードに戻ることができます。この **set** が正常に実行されると、データベースでその文を実行しているユーザが所有するすべてのテーブルの **DML** では、現在のセッションで **set dml\_logging default** が実行されるまで、最低限のログを取ります。
- セッションに対して **minimal** ロギングが有効化されている場合、**DML** ロギングは、デフォルトでデータベースレベルおよびテーブルレベルの設定になりますが、**DML** はセッションのユーザが所有していないテーブルを処理します。
- セッションまたはプロシージャの **DML** ロギング設定は、プロシージャから継承されますが、セッションを実行しているユーザが所有するテーブルにのみ影響します。

```
set encryption passwd 'password_phrase'
for {key | column} {keyname | column_name}
```

**insert**、**update**、**delete**、**select**、**alter table**、または **select into** 文でデータを暗号化または復号化する暗号化キーのパスワードを作成します。

- **password\_phrase** - キーを保護するために **create encryption key** または **alter encryption key** コマンドで指定された明示的なパスワードです。
- **key** — 名前付きキーによって暗号化されたカラムにアクセスするときに Adaptive Server がこのパスワードを使用してキーを復号化することを示します。
- **keyname** — 完全修飾名として入力できます。次に例を示します。  
[[[database.][owner].]keyname
- **column** — 名前付きカラムを暗号化または復号化するコンテキスト内でのみ Adaptive Server がこのパスワードを使用することを示します。エンド・ユーザは、カラムを暗号化するキーの名前を必ずしも知っておく必要はありません。
- **column\_name** — 暗号化パスワードを設定するカラムの名前です。 **column\_name** は次のように入力します。

```
[[ [ database. ] [ owner. ] table_name.column_name
```

#### set export\_options { on | off }

デフォルトでは、Adaptive Server は、トリガまたはシステム・プロシージャの実行が完了した後、これらによって設定された set パラメータの変更をリセットします。set export\_options を有効にすると、システム・プロシージャまたはトリガによって設定されたセッション設定をそのセッション中保持できます。

たとえば、次の例は set export\_options を有効にします。

```
set export_options on
```

次の例は、set export\_options を無効にして Adaptive Server をデフォルト動作に戻します。

```
set export_options off
```

#### set fipsflagger { on | off }

初級レベルの ANSI SQL の Transact-SQL 拡張機能を使用したときに、Adaptive Server が警告メッセージを表示するかどうかを決定します。デフォルトでは、非標準 SQL を使用する場合に警告メッセージは表示されません。このオプションを使用しても SQL 拡張機能は無効にはなりません。処理は、ANSI 以外の SQL コマンドを発行すると完了します。

#### set flushmessage { on | off }

Adaptive Server がユーザにメッセージを返す時期を決定します。デフォルトにより、メッセージは、それらを生成したクエリが完了するか、バッファの容量が満杯になるまで、バッファ内に保管されます。メッセージを生成されると同時にユーザに返すには、set flushmessage on を使用してください。

#### set fmtonly { on | off }

実際に実行せずに、ストアド・プロシージャ内のプランを取得します。

#### set forceplan { on | off }

クエリ・オブティマイザで、クエリ内の from 句のテーブル順序が、クエリ・プランのジョイン順序として使用されます。forceplan は通常、オブティマイザが適切なプランを選択できない場合に使用します。無効なプランを強制的に使用すると、I/O とパフォーマンスに深刻な影響が及ぶ可能性があります。『パフォーマンス & チューニング・シリーズ』を参照してください。

**注意** 次のように外部ジョインで不正なジョイン順を指定しても、クエリ・オプティマイザは無視します。

```
11> set forceplan on
2> select * from table1, table2
    where table2.id *= table1.id
```

`set identity_insert [database.[owner.]]table_name {on | off}`

テーブルの **IDENTITY** カラムに対する明示的な挿入操作を実行できるかどうかを判断します。このオプションはベース・テーブルでのみ使用できます。このオプションをビューで使用したり、トリガ内に設定したりすることはできません。

テーブルに対して `identity_insert on` を設定すると、テーブル所有者、またはカラムに対する `insert` パーミッションを持つユーザは、5 より大きい任意の有効な値を手動で挿入できます。たとえば、次のように値 55 を挿入すると、**IDENTITY** カラム値に大きなギャップが発生します。

```
insert stores_cal
  (syb_identity, stor_id, stor_name)
values (55, "5025", "Good Reads")
select syb_identity from stores_cal
id_col
-----
      1
      5
     55
```

(3 rows affected)

その後、`identity_insert` が `off` に設定されると、Adaptive Server は、次の挿入で **IDENTITY** カラムに  $55 + 1$ 、つまり、値 56 を割り当てます。`insert` 文を含むトランザクションがロールバックされると、Adaptive Server は値 56 を廃棄して、次の挿入に 57 を使用します。

**IDENTITY** カラムでユニーク・インデックスを作成していないかぎり、Adaptive Server は挿入された値の一意性を確認することができません。任意の正の整数を挿入できます。

`identity_insert table_name off` を設定すると、**IDENTITY** カラムへの明示的な挿入操作が禁止され、デフォルト動作がリストアされます。セッション中は、いつでも単一データベース・テーブルに対して `set identity_insert table_name on` を実行できます。

細密なパーミッションが有効	<p><code>set identity_insert table_name on off</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。</p> <p>細密なパーミッションが有効の場合、テーブル所有者であるかテーブルに対する <code>identity_insert</code> パーミッションを持つユーザであることが必要。テーブル所有者、または <code>manage any object permission</code> 権限を持つユーザは、他のユーザにパーミッションを付与できる。</p>
細密なパーミッションが無効	<p>細密なパーミッションが無効の場合、テーブル所有者、データベース所有者、または <code>sa_role</code> が付与されたユーザであることが必要。<code>identity_insert</code> パーミッションは譲渡できない。</p>

#### `set identity_update table_name {on | off}`

`set identity_update` を `on` にすると、テーブル上の `IDENTITY` カラムの値を明示的に更新できます。`identity_update` は、該当するローの `identity` カラムの値を変更します。`identity_update` を有効にすると、`identity` の値を 0 より大きい任意の値に更新できます。ただし、入力値が `identity burn max` の値より大きい場合は、新しい ID 値が割り付けられ、OAM ページ上の `identity burn max` 値も、それに従って更新されます。トランザクションに `update` が含まれている場合、新しい `identity burn max` 値はロールバックできません。`syb_identity` を使用すると、`update` の対象の `identity` カラムを指定することができます。次に例を示します。

```
update table_name set syb_identity = value
where 句
```

Adaptive Server では、重複したエントリをチェックしたり、エントリがユニークであることを確認したりしません。宣言されたカラムの精度で許可されている範囲内で、任意の正の整数に既存の値を更新できます。`identity` カラムにユニーク・インデックスを作成すると、重複エントリをチェックできます。

`set identity_update table_name on|off` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	<p>細密なパーミッションが有効の場合、テーブル所有者であるかテーブルに対する <code>identity_update</code> パーミッションを持つユーザであることが必要。テーブル所有者、または <code>manage any object permission</code> 権限を持つユーザは、他のユーザにパーミッションを付与できる。</p>
細密なパーミッションが無効	<p>細密なパーミッションが無効の場合、テーブル所有者、データベース所有者、または <code>sa_role</code> が付与されたユーザであることが必要。<code>identity_update</code> パーミッションは譲渡できない。</p>

**set index\_union on | off**

この設定を有効にすると、or 句によってテーブルのスキャンが制限されます。

インデックス和集合 (または or 方式) は、or 句を含むクエリに使用されます。次に例を示します。

```
select * from titleauthor where au_id = "409-56-7008" or title_id = "PC8888"
```

index\_union の設定に応じて、次のように動作します。

- 有効 — この例では、au\_id のインデックスを使用して au\_id = "409-56-7008" のすべての titleauthor タブルのロー ID (RID) を検出し、title\_id のインデックスを使用して title\_id = "PC8888" のすべての titleauthor タブルの RID を検出します。次に、Adaptive Server はすべての RID に対して和集合を行って重複を削除します。結果の RID は RidJoin とジョインされてデータ・タブルにアクセスします。
- 無効 — Adaptive Server はテーブル・スキャンを制限するためにクエリでインデックス和集合方式を使用しません。代わりに、テーブルのその他のアクセス・パスを使用し (前述の例では、テーブル titleauthor にテーブル・スキャンを使用する)、or 句を scan 演算子のフィルタとして適用します。

**set literal\_autoparam on | off**

デフォルトでは on になっています。literal\_autoparam のサーバ・レベルの設定が on の場合、このオプションでその機能の使用の有効と無効を切り替えます。サーバ・レベルの設定が off である場合、この設定は無効です。

**set lock {wait [numsecs] | nowait}**

ロックの設定を指定します。

- wait — コマンドがアボートしてエラーを返すまで、ロックの取得を待機する時間の長さを指定します。
- numsecs — コマンドがロックの取得を待機する秒数を指定します。有効な値は 0 ~ 2147483647 (整数の最大値) です。
- lock nowait — コマンドがロックをすぐに取得できない場合に、エラーを返して失敗するように指定します。set lock nowait は、set lock wait 0 と同じです。

**set logbulkcopy {on | off}**

高速にログが記録される bcp をセッションに設定します。

**set materialized\_view\_optimization {disable | fresh | stale}**

クエリ最適化時に考慮される事前計算済み結果セットを決定します。次のいずれかになります。

- **disabled** — (デフォルト) クエリの最適化に事前計算済み結果セットは使用されません。
- **fresh** — **immediate refresh** 方式が指定されている場合のみ、事前計算済み結果セットが考慮されます。
- **stale** — 事前計算済み結果セットが古い場合でも、有効化されているものはすべてクエリの最適化で考慮されます。

**set metrics\_capture {on | off}**

セッション・レベルでクエリ処理 (QP) 測定基準を取得できるようにし、この取得を **on** に設定します。QP 測定基準とは、クエリの実行における経験的な測定基準値を識別し、比較するためのものです。クエリの実行時に、QP 測定基準の比較の基礎となる一連の定義済みの測定基準がクエリに関連付けられます。

**set mon\_stateful\_history on | off**

無効にすると、履歴モニタリング・テーブル (**monSysStatement**、**monErrorLog**、**monSysSQLText**、**monSysPlanText**、**monDeadLock**) に対するクエリでテーブル・バッファ内のすべてのローが返されます。

有効にすると、履歴モニタリング・テーブルに対するクエリでは、**mon\_stateful\_history** の無効化以降にテーブルに追加されたローのみが返されます。

**set nodata**

クエリ実行の完了時に、クライアントに転送されるデータはないと指定します。**set no data on** を指定すると、TDS フォーマット・ストリームのみがクライアントに送信され、クエリの動作は該当するローがない場合と同様になります。

**set offsets {select, from, order, compute, table, procedure, statement, param, execute} {on | off}**

Transact-SQL 文に指定されたキーワードの位置を (クエリの最初と関連させて) 返します。キーワード・リストは、カンマで区切られたリストであり、Transact-SQL 文 **select**、**from**、**order**、**compute**、**table**、**procedure**、**statement**、**param**、**execute** を含めることができます。エラーがない場合には、Adaptive Server はオフセットを返します。

このオプションは、Open Client DB-Library でのみ使用されます。

`set option show_option {normal | brief | long | on | off}`

診断出力をテキスト・フォーマットで生成します。

`show_option` の有効な値は、次のとおりです。

- `show` — すべてのモジュールに共通の基本構文を表示します。
- `show_lop` — 使用された論理演算子 ( スキャン、ジョインなど ) を表示します。
- `show_managers` — 最適化で使用されたデータ構造マネージャを表示します。
- `show_log_props` — 評価された論理プロパティ ( ロー・カウント、選択性など ) を表示します。
- `show_parallel` — 並列クエリの最適化に関する詳細を表示します。
- `show_histograms` — 探索索引数 / ジョイン・カラムに関連付けられたヒストグラムの処理を表示します。
- `show_abstract_plan` — 抽象プランの詳細を表示します。
- `show_search_engine` — ジョイン順を決定するアルゴリズムの詳細を表示します。
- `show_counters` — 最適化カウンタを表示します。
- `show_best_plan` — オプティマイザで選択された最適なクエリ・プランの詳細を表示します。
- `show_pio_costing` — 物理入出力 ( ディスク読み込みと書き込み ) の見積もりを表示します。
- `show_lio_costing` — 論理入出力 ( メモリ読み込みと書き込み ) の見積もりを表示します。
- `show_elimination` — パーティション削除を表示します。
- `show_missing_stats` — 探索索引数 / ジョイン・カラムから欠落した有用な統計の詳細を表示します。

詳細については、『クエリ・オプティマイザ』の「クエリ最適化方式と見積もりの表示」を参照してください。

`set option show_option` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッション  
が有効

細密なパーミッションが有効の場合、`set tracing` 権限または `monitor qp performance` 権限を持つユーザであることが必要。

細密なパーミッションが無効

細密なパーミッションが無効の場合、**set tracing** 権限を持つか、**sa\_role** が付与されたユーザであることが必要。

#### set optimeoutlimit

オペティマイザのタイムアウト制限を設定します。**optimeoutlimit** の有効な値の範囲は 0 ~ 4000 ミリ秒で、0 は最適化の制限がないことを示します。

#### set parallel\_degree number

クエリの並列実行時に使用されるワーカー・プロセスの最大数を指定します。**max parallel degree** 設定パラメータの現在の値以下の数値を指定してください。**@@parallel\_degree** グローバル変数には、現在の設定が格納されます。

#### set plan {dump | load} [group\_name] {on | off}

抽象プラン・コマンドを導入します。

- **dump** — 現在の接続の抽象プランの取得を有効または無効にします。**group\_name** を指定しないと、プランはデフォルト・グループの **ap\_stdout** に保管されます。
- **load** — 現在の接続の抽象プランのロードを有効または無効にします。**group\_name** を指定しないと、デフォルト・グループの **ap\_stdin** からプランがロードされます。
- **group\_name** — プランのロードと保管に使用する抽象プラン・グループの名前です。

『パフォーマンス&チューニング・ガイド』の「抽象プランの作成と使用」を参照してください。

#### set plan exists check { on | off }

**set plan load** とともに指定すると、抽象プラン・グループから最大 20 個のクエリのハッシュ・キーをユーザ単位キャッシュに保管します。

#### set plan for show

診断出力の XML ドキュメントを生成します。**show** の有効な値は次のとおりです。

- **show\_exec\_xml** — コンパイル済みのプランの出力を XML 形式で取得し、各クエリ・プランの演算子を表示します。
- **show\_execio\_xml** — I/O の見積もり値と実測値を含めてプランの出力を取得します。クエリ・テキストも含まれます。



- `show_opt_xml` — オプティマイザ診断出力を取得します。この出力には、論理演算子、マネージャからの出力、検索エンジン診断、最適なクエリ・プランなど、すべてのコンポーネントに関する情報が含まれます。
- `show_lop_xml` — 出力の論理演算子ツリーを XML 形式で取得します。
- `show_managers_xml` — クエリ・オプティマイザの準備フェーズに使用された複数のコンポーネント・マネージャの出力を表示します。
- `show_log_props_xml` — 特定の同等クラス (クエリ内の 1 つ以上の関係のグループ) の論理プロパティを表示します。
- `show_parallel_xml` — 並列クエリ・プラン生成中のオプティマイザに関連する診断を表示します。
- `show_histograms_xml` — ヒストグラムおよびヒストグラムのマージに関連する診断を表示します。
- `show_abstract_plan_xml` — 抽象プラン生成 / アプリケーションを表示します。
- `show_search_engine_xml` — 診断に関連する検索エンジンを表示します。
- `show_counters_xml` — プラン・オブジェクト作成 / 破棄カウンタを表示します。
- `show_best_plan_xml` — 最適なプランを XML 形式で表示します。
- `show_pio_costing_xml` — 実際の PIO コストを XML 形式で表示します。
- `show_lio_costing_xml` — 実際の LIO コストを XML 形式で表示します。
- `show_elimination_xml` — パーティション削除を XML 形式で表示します。
- `client` — 指定すると、出力はクライアントに送信されます。
- `message` — 指定すると、出力は内部メッセージ・バッファに送信されます。

詳細については、『パフォーマンス&チューニング・ガイド：クエリ処理と抽象プラン』の「クエリ最適化方式と見積もりの表示」を参照してください。

`set plan for show` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>set tracing</code> 権限または <code>monitor qp performance</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>set tracing</code> 権限を持つか、 <code>sa_role</code> が付与されたユーザであることが必要。

`set plan optgoal {allrows_oltp | allrows_mix | allrows_dss | user_defined_goal_identifier}`

`optimization goal` を設定します。

- `allrows_mix` — デフォルトの最適化目標です。混合クエリ環境では、これが最も実用的な最適化目標です。OLTP クエリ環境と DSS クエリ環境のニーズのバランスを取ります。
- `allrows_dss` — 複雑さが中程度以上である業務的 DSS クエリを実行する場合に最も便利な最適化目標です。現時点では、この目標は試験的に提供されています。

最適化プランの詳細については、『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「Adaptive Server におけるクエリ処理 (QP) について」を参照してください。

`set plan optlevel value`

セッションの最適化レベルを設定します。Adaptive Server リリースまたは ESD のそれぞれに、新しい最適化レベルが組み込まれている場合があります。次に例を示します。

- `ase_current` — 現在のリリースまでのすべてのオプティマイザの変更を有効にします。
- `ase_default` — バージョン 15.0.3 ESD #1 以降のオプティマイザの変更を無効にします。
- `ase1503esd2` — バージョン 15.0.3 ESD #2 までのすべてのオプティマイザの変更を有効にします。
- `ase1503esd3` — バージョン 15.0.3 ESD #3 までのすべてのオプティマイザの変更を有効にします。

『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「最適化の制御」を参照してください。

**set plan optimeoutlimit *number***

タイムアウトをセッション・レベルで設定します。ここで、*n* は、0 ~ 1000 の整数です。詳細については、『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「Adaptive Server におけるクエリ処理 (QP) について」を参照してください。

**set plan replace { on | off }**

プラン取得モードでの既存の抽象プランの置換を有効または無効にします。デフォルトでは、プランの置換はオフになります。

**set prefetch {on | off}**

データ・キャッシュへの大容量 I/O を有効または無効にします。

**set print\_minlogged\_mode\_override**

トレース情報をセッション出力に生成します。参照整合性制約の存在、遅延モードの選択、影響を受けるテーブルの名前、影響を及ぼす規則の説明などの規則によって、テーブルの最低限のログを取るモードが上書きされる文についてレポートします。

**set proc\_output\_params {on | off}**

ストアド・プロシージャで生成された出力パラメータをクライアントへ送信する処理を制御します。set proc\_output\_params off を設定すると、出力パラメータはクライアントに送信されません。このパラメータのデフォルトは on です。

**set proc\_return\_status {on | off}**

リターン・ステータス TDS トークンをクライアントへ送信する処理を制御します。set proc\_return\_status off を設定すると、リターン・ステータス・トークンはクライアントに送信されず、isql クライアントには (return status = 0) メッセージが表示されません。このパラメータのデフォルトは on です。

---

**警告！** プロシージャを実行するクライアント・アプリケーションが、リターン・ステータスに基づくプロシージャの成功または失敗に依存する場合は、set proc\_return\_status off オプションは使用しないでください。

---

`set process_limit_action { abort | quiet | warning }`

使用可能なワーカー・プロセスの数が不十分なときに、並列クエリを実行するかどうかを指定します。このような状況では、次のように設定できます。

- `process_limit_action` を `quiet` に設定すると、Adaptive Server では、使用可能なプロセスの数を超えない程度の並列度を使用するようにプランが暗黙的に調整されます。
- `process_limit_action` を `warning` に設定すると、使用可能なワーカー・プロセスの数が不十分な場合、Adaptive Server はプランの調整時に警告メッセージを発行します。
- `process_limit_action` を `abort` に設定すると、Adaptive Server はクエリをアボートし、使用可能なワーカー・プロセスの数が不十分であることを示すメッセージを発行します。

`set proxy login_name`

`login_name` のパーミッション、ログイン名、および `suid` (サーバ・ユーザ ID) を引き受けられるようにします。`login_name` には、`master.syslogins` の有効なログインを引用符で囲んで指定します。元のログイン名と `suid` を復元するには、元の `login_name` を指定した `set proxy` を使用します。

---

**注意** `set proxy login_name` を使用するには、システム・セキュリティ担当者を含めたユーザにパーミッションが明示的に付与されている必要があります。明示的なパーミッションがないと、“`sa_role`” も “`sso_role`” も `set proxy login_name` コマンドを発行できません。

---

詳細については「[プロキシの使用](#)」(764 ページ)を参照してください。

`set quoted_identifier { on | off }`

Adaptive Server で二重引用符内の区切り識別子が認識されるかどうかを決定します。デフォルトでは、`quoted_identifier` は `off` であり、識別子は次の条件を満たす必要があります。

- 有効な識別子の規則に従っている。
- カッコで囲まれている。

`set quoted_identifier on` を使用すると、二重引用符はカッコと同じ効力を持ちます。また、非アルファベット文字で始まるか、このオプションを設定しないと使用できない文字を含むか、または予約語であるテーブル、ビュー、およびカラム名を、二重引用符で識別子を囲むことによって使用できます。区切り識別子は 28 バイトを超過してはならず、どのフロントエンド製品でも認識できないようにする必要があります。また、システム・プロシージャへのパラメータとして使用された場合、予期しない結果が生じる場合があります。

`quoted_identifier` が `on` の場合、二重引用符で囲まれる文字列はすべて識別子として処理されます。文字列またはバイナリ文字列は一重引用符で囲んでください。

#### `set repartition_degree number`

セマンティックに利用される中間データ・ストリームが再分割されるときの最大レベルです。セッションに対する `max repartition degree` の値の設定の詳細については、『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「並列クエリ処理」を参照してください。

#### `set rephreshold number`

`SQL` 複写スレッシュホールドをセッション・レベルで設定します。ストアド・プロシージャで `set rephreshold` を呼び出す場合、そのスコープは、プロシージャのスコープになります。ユーザ・セッションで `set rephreshold` を呼び出す場合、そのスコープは、セッションのスコープになります。

ユーザは、ログイン・トリガを使用して、スレッシュホールドのスコープを変更してセッション・スレッシュホールドを設定できます。その場合、ログイン時にセッション・スレッシュホールドを明示的に設定する必要はありません。

次に例を示します。

```
create proc myproc
as
    set rephreshold 777
-----

alter login sa modify login script 'myproc'
-----
option changed.
```

```
(Return status = 0
```

ユーザ **sa** がログインするたびにプロシージャ ' **myproc** ' が呼び出され、セッション全体で複製スレッシュールドが **777** に設定されます。

スレッシュールドのスコープは、次のように **set export\_options** を使用しても変更できます。

```
create proc p2
as
    set repthreshold222
    set export_options on
-----
```

**p2** が実行された後も、スレッシュールドは **222** のままになります。スレッシュールドの階層は、セッション>テーブル>データベースです。

セッション・スレッシュールドを **0** に設定すると、テーブルとデータベースのスレッシュールドが指定されている場合は、それらがその順番で有効な複製スレッシュールドになります。スレッシュールドを設定しない場合、スレッシュールドはデフォルトで **50** 個のローになります。

テーブル・レベルのスレッシュールドを設定するには、『リファレンス・マニュアル：プロシージャ』の「**sp\_setrepdbmode**」を、データベース・レベルのスレッシュールドを設定するには、同じマニュアルの「**sp\_setrepdefmode**」を参照してください。

セッション・スレッシュールドはエクスポートできます。ストアド・プロシージャでスレッシュールドを設定し、**export options** 設定を **ON** にします。**Adaptive Server** は、呼び出し元のプロシージャまたはセッションで新しいスレッシュールドを設定します。

**set repthreshold** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

細密なパーミッションが有効の場合、**manage replication** 権限を持つユーザであることが必要。

細密なパーミッションが無効

細密なパーミッションが無効の場合、**replication\_role** が付与されたユーザであることが必要。

#### **set resource\_granularity number**

グローバル値 **max resource granularity** を上書きしてセッション固有の値に設定します。この値は、**Adaptive Server** でメモリを集中的に使用する操作が実行されているかどうかで異なります。詳細については、『クエリ・プロセッサ』の「並列クエリ処理」を参照してください。

```
set role {role_name [with passwd "password"]} {on | off}
```

必要に応じて、役割をオフにするには `set role role_name off`、役割を再びオンにするには `set role role_name on` を使用します。

- `role_name` は、システム標準の役割またはユーザ定義の役割の名前のいずれかにすることができます。ユーザがログインしたとき、ユーザに付与されているすべてのシステム標準の役割が自動的にアクティブになります。`sa_role` または `sso_role` が付与されている場合は、データベースの指定ユーザまたはエイリアス・ユーザでない限り、または "guest" ユーザが存在するのでない限り、これらの役割をオフに設定することはできません。
- `role_name` — システム・セキュリティ担当者によって作成されたユーザ定義役割の名前です。ユーザ定義の役割は、デフォルトではオンになりません。ユーザ定義の役割をログオン時にアクティブにするように設定するには、システム・セキュリティ担当者がログインやログイン・プロファイルの作成または変更時に、その役割をデフォルトまたは自動アクティブ化の役割として指定する必要があります。
- `with passwd` — 役割をアクティブにするためのパスワードを指定します。ユーザ定義の役割に付加されたパスワードがある場合は、このパスワードを指定して役割をアクティブにする必要があります。

`set role` を使用するには、この役割が付与されている必要があります。特定の役割が割り当てられていることでデータベースへの入力操作ができる場合には、データベースの使用中にはその役割をオフにできません。

アクティブ化述部を使用して役割が付与されている場合、`set role on` の実行時に **Adaptive Server** によって述部が評価されます。述部が `false` と評価されると、**Adaptive Server** はエラーを返し、役割をアクティブ化しません。

```
set {rowcount number, textsize number}
```

指定した数のローが影響を受けた後で、**Adaptive Server** によるクエリ (`select`、`insert`、`update`、または `delete`) の処理を停止します。この `number` は、小数点のない数値リテラル、または `integer` 型のローカル変数です。このオプションをオフにするには、次のコマンドを使用します。

```
set rowcount 0
```

`set rowcount` の現在の値は、`@@setrowcount` グローバル変数を使用して確認できます。次に例を示します。

```
select @@setrowcount
```

---

37

#### set scan\_parallel\_degree *number*

ハッシュベース・スキャン (非分割テーブルに対する並列インデックス・スキャンと並列テーブル・スキャン) のセッション固有の最大並列度を指定します。max scan parallel degree 設定パラメータの現在の値以下の数値を設定してください。@@scan\_parallel\_degree グローバル変数には、現在の設定が格納されます。

#### set send\_locator {on | off }

クライアントに送信される結果セット内で LOB または LOB を参照するロケータを送信するかどうかを指定します。オプションが off (デフォルト) のときは、Adaptive Server は LOB を送ります。

#### set session authorization *login\_name*

is identical to set proxy, with this exception: set session authorization は SQL 規格に準拠し、set proxy は Transact-SQL 拡張機能となります。

#### set show\_exec\_info

コマンドの実行時に追加情報が生成されます。

- on — DML 文のロギング・モードに関する追加の診断情報が生成されます。現在の文およびセッションに選択されているロギング・モードと、DML を実行しているユーザが表示されます。
- off — disables show\_exec\_info が無効になります。

#### set show\_permission\_source, {on|off}

付与対象者、付与対象者のタイプ、付与者、アクション、オブジェクト、および述部を表形式で表示します。付与対象者は、*user\_name*、*role\_name*、*group\_name* のいずれかにすることができます。Type of grantee カラムには、付与対象者がユーザ、役割、グループのいずれであるかが表示されます。述部がない場合、Adaptive Server から NULL が返されます。

set show\_permission\_source, {on|off} の使用時は、以下について考慮が必要です。

- 階層に含まれる役割にパーミッションを付与する場合はパーミッションがユーザに直接付与されるわけではありません。たとえば、role1 のパーミッションを role2 に付与され、それが次に role3 に付与され、そのパーミッションがユーザに付与されます。role3 ではなく role1 にパーミッションが付与されているため、set show\_permission\_source では grantee カラムに role3 ではなく、"role1" が表示されます。role3 に関する情報を表示するには、sp\_displayroles ... expand\_up を使用します。



- 複数の付与者が特定のアクションのパーミッションを付与すると、`set show_permission_source` には、最も上位のユーザ ID を持つ付与者に関連付けられたパーミッションが表示されます。
- 同一のオブジェクトに対して、複数の付与者による複数のカラムに関するパーミッションに対応する複数のローが存在する可能性があるため、`set show_permission_source` にはカラム・レベルではなくオブジェクト・レベルの情報が表示されます。
- 特定のアクションと付与者の組み合わせごとに、1 つのオブジェクトに対して複数の述部があるパーミッションが複数存在する場合、`set show_permission_source` はすべての述部名を述部ごとに 1 ローずつ使用して表示します。

#### `set show_sqltext {on | off}`

アドホック・クエリ、ストアド・プロシージャ、カーソル、動的準備文の SQL テキストを出力できます。

`set showplan on` などのコマンドで行うように、クエリを実行して SQL セッションの診断情報を収集する前に、`set show_sqltext` を有効にする必要はありません。その代わりに、各コマンドの実行中にこのコマンドを有効にして、どのクエリが適切に実行されていないかを判断し、その問題を診断できます。

`show_sqltext` を有効にする前に `dbcc traceon` を有効にして、次のように標準出力への出力を表示します。

```
dbcc traceon(3604)
```

`set show_sqltext` の構文は、次のとおりです。

```
set show_sqltext {on | off}
```

たとえば、次の例は `show_sqltext` を有効にします。

```
set show_sqltext on
```

`set show_sqltext` が有効になると、入力した各コマンドとシステム・プロシージャに対するすべての SQL テキストが標準出力に出力されます。実行するコマンドまたはシステム・プロシージャに応じて、この出力は長くなる場合があります。

`show_sqltext` を無効にするには、次のように入力します。

```
set show_sqltext off
```

**set statement\_cache on | off**

デフォルトでは **on** になっています。**statement\_cache** のサーバ・レベルの設定が **on** である場合、このオプションでその機能の使用を有効または無効にします。サーバ・レベルの設定が **off** である場合、この設定は無効です。

**set show\_transformed\_sql, {on|off}**

Adaptive Server のクエリ処理時にクエリの中間フォームを **SQL** テキスト形式で表示します。これは、ビュー、述部付き権限、暗号化などのクエリの変形の実行後で、サブクエリのクエリの変形前のフォームです。

**set statistics {io, subquerycache, time, plancost, simulate} {on | off}**

さまざまな統計情報を表示します。

- **io** — 文で参照されたテーブルごとに、統計情報を表示します。
  - テーブルのアクセス回数 ( スキャン回数 )
  - 論理読み込み数 ( メモリ内でアクセスされたページ )
  - 物理読み込み数 ( データベース・デバイス・アクセス )

**statistics io** は、書き込みのあったバッファの数をコマンド別に表示します。

Adaptive Server がリソースの制限を実施するように設定されている場合は、**statistics io** でも **I/O** コストの合計が表示されます。

- **subquerycache** — サブクエリごとのキャッシュ・ヒット数、キャッシュ・ミス数、およびサブクエリ・キャッシュ内のローの数を表示します。
- **time** — Adaptive Server による各コマンドの解析とコンパイルの所要時間を表示します。**statistics time** は、コマンドの各ステップについて、そのコマンドの実行に Adaptive Server が要した時間を表示します。時間はミリ秒とタイムチックで示されます。その正確な値はマシンに依存します。
- **plancost** — クエリ統計をツリー形式で表示します。

---

**注意** **set statistics plancost** を有効にすると、Adaptive Server では **lio**、**pio**、および **row** の名前が、それぞれ **l**、**p**、および **r** に省略されます。

---

- **simulate** — オプティマイザが、クエリを最適化するためにシミュレートした統計を使用することを指定します。

『パフォーマンス&チューニング・シリーズ：統計的分析によるパフォーマンスの向上』の「set statistics コマンドの使用」を参照してください。

**set strict\_dtm\_enforcement { on | off }**

サーバが、Adaptive Server トランザクション調整サービスをサポートしないサーバにトランザクションを伝達するかどうかを指定します。デフォルト値は、strict dtm enforcement 設定パラメータの値から継承されます。

**set string\_truncation { on | off }**

insert または update コマンドによって char、unichar、varchar、または univarchar 文字列がトランケートされたときに、Adaptive Server が SQLSTATE 例外を出すかどうかを指定します。トランケートされた文字がスペースだけからなる場合、例外は出されません。デフォルト設定の off では SQLSTATE 例外は出されず、文字列は暗黙的にトランケートされます。

**set system\_view {instance | cluster | clear}**

(クラスタのみ)セッションのシステム・ビューを指定し、sp\_who などのストアド・プロシージャの出力に影響する偽のテーブルのメテリアライズを制御します。

- **instance** — 現在のインスタンスのシステム・ビューを設定します。
- **cluster** — クラスタのシステム・ビューを設定します。
- **clear** — セッションレベルの設定がすべてクリアされ、その spid をホストしている論理クラスタの system\_view 設定に戻ります。現在の値を確認するには、「select @@system\_view」と入力します。

**set switch [serverwide] {on | off} trace\_flag[,trace\_flag] [,with option [, option]>**

トレース・フラグおよびスイッチ名をローカルに設定することも、サーバワイドに設定することもできます。

- **serverwide** — サーバワイドにスイッチを ON または OFF に設定します(省略可能)。デフォルトは、セッションによって異なります。
- **on** — トレース・フラグがオンに設定されます。
- **off** — トレース・フラグがオフに設定されます。
- **trace\_flag** — 数字列(古いトレースフラグの番号)かスイッチ名、またはその両方。

- **option** — スイッチ・オプションの文字列 (省略可能)。有効な値は次のとおりです。
  - **override** — ドキュメント化されていないスイッチ名またはトレース・フラグを有効にするには、このオプションが必要である。
  - **no\_info** — このオプションは、情報や警告を非表示にする場合に使用する。

**set switch** のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

細密なパーミッションが有効化されている場合は、次のようになる。

- トレースフラグ 3604/3605 を設定するには、**monitor qp performance** 権限または **set switch** 権限が必要。

その他のすべてのトレースフラグについては、**set switch** 権限が必要。

細密なパーミッションが無効

細密なパーミッションが無効の場合、**sa\_role** が付与されていることが必要。

#### **set textsize** [*number*]

**select** 文によって返される **text**、**unitext**、または **image** 型のデータの最大サイズ (バイト) を指定します。@@**textsize** グローバル変数には、現在の設定が格納されます。

**isql** のデフォルト設定は 32K です。一部のクライアント・ソフトウェアでは、32K 以外のデフォルト値が設定されています。**textsize** をデフォルト・サイズ (32K) にリセットするには、次のコマンドを使用します。

```
set textsize 0
```

#### **set tracefile** [*filename*] [*off*] [*for spid*]

有効にすると、現在のセッションの **SQL** テキストすべてを指定のファイルに保存します。各 **SQL** テキストのバッチは、以前のバッチに追加されます。

トレースを有効にする構文は次のとおりです。

```
set tracefile file_name [off] [for spid]
```

トレースを無効にする構文は次のとおりです。

`set tracefile off [for spid]`

各パラメータの意味は次のとおりです。

- *file\_name* – SQL テキストの保存先ファイルのフル・パス。ディレクトリ・パスを指定しない場合は、*\$SYBASE* 内にファイルが作成される。

---

**注意** *file\_name* に英数字以外の特殊文字 (":", "/" など) が含まれている場合は、*file\_name* を引用符で囲んでください。たとえば、次のディレクトリ構造の場合は “/” が含まれているので、この *file\_name* を引用符で囲む必要があります。

```
set tracefile '/tmp/mytracefile.txt' for 25
```

*file\_name* に特殊文字が含まれておらず、ファイルを *\$SYBASE* に保存する場合は、引用符で囲む必要はありません。たとえば、次の *file\_name* は引用符で囲む必要はありません。

```
set tracefile mytracefile.txt
```

---

- `off` – このセッションまたは *spid* のトレースを無効にします。
- *spid* – SQL テキストをトレース・ファイルに保存するサーバ・プロセス ID。sa またはシステム・セキュリティ担当者の役割を持つユーザだけが、他の *spid* のトレースを有効にできます。システム・タスク (ハウスキーピングやポート・マネージャなど) の SQL テキストを保存することはできません。

---

**注意** 特定のセッションで `set tracefile` を使用すると、後続のすべての `set` コマンドまたは `DBCC` トレースの診断出力は、トレース・ファイルにリダイレクトされます。

`set tracefile off` を発行する前に、オンにしたすべての診断コマンドをオフに切り替えてください。これを行わないと、トレース・ファイルに出力するはずの内容がクライアントに送信されます。

---

`set tracefile` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッション  
が有効

細密なパーミッションが有効の場合、所有するセッションのトレースファイルを設定するには `set tracing` 権限または `monitor qp performance` 権限が必要。他のユーザのセッションにトレースファイルを設定するには `set tracing any process` 権限または `monitor qp performance` 権限が必要。

---

細密なパーミッションが無効

細密なパーミッションが無効の場合、所有するセッションのトレースファイルを設定するには **set tracing** 権限が必要、他のユーザのセッションにトレースファイルを設定するには、**sa\_role** または **sso\_role** が必要。

**set transaction isolation level** {[read uncommitted | 0] | [read committed | 1] | [repeatable read | 2] | [serializable | 3]}

セッションのトランザクション独立性レベルを設定します。このオプションを設定すると、現在または今後のトランザクションは、この独立性レベルで動作します。

- **read uncommitted | 0** — 独立性レベル 0 でのスキャンでは、ロックは取得されません。したがって、レベル 0 のスキャンの結果セットは、スキャンの進行中に変更される場合があります。基本となるテーブルのデータが変更されたためにスキャン位置が不明になった場合は、スキャンの再実行にはユニーク・インデックスが必要です。ユニーク・インデックスを作成しておかないと、スキャンがアボートする場合があります。

デフォルトでは、読み込み専用データベース上にないテーブルを独立性レベル 0 でスキャンする場合、ユニーク・インデックスが必要です。この要件を無効にするには、次のように、**Adaptive Server** がユニークでないインデックスまたはテーブル・スキャンを選択するように設定します。

```
select * from table_name (index table_name)
```

基本となるテーブルのアクティビティによってはスキャンが完了する前に、アボートする場合があります。

- **read committed | 1** — デフォルトでは、**Adaptive Server** のトランザクションの独立性レベルは **read committed** または 1 であり、データに対する共有読み込みロックが許可されます。
- **repeatable read | 2** — 繰り返し不可能読み出しを禁止します。
- **serializable | 3** — 独立性レベル 3 を指定すると、**Adaptive Server** では、トランザクション内のすべての **select** および **readtext** オペレーションに **holdlock** が適用されます。これにより、トランザクションの最後までクエリの読み込みロックが設定されます。連鎖モードも設定すると、この独立性レベルは、暗黙的にトランザクションを開始するすべてのデータ検索またはデータ修正文に影響を持ち続けます。

```
set transactional_rpc { on | off }
```

リモート・プロシージャ・コールの処理を制御します。このオプションを on に設定すると、トランザクションが保留中の場合、RPC は Adaptive Server によって調整されます。このオプションを off に設定すると、リモート・プロシージャ・コールは、Adaptive Server のサイト・ハンドラによって処理されます。デフォルト値は、enable xact coordination 設定パラメータの値から継承されます。

例

**例 1** Adaptive Server に対し、等式 (=) と不等式 (!=) の比較演算子、および集合関数の NULL 値のオペランドを初級レベルの ANSI SQL 規格に従って評価するよう指示します。

```
set ansinull on
```

set ansinull on を使用すると、集合関数およびローの集合関数は、Adaptive Server が 1 つ以上のカラムやローで null 値を検出すると、次の SQLSTATE 警告を出します。

```
Warning - null value eliminated in set function
```

等式オペランドまたは不等式オペランドのいずれかの値が NULL の場合、比較結果は UNKNOWN になります。たとえば、ansinull モードで次のクエリを実行しても、ローは返されません。

```
select * from titles where price = null
```

set ansinull off を使用する場合、同じクエリを実行すると price が NULL のローが返されます。

**例 2** 文字セット変換をアクティブ化して、クライアントで使用されている文字セットに基づいて文字セット変換のデフォルトを設定します。クライアントの文字セットへ変換できない文字があると、Adaptive Server からクライアントまたはアプリケーションにその旨通知されます。

```
set char_convert on with error
```

**例 3** デフォルトで、アウトバウンド RPC 要求をコンポーネント統合サービスで処理することを指定します。

```
set cis_rpc_handling on
```

**例 4** このユーザにクライアント名 alison、ホスト名 money1、およびアプリケーション名 webserver2 を割り当てます。

```
set clientname 'alison'
set clienthostname 'money1'
set clientapplname 'webserver2'
```

**例 5** `test_cursor` を使用して、クライアントによって要求された後続の各 `fetch` 文に対して、5 つのローを返します。

```
set cursor rows 5 for test_cursor
```

**例 6** システム・プロシージャまたはトリガによって設定されたセッション設定をそのセッション中保持するように Adaptive Server に指示します。

```
set export_options on
```

`set export_options` を無効にして Adaptive Server をデフォルト動作に戻すには、次のコマンドを使用します。

```
set export_options off
```

これらの最適化設定は、`set export_options on` を使用してエクスポートできます。

---

**注意** デフォルトでは、`set export_options` はログイン・トリガで有効です。

---

**例 7** Transact-SQL 拡張機能を使用している場合、Adaptive Server に警告メッセージを表示するよう指示します。

```
set fipsflagger on
```

次のように標準でない SQL を使用したとします。

```
use pubs2
go
```

Adaptive Server は次のメッセージを表示します。

```
SQL statement on line number 1 contains Non-ANSI text.
The error is caused due to the use of use database.
```

**例 8** 値 100 が `stores_south` テーブルの `IDENTITY` カラムに挿入され、今後このカラムに対する明示的な挿入が禁止されます。`syb_identity` キーワードが使用されていることに注意してください。Adaptive Server では、このキーワードが `IDENTITY` カラムの名前によって置き換えられます。

```
set identity_insert stores_south on
go
insert stores_south (syb_identity)
values (100)
go
set identity_insert stores_south off
go
```



**例 9** `identity_update` を有効にし、テーブルを値 1 および 10 でそれぞれ更新して、`identity_update` を無効にします。

```
set identity_update t1 on
update t1 set c2 = 10 where c1 =1
select * from t1
c1                c2
-----          -
1                  10
```

```
set identity_update t1 off
```

**例 10** セッションまたはストアド・プロシージャの次のコマンドは、要求されたロックをすぐに取得できないと、エラーを返して失敗します。

```
set lock nowait
```

**例 11** 現在のセッションまたはストアド・プロシージャの次のコマンドは、ロックを取得するまで無制限に待機します。

```
set lock wait
```

**例 12** セッションまたはストアド・プロシージャの後続のコマンドは、ロックを取得するまでに 5 秒間待機した後、エラー・メッセージを生成して失敗します。

```
set lock wait 5
```

**例 13** `dev_plans` グループへの抽象プランの取得を有効にします。

```
set plan dump dev_plans on
```

**例 14** 現在のセッションのクエリで、`dev_plans` グループからの抽象プランのロードを有効にします。

```
set plan load dev_plans on
```

**例 15** パラメータ情報を出力しません。

```
11> create procedure sp_pout (@x int output) as select
    @x = @x + 1
2> go

1> set proc_output_params off
2> go
1> declare @x int
2> select @x = 1
3> exec sp_pout @x output
4> print "Value of @x returned from sproc is:%1!", @x
5> go
```

```
(1 row affected)
(return status = 0)
```

Value of @x returned from sproc is:1

set proc\_output\_params off を実行しないと、(return status = 0) の後の出力は次のようになります。

Return parameters:

```
-----
                2
```

**例 16** パラメータとリターン・ステータス TDS トークンをいずれも出力しません。

```
set proc_output_params OFF
go

set proc_return_status OFF
go

declare @x int
select @x = 2
exec sp_pout @x output
print "Value of @x returned from sproc is:%1!", @x
go

(1 row affected)
Value of @x returned from sproc is:2
(1 row affected)
```

さらに、set nocount on を使用してからこのバッチを実行すると、影響を受けるローの数をレポートする行を表示せず、余分なメッセージのない出力を生成できます。

**例 17** このコマンドを実行するユーザは、サーバ内ではログイン "mary"、および Mary のサーバ・ユーザ ID として操作できるようになります。

```
set proxy "mary"
```

**例 18** insert、update、delete、select の文のそれぞれに対して、Adaptive Server は、最初の 4 ロウを処理してから、クエリの処理を停止します。次に例を示します。

```
select title_id, price from titles
title_id price
```

```

-----
BU1032          19.99
BU1111          11.95
BU2075          2.99
BU7832          19.99

```

```
(4 rows affected)
```

```
set rowcount 4
```

**例 19** Adaptive Server に対して、二重引用符で囲まれた文字列をすべて識別子として処理するように指示します。quoted\_identifier が on の場合、テーブル名 “!\*&strange\_table” とカラム名 “emp's\_name” は有効な識別子名です。

```

set quoted_identifier on
go
create table "!*&strange_table"
    ("emp's_name" char(10), age int)
go
set quoted_identifier off
go

```

**例 20** カッコで囲まれた文字列を識別子として扱います。quoted\_identifier が off であっても、テーブル名 [!\*&strange\_table] とカラム名 [emp's\_name] は、どちらも角カッコで囲まれているので、有効な識別子名です。

```

set quoted_identifier off
go
create table [!*&strange_table]
    ([emp' s_name] char (10), age int)
go

```

角カッコ識別子の詳細については、「[区切り識別子](#)」(759 ページ)を参照してください。

**例 21** 役割 "doctor" をアクティブにします。このコマンドは、ユーザがアクティブにする役割を指定するときに使用します。

```
set role doctor_role on
```

**例 22** 現在のセッションで、ユーザに割り当てられているシステム管理者の役割を非アクティブにします。

```
set role "sa_role" off
```

**例 23** ユーザがパスワードを入力すると、役割 "doctor" がアクティブになります。

```
set role doctor_role with passwd "physician" on
```

**例 24** 役割 "doctor" を非アクティブにします。

```
set role doctor_role off
```

**例 25** 非分割テーブルにおける並列インデックス・スキャンと、並列テーブル・スキャンの最大並列処理を 4 と指定します。

```
set scan_parallel_degree 4
```

**例 26** 上記の例 5 と同じです。

```
set session authorization "mary"
```

**例 27** 各クエリについて処理プランの説明が返されますが、クエリは実行されません。

```
set showplan, noexec on
go
select * from publishers
go
```

**例 28** クエリの統計をツリー形式で表示します。

```
set statistics plancost on
```

```
select * from authors
```

au_id	au_lname	au_fname	phone	address
city	state	country	postalcode	
ite	Johnson	408 496-7223	10932 Bigge Rd.	172-32-1176 Wh
Menlo Park	CA	USA	94025	
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411
Oakland	CA	USA	94618	
. . . .				
998-72-3567	Ringer	Albert	801 826-0752	67 Seventh Av.
Salt Lake City	UT	USA	84152	

```
===== Lava Operator Tree =====
```

```
Emit
(VA = 1)
23 rows est:23
cpu:0
```

```

/
TableScan
authors
  (VA = 0)
23 rows est:23
lio:1 est:2
pio:0 est:2

```

```
=====
```

(23 rows affected)

**例 29** char、unichar、または nchar 文字列がトランケートされるときに、Adaptive Server に例外を生成させます。

```
set string_rtruncation on
```

insert または update 文によって文字列がトランケートされると、Adaptive Server から次のメッセージが表示されます。

```
string data, right truncation
```

**例 30** select 文によって返される text、unitext、または image データのサイズを 100 バイトに制限します。

```
set textsize 100
```

**例 31** トレース・フラグをドキュメント化されていないトレース・フラグ 110 に設定するサーバワイド・スイッチをオンに設定します。追加情報や警告は表示しません。

```
set switch serverwide on 110 with override, no_info
```

**例 32** 現在のセッションに対して *sql\_text\_file* という名前のトレース・ファイルを開きます。

```
set tracefile '/var/sybase/REL1502/text_dir/sql_text_file'
```

set showplan、set statistics io、dbcc traceon(100) からの後続出力は、*sql\_text\_file* に保存されます。

**例 33** ディレクトリ・パスは指定しないので、*\$\$SYBASE/sql\_text\_file* 内にトレース・ファイルが保存されます。

```
set tracefile 'sql_text_file' for 11
```

spid 11 で実行される SQL は、このトレース・ファイルに保存されます。

**例 34** spid 86 の SQL テキストを保存します。

```
set tracefile '/var/sybase/REL1502/text_dir/sql_text_file' for 86
```

**例 35** トランザクションが保留中のとき、Adaptive Server サイト・ハンドラではなく、コンポーネント統合サービスのアクセス・メソッドで RPC を処理することを指定します。

```
set transactional_rpc on
```

**例 36** セッション内の後続のクエリはすべて、繰り返し読み出しトランザクション独立性レベルで実行されます。

```
set transaction isolation level 2
```

**例 37** トランザクションの `select` 文のそれぞれによって、そのトランザクション実行中に保持される読み込みロックが実装されます。

```
set transaction isolation level 3
```

**例 38** 次の例では、トランザクションの開始時にテーブルが最低限のロギングで実行されていた場合、同じテーブルで複数の文の DML ロギング・モードがそのままになることを示します。

- 1 トランザクションを開始し、DML ロギングを最低限に設定します。

```
begin tran
set dml_logging minimal
```

- 2 `insert` コマンドを実行します。

```
insert into tab1 values(1)
```

- 3 DML ロギングをデフォルト設定に戻します。

```
set dml_logging default
```

DML ロギングはデフォルトにリセットされますが、`t1` は前にこのトランザクションで最低限のロギングで実行されたため、次の `insert` は最低限のロギングで実行されます。

```
insert into tab1 values(1)
```

エラー・ログには、ロギング・モードの選択が上書きされた理由が示されます。

**例 39** 次の例では、同じセッション内で `show_exec_info` を `minimal` から `full` に変更します。

- 1 Adaptive Server へログインします。

```
isql -Ubob -Pbob123
use myimdb
```

- 2 テーブル `tab1` を作成します。

```
create table tab1(coll int)
```

- 3 `show_exec_info` を有効にし、DML ロギングを `minimal` に設定します。

```
set show_exec_info on
set dml_login minimal
```

- 4 `tab1` に値を挿入します。

```
insert into tab1 values(1)
```

- 5 Adaptive Server は、テーブルとデータベースの名前、コマンドを実行している ユーザ ID、および使用されるロギング・モードを表示します。

```
Operating on the table 'tab1', database 'myimdb' (owner ID 3) in
'minimal' logging mode by user ID 3.
```

- 6 DML ロギングをデフォルト設定に戻します。

```
set dml_logging default
```

- 7 `tab1` に追加の値を挿入します。

```
insert into tab1 values(1)
```

- 8 Adaptive Server は、テーブルとデータベースの名前、コマンドを実行している ユーザ ID、および使用されるロギング・モードを表示します。

```
Operating on the table 'tab1', database 'myimdb' (owner ID 3) in 'full'
logging mode by user ID 3.
```

## 使用法

`fipsflagger`、`string_rtruncation`、`ansinull`、`ansi_permissions`、`arithabort`、`arithignore` によって、Adaptive Server のエラー処理の内容や、SQL 規格への対応が変わります。

- `cis_rpc_handling` オプションと `transactional_rpc` オプションは、コンポーネント統合サービス (CIS) が有効な場合にのみ使用できます。
- `async log service` オプションと `delayed_commit` は同時に指定できません。`async log service` が “true” に設定されている場合、`delayed_commit` は動作しません。
- Adaptive Server に並列処理が設定されている場合には、`parallel_degree` と `scan_parallel_degree` によってクエリの並列度が制限されます。これらのオプションを使用すると、オペティマイザに対し、設定パラメータの指定値よりも少ない数のワーカー・プロセスを並列クエリで使用するよう指示が出されます。これらのパラメータを 0 に設定すると、サーバワイドの設定値がリストアされます。

設定パラメータで指定された数より大きい値を指定すると、**Adaptive Server** は警告メッセージを表示し、設定パラメータで指定した数を使用します。

- トリガやストアド・プロシージャ内で **set** コマンドを使うと、トリガやプロシージャの実行後にほとんどの **set** オプションが元の設定に復元されます。

次のオプションは、プロシージャまたはトリガの実行後もその以前の設定には復元されず、**Adaptive Server** セッション全体を通して、または明示的にリセットされるまでそのままです。

- **datefirst**
- **dateformat**
- **identity\_insert**
- **language**
- **quoted\_identifier**
- 複数の **set** オプションを指定した場合、最初に発生した構文エラーが原因で、後続のオプションはすべて無視されます。ただし、エラーの発生位置より前に指定されていたオプションは実行され、新しいオプション値が設定されます。
- ユーザにクライアント名、ホスト名、またはアプリケーション名を割り当てる場合、これらの割り当ては現在のセッションの間だけアクティブになります。次にユーザがログインするときにこれらの名前を再び割り当てる必要があります。新しい名前は **sysprocesses** に表示されますが、パーミッション検査には使用されず、**sp\_who** では、元のログインに属するものとして引き続きクライアント接続が表示されます。ユーザ・プロセスの設定の詳細については、『システム管理ガイド』を参照してください。
- **showplan** と **char\_convert** を除くすべての **set** オプションは、ただちに有効になります。**showplan** は、次のバッチ処理で有効になります。以下に **set showplan on** を使用した例を2つ示します。

```
set showplan on
select * from publishers
go
```

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)

しかし、

```
set showplan on
go
select * from publishers
go
QUERY PLAN FOR STATEMENT 1 (at line 1).
STEP 1
    The type of query is SELECT

    FROM TABLE
        publishers
    Nested iteration
    Table Scan
    Ascending Scan.
    Positioning at start of table.
```

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)

- Adaptive Server は1つ以上のスペースを `clientname`、`clienthostname`、`clientappname` カラムに自動的に格納します。したがって、これら3つのカラムのいずれかに対して "is null" を使用するクエリでは、予期する結果セットが返されません。
- `set fipsflagger` オプションが有効になっているときに `set proxy` を発行すると、次の警告が発行されます。
 

```
SQL statement on line number 1 contains Non-ANSI
text.The error is caused due to the use of DBCC.
```
- ログイン・トリガを使用して現在の実行プロパティを設定した場合、ログイン・トリガ内で有効または無効にしたエクスポート可能な `set` オプションは、現在の処理に反映されます。
- 一部の `set` オプションは、次のように分類できます。

- `parseonly`、`noexec`、`prefetch`、`showplan`、`rowcount`、`nocount` は、クエリの実行方法を制御する。`parseonly` と `noexec` の両方をオンに設定しても意味はない。`rowcount` のデフォルト設定は 0 である (すべてのローが返される)。それ以外のオプションのデフォルト値は `off` である。
- `statistics` オプションは、各クエリの後でパフォーマンス統計を表示する。`statistics` オプションのデフォルト設定は `off` である。`noexec`、`prefetch`、`showplan`、`statistics` の詳細については、『パフォーマンス&チューニング・ガイド』を参照してください。
- `set` 句では、サブクエリから返されたリテラル、変数、または式を使用して、最大 1024 のカラムを更新できる。
- `offsets` は、Adaptive Server から返された結果を解釈するため、DB-Library で使用される。このオプションのデフォルト設定は `on` である。
- `datefirst`、`dateformat`、`language` は、日付関数、日付順、メッセージ表示に作用する。トリガまたはストアド・プロシージャ内でこれらのオプションを使用した場合、その以前の設定は復元されない。

デフォルト言語である `us_english` では、`datefirst` は 1 (日曜)、`dateformat` は `mdy` であり、メッセージは `us_english` で表示される。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 となり、他の言語では月曜日が 1、火曜日が 2 となる。

`set language` では、Adaptive Server は指定される言語での最初の曜日と日付フォーマットを使用する必要があるが、現在のセッション内ですでに実行されている、明示的な `set datefirst` または `set dateformat` コマンドがこのコマンドによって上書きされることはない。

- `cursor rows` と `close on endtran` は、Adaptive Server によるカーソルの処理方法に影響する。すべてのカーソルで `cursor rows` のデフォルト設定は 1 である。`close on endtran` のデフォルト設定は `off` である。
- `chained` と `transaction isolation level` により、Adaptive Server は SQL 標準に準拠した方法でトランザクションを扱う。

#### コンパイル時の set パラメータの変更点

バージョン 15.0.2 以降の Adaptive Server では、一部の抽象プランの `set` パラメータで、ストアド・プロシージャの作成や、Transact-SQL バッチの実行に使用する際のコンパイル時間の動作が変更されています。

Adaptive Server の以前のバージョンでは、`set` パラメータは、ストアド・プロシージャの実行後または再コンパイル後に有効になっていました。Adaptive Server 15.0.2 では、コンパイル時に `set` オプティマイザ・パラメータを使用することによって、ストアド・プロシージャやバッチ内のオプティマイザに影響を与えることができます。

**注意** この動作変更は、結果セットの構成に影響を与える場合があります。15.0.2 バージョンの `set` パラメータで生成された結果セットを見直してから、運用システムでそれらを使用するようにおすすめします。

ストアド・プロシージャから戻る前に、`set` パラメータをリセットしてください。そうしないと、その後のストアド・プロシージャの実行に影響を与える可能性があります。この変更を後続のストアド・プロシージャに反映させるには、`export_options` パラメータを使用します。

表 1-31 は、`set export_options on` を使用したときにエクスポートできるオプティマイザのオプションを示しています。

表 1-31: `set export_options on` によりエクスポートできるオプティマイザのオプション

<code>optgoal</code>	<code>store_index</code>	<code>showmanagers</code>
<code>opttimeout</code>	<code>bushy_space_search</code>	<code>showlogprops</code>
<code>merge_join</code>	<code>parallel_query</code>	<code>showparallel</code>
<code>hash_join</code>	<code>replicated_partitioning</code>	<code>showhistograms</code>
<code>nl_join</code>	<code>basic_optimization</code>	<code>showabstractplan</code>
<code>distinct_sorted</code>	<code>index_intersection</code>	<code>showsearchengine</code>
<code>distinct_sorting</code>	<code>index_union</code>	<code>showcounters</code>
<code>distinct_hashing</code>	<code>multi_gt_store_index</code>	<code>showbestplan</code>
<code>group_sorted</code>	<code>opportunistic_grouping</code>	<code>showfinalplan</code>
<code>group_hashing</code>	<code>opportunistic_distinct</code>	<code>showcodegen</code>
<code>group_inserting</code>	<code>auto_query_tuning</code>	<code>showpiocosting</code>
<code>order_sorting</code>	<code>streaming_sort</code>	<code>showliocosting</code>
<code>addend_union_all</code>	<code>nary_nl_join</code>	<code>showelimination</code>
<code>merge_union_all</code>	<code>query_tuning_mem_limit</code>	<code>showpllcosting</code>
<code>merge_union_distinct</code>	<code>query_tuning_time_limit</code>	<code>shownostats</code>
<code>hash_union_distinct</code>	<code>showlop</code>	<code>showexecio</code>

### 集合動作

`ansinull` は、集合関数での `NULL` 値オペランドの評価が `ANSI SQL` 規格に準拠しているかどうかを判断します。`set ansinull on` を使用すると、集合関数が `null` 値オペランドを計算から削除した場合に `Adaptive Server` で警告が生成されます。

たとえば、`titles` テーブルに対して、`set ansinull off` (デフォルト値) を指定して次のクエリを実行するとします。

```
select avg (total_sales) from titles
```

`Adaptive Server` は次の値を返します。

```
-----  
6090
```

ただし、同じクエリを `set ansinull on` を使用して実行すると、`Adaptive Server` は次の出力を返します。

```
1> use pubs2  
2> go  
1> select avg (total_sales) from titles  
2> go  
  
-----  
6090  
(1 row affected)  
  
1>set ansinull on  
2> go  
1> select avg (total_sales) from titles  
2> go  
  
-----  
6090  
Warning - null value eliminated in set function  
(1 row affected)
```

このメッセージは、`total_sales` の一部のエントリに実際の金額ではなく `NULL` が含まれているため、このテーブルのすべての本の総売上げのデータが完全ではないことを示します。ただし、使用可能なデータの最大値が返されます。

### 比較動作

SQL 標準では、等式比較の 2 つのオペランドのいずれか一方が null である場合、結果は UNKNOWN になります。Transact-SQL では、null 値の扱いが異なります。一方のオペランドがカラム、パラメータ、または変数であり、もう一方のオペランドが NULL 定数、または値が NULL のパラメータまたは変数である場合、結果は TRUE または FALSE になります。

- Sybase NULL モード – “val” が NULL の場合、“val = NULL” は true
- ANSI NULL モード – “val” が NULL である場合 “val = NULL” は unknown

where 句および on 句の ANSI 規則では、true であるローが返され、false でありかつ unknown であるローは拒否されます。

check 定数の ANSI 規則では、false である値は拒否されます。このため、unknown や true の結果は拒否されません。

状況に応じて、次のようにします。

- ansinull モードを有効にする場合 – Sybase NULL 比較 (val = NULL または val != NULL) は使用しないでください。
- insert および update で ANSI-null モードの使用が予定される場合 – check 制約では Sybase null 比較を使用しないでください。

代わりに、ANSI IS NULL 構文または IS NOT NULL 構文を使用して、予期しない結果を招かないようにしてください。

### 区切り識別子

文の構文で識別子を含む文字列を引用符で囲む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。次に例を示します。

```
set quoted_identifier on
create table "lone" (c1 int)
```

ただし、object\_id には文字列が必要であるため、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記することが要求されている場合に、引用符を2つ使用する必要はありません。

```
select object_id ('embedded"quote')
```

**角カッコ識別子** Adaptive Server では、引用符識別子の代わりとして、角カッコで識別子を囲む書式も使用できます。角カッコ識別子の動作は、この識別子を使用するために `set quoted_identifier on` を使用する必要がない点を除けば、引用符付き識別子と同じです。

引用符識別子の代わりに角カッコ識別子を使用してオブジェクトを作成するときは、次のような有効な文字を少なくとも1つオブジェクト名に含める必要があります。

```
create table [table name]
```

```
create database [database name]
```

オブジェクト名からは末尾のスペースが削除されるので、次の名前はすべて同じ名前として扱われます。

```
[tabl<space><space>]
[tabl<space><space>]
[tabl]
[tabl<space><space><space>]
tabl
```

これは、角カッコ識別子を使用して作成できるオブジェクトに適用されます。

Adaptive Server で区切り識別子を使用するときの制限事項を次に示します。

- 区切られている場合でも、識別子の名前にドット (.) を含めることはできません。
- オブジェクト名をストアド・プロシージャ・パラメータとして使用する場合、Adaptive Server のストアド・プロシージャ・オブジェクト名は文字列として扱われるため、デリミタは不要です。たとえば、テーブル名 `table` が実際に存在する場合、次の指定は正しい結果になります。

```
exec sp_help 'dbo.table'
```

ただし、次のように指定した場合に角カッコはオブジェクト名から除外されません。

```
exec sp_help 'dbo.[table]'
```

### 役割と set オプション

- ユーザが **Adaptive Server** にログインすると、ユーザに付与されているすべてのシステム定義役割が自動的にアクティブになります。ユーザに付与されているユーザ定義役割は自動的にアクティブになりません。割り当てられたユーザ定義の役割を自動的にアクティブ化するには、**sp\_modifylogin** を使用します。詳細については、『リファレンス・マニュアル：プロシージャ』の **sp\_modifylogin** を参照してください。役割のオンとオフを切り替えるには、**set role role\_name on** または **set role role\_name off** を使用します。

たとえば、システム管理者の役割を付与された場合は、現在のデータベース内のデータベース所有者の **ID** (およびユーザ **ID**) を想定します。実際のユーザ **ID** を想定するには、次のコマンドを実行します。

```
set role "sa_role" off
```

現在のデータベースのユーザではなく、**"guest"** ユーザがない場合は、**sa\_role off** を設定できません。

- 有効にしたいユーザ定義役割にパスワードが設定されている場合、その役割をオンにするにはパスワードを指定する必要があります。次のように入力します。
 

```
set role "role_name" with passwd "password" on
```
- **set role** の際に、役割の有効化の失敗回数が **max failed logins** で設定した回数に達すると、**Adaptive Server** によってその役割がロックされます。このような状況が発生した場合は、**locksuid**、**locdate**、**lockreason** は **sysssrvroles** で更新されます。

### インメモリ・データベースおよびリラックス持続性データベース

- ログギング・レベルを **minimal** に設定した場合、現在のユーザが所有するオブジェクトのログギング・モードにのみ影響します。ただし、ユーザがシステム管理者権限を持っている場合は、ログギングを **minimal** に設定すると、ユーザのセッション内のすべてのオブジェクトのログギング・モードに影響します。
- **select into** で作成されたテーブルにコピーされたデータについては、最低限のログを取ります。**with dml\_logging = minimal** を使用すると、このテーブルでの今後の **DML** オペレーションのログギング・モードが指定されます。
- **show\_exec\_info** では、ユーザが選択した最小限のログギング・モードを **Adaptive Server** が上書きした理由は表示されません。上書きの理由を表示するには、**set switch print\_minlogged\_mode\_override** を使用します。

- ログイン・モードのセッション固有の設定は、テーブルおよびデータベースのレベルで設定されたログイン・オプションを上書きしますが、次のような制限があります。
  - データベースワイドの設定
  - ログイン・モードに基づき、現在のセッションの DML ログインを無効にする
    - データベースワイドのログイン・モード設定
    - テーブル固有のログイン・モード設定
    - 更新対象のテーブルの所有権
  - セッション固有の DML ログインを `minimal` に設定して、`set dml_logging default` を実行すると、影響を受けるテーブルのログイン・モードは、テーブルおよびデータベースワイドの設定に応じたデフォルトのログイン・モードに戻ります。
  - データベースまたはテーブルの所有者が、テーブルが最低限のログインで実行されるようにすでに設定している場合は、`set dml_logging` を使用して完全なログを取る DML を実行することはできません。

#### セッションの圧縮レベルの設定

- セッションの圧縮の有効化または無効化は、既存データの圧縮レベルを変更しません。
- 更新されたローは非圧縮のままになります。Adaptive Server は、更新中にすべての圧縮ローを解凍します。
- `set compression` が `off` に設定されていると、データのコピーを必要とするコマンド (たとえば、`reorg rebuild` や `alter table`) はデータ・ローを解凍します。
- `set compression on` を設定すると、その後の更新ではパーティションまたはテーブルの圧縮レベルが使用されます。

#### 述部付きの権限の使用

- `set noexec` を有効にして `set show_transformed_sql` を使用しても、クエリは実行されず、SQL テキストのテキストが表示されます。
- セッション、DML、または `select` コマンドに対して `show_transformed_sql` を有効化すると、次の SQL テキストが表示されます。



- SQL コマンドのテーブル内に述部が存在する場合は、述部テキスト。述部が存在しない場合は、`show_transformed_sql` で `NULL` を表示します。
- ユーザ・クエリ・テキスト
- 集約処理、表示処理、暗号化、述部マージ後の SQL テキスト。述部にアクセスしないクエリは、SQL テキストで表示処理、暗号化などの後にテキストを表示します。
- サブクエリ処理後のクエリの SQL テキスト。

#### 分散トランザクション、コンポーネント統合サービス、`set` オプション

- ASTC の導入により、`cis rpc handling` 設定プロパティと `set transactional_rpc` コマンドの動作が変更されました。12.0 より前のバージョンでは、`cis rpc handling` を有効にすると、すべての RPC がコンポーネント統合サービスの Client-Library 接続経由で送信されていました。その結果、`cis rpc handling` が有効になっている場合には、`transactional_rpc` の動作は特に設定しなくても必ず発生していました。Adaptive Server 12.0 では、この動作が変更されています。`cis rpc handling` が有効で、`transactional_rpc` が `off` の場合、トランザクション内の RPC はサイト・ハンドラを介してルート指定されません。トランザクションの外部で実行された RPC は、コンポーネント統合サービスの Client-Library 接続経由で送信されます。
- Adaptive Server の分散トランザクション管理サービスが有効な場合は、トランザクション内に RPC を配置できます。これらの RPC は、「トランザクション指向 RPC」と呼ばれます。トランザクション指向 RPC とは、その作業を現在のトランザクションのコンテキストに含めることができる RPC です。このリモート作業単位は、ローカル・トランザクションによって実行される作業とともにコミット、またはロールバックすることができます。

トランザクション指向 RPC を使用するには、`sp_configure` を使用してコンポーネント統合サービスと分散トランザクション管理を有効にしてから、`set transactional_rpc` コマンドを発行します。`set transactional_rpc` が `on` で、トランザクションが保留中の場合は、Adaptive Server (Adaptive Server サイト・ハンドラではなく) によって RPC が調整されます。

`set transactional_rpc` コマンドのデフォルト設定は `off` です。`set cis_rpc_handling` コマンドは、`set transactional_rpc` コマンドを無効にします。`cis_rpc_handling on` を設定すると、すべてのアウトバウンド RPC がコンポーネント統合サービスによって処理されます。

- `set transactional_rpc`、`set cis_rpc_handling`、`sp_configure` の使用方法については、『コンポーネント統合サービス・ユーザーズ・ガイド』を参照してください。

## プロキシの使用

**注意** 明示的なパーミッションがないと、`"sa_role"` も `"sso_role"` も、`set proxy login_name` コマンドを発行できません。システム・セキュリティ担当者を含め、`set proxy login_name` のユーザには、システム・セキュリティ担当者によって明示的にパーミッションが付与されている必要があります。

- `set proxy` コマンドや `set session authorization` コマンドを使用するには、`master` データベースで `set proxy` 権限または `set session authorization` 権限が付与されている必要があります。
- サーバ・ユーザ ID を他のサーバ・ログインに切り替え、その使用をターゲット・ログインの役割に基づいて制限するには、次の構文を使用します。

```
grant set proxy to user_or_role_list  
[restrict role role_list | all | system]
```

詳細については [「grant」\(486 ページ\)](#) を参照してください。

- 元の `login_name` を指定した `set proxy` コマンドや `set session authorization` コマンドを実行すると、元のログイン ID に戻ります。
- `set proxy` と `set session authorization` を 1 つのトランザクション内で実行することはできません。
- Adaptive Server では、1 レベルでのみログイン ID を変更できます。したがって、`set proxy` または `set session authorization` を使用して ID を変更した場合、この ID を再度変更するには、変更した ID を元の ID に戻す必要があります。たとえば、ログイン名が `"ralph"` であるとし、`"mary"` としてテーブルを作成し、`"joe"` としてビューを作成してから、元のログイン ID に戻るには、次の文を使用します。

```
set proxy "mary"  
create table mary_sales  
  (stor_id char (4),  
   ord_num varchar (20),  
   date datetime)  
grant select on mary_sales to public  
set proxy "ralph"  
set proxy "joe"
```

```

create view joes_view (publisher, city,
state)
as select stor_id, ord_num, date
from mary_sales
set proxy "ralph"

```

- ユーザが `set proxy` を発行して、別のユーザのパーミッション、ログイン名、`suid` を代用する場合、Adaptive Server は、元のユーザのアクセスではなく、データベース・オブジェクトへのプロキシ・ユーザのアクセスをチェックします。

Adaptive Server は、ログインしているユーザの名前とパスワード情報を使用して、ログイン・クレデンシャルを使用した暗号化キーへの自動アクセスをチェックします。Adaptive Server にはプロキシ・ユーザのパスワードへのアクセス権がありません。ログイン・パスワードによるキーへのアクセスは、エイリアス、`set proxy`、または `setuser` によって想定されたユーザの代わりではなく、ログインしたユーザの代わりに行うものです。暗号化キーがログインへの関連付けのために設定され、現在もシステム暗号化パスワードまたはマスタ・キーによって暗号化されている場合、そのコピーへのアクセスも同様に処理されます。

#### lock wait の使用

- デフォルトにより、すぐにロックを取得できない Adaptive Server のタスクは、互換性のないロックが解放されるまで待機して、処理を継続します。これは、`set lock wait` で `numsecs` パラメータに値を指定しない場合と同じです。
- サーバワイドなロック待機期間は、`lock wait period` オプションを指定した `sp_configure` を使用して設定できます。
- セッションレベルの設定 `set lock wait nnn` が指定された `lock wait period` は、ユーザ定義テーブルにのみ適用されます。これらの設定は、システム・テーブルには影響しません。
- `set lock` によってセッション・レベルまたはストアド・プロシージャでロック待機期間を定義すると、サーバ・レベルのロック待機期間が無効になります。
- `set lock wait` を単独で使用し、`numsecs` に値を指定しないと、現在のセッションの後続コマンドはすべて、要求されたロックを取得するまで無期限に待機します。
- `sp_sysmon` は、ロック待機中のタスクが待機期間内にロックを取得できなかった回数をレポートするものです。

### 繰り返し読み出しトランザクション独立性レベル

- 繰り返し読み出し独立性レベルは、トランザクション独立性レベル2とも呼ばれ、トランザクションが完了するまで、文によって読み込まれたすべてのページにロックを設定します。
- 繰り返し不可能読み出しは、あるトランザクションがテーブルからローを読み出したときに、2番目のトランザクションがそれと同じローを変更して、最初のトランザクションが完了するまでにその変更をコミットできる場合に発生します。最初のトランザクションがこれらのローを再度読み込んだ場合、その値は変更されているため、最初の読み出しを繰り返すことはできなくなります。繰り返し読み出しでは、トランザクションの期間中に共有ロックが設定されて、ロックされたページ上のロックされたローを更新するトランザクションが阻止されます。

### シミュレートされた統計値の使用

optdiag ユーティリティ・プログラムの `simulate` モードを使用すると、シミュレートした統計をデータベースにロードできます。`set statistics simulate on` がセッションで発行された場合、クエリは、テーブルの実際の統計ではなく、シミュレートした統計を使用して最適化されます。

### set オプションの影響を受けるグローバル変数

表 1-32 は、`set` コマンドによって制御されるセッション・オプションの情報が含まれているグローバル変数のリストです。

表 1-32: セッション・オプションが含まれるグローバル変数

グローバル変数	説明
<code>@@char_convert</code>	文字セットの変換が無効な場合は 0。文字セット変換が有効な場合は 1 が格納される。
<code>@@client_csexpansion</code>	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、 <code>@@client_csexpansion</code> の値が 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。
<code>@@cursor_rows</code>	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。値として -1 を返す。
<code>@@datefirst</code>	<code>set datefirst n</code> を使用して設定する。n には 1 ~ 7 までの値が入る。tinyint で表される各週の指定された最初の曜日を示す <code>@@datefirst</code> 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、 <code>set datefirst 7</code> を指定して設定する。設定と値の詳細については、 <code>set</code> コマンドの「 <code>datefirst</code> オプション」を参照。
<code>@@isolation</code>	Transact-SQL プログラムの現在の独立性レベル。 <code>@@isolation</code> はアクティブ・レベルの値 (0、1、または 3) をとる。

グローバル変数	説明
@@lock_timeout	set lock wait n を使用して設定する。現在の lock_timeout 設定 (ミリ秒) を返す。@@lock_timeout 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 set lock wait n が実行されない場合、@@lock_timeout は -1 を返す。
@@options	セッションの et オプションの 16 進表現。
@@parallel_degree	現在の最大並列度の設定。
@@rowcount	最後のクエリによる影響を受けたローの数。@@rowcount は、if、update、delete 文など、ローを返さないコマンドによって 0 に設定される。カーソルでは、@@rowcount は、最後の fetch 要求までにカーソル結果セットからクライアントに返されたローの累積数を表す。 @@rowcount は、nocount がオンの場合も更新される。
@@scan_parallel_degree	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。
@@textsize	select が返す text、unitext、または image データのバイト数に対する制限が入る。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。セッションごとに set textsize を使用して変更できる。 enable surrogate processing を使用する場合、実際に返されるサイズが @@text サイズ値より小さい場合も、Unicode サロゲート (2 つの 16 ビット値) は 1 文字として返される。
@@tranchained	Transact-SQL プログラムの現在のトランザクション・モード。 @@tranchained は、非連鎖モードの場合は 0、連鎖モードの場合は 1 を返す。

表 1-33 は、@@options で機能する set オプションと値のリストです。

表 1-33: set オプションと @@options の値

数値	16 進数値	set オプション
4	0x04	showplan
5	0x05	noexec
6	0x06	arithignore
8	0x08	arithabort
13	0x0D	control
14	0x0E	offsets
15	0x0F	statistics io と statistics time
16	0x10	parseonly
18	0x12	procid
20	0x14	rowcount
23	0x17	nocount
77	0x4D	opt_sho-fi
78	0x4E	select
79	0x4F	set tracefile

データベースにおける Java での *fipsflagger* の使用

- *fipsflagger* がオンの場合、Adaptive Server では、次の拡張機能を使用すると警告メッセージが表示されます。
  - installjava ユーティリティ
  - remove java コマンド
  - Java クラスをデータ型として参照するカラムと変数宣言
  - メンバ参照として Java-SQL 式を使用する文
- *fipsflagger* のステータスは、Java メソッドによって実行される算術式に影響しません。
- データベース内の Java の詳細については、『Adaptive Server Enterprise における Java』を参照してください。
- set tracefile の使用の制限
- システム・タスク (ハウスキーピングやポート・マネージャなど) の SQL テキストは保存できない。
- トレースを有効または無効にするには、sa または sso の役割があるか、set tracing パーミッションが付与されている必要がある。
- set tracefile を使用して、既存のファイルをトレース・ファイルとして開くことはできません。

- SA または システム・セキュリティ担当者セッション中に、特定の `spid` に対して `set tracefile` を有効にした場合、その後に実行したすべてのトレース・コマンドは、SA またはシステム・セキュリティ担当者の `spid` ではなく、この `spid` に対して有効になる。
- トレース・ファイルへの書き込み中に、Adaptive Server にファイルの領域がなくなった場合は、ファイルが閉じられ、トレースが無効になる。
- `isql` セッションで `spid` のトレースを開始した後、このトレースを無効にしないまま `isql` セッションが終了した場合は、別の `isql` セッションでこの `spid` のトレースを開始できる。
- トレースは、このトレースを有効にしたセッションではなく、トレースが有効になっているセッションに対してのみ行われる。
- 単一の `sa` または `sso` セッションで一度に複数のセッションをトレースすることはできない。トレース・ファイルがすでに開いているセッションに対してトレース・ファイルを開こうとすると、次のエラー・メッセージが表示される。`tracefile is already open for this session.`
- 複数の `sa` または `sso` セッションから1つのセッションをトレースすることはできない。
- トレース出力を保存しているファイルは、トレースしているセッションが終了するか、トレースが無効になったときに閉じられる。
- トレース用のリソースを割り当てる場合、各トレースには1つのエンジンごとに1つのファイル記述子が必要であることを注意しておくこと。

#### トレース・ファイルに診断情報を保存するオプションの設定

`set tracefile` は、診断情報を提供する他の `set` コマンドやオプションと結合して使用すると、低速実行クエリに関する理解を深めることができます。診断情報をファイルに保存する `set` コマンドとオプションは次のとおりです。

- `set show_sqltext [on | off]`
- `set showplan [on | off]`
- `set statistics io [on | off]`
- `set statistics time [on | off]`
- `set statistics plancost [on | off]`

`set` オプションは次のとおりです。

- `set option show` [normal | brief | long | on | off]
- `set option show_lop` [normal | brief | long | on | off]
- `set option show_parallel` [normal | brief | long | on | off]
- `set option show_search_engine` [normal | brief | long | on | off]
- `set option show_counters` [normal | brief | long | on | off]
- `set option show_managers` [normal | brief | long | on | off]
- `set option show_histograms` [normal | brief | long | on | off]
- `set option show_abstract_plan` [normal | brief | long | on | off]
- `set option show_best_plan` [normal | brief | long | on | off]
- `set option show_code_gen` [normal | brief | long | on | off]
- `set option show_pio_costing` [normal | brief | long | on | off]
- `set option show_ljo_costing` [normal | brief | long | on | off]
- `set option show_log_props` [normal | brief | long | on | off]
- `set option show_elimination` [normal | brief | long | on | off]

#### `show_sqltext` の制限

- `show_sqltext` を実行するには、sa または sso の役割が必要です。
- `show_sqltext` を使用してトリガの SQL テキストを出力することはできません。
- `show_sqltext` を使用して、バインド変数または表示名を示すことはできません。

#### ログイン・トリガからの `set` オプションのエクスポート

Adaptive Server では、ログイン・トリガ内の `set` オプションをユーザ・セッション全体で有効にできます。次の `set` オプションは自動的にエクスポートされます。



- altnames
- ansi\_permissions
- ansinull
- arithabort [overflow | numeric\_truncation]
- arithignore [overflow]
- cis\_rpc\_handling
- close on endtran
- colnames
- command\_status\_reporting
- dup\_in\_subquery
- explicit\_transaction\_required
- fipsflagger
- flushmessage
- fmtonly
- forceplan
- format
- nocount
- or\_strategy
- prefetch
- proc\_output\_params
- proc\_return\_status
- procid
- quoted\_identifier
- raw\_object\_serialization
- remote\_indexes
- replication
- rowcount
- self\_recursion
- showplan
- sort\_resources
- statistics io
- statement\_cache
- strict\_dtm\_enforcement
- string\_rtruncation
- textptr\_parameters
- transactional\_rpc
- triggers

## 標準規格

## ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

ANSI SQL 規格により指定される動作は、Adaptive Server の 15.7 よりも前のバージョンでの Transact-SQL の動作とは異なります。すべての Embedded-SQL プリコンパイラ・アプリケーションに対し、この標準に準拠した動作はデフォルトで有効です。他のアプリケーションでこの標準の動作に対応する必要がある場合には、set オプションを使用します。

表 1-34: 初級レベル ANSI SQL に準拠するための set オプション

オプション	設定
ansi_permissions	on
ansinull	on
arithabort	off
arithabort numeric_truncation	on
arithignore	off
chained	on
close on endtran	on
fipsflagger	on
quoted_identifier	on
string_rtruncation	on
transaction isolation level	3

## パーミッション

パーミッション・チェックは、細密なパーミッションの設定によって異なります。**set** パーミッションは、通常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に特別なパーミッションは必要ありません。例外には、**set identity\_insert**、**set identity\_update**、**set option show\_option**、**set plan for show**、**set proxy**、**set rephreshold**、**set role**、**set session authorization**、**set tracefile** および **set switch** が含まれます。例外それぞれのパーミッション要件については、前述コマンドの説明を参照してください。

## 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
88	security	set proxy または set session authorization	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – 以前の <b>suid</b></li> <li>• <i>Current value</i> – 新しい <b>suid</b></li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – <b>set proxy</b> または <b>set session authorization</b> にパラメータがない場合は、元のログイン名。それ以外は <b>null</b>。</li> </ul>

## 参照

コマンド [create trigger](#)、[fetch](#)、[grant](#)、[insert](#)、[lock table](#)、[revoke](#)、[set](#)

関数 [convert](#)

ユーティリティ [isql](#)、[optdiag](#)

ストアド・プロシージャ [sp\\_setrepcbmode](#)、[sp\\_setrepcbmode](#)

## setuser

**説明** データベース所有者が別のユーザになり代わることを可能にします。

**構文** `setuser ["user_name"]`

**例** データベース所有者は、Mary が所有するテーブル `authors` に対するパーミッションを Joe に付与するために、データベースで一時的に Mary になります。

```
setuser "mary"  
go  
grant select on authors to joe  
setuser  
go
```

**使用法**

- データベース所有者は、`setuser` を使用して、他のユーザのデータベース・オブジェクトを使用したり、パーミッションを付与したり、オブジェクトを作成したり、その他の必要に応じて、他のユーザになりかわることができます。
- ログイン・アカウント "sa" で実行されるセッションを除き、データベース所有者が `setuser` コマンドを使用すると、データベース所有者のパーミッションではなく、ID を使用されるユーザのパーミッションが検査されます。ID を使用されるユーザは、データベースの `sysusers` テーブルにリストされているユーザである必要があります。
- `setuser` は、ローカル・データベースのパーミッションにのみ影響します。リモート・プロシージャ・コールや、その他のオブジェクトへのアクセスには影響しません。
- `setuser` は、別の `setuser` コマンドが実行されるまで、または `use` コマンドによって現在のデータベースが変更されるまで有効です。
- `setuser` はデータベースの作成時には影響しません。
- ユーザ名を指定せずに `setuser` を実行すると、元のデータベース所有者に戻ります。
- システム管理者は、`setuser` を使用して、別のユーザが所有するオブジェクトを作成できます。ただし、システム管理者はパーミッション・システムの外部で操作するため、`setuser` を使用して別のユーザのパーミッションを取得することはできません。

**標準規格**

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

**パーミッション**

`setuser` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>setuser</code> の実行には <code>setuser</code> 権限が必要。 <code>setuser</code> 権限はデフォルトでデータベース所有者に付与されている。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>setuser</code> 権限のデフォルトでデータベース所有者に設定され、譲渡は不可。

監査 `sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
84	setuser	setuser	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – 設定されたユーザ名</li> <li>• <i>Proxy information</i> – <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

参照 コマンド [grant](#), [revoke](#), [use](#)

## shutdown

**説明** コマンドの発行元 Adaptive Server、そのローカル Backup Server、またはリモート Backup Server を停止します。

**構文** `shutdown [srvname] [with {wait [= "hh:mm:ss"] | nowait}]`

クラスタの構文：

```
shutdown {cluster | [instance_name]} [with {wait | nowait}]
```

**パラメータ**

*srvname*

Adaptive Server の `sys.servers` システム・テーブルで Backup Server を認識する論理名です。このパラメータは、ローカル Adaptive Server を停止する場合は不要です。

*with wait*

デフォルトです。このオプションは、Adaptive Server または Backup Server を停止します。

*hh:mm:ss*

実行中またはスリープ中のすべてのプロセスがジョブを終了するまでにサーバが待つ最大時間を指定します。この設定は省略可能です。

*with nowait*

現在実行している文の終了を待たずに、Adaptive Server または Backup Server をただちに停止します。

---

**警告！** `shutdown with nowait` は、特殊な状況でのみ使用します。Adaptive Server では、`checkpoint` コマンドを発行してから、`shutdown with nowait` を実行してください。`shutdown with nowait` を使用すると、IDENTITY カラムの値にギャップが発生する可能性があります。

---

*cluster*

(共有ディスク・クラスタのみ) クラスタ内のすべてのインスタンスを停止するように指定します。

*instance\_name*

(共有ディスク・クラスタのみ) 停止する特定のインスタンスの名前です。

**例**

**例 1** `shutdown` コマンドを発行した Adaptive Server を停止します。

```
shutdown
```

**例 2** Adaptive Server をただちに停止します。

```
shutdown with nowait
```

**例 3** ローカルの Backup Server を停止します。

```
shutdown SYB_BACKUP
```

**例 4** リモート Backup Server の REM\_BACKUP を停止します。

```
shutdown REM_BACKUP
```

**例 5** 現在のクラスタを停止します。

```
shutdown cluster
```

**例 6** インスタンス "ase1" を停止しますが、クラスタは実行したままにします。

```
shutdown ase1
```

#### 使用法

- `nowait` オプションを使用しないかぎり、`shutdown` を実行すると Adaptive Server は次の順序で安全に停止します。
  - (システム管理者以外のユーザの) ログインを禁止します。
  - 各データベースにおいてチェックポイントを実行します。
  - 現在実行中の SQL 文またはストアド・プロシージャの終了を待ちます。

---

**注意** データサーバで通常の停止を実行する場合、無限ループを含む SQL と `waitfor delay` コマンドを含む Transact-SQL のバッチは強制終了されません。

---

`nowait` オプションを指定しないで停止すると、自動リカバリ処理が行う必要のある作業量を最小限に抑えることができます。

安全な停止の場合、多くの文が、完了でコミットされるアトミック・トランザクションであると想定して現在実行中の文の完了を待ちます。ただし、安全な停止でも、セッションがトランザクション内で "awaiting command" 状態になると、長時間を要するトランザクションの完了は待ちません。代わりに、停止処理が開始され、トランザクションはリカバリでロールバックされます。

`shutdown` を発行する前に、`master.syslogshold` で実行に長時間かかる未処理トランザクションを調べて対処することをおすすめします。このようなトランザクションには、実行しているユーザに連絡し、トランザクションのコミットが可能であるか、強制終了してロールバックを待つかを確認することで対処します。アクティブなトランザクションのロールバックは、シャットダウン前に実行したほうが、リカバリ後の実行より短時間で済みます。これは多くのページがキャッシュに存在した状態であり、必要なログのスキャンが少なく済むためです。長時間実行中のトランザクションが複数存在する場合は、非常に古いトランザクションが検出されなくなるまで、`syslogshold` の検査を繰り返し実行します。

---

**注意** サーバ上で長時間実行中の未処理トランザクションが存在する状態で停止すると、リカバリに必要な時間が長くなる場合があります。

---

- `nowait` オプションを使用しないと、`shutdown backup_server` は、実行中のダンプやロードが完了するまで待機します。Backup Server に対して `shutdown` コマンドを発行すると、それ以降、この Backup Server を使用するダンプやロードが新規に実行されることはありません。
- `shutdown` を使用して停止できるのは、ローカルの Adaptive Server だけです。リモートの Adaptive Server は停止できません。
- Backup Server は、次の場合にのみ停止できます。
  - Backup Server が、`syservers` テーブルにリストされている場合。`sp_addserver` を使用して、`syservers` にエントリを追加する。
  - Backup Server がコマンドを実行する Adaptive Server の `interfaces` ファイルにリストされている場合。
- Adaptive Server で認識されている Backup Server の名前を確認するには、`sp_helpserver` を使用します。`srvname` パラメータには、Backup Server の `network_name` ではなく、`name` を指定します。次に例を示します。

```
sp_helpserver
name          network_name  status                                     id
-----
REM_BACKUP    WHALE_BACKUP  timeouts, no net password encryption    3
SYB_BACKUP    SLUG_BACKUP   timeouts, net password encryption       1
eel           eel           0
whale         whale         timeouts, no net password encryption    2
```

WHALE\_BACKUP という名前のリモート Backup Server を停止するには、次のコマンドを使用します。

```
shutdown REM_BACKUP
```

### 待ち時間の指定

サーバを停止する準備ができると、次が実行されます。

- 1 すべてのデータベースで **checkpoint** を実行します。
- 2 新しいユーザがログインできないようにします。
- 3 実行中またはスリープ中のプロセスのジョブが終了するまで待ちます。
- 4 データベースに別の **checkpoint** を実行します。今回は、フラッシュする必要がある以下の項目にフラグを追加します。
  - 混合ログデータ・データベースにあるすべての動的スレッシュホルド
  - すべてのオブジェクト統計
  - リカバリ後の欠損を避けるための **identity** フィールドの値

**with wait** オプションと **hh:mm:ss** オプションを一緒に使用する場合、指定する時間は、Adaptive Server 自体を停止するために要する最大合計時間になりません。Adaptive Server は、最初の **checkpoint** を実行するためにかかる時間を計算し、指定された時間からこの値を自動的に差し引きます。

たとえば、最大待ち時間を 20 分に指定し、最初のチェックポイントの実行に 3 分かかった場合、Adaptive Server はプロセスの終了を最大 17 分まで待ちます。ただし、なんらかの理由で 2 回目の **checkpoint** の実行時間が長くなった場合、この値は指定した **with wait hh:mm:ss** パラメータに算入されません。

また、Adaptive Server は **with wait hh:mm:ss** で指定したよりも長時間かかる **checkpoint** にも対応できます。たとえば、待ち時間を 10 分に指定し、最初の **checkpoint** を完了するのに 20 分かかる場合、Adaptive Server は **checkpoint** を中断せずに、**checkpoint** が完了するまで待ちます。この場合、**checkpoint** が完了した時点で指定した時間が過ぎているため、Adaptive Server はただちに停止処理を開始し、フラッシュする項目を示すフラグを追加する最後の **checkpoint** を実行します。

### クラスタ環境での停止

- 次のように、オプションを指定しない **shutdown** コマンドは、クラスタ環境では無効です。

```
shutdown  
go
```



標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>shutdown</code> のパーミッション・チェックは、細密なパーミッションの設定によって異なります。
細密なパーミッションが有効	細密なパーミッションが有効化されている場合は、次のようになる。 <ul style="list-style-type: none"> <li>• SMP サーバ、クラスタ・インスタンス、またはバックアップ・サーバを停止するには、<code>shutdown</code> 権限を持つユーザであることが必要。</li> <li>• クラスタを停止するには、<code>shutdown</code> 権限と <code>manage cluster</code> 権限を持つユーザであることが必要。</li> </ul>
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> が付与されたユーザであることが必要。

**監査** `sysaudits` の event カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	<code>extrainfo</code> の情報
51	security	サーバの停止	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — <code>shutdown</code></li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

**参照** コマンド [alter database](#)  
システム・プロシージャ [sp\\_addserver](#), [sp\\_helpserver](#)

## transfer table

説明	テーブルの増分転送を開始します。
構文	<pre>transfer table [[db.]owner.]table [to   from] destination_file   [ for { ase   bcp   iq   csv } ]   [ with {column_separator=string}, {column_order=option},     {encryption=option}, {row_separator=string},     {resend=id}, {progress=sss}, {tracking_id=nnn}     {sync = true   false}, {fixed_length = true   false}     , null_byte = true   false}</pre>
パラメータ	<p><b>table</b> Adaptive Server 内の無効なテーブルです。transfer table を実行するには sa_role またはテーブルの所有権が必要です。テーブルの所有者は、所有するテーブルの transfer table 権限を付与できます。</p> <p><b>to   from</b> 転送の方向を表します。transfer table...to で使用するパラメータの一部は、transfer table...from に使用できません。from パラメータに使用できるパラメータは次のとおりです。</p> <ul style="list-style-type: none"> <li>• column_order=option (for ase を使用したロードには適用されません。今後のために予約済み。)</li> <li>• column_separator=string (for ase を使用したロードには適用されません。今後のために予約済み。)</li> <li>• encryption={true   false} (for ase を使用したロードには適用されません。今後のために予約済み。)</li> <li>• progress=nnn</li> <li>• row_separator=string (for ase を使用したロードには適用されません。今後のために予約済み。)</li> </ul> <p><b>destination_file</b> オペレーティング・システムに有効で、Adaptive Server がアクセスできるファイルまたはパス名です。ファイルが相対ファイル・パスである場合、Adaptive Server はその絶対パスを指定します。destination_file を開くことができないと、Adaptive Server はエラー・メッセージを返します。</p> <p>次のすべての条件を満たす場合、Adaptive Server は転送先ファイルを削除します。</p> <ul style="list-style-type: none"> <li>• ファイルは正規ファイルであり、名前付きパイプ、シンボリック・リンク、またはその他の特殊ファイルではない。</li> <li>• Adaptive Server がこの転送でファイルを開く。</li> <li>• 転送が失敗する、またはローを送信しない。</li> </ul>

## for 句

転送先のデータ・フォーマットのいずれかに名前を指定します。for 句を省略すると、指定されたテーブルの最初の転送のデフォルト値は **for ase** になります。後続の転送のフォーマットは、デフォルトで、前に正常に実行された転送で使ったフォーマットになります。ただし、コマンドがフォーマット **with resend = id** を指定する場合は、指定された前の転送フォーマットがデフォルトで使用されます。

- **ase** – データを Adaptive Server にインポートするためのフォーマット。この出力フォーマットはライセンスの必要な機能で、RAP ユーザと、インメモリ・データベースのライセンスを持つユーザが使用できます。データ変換は適用されません。このファイル・フォーマットには、ソース・コンピュータのバイト順序、文字セット、およびデフォルトのソート順があるテーブルを記述するヘッダが含まれています。これは、まだ転送が正常に実行されていないテーブルのデフォルト値です。
- **bcp** – データを bcp バイナリフォーマットのデータとしてインポートするためのフォーマット。ローは、bcp を使用してロードできるバイナリ・データとして出力されます。データ変換は適用されません。転送で、Adaptive Server はフォーマット・ファイルを作成します。このフォーマット・ファイルは、bcp がデータを記述するのに使用され、出力ファイルと同じディレクトリに配置されます。

名前付きパイプへの転送に for bcp を使用することはできません。

出力ファイルが名前付きパイプ以外のファイル・タイプである場合、Adaptive Server は次の命名規則をフォーマット・ファイルに使用します。

```
{{table_name},{database_id},{object_id}.fmt
```

- **iq** – IQ の load table コマンドを使用して、Sybase IQ へのロードに適した形式でデータを書き込みます。Adaptive Server は、データをバイナリ・フォーマットでファイルに書き込み、データ型を IQ と互換性のあるバージョンに変換するのに必要なデータ変換を適用します。with fixed\_length='true' または with null\_byte='true' 変更子が指定されていない場合、for iq はデータをデフォルトのフォーマットで書き込みます。
  - デフォルト・フォーマット – null を入力可能なデータには、次の「null バイト」(1 バイトのインジケータ)が含まれる。
    - カラムが null でない場合は 0。
    - カラムが null の場合は 1。

null 入力不可のデータには、この null バイトは含まれない (load table については IQ のマニュアルを参照)。可変長文字列の前には、文字列の長さを示す 1 または 2 バイトを付加する。このときに、プレフィックスのバイト数はカラムの最大長によって異なる。最大 255 バイトの文字列には 1 バイト、256 バイト以上の文字列には 2 バイトを付加する (Adaptive Server は最大約 16000 バイトの文字列をサポートする)。文字列を除いて、各カラムは固定長として転送され、必要に応じて、この固定長サイズになるように埋め込まれる。

- これらの変更子をデータのフォーマットを判別するために使用する。
  - `with fixed_length='true'` — 文字列を含むすべてのカラムは、カラム幅いっぱいまで埋め込まれる。文字列には空白が埋め込まれる。その他のカラムには <NUL> または 0x00 が埋め込まれる。長さのインジケータを含むカラムはない。
  - `with null_byte='true'` — すべてのカラムに、null を入力可能かどうかに関係なく、null バイトが含まれる。この修飾子を指定すると、`for iq` はコマンドで指定した内容に関係なく、`fixed_length='true'` 変更子を使用する。
- `csv` — テキスト化された値のフォーマットです。ローは、文字コード化されたデータとして出力されます。カラムは、指定されたカラム・セパレータによって区切られます。ローは、指定されたロー・ターミネータで終了します。セパレータとターミネータはユーザ定義されます。

#### with 句

コマンドの操作を変更するオプションを指定します。

#### `column_separator = string`

出力カラム間で書き込まれる文字列を csv フォーマットで宣言し、デフォルトを置き換えます。後続の転送用に書き込まれる文字列は、デフォルトで前に指定された `column_separator` になります。

#### `column_order = option`

カラム・データが出力に書き込まれる順序を宣言します。オプションは次のとおりです。

- `id` — `syscolumns` で指定されたカラム ID に基づく順序です。転送が `for bcp` の場合に使用できる唯一のカラム順で、これらの転送のデフォルト値です。

- **name** — Adaptive Server の現在の文字セットとソート順を使用した、**syscolumns** で指定されたカラム名に基づく順序です。
- **name\_utf8** — ソートする前にカラム名を UTF8 文字に変換した、**syscolumns** で指定されたカラム名に基づく順序です。
- **offset** — データ・ロー内のカラム・オフセットに基づく順序です。転送が **for ase** の場合に使用できる唯一のカラム順で、これらの転送のデフォルト値です。

for 句のフォーマットに一致しない **column\_order** を使用する場合、Adaptive Server はエラー・メッセージを出力します。カラム順は次のとおりです。

- **for ase** — **offset** カラム順を使用します。
- **for bcp** — **id** カラム順を使用します。

#### **encryption = option**

コマンドが暗号化カラムを処理する方法を指定します。オプションは次のとおりです。

- **true** — 転送前にカラムを復号化します。これはデフォルト値です。ユーザには、暗号化カラムを復号化するパーミッションが必要です。
- **false** — データ・ローに表示されるとおりに暗号化カラムを転送します。

---

**注意** データをリカバリするには、受信側が暗号化キーと暗号化に使用するアルゴリズムを把握する必要があります。Adaptive Server は暗号化データをファイルに書き込む際、テーブルに最初に格納したときに暗号化されたとおりにデータを書き込みます。データを転送しても変わりません。データをリカバリするには、受信側はデータを暗号化したキーと、暗号化アルゴリズムの特殊な設定 (初期化ベクトルを使用したかどうかなど) を把握する必要があります。

---

#### **progress = sss**

転送処理中に **sss** 秒ごとに進行メッセージが生成されることを示します。デフォルトでは、進行メッセージは省略されます。

**row\_separator = string**

各出力ローの改行コードを `csv` フォーマットで宣言し、デフォルトを置き換えます。転送が `for csv` である場合を除き、このオプションは無効です。`column_separator` と同様に、2 回目以降、`csv` モードでのすべての転送のデフォルトは、正常に実行された最後の転送のデフォルト値になります。デフォルトのロー・セパレータはプラットフォームによって異なります。**Linux** および **UNIX** では改行 ([`Ctrl`] キーを押しながら [`J`] キー) を使用し、**Windows** では復帰改行と改行 ([`Ctrl`] キーを押しながら [`M`] キー、 [`Ctrl`] キーを押しながら [`J`] キー) を使用します。

**resend =id**

シーケンス ID カラムがこのデータ転送の開始時点のタイムスタンプを取得するテーブルの履歴エントリを示します。このオプションは前に送信したデータを再送信します。コマンドで指定されたテーブルに増分転送のマークが付いている場合を除き、`resend =id` は無視されます。指定された `sequence ID` がこのテーブルに存在しない場合、**Adaptive Server** はテーブル全体を再送信します。

**Adaptive Server** は、この転送の開始時点のタイムスタンプとして、指定されたエントリの開始時点のタイムスタンプを選択し、デフォルトの転送先の型として、指定されたエントリの転送先の型 (`ase`、`bcp` など) を選択します。

`id` が負の値の場合は、前に正常に完了した、指定されたテーブルの転送の履歴エントリを取得します。`-1` は正常に完了した最後の転送、`-2` は正常に完了した 2 つ前の転送を示し、以降も同様です。転送履歴テーブルは、正常に実行された転送と失敗した転送の両方のエントリを格納します。

**tracking\_id =nnn**

特定の転送を追跡するのに役立つ、オプションの整数識別子を指定します。`spt_TableTransfer.tracking_id` カラムを使用して、`nnn` の値を特定し、クエリでその値を使用します。次の例では、追跡 ID 番号 123 の終了ステータスとシーケンス ID が、出力データ・ファイルの完全パスとともに返されます (これらの値が存在しない場合は `NULL` が返されます)。

```
select end_code, sequence_id, pathname from spt_TableTransfer
where id = object_id('mytable') and tracking_id = 123
```

**Adaptive Server** では、`tracking_id` を制御しません。また、追跡 ID がユニークである必要もありません。

---

**注意** この追跡 ID は `resend =id` に使用されるシーケンス ID ではありません。

---

**sync = true | false**

トランザクションに対する転送の対応を決定します。オプションは次のとおりです。

- **true** — 転送に含まれているテーブルからローをグループとして取得するように、転送が同期されます。このテーブルに影響するすべてのトランザクションが終了した後で、**transfer** が開始します。**transfer** が開始を待機している間は、このテーブルに影響する新しいトランザクションがテーブルを変更することはありません。**transfer** が開始するまで、トランザクションは待機します。**transfer** の進行中は、**transfer** による検査が終わるまで、トランザクションがこのテーブル内のローを変更することはありません。
- **false** — 転送は同期されません。**transfer** は、テーブルの別のローを送信するかどうかに関係なく、選択されたタイムスタンプ範囲内のローを送信します。これはデフォルトの動作です。

---

**注意** **sync** は転送対象のテーブルのみに影響します。**transfer** ではテーブル間の制約は考慮されません。

---

**fixed\_length = true | false**

出力ファイルで **transfer..for iq** がすべてのカラムを固定長フィールドとして転送するかどうかを指定します。通常、**Adaptive Server** は 1 または 2 バイトのプレフィクス長を持つ可変長文字列を転送します。**fixed\_length** を **true** に設定すると、**Adaptive Server** はカラムの最大幅に達するまで、文字列にブランクを埋め込みます。**for iq** パラメータを持つパラメータを使用する必要があります。**fixed\_length** の設定は次のとおりです。

- **true** — **Adaptive Server** はプレフィクス長を使用しないで、文字列に幅いっぱいまで埋め込みます。
- **false** — **Adaptive Server** はデフォルトの動作、つまり、プレフィクス長を使用して文字列を送信します。

**null\_byte = true | false**

カラムが **null** であるかどうかを示すため、**transfer..for iq** が送信する各カラムの末尾にバイトを追加するかどうかを指定します。通常、**Adaptive Server** は **null** を入力可能なカラムにのみこのバイトを附加します。オプションは次のとおりです。

- **true** — Adaptive Server は、すべてのカラムの最後に **null** バイトを付加します。カラムに **null** を入力可能かどうかに関係なく、カラムが **null** の場合は 0、**null** でない場合は 1 を付加します。**true** の場合、**for iq** は **transfer** コマンドで指定した内容に関係なく、**fixed\_length='true'** 変更子を使用します。
- **false** — Adaptive Server は **null** を入力可能なカラムにのみ **null** バイトを付加します。

---

**注意** **true** または **false** に設定するかどうかに関係なく、**null\_byte** は **for iq** 句を含む転送にのみ適用されます。

---

**例** **例 1 mytable** テーブルを転送するパーミッションをユーザ “john” に付与します。

```
grant transfer table on mytable to john
```

**例 2 mytable** を Sybase IQ へのロード用にフォーマットされた出力ファイルに転送します。この例で **name\_utf8** を指定しない場合、デフォルトの順序はカラム ID 順になります。

```
transfer table mytable to '/path/to/file' for iq
with column_order = 'name_utf8'
```

**例 3 offset** のカラム出力順を使用する Adaptive Server ファイル・フォーマット用にフォーマットされた **mytable** を転送します。次の例では、存在しない履歴エントリの **resend** を要求するため、テーブル全体が転送されます。

```
transfer table mytable to '/path/to/file3/' for ase
with resend=10
```

この例では、**for ase** 転送のデフォルトのカラム順が変わります。転送後、デフォルトの受信側は **ase**、カラム順は **offset**、カラムとローのセパレータは **null** になります。

#### 使用法

- **transfer table** は、前の転送以降に変更された、コミットされたデータのみを送信します。
- **with** 句内の **column\_separator** と **row\_separator** への **string** 引数の長さの最大値は 64 バイトで、次のフォーマット指示が含まれています。
  - "¥b" はバックスペース <BS> ([Ctrl] キーを押しながら [H] キー) を示す。
  - "¥n" は改行 <LF> ([Ctrl] キーを押しながら [J] キー) を示す。



- "␣" は復帰改行 <CR> ([Ctrl] キーを押しながら [M] キー) を示す。
- "␣t" は <TAB> ([Ctrl] キーを押しながら [I] キー) を示す。
- "␣␣" は円記号を示す。
- これらのいずれかのシーケンスに含まれていない文字列内の "␣" は実際の円記号であり、文字列にそのように表示される。
- `transfer table .. from` は、更新や挿入の実行中はトリガを起動しません。
- `transfer table` で (重複キーなどの) エラーが発生した場合、Adaptive Server はエラー番号のみを表示するため、エラーの原因を理解することが難しくなります。次に例を示します。

```
Msg 2633, Level 20, State 1
Server 'SYB155', Line 1
TRANSFER TABLE failed to insert a row to table 'my_tab'.
The indicated error was 2601.
Msg 16025, Level 16, State 1
Server 'SYB155', Line 1
TRANSFER TABLE my_tab:command failed with status 2633.
```

エラー・メッセージを取得するには、手動で `master.sysmessages` に対するクエリを実行します。たとえば、2601 がエラー番号である場合は、次のように入力します。

```
select * from master..sysmessages where error = 2601
```

エラー 2601 の詳細については、『トラブルシューティング・ガイド』を参照してください。

### 増分転送のマークが付いていないテーブルの転送

増分転送のマークが付いていないテーブルに対して `transfer table` を使用できますが、次の制限があります。

- 一部のローが転送されない場合がある。ユーザが転送処理中にテーブルを更新すると、更新されたローは転送されない場合があります。
- 増分転送は実行されない。転送できるのはテーブル全体のみです。この転送について、後続の転送には通知されません。
- `spt_TableTransfer` に履歴エントリは書き込まれません。転送中は `monTableTransfer` に転送が表示されますが、転送が完了すると、レコードは表示されなくなります。

## パーミッション

テーブルを転送するパーミッションは、対象のテーブル内のデータを復号化するパーミッションを自動的に付与しません。暗号化カラムを復号化するには、テーブル所有者から特定のパーミッションを付与される必要があります。

細密なパーミッションの設定によって異なる **transfer table** のパーミッション・チェックについて、次に説明します。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であるかテーブルに対する <b>transfer table</b> パーミッションを持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者、 <b>transfer table</b> パーミッションを持つユーザ、または <b>sa_role</b> が付与されたユーザであることが必要。

## 監査

**sysaudits** の **event** カラムと **extrainfo** カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
136	transfer table	transfer table	

## 参照

コマンド `create table`、`alter table`

## truncate lob

説明	LOB を指定の長さにトランケートする。
構文	<code>truncate lob locator_descriptor [( result_length)]</code>
パラメータ	<i>locator_descriptor</i> 有効なロケータです。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。 <i>result_length</i> <code>text</code> と <code>unitext</code> ロケータでは文字単位で、 <code>image</code> ロケータではバイト単位の長さです。
例	テキスト・ロケータ <code>@w</code> によって参照される LOB を 20 文字にトランケートします。 <pre>truncate lob @w (20)</pre>
使用法	<i>result_length</i> を指定しない場合や 0 (ゼロ) の場合、Adaptive Server は LOB メモリの割り付けを解除し、ロケータは <code>null</code> の LOB を指します。
パーミッション	すべてのユーザが <code>truncate lob</code> を実行できます。
参照	コマンド <code>deallocate locator</code> Transact-SQL 関数 <code>locator_valid</code> 、 <code>return_lob</code> 、 <code>create_locator</code>

## truncate precomputed result set

説明	事前計算済み結果セットのデータをトランケートする。
構文	<pre>truncate {precomputed result set   materialized view}         [owner_name.]prs_name</pre>
パラメータ	<p>precomputed result set   materialized view マテリアライズされたビューまたは事前計算済み結果セットのデータをトランケートするかどうかを指定します。</p> <p><i>prs_name</i> 事前計算済み結果セットの名前。完全修飾された <i>prs_name</i> にサーバまたはデータベースの名前を含めることはできません。</p>
例	<p>authors_prs 事前計算済み結果セットをトランケートします。</p> <pre>truncate precomputed result set authors_prs</pre>
使用法	truncate precomputed result set では、disable に設定された事前計算済み結果セットが自動的に変更されます。事前計算済み結果セットを再有効化するには、refresh precomputed result set を発行します。
パーミッション	truncate precomputed result set を実行するには、事前計算済み結果セットの所有者である必要があります。

## truncate table

説明	テーブルまたはパーティションからすべてのローを削除します。
構文	<code>truncate table [[database.]owner.]table_name [partition partition_name]</code>
パラメータ	<p><i>table_name</i> トランケートされるテーブルの名前です。テーブルが別のデータベース内にある場合はデータベース名を指定し、データベース内にその名前のテーブルが複数ある場合は所有者名を指定します。 <i>owner</i> のデフォルト値は現在のユーザで、<i>database</i> のデフォルト値は現在のデータベースです。</p> <p><i>partition_name</i> トランケートするパーティションの名前を指定します。</p>
例	<p><b>例 1</b> テーブル <code>authors</code> からすべてのデータを削除します。</p> <pre>truncate table authors</pre> <p><b>例 2</b> <code>titles</code> テーブルの <code>smallsales</code> パーティションからすべてのデータを削除します。</p> <pre>truncate table titles partition smallsales</pre>
使用法	<ul style="list-style-type: none"> <li><code>truncate table</code> は、テーブルのすべてのローを削除します。テーブルの構造とインデックスはすべて、<code>drop table</code> コマンドが発行されるまで引き続き存在します。カラムにバインドされたルール、デフォルト、および制約はバインドされた状態であり、トリガも引き続き有効です。</li> <li><code>Adaptive Server</code> ではディストリビューション・ページが使用されなくなり、統計情報はテーブル <code>sysstatistics</code> と <code>systabstats</code> に格納されるようになりました。 <code>truncate table</code> の実行中に統計情報が削除 (割り付け解除) されなくなったため、データを追加した後、<code>update statistics</code> を実行する必要はありません。 <code>truncate table</code> はテーブルの統計情報を削除しません。</li> <li><code>truncate table</code> は、<code>where</code> 句が指定されていない <code>delete</code> コマンドと同じですが、処理速度はより早くなります。<code>delete</code> は、ローを 1 つずつ削除し、削除されたローをそれぞれ 1 つのトランザクションとしてログに取ります。<code>truncate table</code> は、データ全体の割り付けを解除するため、ログ・エントリ数がより少なくなります。<code>delete</code> と <code>truncate table</code> は両方とも、データとその関連するインデックスによって使われていた領域を再利用します。</li> </ul>

- パーティションをトランケートしても、他のパーティションのデータに影響はありません。
- 一度にトランケートできるパーティションは1つだけです。
- テーブルをトランケートすると、その処理が完了するまでテーブル全体がロックされます。
- 削除されたローは個別にログが残らないので、`truncate table` ではトリガは実行されません。
- 別のテーブルのローにこれを参照するローがある場合、`truncate table` は使用できません。外部テーブルのローを削除するか、または外部テーブルをトランケートしてから、プライマリ・テーブルをトランケートしてください。
- `grant` コマンドと `revoke` コマンドを使用して、テーブルに `truncate table` を使用するパーミッションをユーザと役割に付与したり、取り消したりできます。

## 標準規格

ANSI SQL — 準拠レベル：初級レベル

## パーミッション

`truncate table` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブル所有者であるかテーブルに対する <code>truncate table</code> パーミッションを持つユーザであることが必要。監査テーブルをトランケートするには、 <code>manage auditing</code> 権限または <code>truncate any audit table</code> 権限が必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブル所有者、テーブルに対する <code>transfer table</code> パーミッションを持つユーザ、 <code>replication_role</code> が付与されたユーザ、または <code>sa_role</code> が付与されたユーザであることが必要。監査テーブルをトランケートするには、 <code>ssr_role</code> が付与されたユーザであることが必要。

## 監査

`sysaudits` の `event` カラムと `extrainfo` カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
64	truncate	truncate table	<ul style="list-style-type: none"> <li>• <i>Roles</i> — 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> — NULL</li> <li>• <i>Previous value</i> — NULL</li> <li>• <i>Current value</i> — NULL</li> <li>• <i>Other information</i> — NULL</li> <li>• <i>Proxy information</i> — <code>set proxy</code> が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [alter table](#), [create table](#), [create trigger](#), [delete](#), [drop table](#), [grant](#), [revoke](#)

## union operator

説明	複数のクエリ結果を結合する単一の結果セットを返します。all キーワードを指定しないかぎり、重複するローは結果セットから削除されます。
構文	<pre>select [top unsigned_integer] select_list       [into clause] [from clause] [where clause]       [group by clause] [having clause]       [union [all]        select [top unsigned_integer] select_list        [from clause] [where clause]        [group by clause] [having clause]]...       [order by clause]       [compute clause]</pre>
パラメータ	<p><b>top unsigned_integer</b> top の制限は union を構成する個々の select に作用し、union 全体には作用しません。</p> <p><b>into</b> select リストで指定されたカラムと where 句で選択されたローに基づいて、新しいテーブルを作成します。union 演算子の最初のクエリにのみ、into 句を指定できます。</p> <p><b>union</b> 2 つの select 文で指定されたデータの共用体を作成します。</p> <p><b>all</b> すべてのローを結果に含めます。重複ローは削除されません。</p>
例	<p><b>例 1</b> 結果セットには、sales および sales_east テーブル両方の stor_id カラムおよび stor_name カラムの内容が含まれます。</p> <pre>select stor_id, stor_name from sales union select stor_id, stor_name from sales_east</pre> <p><b>例 2</b> 最初のクエリにある into 句は、テーブル results に、publishers、stores、stores_east テーブルの指定カラムの共用体の最終結果セットを保持するように指定します。</p> <pre>select pub_id, pub_name, city into results from publishers union select stor_id, stor_name, city from stores union select stor_id, stor_name, city from stores_east</pre>

**例 3** まず、`sales` および `sales_east` テーブル内の指定されたカラムの `union` が生成されます。次に、`publishers` によるその結果の `union` が生成されます。最後に、2 番目の結果と `authors` の `union` が生成されます。

```
select au_lname, city, state from authors
union
((select stor_name, city, state from sales
union
select stor_name, city, state from sales_east)
union
select pub_name, city, state from publishers)
```

**例 4** 6 つのローを返します。`top` の制限は `union` を構成する個々の `select` に作用し、`union` 全体には作用しません。

```
select top 3 au_lname from authors
union all
select top 3 title from titles
```

#### 使用法

- `union` の片側にあるサブクエリの最大数は 50 です。
- `union` クエリのすべてのサイドに表示できるテーブルの合計数は、256 です。
- `select` 文で `union` を使用できます。次に例を示します。

```
create view
select * from Jan1998Sales
union all
select * from Feb1998Sales
union all
```

- `order by` 句と `compute` 句は `union` 文の最終部分だけに記述することができ、最終結果の順序の決定、または合計値の計算に使用されません。
- `group by` 句と `having` 句は、個々のクエリ内でだけ使用でき、最終結果セットには影響を与えません。
- `union` 演算子を含む SQL 文のデフォルトの評価順序は、左から右です。
- `union` はバイナリ演算であるため、3 つ以上のクエリを含む式の場合は、カッコを追加して、評価の順序を指定します。
- `union` 文の最初のクエリには、最終的な結果セットを保持するテーブルを作成する `into` 句を含めることができます。`into` 文は、最初のクエリに含める必要があります。そうしないと、エラー・メッセージが出されます (例 2 を参照)。



- union 演算子は、insert...select 文内で使用できます。次に例を示します。

```
insert into sales.overall
select * from sales
union
select * from sales_east
```

- SQL 文の select リストすべてに、同じ数の式 (カラム名、算術式、集合関数など) が指定されている必要があります。たとえば、次に示す文では、1 番目の select リストの式の数が 2 番目の select リストの式の数よりも多いため、この文は無効です。

```
/* Example of invalid command--shows imbalance */ /*
in select list items */
select au_id, title_id, au_ord from titleauthor
union
select stor_id, date from sales
```

- union 文の select リストでは、対応するカラムが同じ順序で指定されている必要があります。これは、union が、各クエリで指定された順序で、カラムを 1 対 1 で比較するためです。
- union の実行結果として作成されるテーブルのカラム名は、union 文の最初のクエリから取得されます。結果セットに新規のカラム見出しを定義する場合は、1 つ目のクエリ内で行います。また、たとえば order by 文中など、結果セット内のカラムを新しい名前参照する場合は、最初の select 文で、その名前によってカラムを参照します。たとえば、次のクエリは有効です。

```
select Cities = city from stores
union
select city from stores_east
order by Cities
```

- union 演算の構成部分であるカラムの記述は、必ずしも同じである必要はありません。表 1-35 に、結果テーブルのデータ型とそれに対応するカラムの規則を示します。

表 1-35: union 演算の結果データ型

union 演算におけるカラムのデータ型	結果テーブル内の対応するカラムのデータ型
データ型の互換性なし (データ変換は、Adaptive Server によって暗黙には処理されない)	Adaptive Server から返されるエラー。
両方とも L1 または L2 の長さを持つ固定長文字	固定長文字。長さは、L1 と L2 の大きい方に等しい。

union 演算におけるカラムのデータ型	結果テーブル内の対応するカラムのデータ型
両方とも L1 または L2 の長さを持つ固定長バイナリ	固定長バイナリ。長さは、L1 と L2 の大きい方に等しい。
どちらか一方、または両方が可変長文字	可変長文字。長さは、union のカラムに指定された最大長に等しい。
どちらか一方、または両方が可変長バイナリ	可変長バイナリ。長さは、union のカラムに指定された最大長に等しい。
両方ともに数値データ型 (smallint、int、float、money など)	2つのカラムの最大精度と同等のデータ型。たとえば、テーブル A のカラムが int タイプで、対応するテーブル B のカラムが float タイプである場合、float は int より精度が高いため、結果テーブルの対応するカラムのデータ型は、float になる。
両方のカラムの記述が NOT NULL	NOT NULL を指定する。

#### 制限事項

- union 演算子は、サブクエリ内では使用できません。
- union 演算子を for browse 句とともに使用することはできません。

#### 標準規格

ANSI SQL — 準拠レベル：初級レベル

Transact-SQL 拡張機能は次のとおりです。

- union 文の select 句での union の使用
- select 文に union 演算子が存在する場合の、select 文における order by 句による新しいカラム見出しの指定

#### 参照

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

関数 [convert](#)

## umount

### 説明

データベースを停止し、Adaptive Server から削除します。デバイスも非アクティブ化され、削除されます。データベースおよびデータベース内のページは、マウント解除されているときには変更できません。データベース・ページは OS デバイス上に残ります。umount コマンドが完了すると、必要に応じて、ソースの Adaptive Server でデバイスを接続解除し、移動することができます。セカンダリ Adaptive Server で使用するマニフェスト・ファイルを作成するには、*manifest\_file* 拡張機能を使用します。

1 つの umount コマンドで指定できるデータベースの数は 8 個に制限されています。

---

**警告！** umount コマンドを実行すると、データベースとそのデータベースのすべての情報が Adaptive Server から削除されます。umount コマンドは、任意の Adaptive Server から別の Adaptive Server へデータベースを移動する場合にのみ使用してください。

---

### 構文

```
umount database dbname_list to manifest_file
```

### パラメータ

*dbname\_list*

マウント解除するデータベースです。複数のデータベースをマウント解除 (umount) できます。

*manifest\_file*

一連のデータベース・デバイスに存在するデータベースを記述するバイナリ・ファイルです。該当のデバイスを占有する一連のデータベースが、切り離されていて、デバイス上で独立している場合にのみ作成できます。

マニフェスト・ファイルはバイナリ・ファイルであるため、ファイルの内容の文字変換が可能な操作 (ftp など) は、バイナリ・モードで実行しないと、ファイルが破損します。

### 例

**例 1** Adaptive Server からデータベースの umount (マウント解除) を実行して、そのデータベースのマニフェスト・ファイルを作成します。

```
umount database pubs2 to "/work2/Devices/Mpubs2_file"
```

**例 2** *key\_db* に作成されている暗号化キーが、*col\_db* のカラムの暗号化に使用されています。これらのコマンドによって、指定したデータベースが正常にマウント解除されます。

```
umount database key_db, col_db
umount database key_db with override
umount database col_db with override
```

使用法

次の処理はできません。

- システム・データベースのマウント解除。ただし、`sysystemprocs` のマウント解除は可能。
- プロキシ・データベースやユーザが作成したテンポラリ・データベースのマウント解除。
- トランザクション内での `unmount` コマンドの使用。
- HA 設定サーバでのデータベースのマウント解除。

Cluster Edition

`mount database` と `unmount database` は Cluster Edition でサポートされません。このどちらかのコマンドが進行中にインスタンスでエラーが発生した場合は、コマンドがアボートされる可能性があります。この場合、インスタンスのフェールオーバー・リカバリが完了したら、`mount database` または `unmount database` を再発行する必要があります。

暗号化カラムと `unmount database`

- カラムが別のデータベースのキーで暗号化されているときは、関連するすべてのデータベースをまとめてマウント解除します。暗号化カラムを含むデータベースとキーを含むデータベースの依存関係は、参照整合性を使用するデータベース間の依存関係と似ています。
- `override` オプションを使用して、別のデータベースのキーで暗号化されているカラムを含むデータベースに `unmount` を実行します (Adaptive Server が警告メッセージを発行しますが、操作は成功します)。
- `with override` を使用しない場合、コマンドは正常に実行されず、エラー・メッセージが返されます。

次のコマンドは、`override` がないためエラー・メッセージが生成されて失敗します。

```
unmount database key_db
unmount database col_db
```

標準規格

ANSI SQL — 準拠レベル：初級レベル

パーミッション

`unmount` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	細密なパーミッションが有効の場合、 <code>unmount any database</code> 権限を持つユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、 <code>sa_role</code> が付与されたユーザであることが必要。

## 監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
102	unmount	unmount database	<ul style="list-style-type: none"> <li>• <i>Roles</i> – 現在のアクティブな役割</li> <li>• <i>Keywords</i> または <i>options</i> – NULL</li> <li>• <i>Previous value</i> – NULL</li> <li>• <i>Current value</i> – NULL</li> <li>• <i>Other information</i> – NULL</li> <li>• <i>Proxy information</i> – set proxy が有効な場合は元のログイン名</li> </ul>

## 参照

コマンド [mount](#), [quiesce database](#)

## update

**説明** データを追加するか、既存のデータを変更して、既存のローのデータを変更します。

**構文**

```
update [top unsigned_integer]
      [[database.]owner.]{table_name | view_name}
set [[[database.]owner.]{table_name.|view_name.}
     column_name1 =
     {expression1 | NULL | (select_statement)} |
     variable_name1 =
     {expression1 | NULL | (select_statement)}
     [, column_name2 =
     {expression2 | NULL | (select_statement)}]...
     [, variable_name2 =
     {expression2 | NULL | (select_statement)}]...

[from [[database.]owner.]{view_name [readpast]]
     table_name
     [(index {index_name | table_name}
      [prefetch size][lru|mru])]}]
     [readpast]
     [, [[database.]owner.]{view_name [readpast] | table_name
     [(index {index_name | table_name}
      [prefetch size][lru|mru])]}]
     [readpast] ...]
[where search_conditions]
[plan "abstract plan"]

update [[database.]owner.]{table_name | view_name}
set [[[database.]owner.]{table_name.|view_name.}
     column_name1 =
     {expression1 | NULL | (select_statement)} |
     variable_name1 =
     {expression1 | NULL | (select_statement)}
     [, column_name2 =
     {expression2 | NULL | (select_statement)}]...
     [, variable_name2 =
     {expression2 | NULL | (select_statement)}]...
     where current of cursor_name
```

**パラメータ**

*table\_name | view\_name*

トランケートされるテーブルまたはビューの名前です。テーブルまたはビューが別のデータベース内にある場合は、データベース名を指定します。データベース内に同じ名前のテーブルまたはビューが複数ある場合は、所有者の名前を指定します。*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。

**top unsigned\_integer**

top *n* 句をキーワードの直後に挿入し、更新されるローの数を制限します。

**set**

カラム名または変数名を指定して、新しい値を割り当てます。この値には、式または **null** を指定できます。複数のカラム名または変数名と値をリストする場合は、カンマで区切る必要があります。

**from**

他のテーブルまたはビューのデータを使用して、更新するテーブルまたはビューのローを変更します。

**readpast**

**update** コマンドを実行し、データページ・ロックが設定されているテーブルに対し、データロー・ロック設定テーブル内でのみロックが解除されているロー、またはロックが解除されているページのローを変更します。**update...readpast** は、ロックされているローまたはページがあると、ロック解除まで待機せずに、暗黙的にこれらのローまたはページを省略します。

**where**

標準的な **where** 句です (**where** 句を参照)。

**index {index\_name | table\_name}**

*index\_name* は、*table\_name* へのアクセスに使用するインデックスを指定します。ビューを更新するときはこのオプションを使用できません。

**prefetch size**

大容量 I/O が設定されているキャッシュにバインドされたテーブルの I/O サイズをキロバイト単位で指定します。ビューを更新するときは、このオプションを使用できません。**sp\_helpcache** は、オブジェクトがバインドされているキャッシュまたはデフォルトのキャッシュの有効な値を示します。データ・キャッシュ・サイズを設定するには、**sp\_cacheconfigure** を使用します。

**prefetch** を使用してプリフェッチ・サイズ (*size*) を指定するとき、最小値は 2K、および 16K までの各論理ページ・サイズに基づいた 2 の累乗になります。キロバイト単位の **prefetch** サイズ・オプションは、次のとおりです。

論理ページ・サイズ	プリフェッチ・サイズ・オプション
2	2, 4, 8 16
4	4, 8, 16, 32
8	8, 16, 32, 64

論理ページ・サイズ	プリフェッチ・サイズ・オプション
16	16, 32, 64, 128

クエリで指定される `prefetch` サイズは、1 つの候補にすぎません。サイズ設定を有効にするには、そのサイズでデータ・キャッシュを設定します。データ・キャッシュを特定のサイズに設定していない場合、デフォルトの `prefetch` サイズが使用されます。

コンポーネント統合サービスが有効な場合は、リモート・サーバに対して `prefetch` を使用できません。

#### `lru` | `mru`

テーブルに対して使用するバッファ置換方式を指定します。**MRU/LRU** (最も最近に使用された/最も長い間使用されていない) チェーン上のキャッシュへのテーブルの読み込みを最適化に強制的に実行させるには、`lru` を使用してください。キャッシュからバッファを解放し、解放したバッファをテーブルの次のバッファに置き換えるには、`mru` を使用してください。ビューを更新するときはこのオプションを使用できません。

#### `where current of`

**Adaptive Server** が、`cursor_name` の現在のカーソル位置が示すテーブルまたはビューのローを更新します。

#### `index_name`

更新するインデックス名です。インデックス名が指定されないと、指定されたテーブルのすべてのインデックスについて、分散統計値が更新されます。

#### `plan "abstract plan"`

クエリを最適化するために使用する抽象プランを指定します。抽象プラン言語で指定された完全プランまたは部分プランを指定できます。詳細については、『パフォーマンス&チューニング・シリーズ: クエリ処理と抽象プラン』の「抽象プランの作成と使用」を参照してください。

**例** **例 1** `authors` テーブルにある `McBadden` をすべて `MacBadden` に変更します。

```
update authors
set au_lname = "MacBadden"
where au_lname = "McBadden"
```

**例 2** `sales` および `salesdetail` テーブルに記録された最新の売り上げを反映するように、`total_sales` カラムを修正します。この例では、指定された日付に指定されたタイトルの売り上げが 1 セットだけ記録されていて、更新が現在のものであることを想定しています。



```

update titles
set total_sales = total_sales + qty
from titles, salesdetail, sales
where titles.title_id = salesdetail.title_id
and salesdetail.stor_id = sales.stor_id
and salesdetail.ord_num = sales.ord_num
and sales.date in
(select max(sales.date) from sales)

```

**例 3** 現在 `title_crsr` で示されている `titles` テーブルの本の価格を、24.95 ドルに変更します。

```

update titles
set price = 24.95
where current of title_crsr

```

**例 4** `IDENTITY` カラムの値が 4 のローを検索し、本の価格を 18.95 ドルに変更します。Adaptive Server により、`syb_identity` キーワードが `IDENTITY` カラムの名前に置き換えられます。

```

update titles
set price = 18.95
where syb_identity = 4

```

**例 5** 宣言された変数を使用して、`titles` テーブルを更新します。

```

declare @x money
select @x = 0
update titles
set total_sales = total_sales + 1,
  @x = price
where title_id = "BU1032"

```

**例 6** 別のタスクがロックを設定していないローを更新します。

```

update salesdetail set discount = 40
from salesdetail readpast
where title_id like "BU1032"
and qty > 100

```

#### 使用法

- すでに挿入されているローの値を変更する場合は、`update` を使用します。新しいローを追加する場合は、`insert` を使用します。
- `update` 文では、最高 15 テーブルを参照できます。
- `update` は、`create index` コマンドに設定されている `ignore_dup_key`、`ignore_dup_row`、`allow_dup_row` オプションと相互に作用します。詳細については、「`create index`」を参照してください。

- 指定のテーブルまたはテーブル内の指定のカラムで `update` コマンドを発行したときに指定の動作を実行するトリガを定義できます。
- 12.5.2 より前のバージョンの Adaptive Server では、クエリで、`union all` 句を含むビューに対して `update` や `delete` を使用すると、ワーク・テーブルを使用せずに解析され、結果に誤りが生じることがありました。Adaptive Server 12.5.2 では、`union all` 句を含むビューに対して `update` や `delete` を使用するクエリは、必ず `tempdb` のワーク・テーブルを使用して解析されます。

#### update 文での変数の使用

- 変数は `update` 文の `set` 句に割り当てることができます。 `select` 文にも同様に設定できます。
- 例 5 に示すように、`update` 文で変数を使用するには、あらかじめ `declare` でその変数を宣言しておき、`select` で初期化しておく必要があります。
- 更新時には、条件を満たすすべてのローで変数の割り当てが行われます。
- `update` 文内で、割り当ての右側で変数を参照すると、それぞれのローを更新するたびに変数の現在の値が変更されます。**現在の値**とは、現在の行を更新する直前の変数の値です。次の例は、ローが更新されるたびに現在の値がどのように変更されるかを示したものです。

次のような文があるとします。

```
declare @x int
select @x=0
update table1
    set C1=C1+@x, @x=@x+1
    where column2=xyz
```

更新開始前の C1 の値は 1 です。次の表は、更新のたびに @x 変数の現在の値がどのように変化するかを示します。

ロー	最初の C1 の値	最初の @x の値	計算 : C1+@x= 更新後の C1	更新後の C1 の値	計算 : @x+1= 更新後の @x	更新後の値
A	1	0	1+0	1	0+1	1
B	1	1	1+1	2	1+1	2
C	2	2	2+2	4	2+1	3
D	4	3	4+3	7	3+1	4

- 同じ `update` 文で複数の変数を割り当てると、変数に割り当てられる値は割り当てリストの順序によって変わることがあります。ただし、常に値が変化するわけではありません。一番問題がないのは、指定された値の判断を場所に頼らないことです。
- 複数のローが返され、変数に対するカラムの非集合割り当てが発生する場合、変数の最終的な値は最後に処理されるローになるため、これが有益ではない場合もあります。
- 変数の値を割り当てる `update` 文では、条件を満たすローの値を設定する必要はありません。
- 更新の条件を満たすローがない場合、変数の割り当ては行われません。
- `update` 文のサブクエリでは、サブクエリの位置に関係なく、同じ `update` 文で値が割り当てられた変数を参照できません。
- `update` 文で値が割り当てられた変数は、同じ `update` 文の `where` 句や `having` 句では参照はできません。
- ジョインで行われた更新では、`update` 文の右側の値が割り当てられた変数は、更新されていないテーブルのカラムを使用します。結果の値は、更新のために選んだジョインの順序と、ジョインド・テーブルのローの数によって異なります。
- `update` 文をロールバックしても、変数の更新には影響がありません。これは、更新される変数の値がディスクに保存されていないためです。

#### トランザクションに対する `update` の使用

`chained transaction mode on` を設定し、現在アクティブ状態のトランザクションがない場合、Adaptive Server は、`update` 文を使用してトランザクションを暗黙的に開始します。更新を完了するには、トランザクションをコミット (`commit`) するか、または変更内容をロールバック (`rollback`) する必要があります。次に例を示します。

```
update stores set city = 'Concord'
  where stor_id = '7066'
if exists (select t1.city, t2.city
  from stores t1, stores t2
  where t1.city = t2.city
  and t1.state = t2.state
  and t1.stor_id < t2.stor_id)
  rollback transaction
else
  commit transaction
```

このバッチでは、(連鎖トランザクション・モードを使用して)トランザクションを開始し、**stores** テーブルのローを更新します。テーブル内の別の店と同じ都市と州の情報を持つローが更新されると、**stores** テーブルの変更内容をロールバックして、トランザクションを終了します。そうでない場合は、更新をコミットして、トランザクションを終了します。

**Adaptive Server** では、1 つのローを同じトランザクションで複数回更新する **update** 文の実行は禁止されていません。たとえば、次の更新のいずれでも、**title\_id MC2022** の本のタイプ ID が "mod\_cook" であるため、その価格が影響を受けます。

```
begin transaction
update titles
set price = price + $10
where title_id = "MC2222"
update titles
set price = price * 1.1
where type = "mod_cook"
```

#### 更新でのジョインの使用

**update** の **from** 句でのジョインの実行は、更新を行う ANSI 準拠の SQL 構文に対する **Transact-SQL** の拡張機能です。**update** 文の処理方法により、単一の文による更新は累積されません。つまり、**update** 文にジョインが含まれ、このジョインで他のテーブルのジョイン・カラムに複数の一致する値がある場合、2 番目の更新は、最初の更新による新しい値ではなく、元の値に基づいて行われます。結果は、処理の順序に依存するので、予測できなくなります。次のジョインについて考えてください。

```
update titles set total_sales = total_sales + qty
  from titles t, salesdetail sd
  where t.title_id = sd.title_id
```

**total\_sales** の値は、**titles** 内の **title\_id** ごとに、**salesdetail** の一致する 1 つのローについて、1 回だけ更新されます。クエリのジョイン順序によっては、テーブル分割または使用可能なインデックスで、結果が毎回異なる可能性があります。しかし、毎回、**salesdetail** の 1 つの値だけが **total\_sales** 値に加算されます。

ジョイン・カラムに一致する値の合計を返すことが目的である場合、次のクエリは、サブクエリを使用して正しい結果を返します。

```
update titles set total_sales = total_sales +
  (select isnull (sum (qty),0)
   from salesdetail sd
   where t.title_id = sd.title_id)
  from titles t
```

### 文字データに対する *update* の使用

- 可変長文字データ、つまり *text* カラムまたは *unitext* カラムを空文字列 ("") で更新すると、シングル・スペースが挿入されます。固定長文字カラムは、定義された長さまで埋め込まれます。
- スペースだけからなる文字列の場合を除き、可変長のカラム・データからは後続スペースがすべて削除されます。スペースだけからなる文字列は、トランケートされて 1 つのスペースになります。 *string\_truncation on* を設定していない場合は、*char*、*nchar*、*unichar*、*varchar*、*univarchar* または *nvarchar* カラムの指定の長さよりも長い文字列は、暗黙的にトランケートされます。
- *text* または *unitext* カラムを更新 (*update*) すると、*text* カラムまたは *unitext* カラムが初期化され、有効なテキスト・ポインタが割り当てられ、少なくとも 1 テキスト・ページが割り当てられます。

### カーソルに対する *update* の使用

- スクロール可能カーソルは更新できません。
- カーソルを使用してローを更新するには、*declare cursor* でカーソルを定義してからカーソルをオープンしてください。カーソル名には、*Transact-SQL* パラメータまたはローカル変数は使用できません。カーソルは更新可能である必要があります。更新可能でない場合には、*Adaptive Server* はエラーを返します。カーソル結果セットへの更新は、カーソルのローが取り出されたベース・テーブルのローにも影響します。
- *update...where current of* で指定する *table\_name* または *view\_name* は、カーソルを定義する *select* 文の最初の *from* 句で指定したテーブルまたはビューでなければなりません。この *from* 句が、複数のテーブルまたはビューを (ジョインを使用して) 参照する場合は、更新されるテーブルまたはビューだけを指定できます。

更新後もカーソル位置は変わりません。他の *SQL* 文がカーソルの位置を変更しないかぎり、そのカーソル位置のローを引き続き更新できます。

- *Adaptive Server* では、カーソルの *select\_statement* のカラムのリストに指定されていないカラムでも、*select\_statement* で指定されたテーブルの一部であれば、このカラムを更新できます。ただし、*for update* で *column\_name\_list* を指定してからカーソルを宣言する場合は、更新できるのはこれらの指定カラムだけです。

## IDENTITY カラムの更新

ベース・テーブルでもビューでも、IDENTITY プロパティが設定されているカラムは更新できません。定義時に IDENTITY プロパティが設定されているカラムであるかどうかを判断するには、そのカラムのベース・テーブルに対して `sp_help` を使用します。

結果テーブルに選択される IDENTITY カラムは、IDENTITY プロパティの継承に関して次の規則に従います。

- IDENTITY カラムは、複数回選択されると、新しいテーブル内で NOT NULL として定義されます。IDENTITY プロパティは継承されません。
- 式の一部として IDENTITY カラムが選択される場合、結果カラムは IDENTITY プロパティを継承しません。結果カラムは、null を指定できるカラムが式に含まれている場合には null として作成されます。それ以外の場合は NOT null として作成されます。
- select 文に group by 句または集合関数が含まれている場合、結果カラムには IDENTITY プロパティは継承されません。IDENTITY カラムの集約が含まれているカラムは null として作成され、その他のカラムは NOT null として作成されます。
- union またはジョインを使用してテーブルに選択された IDENTITY カラムでは、IDENTITY プロパティが保持されません。テーブルに IDENTITY カラムの union および NULL カラムが含まれる場合、新しいカラムは NULL として定義されます。それ以外の場合は、新しいカラムは NOT NULL として定義されます。

## ビューを使用したデータの更新

- distinct 句で定義されたビューに update を実行することはできません。
- ビューが with check option 付きで作成された場合、ビューで更新される各ローは、更新操作後もビュー上に表示されている必要があります。たとえば stores\_cal ビューには、stores テーブルで、state の値が "CA" のローがすべて含まれています。with check option 句により、各 update 文がビューの選択基準を満たしているかどうかチェックされます。

```
create view stores_cal
as select * from stores
where state = "CA"
with check option
```

次のような update 文は、state を "CA" 以外の値に変更すると失敗します。

```
update stores_cal
set state = "WA"
where store_id = "7066"
```

- ビューを `with check option` 付きで作成する場合、ベース・ビューから抽出されたビューはビューの選択基準を満たしている必要があります。抽出ビューで更新されたローのそれぞれは、更新後もベース・ビューから参照できなければなりません。

`stores_cal` から抽出されたビュー `stores_cal30` を考えてみます。新しいビューには、"Net 30" という支払期限が指定されているカリフォルニア州にある支店の情報が含まれます。

```
create view stores_cal30
as select * from stores_cal
where payterms = "Net 30"
```

`stores_cal` は `with check option` 付きで作成されているため、`stores_cal30` で更新されるローはすべて、更新後も `stores_cal` に表示される必要があります。`state` を "CA" 以外の値に変更するローは、拒否されます。

`stores_cal30` には、それ自体の `with check option` 句がないことがわかります。したがって、`payterms` が "Net 30" 以外の値のローを `stores_cal30` で更新できます。たとえば、`stores_cal30` に表示されないローの場合でも、次の `update` 文は正常に実行されます。

```
update stores_cal30
set payterms = "Net 60"
where stor_id = "7067"
```

- 次の条件が両方とも真でないかぎり、複数のテーブルのカラムをジョインするビューを通してローを更新することはできません。
  - ビューに `with check option` 句がない
  - 更新するカラムがすべて同じベース・テーブルに属する
- `update` 文は、`with check option` 句を含んだジョイン・ビューで使用できます。影響を受けるカラムが、複数のテーブルからのカラムを含む式の `where` 句に表示される場合、更新は失敗します。
- ジョイン・ビューを通して更新が実行された場合、影響を受けるカラムは、同じベース・テーブルに属していなければなりません。

*index*、*prefetch*、または *lru | mru* の使用

*index*、*prefetch*、*lru | mru* は、Adaptive Server オプティマイザが選択した内容を上書きします。これらのオプションは注意して使用し、*set statistics io on* を使用してパフォーマンスへの影響を常にチェックしてください。これらのオプションの使用の詳細については、『パフォーマンス&チューニング・ガイド』を参照してください。

*readpast* の使用

- *readpast* オプションは、データオンリーロック・テーブルのみに適用されます。全ページロック・テーブルに指定されている *readpast* は無視されます。
- *readpast* オプションは、*holdlock* オプションと同時に使用できません。両方のオプションを同じ *select* コマンドに指定すると、エラーが生成されて、コマンドは終了します。
- セッション・ワイドな独立性レベルが 3 の場合、*readpast* オプションは無視されます。
- セッションのトランザクション独立性レベルが 0 の場合、*readpast* を使用する *update* コマンドは警告メッセージを発行しません。データページロック・テーブルの場合、これらのコマンドは、互換性のないロックによってロックされていない全ページのすべてのローを修正します。データローロック・テーブルの場合、これらのコマンドは、互換性のないロックによってロックされていないすべてのローに影響します。
- *readpast* オプション付きの *update* コマンドが複数の *text* カラムに適用され、そのときにチェックされた最初の *text* カラムに非両立ロックが保持されている場合、読み飛ばしロックはそのローを省略します。カラムに互換性のないロックが設定されていない場合、コマンドはロックを取得し、カラムを修正します。その後、ローの後続の *text* カラムに互換性のないロックが設定されている場合、ロックを取得し、カラムを修正できるまで、コマンドはブロックします。
- *readpast* の詳細については、『パフォーマンス&チューニング・ガイド』を参照してください。

## 標準規格

## ANSI SQL — 準拠レベル：初級レベル

Transact-SQL 拡張機能は次のとおりです。

- *from* 句や、テーブルまたはカラムの修飾名の使用は、FIPS フラガによって検出される Transact-SQL 拡張機能です。ターゲット・リストに式が含まれているビューまたはジョイン・ビューでの更新は、FIPS フラガによって通知されず、実行時まで検出できない Transact-SQL 拡張機能です。



- 変数の使用
- readpast

パーミッション

set ansi\_permissions が on の場合、更新するテーブルの update パーミッションに加えて、where 句に表示されるすべてのカラムおよび set 句の右側のすべてのカラムの select パーミッションが必要です。デフォルトでは、ansi\_permissions は off です。

細密なパーミッションの設定によって異なる update のパーミッション・チェックについて、次に説明します。

細密なパーミッションが有効	細密なパーミッションが有効の場合、テーブルまたはビューの所有者であるか、update パーミッションが付与されたユーザであることが必要。
細密なパーミッションが無効	細密なパーミッションが無効の場合、テーブルまたはビューの所有者、update パーミッションを持つユーザ、または sa_role が付与されたユーザであることが必要。

監査

sysaudits の event カラムと extrainfo カラムの値は次のとおりです。

イベント	監査オプション	監査対象のコマンドまたはアクセス	extrainfo の情報
70	update	テーブルの update	<ul style="list-style-type: none"> <li>• Roles — 現在のアクティブな役割</li> <li>• Keywords または options — update または writetext</li> <li>• Previous value — NULL</li> <li>• Current value — NULL</li> <li>• Other information — NULL</li> <li>• Proxy information — set proxy が有効な場合は元のログイン名</li> </ul>
71	update	ビューの update	<ul style="list-style-type: none"> <li>• Roles — 現在のアクティブな役割</li> <li>• Keywords または options — update または writetext</li> <li>• Previous value — NULL</li> <li>• Current value — NULL</li> <li>• Other information — NULL</li> <li>• Proxy information — set proxy が有効な場合は元のログイン名</li> </ul>

参照

コマンド [alter table](#), [create default](#), [create index](#), [create rule](#), [create trigger](#), [insert](#), [where 句](#)

システム・プロシージャ [sp\\_bindefault](#), [sp\\_bindrule](#), [sp\\_help](#), [sp\\_helppartition](#), [sp\\_helpindex](#), [sp\\_unbindefault](#), [sp\\_unbindrule](#)

## update all statistics

**説明** インデックスの有無に関係なく、全カラムのヒストグラムなど、特定のテーブルのすべての統計情報を更新します。update all statistics は単一のデータ・パーティションに対して実行できます。

**構文**

```
update all statistics
    table_name [partition data_partition_name ]
    [using step values]
    [with consumers = consumers]
    [, sampling=N [percent]]
    [, no_hashing | partial_hashing | hashing]
      [, max_resource_granularity = N [percent]]
      [, histogram_tuning_factor = int ]
    [, print_progress = int]
```

**パラメータ**

*table\_name*

統計を更新するテーブルの名前です。

*data\_partition\_name*

更新するパーティション名です。データ・パーティションの各ローカル・インデックス・パーティションの統計値が更新されます。グローバル・インデックスの統計値は更新されません。

*using step values*

ヒストグラムのステップの数を指定します。統計が存在しないカラムのデフォルト値は 20 です。このデフォルトを変更するには、**sp\_configure** を使用して、*number of histogram steps* パラメータを変更します。カラムの統計が **sysstatistics** にすでに存在する場合、デフォルト値は現在のステップ数になります。

このステップは、パーティション・テーブルの各パーティションに適用されます。たとえば、**update index statistics** では、統計を更新するスキャンに関係する各データ・パーティションとインデックス・パーティションに、デフォルト値の 20 ステップを使用します。グローバル統計がグローバル・インデックスのインデックス・スキャンによって生成される場合、デフォルトで 20 ステップが適用されます。データ・スキャンまたはローカル・インデックス・スキャンによってパーティション統計が生成される場合、各パーティションにはデフォルトで 20 ステップが適用されます。

*using step values* で指定するヒストグラム・ステップを *M* とし、*histogram\_tuning\_factor* パラメータを *N* とすると、**update index statistics** では  $0 \sim M \cdot N$  のステップ数が使用されます。実際のステップ数は、**update index statistics** によって分離された頻度セルの数と範囲セルの存在の有無によって決まります。

`with consumers = consumers`

`column_list` が指定されていて、並列クエリ処理が有効な場合にソートに使用されるコンシューマ・プロセスの数を指定します。

`consumers` オプションは、1 つのデータ・パーティションに対する統計更新時のソートに適用される並列度を指定します。たとえば、カラム・リストのある `update statistics` をデータ・パーティションが 3 つあるテーブルに適用すると、3 つのパーティションそれぞれのデータは別々にソートされ、それぞれのソートの際には `consumers` オプションが適用されます。この 3 つのソート自体は並列実行が行われません。

---

**注意** `max parallel degree` 設定パラメータの値は、`with consumer` の値より大きくする必要があります。たとえば、`with consumers` が 2 に設定されている場合、`max parallel degree` は 3 以上にする必要があります。

---

`with sampling = N percent`

統計を収集するときにランダムなサンプリングの対象とするカラムの割合を指定します。`N` は、1 ~ 100 の整数値です。

`[no_hashing | partial_hashing | hashing]`

`update all statistics` が収集するハッシュベース統計のレベルを指定します。次のいずれかになります。

- `no_hashing` — `updates all statistics` は、バージョン 15.7 より前の Adaptive Server のアルゴリズムを使用して、ソートベースの統計を収集します。
- `partial_hashing` — `updates all statistics` はユニーク値の数が 65536 未満のカラムにアルゴリズムを使用します。`updates all statistics` で検出されたユニークなカラム数がスレッショルドの 65536 以上である場合は、ソートによる追加のスキャンが実行されます。
- `hashing` — `updates all statistics` は低ドメインと高ドメインのハッシュを使用して、ヒストグラムを作成します。

これらのパラメータのデフォルトは、`update statistics hashing` の設定値です。

`max_resource_granularity = N percent`

`update all statistics` およびハッシュで使用される `tempdb` バッファ・キャッシュの量を制限します。

`histogram_tuning_factor = integer`

`update all statistics` の分布密度を決定します。

```
print_progress = int
```

`update all statistics` で進行メッセージを表示するかどうかを決定します。

- 0 – (デフォルト) 進行メッセージは表示されません。
- 1 – 進行メッセージを表示します。

例

**例 1** `salesdetail` テーブルのすべての統計を更新します。

```
update all statistics salesdetail
```

**例 2** `salesdetail` テーブルの `smallsales` パーティションのすべての統計値を更新します。

```
update all statistics salesdetail partition smallsales
```

**例 3** `authors` テーブルのハッシュベースの統計を更新します。

```
update index statistics authors with hashing
```

使用法

- `update all statistics` は、指定したテーブルの統計情報をすべて更新します。Adaptive Server にはテーブル内のページ分布に関する統計値が保存されており、クエリ処理で並列スキャンを使用するかどうかの判断が、この統計値をもとに行われます。クエリの最適化は保存されている統計値の精度によって異なります。
- ヒストグラム統計は、先行カラムのインデックス・スキャン、ワーク・テーブルへのカラムの射影とその後のソート、またはハッシュの使用によるスキャンごとの複数のカラムのヒストグラムの同時生成によって、カラムごとに作成されます。
- 統計が更新されたインデックスのカラムのすべてのプレフィックス・サブセットについて密度の統計値を作成します。たとえば、カラム `c1`、`c2`、`c3` にインデックスがある場合、プレフィックス・サブセットは `(c1, c2)` と `(c1, c2, c3)` です。
- `update all statistics` を 1 つのデータ・パーティションに対して実行すると、インデックス・スキャンを使用してローカル・インデックスの先行カラムごとにヒストグラムが生成されます。ハッシュベース統計の収集が有効化されていると、インデックス・スキャン時だけでなく、小さい属性のすべてについても統計が収集されます。グローバル・インデックスの先行カラムを含め、他のすべてのカラムについては、`update all statistics` によってデータ・スキャンの実行とその後のソートが行われるか、ハッシュベースの統計収集を使用するデータ・スキャンが実行されます。

ハッシュベース統計の利点は、1 回のデータ・スキャンですべてのカラムのヒストグラムを収集できることです。ソートベースの統計では、カラムごとのスキャンが使用されます。

- **update statistics** コマンドは、パーティション固有の統計を作成します。グローバルな統計値は、パーティション統計の作成時に暗黙に作成されます。パーティション統計は、グローバル統計を作成する入力値として使用され、パーティション単位での DDL 操作を可能にします。グローバル統計はオブティマイザで使用されません。
- **update all statistics** は、テーブルの各データ・パーティションとインデックス・パーティションの **systabstats** に保管されたテーブル統計を再生成および更新します。**update all statistics** コマンドを特定のデータ・パーティションに対して実行すると、そのデータ・パーティションとローカル・インデックス・パーティションについてだけテーブル統計が生成され、更新されます。グローバル・インデックスはスキップされます。

#### ハッシュベースの統計の使用

- **partial\_hashing** パラメータは、ソートで生成されるヒストグラムと同じ程度の精度と品質のヒストグラムを生成できる場合のみ、カラムにハッシュを使用します (低ドメイン・ケース)。これ以外の場合、**partial\_hashing** パラメータは、高ドメイン・ケースのソートを使用します。
- 一部の状況では (高ドメイン・ケース)、ハッシュで生成されるヒストグラムの精度がソートより低くなります。
- ハッシュベースの統計では、**tempdb** ディスク領域やソートで使用するプロシージャ・キャッシュの必要がありませんが、**tempdb** バッファ・キャッシュ・メモリは大量に使用されます。
- **no\_hashing** パラメータで使用される大規模なソートでは、ソートに必要なプロシージャ・キャッシュ・メモリを解放するため、ステートメント・キャッシュの文とストアド・プロシージャが除去される場合があります。
- **max\_resource\_granularity** は、**hashing** または **partial\_hashing** に使用される **tempdb** バッファ・キャッシュ・メモリの量を制限します。これは、**no\_hashing** パラメータやソートで使用されるメモリ量には影響しません。

- カラムに以前のヒストグラムが存在し、高ドメイン・カラムであることが示される場合に、`partial_hashing` パラメータを指定すると、Adaptive Server はそのカラムにソートベースの統計が必要であるとみなします。カラムに以前のヒストグラムが存在しない場合、Adaptive Server は高ドメイン制限に達するまで、そのカラムが低ドメインであると判断します。

### 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

### パーミッション

`update all statistics` を実行するには、テーブル所有者であるか、テーブルに対する `update statistics` パーミッションを持つユーザであることが必要です。

### 参照

コマンド [update statistics](#), [update index statistics](#), [update statistics](#), [update table statistics](#)

## update index statistics

**説明** インデックス上にある、すべてのカラムの統計を更新します。

**構文**

```
update index statistics
    table_name [[partition data_partition_name] |
    [ [index_name [partition index_partition_name]]]
    [using step values]
    [with consumers = consumers] [, sampling=N [percent]]
    [, no_hashing | partial_hashing | hashing]
    [, max_resource_granularity = N percent]]
    [, histogram_tuning_factor = int]
    [, print_progress = int]
```

**パラメータ**

*table\_name*

`update statistics` とともに使用する場合、*table\_name* は、インデックスが関連付けられるテーブルの名前になります。Transact-SQL ではインデックス名をデータベース内でユニークにする必要がないため、*table\_name* が必要になります。

*data\_partition\_name*

更新するパーティション名です。データ・パーティションの各ローカル・インデックス・パーティションの統計値が更新されます。グローバル・インデックスの統計値は更新されません。

*index\_name*

更新するインデックス名です。インデックス名が指定されないと、指定されたテーブルのすべてのインデックスについて、分散統計値が更新されます。

*index\_partition\_name*

更新するインデックス・パーティション名です。

**using step values**

ヒストグラムのステップの数を指定します。統計が存在しないカラムのデフォルト値は 20 です。このデフォルトを変更する必要がある場合は、`sp_configure` を使用して、`number of histogram steps` パラメータを変更します。カラムの統計が `sysstatistics` にすでに存在する場合、デフォルト値は現在のステップ数になります。

このステップは、パーティション・テーブルの各パーティションに適用されます。たとえば、`update index statistics` では、統計を更新するスキャンに関する各データ・パーティションとインデックス・パーティションに、デフォルト値の 20 ステップを使用します。グローバル統計がグローバル・インデックスのインデックス・スキャンによって生成される場合、デフォルトで 20 ステップが適用されます。データ・スキャンまたはローカル・インデックス・スキャンによってパーティション統計が生成される場合、各パーティションにはデフォルトで 20 ステップが適用されます。

`using step values` で指定するヒストグラム・ステップを  $M$  とし、`histogram_tuning_factor` パラメータを  $N$  とすると、`update index statistics` では  $0 \sim M*N$  のステップ数が使用されます。実際のステップ数は、`update index statistics` によって分離された頻度セルの数と範囲セルの存在の有無によって決まります。

**with consumers = consumers**

`column_list` が指定されていて、並列クエリ処理が有効な場合にソートに使用されるコンシューマ・プロセスの数を指定します。

`consumers` オプションは、1 つのデータ・パーティションに対する統計更新時のソートに適用される並列度を指定します。たとえば、カラム・リストのある `update statistics` をデータ・パーティションが 3 つあるテーブルに適用すると、3 つのパーティションそれぞれのデータは別々にソートされ、それぞれのソートの際には `consumers` オプションが適用されます。この 3 つのソート自体は並列実行が行われません。

---

**注意** `max parallel degree` 設定パラメータの値は、`with consumer` の値より大きくする必要があります。たとえば、`with consumers` が 2 に設定されている場合、`max parallel degree` は 3 以上にする必要があります。

---



`with sampling = N percent`

統計を収集するときにランダムなサンプリングの対象とするカラムのパーセンテージを指定します。*N*は、1 ~ 100 の整数値です。

---

**注意** 現在、ハッシュではサンプリングがサポートされていません。

---

`[no_hashing | partial_hashing | hashing]`

`update index statistics` が収集するハッシュベース統計のレベルを指定します。次のいずれかになります。

- `no_hashing` (デフォルト) `update index statistics` は、バージョン 15.7 より前の Adaptive Server のアルゴリズムを使用して、ソートベース統計を収集します。
- `partial_hashing` — `update index statistics` はユニーク値の数が 65536 未満のカラムにアルゴリズムを使用します。`update index statistics` で検出されたユニークなカラム数がスレッシュホールドの 65536 以上である場合は、ソートによる追加のスキャンが使用されます。
- `hashing` — `update index statistics` は低ドメインと高ドメインのハッシュを使用して、ヒストグラムを作成します。

`max_resource_granularity = N percent`

`update index statistics` およびハッシュで使用される `tempdb` バッファ・キャッシュの量を制限します。

`with histogram_tuning_factor = integer`

`update index statistics` の分布密度を決定します。

`print_progress = int`

`update index statistics` で進行メッセージを表示するかどうかを決定します。

- 0 — (デフォルト) 進行メッセージは表示されません。
- 1 — 進行メッセージを表示します。

例

**例 1** `authors` テーブルのすべてのインデックスにあるすべてのカラムの統計を生成します。

```
update index statistics authors
```

**例 2** `authors` テーブルの `au_names_ix` インデックスにあるすべてのカラムの統計を生成します。

```
update index statistics authors au_names_ix
```

**例 3** サンプリング・レートを 20 パーセントにして、`au_names_ix` インデックスのすべての内部カラムに関する統計値を生成します。

```
update index statistics authors au_names_ix
with sampling = 20 percent
```

`au_names_ix` の先行カラムの統計は、インデックス・ページのフル・スキャンを使用して収集されます。サンプリングはこのカラムには適用されません。

**例 4** インデックス・パーティションにあるすべてのカラムの統計を生成します。

```
update index statistics publishers publish1_idx
partition p1
```

#### 使用法

- `update index statistics` にテーブル名とインデックス名を指定して実行すると、指定したインデックス内のすべてのカラムの統計が更新されます。`update index statistics` にテーブル名だけを指定して実行すると、テーブルのインデックス内にあるすべてのカラムの統計が更新されます。
- `update index statistics` を大きなテーブルに対して実行するときに、`tempdb` がコマンドを処理できるだけの十分な大きさでないと、コマンドはエラー番号 1105 で失敗します。
- インデックスが作成されていないカラムまたはインデックスの先行カラムではないカラムの名前を指定すると、インデックスを作成しないでそのカラムの統計が生成されます。
- 統計が更新されているインデックスの各カラムに、ヒストグラムの統計が作成されます。
- 統計が更新されたインデックスのカラムのすべてのプレフィックス・サブセットについて密度の統計値を作成します。
- 特定のパーティションに `update index statistics` を使用すると、グローバル統計も暗黙に更新されます。
- パーティション統計は、グローバル統計作成の入力値として使用され、パーティション単位での DDL 操作を可能にします。グローバル統計はオプティマイザで使用されます。

- `update index statistics` は、コマンドで更新するテーブルの各データ・パーティションとインデックス・パーティションの `sysstat` に保管されたテーブル統計も再生成および更新します。`update index statistics` コマンドを特定のデータ・パーティションに対して実行すると、そのデータ・パーティションとローカル・インデックス・パーティションについてだけテーブル統計が生成され、更新されます。グローバル・インデックスはスキップされます。`update index statistics` コマンドを特定のインデックス・パーティションに対して実行すると、そのインデックス・パーティションについてだけテーブル統計が更新されます。
- `with consumers` 句は、RAID デバイス上の分割されたテーブルでの使用を目的として設計されています。このデバイスは、`Adaptive Server` では単一の I/O デバイスのようにみえますが、並列ソートに必要な高いスループットを実現します。『パフォーマンス & チューニング・ガイド』の「並列ソート」を参照してください。
- `update index statistics` コマンドによって生成される一連の統計更新オペレーションでは、同等のインデックス・レベルおよびカラム・レベルのコマンドと同じロック、スキャン、およびソートが使用されます。たとえば、`salesdetail` テーブルの `salesdetail(stor_id, ord_num, title_id)` にノンクラスタード・インデックス `sales_det_ix` がある場合、`update index statistics salesdetail` コマンドは次の `update statistics` 操作を実行します。

```
update statistics salesdetail sales_det_ix
update statistics salesdetail (ord_num)
update statistics salesdetail (title_id)
```

#### ハッシュベースの統計の使用

- `partial_hashing` パラメータは、ソートで生成されるヒストグラムと同じ程度の精度と品質のヒストグラムを生成できる場合のみ、カラムにハッシュを使用します(低ドメイン・ケース)。これ以外の場合、`partial_hashing` パラメータは、高ドメイン・ケースのソートを使用します。
- 一部の状況では(高ドメイン・ケース)、ハッシュで生成されるヒストグラムの精度がソートより低くなります。
- ハッシュベースの統計では、`tempdb` ディスク領域やソートで使用するプロシージャ・キャッシュの必要がありませんが、`tempdb` バッファ・キャッシュ・メモリは大量に使用されます。

- `no_hashing` パラメータで使用される大規模なソートでは、ソートに必要なプロシージャ・キャッシュ・メモリを解放するため、ステートメント・キャッシュの文とストアド・プロシージャが除去される場合があります。
- `max_resource_granularity` は、`hashing` または `partial_hashing` に使用される `tempdb` バッファ・キャッシュ・メモリの量を制限します。これは、`no_hashing` パラメータやソートで使用されるメモリ量には影響しません。
- カラムに以前のヒストグラムが存在し、高ドメイン・カラムであることが示される場合に、`partial_hashing` パラメータを指定すると、Adaptive Server はそのカラムにソートベースの統計が必要であるとみなします。カラムに以前のヒストグラムが存在しない場合、Adaptive Server は高ドメイン制限に達するまで、そのカラムが低ドメインであると判断します。

### 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

### パーミッション

`update index statistics` を実行するには、テーブル所有者であるか、テーブルに対する `update statistics` パーミッションを持つユーザであることが必要です。

### 参照

コマンド [delete statistics](#), [update all statistics](#), [update statistics](#), [update table statistics](#)

マニュアル 『パフォーマンス&チューニング・シリーズ』

## update statistics

説明	指定したインデックス、インデックス内のすべてのカラム、テーブル内のすべてのカラム、またはパーティション内のすべてのカラムのキー値の分布に関する情報を更新し、グローバル・ノンクラスタード・インデックスのデータ変更カウンタをリセットします。
構文	<pre>update statistics <i>table_name</i>   [[<i>partition data_partition_name</i>]    [<i>(column1, column2, ...)</i>   (<i>column1</i>), (<i>column2</i>), ...]      <i>index_name</i> [<i>partition index_partition_name</i>]   [using step values   [out_of_range [on   off] default]]]   [with consumers = <i>consumers</i>][, sampling=<i>N</i> percent]   [, no_hashing   partial_hashing   hashing]   [, max_resource_granularity = <i>N</i> [percent]]   [, histogram_tuning_factor = <i>inf</i>]   [, print_progress = <i>inf</i>]</pre>
パラメータ	<p><i>table_name</i></p> <p>update statistics とともに使用する場合、<i>table_name</i> は、インデックスが関連付けられるテーブルの名前になります。Transact-SQL ではインデックス名をデータベース内でユニークにする必要がないため、<i>table_name</i> が必要になります。</p> <p><i>index_name</i></p> <p>更新するインデックス名です。インデックス名が指定されないと、指定されたテーブルのすべてのインデックスについて、分散統計値が更新されます。</p> <p><i>data_partition_name</i></p> <p>更新するパーティション名です。データ・パーティションの各ローカル・インデックス・パーティションの統計値が更新されます。グローバル・インデックスの統計値は更新されません。</p> <p><i>index_partition_name</i></p> <p>更新するインデックス・パーティション名です。</p> <p><i>(column1, column2, ...)</i></p> <p>ローカル・インデックスが作成されカラム・タプルに含まれる場合に作成されるものと同じ統計を作成します。つまり、<i>column1</i> のヒストグラムを作成し、すべてのカラム・プレフィクス (<i>column1</i>、<i>column2</i>)、(<i>column1</i>, <i>column2</i>, <i>column3</i>) の複数属性分布を作成します。</p> <p><i>(column1)</i>, <i>(column2)</i></p> <p>update statistics が実行されるカラムのカンマ区切りリストです。1 つの update statistics コマンドでリスト内の各カラムのヒストグラムが作成されますが、複数属性分布は作成されません。</p>

**using step values**

ヒストグラムのステップの数を指定します。統計が存在しないカラムのデフォルト値は 20 です。このデフォルトを変更する必要がある場合は、`sp_configure` を使用して、`number of histogram steps` パラメータを変更します。カラムの統計が `sysstatistics` にすでに存在する場合、デフォルト値は現在のステップ数になります。

このステップは、パーティション・テーブルの各パーティションに適用されます。たとえば、`update statistics` では、デフォルト値の 20 ステップを、統計を更新するスキャンに関係する各データ・パーティションとインデックス・パーティションに使用します。グローバル統計がグローバル・インデックスのインデックス・スキャンによって生成される場合、デフォルトで 20 ステップが適用されます。データ・スキャンまたはローカル・インデックス・スキャンによってパーティション統計が生成される場合、各パーティションにはデフォルトで 20 ステップが適用されます。

`using step values` で指定するヒストグラム・ステップを  $M$  とし、`histogram_tuning_factor` パラメータを  $N$  とすると、`update statistics` では  $0 \sim M \cdot N$  のステップ数が使用されます。実際のステップ数は、`update statistics` によって分離された頻度セルの数と範囲セルの存在の有無によって決まります。

**out\_of\_range [on | off | default]**

急速に拡大するテーブルのカラム統計は、`update statistics` が完了すると古くなり、カラムのヒストグラムで示されるよりも選択される範囲が大きくなる範囲外 SARG (検索句) が発生することがあります。範囲外 SARG の選択性は 0 です。`out_of_range` ヒストグラム調整機能によってカラムのヒストグラムが調整され、このような SARG に適切な選択性値が割り当てられます。

範囲外 SARG のヒストグラム調整は、デフォルトでサーバワイドで有効化されています。

`out_of_range [on | off | default]` は、カラム・レベルの範囲外ヒストグラム調整を指定します。次のいずれかになります。

- `on` — `column_name` の範囲外ヒストグラム調整を有効にします。
- `off` — `column_name` の範囲外ヒストグラム調整を無効にします。
- `default` — トレース・フラグ 15355 の値に応じて範囲外ヒストグラム調整が作用します。:
  - トレースフラグ 15355 がオンの場合は範囲外ヒストグラム調整を無効にします。
  - トレースフラグ 15355 がオフの場合は範囲外ヒストグラム調整を有効にします。

`with consumers = consumers`

`column_list` が指定されていて、並列クエリ処理が有効な場合にソートに使用されるコンシューマ・プロセスの数を指定します。

`consumers` オプションは、1 つのデータ・パーティションに対する統計更新時のソートに適用される並列度を指定します。たとえば、カラム・リストのある `update statistics` をデータ・パーティションが 3 つあるテーブルに適用すると、3 つのパーティションそれぞれのデータは別々にソートされ、それぞれのソートの際には `consumers` オプションが適用されます。この 3 つのソート自体は並列実行が行われません。

---

**注意** `max parallel degree` 設定パラメータの値は、`with consumer` の値より大きくする必要があります。たとえば、`with consumers` が 2 に設定されている場合、`max parallel degree` は 3 以上にする必要があります。

---

`with sampling = N percent`

統計を収集するときにランダムなサンプリングの対象とするカラムのパーセンテージを指定します。`N` の値は、1 ~ 100 の数値です。サンプリングは、次のすべてのタイプの `update statistics` に適用されます。

- `update statistics table_name (col_name)`
- `update index statistics`
- `update all statistics`

`index`

インデックス内のカラムすべての統計が更新されるように指定します。

`[no_hashing | partial_hashing | hashing]`

`update statistics` が収集するハッシュベース統計のレベルを指定します。次のいずれかになります。

- `no_hashing` — (デフォルト) `update statistics` は、バージョン 15.7 より前の Adaptive Server のアルゴリズムを使用して、ソートベース統計を収集します。
- `partial_hashing` — `update statistics` はユニーク値の数が 65536 未満のカラムにアルゴリズムを使用します。`update statistics` で検出されたユニークなカラム数がスレッショルドの 65536 以上である場合は、ソートによる追加のスキャンが使用されます。
- `hashing` — `update statistics` は低ドメインと高ドメインのハッシュを使用して、ヒストグラムを作成します。

```
max_resource_granularity = N percent
```

update statistics およびハッシュで使用される tempdb バッファ・キャッシュの量を制限します。

```
with histogram_tuning_factor = integer
```

update statistics の分布密度を決定します。

```
print_progress = int
```

update statistics で進行メッセージを表示するかどうかを決定します。

- 0 – (デフォルト) 進行メッセージは表示されません。
- 1 – 進行メッセージを表示します。

例

**例 1** titles テーブルの price カラムの統計を生成します。

```
update statistics titles (price) using 40 values
```

**例 2** データ・パーティション smallsales の統計値を更新します。

Adaptive Server は、このデータ・パーティションの各先行カラムのヒストグラムおよび各ローカル・インデックスの複合カラムの密度を作成します。グローバル・インデックスの統計値は更新されません。

```
update statistics titles partition smallsales
```

**例 3** データ・パーティション smallsales の統計値を更新します。

Adaptive Server は、カラム col1 のヒストグラムを作成し、col1 と col2 の複合カラムの密度を作成します。

```
update statistics titles partition smallsales (col1,  
col2)
```

**例 4** カラムで out\_of\_range (範囲外) SARG が検出されると、オプティマイザによってカラムのヒストグラムが調整され、適切な選択性値が out-of-range 句に割り当てられます。

```
update statistics TOFO_FUOP_ORD(OrdDt) using  
out_of_range on
```

**例 5** トレース・フラグ 15355 がオンの場合、カラムのヒストグラムで範囲外 SARG の調整は行われません。

```
update statistics TOFO_FUOP_ORD(OrdDt) using  
out_of_range default
```

**例 6** histogram\_tuning\_factor を 5 パーセントとして、authors テーブルに対して update statistics を実行します。

```
update index statistics authors with  
histogram_tuning_factor = 5
```



## 使用法

- **Adaptive Server** は、各インデックス内のキー値の分散についての統計値を保持し、これらの統計値を使用して、どのインデックスをクエリ処理で使用するかを決定します。
- データを含むテーブルにノンクラスタード・インデックスを作成すると、新しいインデックスに対して、**update statistics** が自動的に実行されます。データを含むテーブルにクラスタード・インデックスを作成すると、**update statistics** は、すべてのインデックスに対して自動的に実行されます。
- **Running** 空のテーブルに **update statistics** を実行しても、システム・テーブルに影響しません。
- クエリの最適化は、統計の正確さによって異なります。インデックスのキー値に大きな変更がある場合は、**update statistics** をそのインデックスまたはカラムに対して再実行する必要があります。インデックス・カラムで大量のデータが追加、変更、または削除された場合 ( キー値の分布が変化したと考えられる場合 ) には、**update statistics** コマンドを実行してください。
- 多数のローがあるシステム・テーブルでは **update statistics** も実行する必要があります。ユーザ・テーブルでコマンドを実行するパーミッションがある場合も、操作方法はシステム・テーブルと同じです。統計情報がない場合は、システム・ストアード・プロシージャが適切に実行されない可能性があります。
- **update statistics** をデータ・パーティションに対して実行した場合、グローバル・インデックスはスキップされます。
- **update statistics** にテーブル名とインデックス名を指定して実行すると、インデックスの先行するカラムの統計が更新されます。**update statistics** にテーブル名だけを指定して実行すると、テーブル内のすべてのインデックスの先行するカラムの統計が更新されます。
- カンマ区切りリスト (col1), (col2)... が使用され、ハッシュが有効化されている場合、リソース粒度を超過していなければ 1 回のスキャンで統計を収集できます。ソートが有効化されている場合は、それぞれに対して 1 回のスキャンが使用されます。部分的なハッシュを使用すると、リソース粒度を超過していなければ、低ドメイン・カラムに対してスキャンが 1 回行われ、高ドメイン・カラムの場合はそれぞれのソートで 1 回ずつスキャンが行われます (つまり、カラムが 3 つの場合は 3 回のソート)。
- インデックスが作成されていないカラムまたはインデックスの先行カラムではないカラムの名前を指定すると、インデックスを作成しないでそのカラムの統計が生成されます。

- カラム・リスト内の複数のカラムを指定すると (たとえば、(col1, col2, ... ))、最初のカラムのヒストグラム、およびカラムのリストのプレフィクス・サブセットすべての密度統計が生成または更新されます。この場合、ハッシュベースの統計は使用できません。
- `update statistics` を使用して、クラスタード・インデックスの非先行カラムおよびインデックスが作成されないカラムの統計を生成する場合、`update statistics` はテーブルをスキャンしてソートを実行します。

カラム・リスト ((col1), (col2)...、(col1, col2, col3) など) を指定した場合を除き、Adaptive Server では `update statistics` のサンプリングが無視されます。カラム・リスト以外のサンプリングが必要な状況では、`update all statistics` または `update index statistics` を使用してください。カラム・リストを指定し、それらのカラムがクラスタード・インデックスの非先行カラムおよびインデックスが作成されないカラムである場合、`update statistics` はテーブルをスキャンしてソートを実行するか、ハッシュベースのアルゴリズムを使用します。

- 特定のパーティションに `update statistics` を使用すると、グローバル統計も暗黙に更新されます。
- `update statistics` で `consumers` や `sampling` などの他のオプションとともに `out_of_range` オプションを使用すると、Adaptive Server でエラー 16015 が発生します。

現在カラム・レベルの統計がないカラムに対して `out_of_range` オプションを指定すると、エラー 16016 が発生します。

- `update statistics` は、コマンドで更新するテーブルの各データ・パーティションとインデックス・パーティションの `systabstats` に保管されたテーブル統計を再生成および更新します。`update statistics` コマンドを特定のデータ・パーティションに対して実行すると、そのデータ・パーティションとローカル・インデックス・パーティションについてだけテーブル統計が生成され、更新されます。グローバル・インデックスはスキップされます。`update statistics` コマンドを特定のインデックス・パーティションに対して実行すると、そのインデックス・パーティションについてだけテーブル統計が更新されます。

- `with consumers` 句は、RAID デバイス上の分割されたテーブルでの使用を意図して設計されています。このデバイスは、Adaptive Server では単一の I/O デバイスのようにみえますが、並列ソートに必要な高いスループットを実現します。『パフォーマンス & チューニング・シリーズ：クエリ処理と抽象プラン』の「パフォーマンス改善のための統計値の使用」を参照してください。
- 表 1-36 は、`update statistics` の実行時に行われるスキャンのタイプ、取得されるロックのタイプ、およびソートが必要な時点を示しています。

表 1-36: `update statistics` 実行中のロック、スキャン、およびソート

次のものを指定した <code>update statistics</code>	実行されるスキャンとソート	ロック
テーブル名		
全ページロック・テーブル	各ノックラスタード・インデックスのテーブル・スキャンとリーフ・レベル・スキャン	レベル 1 (現在のページに対する意図的共有テーブル・ロック、共有ロック)
データオンリーロック・テーブル	各ノックラスタード・インデックスと、もしあればそのクラスタード・インデックスのテーブル・スキャンとリーフ・レベル・スキャン	レベル 0 (ダーティ・リード)
テーブル名とクラスタード・インデックス名		
全ページロック・テーブル	テーブル・スキャン	レベル 1 (現在のページに対する意図的共有テーブル・ロック、共有ロック)
データオンリーロック・テーブル	リーフ・レベル・インデックス・スキャン <sup>1</sup>	レベル 0 (ダーティ・リード)
テーブル名とノックラスタード・インデックス名		
全ページロック・テーブル	リーフ・レベル・インデックス・スキャン <sup>1</sup>	レベル 1 (現在のページに対する意図的共有テーブル・ロック、共有ロック)
データオンリーロック・テーブル	リーフ・レベル・インデックス・スキャン <sup>1</sup>	レベル 0 (ダーティ・リード)
テーブル名とカラム名		
全ページロック・テーブル	テーブル・スキャン (ワーク・テーブルを作成してワーク・テーブルをソートする)	レベル 1 (現在のページに対する意図的共有テーブル・ロック、共有ロック)
データオンリーロック・テーブル	テーブル・スキャン (ワーク・テーブルを作成してワーク・テーブルをソートする)	レベル 0 (ダーティ・リード)

<sup>1</sup>2つ以上のインデックスに統計のサンプリングが行われるカラムが存在する場合にスキャン対象のインデックスを決定する方法については、『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「第10章 パフォーマンス改善のための統計値の使用」の「Adaptive Server がスキャンとソートを実行する場合」を参照してください。

- **update index statistics** コマンドによって生成される一連の統計更新オペレーションでは、同等のインデックス・レベルおよびカラム・レベルのコマンドと同じロック、スキャン、およびソートが使用されます。たとえば、**salesdetail** テーブルの **salesdetail(stor\_id, ord\_num, title\_id)** にノンクラスタード・インデックス **sales\_det\_ix** がある場合、**update index statistics salesdetail** コマンドは次の **update statistics** 操作を実行します。

```
update statistics salesdetail sales_det_ix
update statistics salesdetail (ord_num)
update statistics salesdetail (title_id)
```

- 以前のバージョンからのアップグレード中には、**master** データベース内のシステム・テーブルに対して **update statistics** は実行されません。インデックスはほとんどのシステム・プロシージャが問い合わせるカラム上に存在しています。通常の使用では、これらのテーブルに対して **update statistics** を実行する必要はありません。ただし、**update statistics** は、通常のテーブルではないテーブルを除いて、すべてのデータベース内にあるすべてのシステム・テーブルに対して実行できます。これらのテーブルは、クエリ実行時に内部困から鉱されるテーブルで、**syscurconfigs**、**sysengines**、**sysgams**、**syslisteners**、**syslocks**、**syslogs**、**syslogshold**、**sysmonitors**、**sysprocesses**、**syssecmechs**、**systestlog**、**systransactions** があります

**Replication Server RSSD** テーブルに対して **update statistics** を実行する必要はありません。**Replication Server** が **RSSD** テーブルにアクセスしようとしているとき、これらのテーブルで **updates statistics** を実行すると、**Replication Server** エラーが発生する可能性があります。**RSSD** テーブルとそのフォーマットは、**Replication Server** の処理に固有です。

- 次の **update statistics** パラメータは、カラムの統計が削除されるか、カラムに対して **sp\_modifystats ... REMOVE\_STICKINESS** を実行するまで、影響が及ぶすべてのカラムの値を保持し、すべての設定を上書きします。
  - **using step values**
  - **out\_of\_range**

- no\_hashing
- partial\_hashing
- hashing
- histogram\_tuning\_factor
- sampling = N percent

---

**注意** consumers と max\_resource\_granularity は、値を保持しません。

---

たとえば、次のコマンドを発行します。

```
update statistics table_name(column1) with
no_hashing
```

このテーブルに対してその後実行される update statistics コマンドは、column1 統計を削除するまで no\_hashing の使用が継続される column1 を除き、すべてのカラムに update statistics hashing のデフォルトの設定値を使用します。

- with consumers と with sampling は、ソートに適用され、ハッシュには適用されません。with consumers パラメータと with sampling パラメータとともに partial\_hashing パラメータを指定できますが、with consumers と with sampling は高ドメイン・カラムのソートにしか適用されません。with consumers パラメータまたは with sampling パラメータとともに with hashing パラメータを明示的に指定すると、これらのパラメータは Adaptive Server で無視されます。

#### ハッシュベースの統計の使用

- partial\_hashing パラメータは、ソートで生成されるヒストグラムと同じ程度の精度と品質のヒストグラムを生成できる場合のみ、カラムにハッシュを使用します(低ドメイン・ケース)。これ以外の場合、partial\_hashing パラメータは、高ドメイン・ケースのソートを使用します。
- 一部の状況では(高ドメイン・ケース)、ハッシュで生成されるヒストグラムの精度がソートより低くなります。
- ハッシュベースの統計では、tempdb ディスク領域やソートで使用するプロシージャ・キャッシュの必要がありませんが、tempdb バッファ・キャッシュ・メモリは大量に使用されます。
- no\_hashing パラメータで使用される大規模なソートでは、ソートに必要なプロシージャ・キャッシュ・メモリを解放するため、ステートメント・キャッシュの文とストアド・プロシージャが除去される場合があります。

- `max_resource_granularity` は、`hashing` または `partial_hashing` に使用される `tempdb` バッファ・キャッシュ・メモリの量を制限します。これは、`no_hashing` パラメータやソートで使用されるメモリ量には影響しません。
- カラムに以前のヒストグラムが存在し、高ドメイン・カラムであることが示される場合に、`partial_hashing` パラメータを指定すると、`Adaptive Server` はそのカラムにソートベースの統計が必要であるとみなします。カラムに以前のヒストグラムが存在しない場合、`Adaptive Server` は高ドメイン制限に達するまで、そのカラムが低ドメインであると判断します。

#### update statistics とサンプリング

`Adaptive Server` はデータ・ページのサンプルをスキャンします。インデックスは、`update statistics` で次のように指定します。

```
update statistics table_name [index_name] with sampling = N percent
```

このコマンドは、指定されたテーブルのすべてのインデックスの先行カラム、または指定されたインデックスの先行カラムについて統計値を作成および更新します。

`sampling = N percent` オプションを `using steps value` とともに使用する場合は、`sampling = N percent` オプションを最後に指定する必要があります。

```
update statistics titles (type)
  using 40 value
  with sampling = 10 percent
```

これを行わない場合、エラー・メッセージが表示されます。

```
update statistics titles (type)
  with sampling = 10 percent
  using 40 value
```

```
Msg 156, Level 15, State 2:
Line 1:
Incorrect syntax near the keyword 'using'.
```

#### インデックスとストアド・プロシージャの作成

`Adaptive Server` では、`update statistics` 文の実行後に、ストアド・プロシージャが自動的に再コンパイルされます。`update statistics` の実行前に開始した `adhoc` クエリは引き続き動作しますが、これらのクエリでは、新しい統計は利用されません。

`Adaptive Server` バージョン 12.5 以前では、`update statistics` は、キャッシュに格納されているストアド・プロシージャでは無視されていました。

---

標準規格	ANSI SQL — 準拠レベル：Transact-SQL 拡張機能
パーミッション	<code>update statistics</code> を実行するには、テーブル所有者であるか、テーブルに対する <code>update statistics</code> パーミッションを持つユーザであることが必要です。
参照	<b>コマンド</b> <a href="#">delete statistics</a> , <a href="#">update all statistics</a> , <a href="#">update index statistics</a> , <a href="#">update table statistics</a> <b>マニュアル</b> 『パフォーマンス&チューニング・シリーズ』 <code>optdiag</code> の構文と使用方法については、『パフォーマンス&チューニング・シリーズ：モニタリングと分析』の「第 2 章 統計テーブルおよび <code>optdiag</code> を使った統計の表示」を参照してください。

## update table statistics

説明	update table statistics は、systabstats テーブルに格納されている、ロー数やクラスタ率などの統計値を更新します。update table statistics は、sysstatistics に格納されているカラムの統計値には影響しません。
構文	<pre>update table statistics <i>table_name</i>                         [partition <i>data_partition_name</i>]                         [<i>index_name</i> [partition <i>index_partition_name</i>]]</pre>
パラメータ	<p><i>table_name</i> 統計を更新するテーブルの名前です。</p> <p><i>data_partition_name</i> 統計を更新するデータ・パーティションの名前を入力します。この指定を省略すると、すべてのデータ・パーティションについてテーブル統計値が更新されます。</p> <p><i>index_name</i> パーティションに関連付けられているインデックスの名前です。</p> <p><i>index_partition_name</i> インデックス・パーティションの名前です。</p>
例	<p><b>例 1</b> <code>smallsales</code> パーティションに対してテーブル統計の更新を実行します。</p> <pre>update table statistics titles partition smallsales</pre> <p><b>例 2</b> <code>titles</code> テーブルのすべてのパーティションに対してテーブル統計の更新を実行します。</p> <pre>update table statistics titles</pre>
使用法	<ul style="list-style-type: none"> <li>• <code>update table statistics</code> は、インデックス・パーティションの統計を更新しません。インデックス・パーティションのテーブルレベルの統計を生成するには、<code>update statistics</code> を使用します。</li> <li>• <code>update table statistics</code> を実行すると <code>update statistics</code> 実行の I/O コストが発生するため、カラム統計とテーブル統計の両方を生成するには <code>update statistics</code> を使用します。</li> </ul> <p>グローバル統計は、グローバル・インデックスを作成してから削除することで生成できます。</p> <p>単一の分割に対して <code>update statistics</code> を実行する場合は、分割統計をマージしてグローバル統計を作成します。ただし、これらのマージされたグローバル統計は、グローバル・インデックスの作成により作成されたグローバル統計よりも正確性に欠けます。前のバージョンのカラム統計がより正確である場合、それを上書きするカラム統計は生成しないでください。</p>



指定する値は次のとおりです。

- *index\_name* — `update table statistics` はインデックスのすべてのインデックス・パーティションに対する統計を更新します。
- *index\_partition* — `update table statistics` は特定のインデックス・パーティションの統計を更新します。

標準規格

ANSI SQL — 準拠レベル：Transact-SQL 拡張機能

パーミッション

`update table statistics` を実行するには、テーブル所有者であるか、テーブルに対する `update statistics` パーミッションを持つユーザである必要があります。

参照

コマンド `update all statistics`, `update index statistics`, `update statistics`

マニュアル 『パフォーマンス&チューニング・シリーズ』

## use

**説明** 作業するデータベースを指定します。

**構文** `use database_name`

**パラメータ** `database_name`  
 オープンするデータベースの名前です。

**例**

```
use pubs2
go
```

現在のデータベースは `pubs2` になります。

- 使用法**
- アーカイブ・データベースで使用できます。
  - データベースのオブジェクトを参照するには、その前に `use` コマンドを実行してください。
  - `use` コマンドを、ストアド・プロシージャまたはトリガ内に含めることはできません。
  - `sp_addalias` で追加されるエイリアスを使用すると、ユーザはデータベースへ別名でアクセスできます。

**標準規格** ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

**パーミッション** `use` のパーミッション・チェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効

細密なパーミッションが有効の場合、有効ユーザ、エイリアス・ユーザまたは "guest" ユーザであるか、データベースに対する `own database` 権限または `use database` 権限を持つユーザであることが必要。

データベースに "guest" アカウントがある場合は、すべてのユーザがそのデータベースを使用できる。データベースに "guest" アカウントがない場合は、そのデータベースの有効なユーザであるか、そのデータベース内にエイリアスを持っているか、データベースに対する `own database` 権限または `use database` 権限を持っていることが必要。

細密なパーミッションが無効

細密なパーミッションが無効の場合、有効ユーザ、エイリアス・ユーザまたは "guest" ユーザであるか、`sa_role` または `sso_role` が付与されたユーザであることが必要。

データベースに "guest" アカウントがある場合は、すべてのユーザがそのデータベースを使用できます。データベースに "guest" アカウントがない場合は、そのデータベースの有効なユーザであるか、そのデータベース内にエイリアスを持っているか、またはシステム管理者かシステム・セキュリティ担当者でなければなりません。

**参照** コマンド [create database](#), [drop database](#)

システム・プロシージャ [sp\\_addalias](#), [sp\\_adduser](#), [sp\\_modifylogin](#)

## waitfor

説明	文ブロック、ストアド・プロシージャ、またはトランザクションを実行するための、特定の時間、時間間隔、またはイベントを指定します。
構文	<code>waitfor {delay <i>time</i>   time <i>time</i>   errexit   processexit   mirrorexit}</code>
パラメータ	<p><code>delay</code> Adaptive Server に、指定の時間 (最大 24 時間) が経過するまで待機するように指示します。</p> <p><code>time</code> Adaptive Server に、指定の時刻まで待機するように指示します。</p> <p><code>time</code> <code>date/time</code> データ、または文字型の変数に有効ないずれかのフォーマットでの時刻。日付は指定できません。つまり、<code>date/time</code> 値の日付要素は使用できません。この情報に関しては、データ型 <code>time</code> を使用できます。</p> <p><code>errexit</code> Adaptive Server に、カーネルまたはユーザのプロセスが失敗するまで待機するように指示します。</p> <p><code>processexit</code> Adaptive Server に、カーネルまたはユーザのプロセスがなんらかの理由で終了するまで待機するように指示します。</p> <p><code>mirrorexit</code> Adaptive Server に、ミラー障害が発生するまで待機するように指示します。</p>
例	<p><b>例 1</b> chess テーブルが、午後 2 時 20 分にユーザの次の処理によって更新され、<code>sendmail</code> というプロシージャが <code>Judy</code> の所有するテーブルにローを挿入し、<code>chess</code> テーブルに新しい処理が行われたことを通知します。</p> <pre>begin     waitfor time "14:20"     insert chess (next_move)         values ('Q-KR5')     execute sendmail 'judy' end</pre> <p><b>例 2</b> 10 秒後に、Adaptive Server は指定したメッセージを出力します。</p> <pre>declare @var char (8) select @var = "00:00:10" begin     waitfor delay @var</pre>

```

        print "Ten seconds have passed.  Your time
              is up."
    end

```

**例 3** なんらかのプロセスが異常終了すると、Adaptive Server は指定されたメッセージを出力します。

```

begin
    waitfor errorexit
    print "Process exited abnormally!"
end

```

#### 使用法

- `waitfor` コマンドを発行すると、指定した時間またはイベントが発生するまで、Adaptive Server に接続できません。
- `waitfor errorexit` を、異常終了したプロセスを強制終了するプロシージャとともに使用することにより、異常になったプロセスに占有されるシステム・リソースを解放できます。
- どのプロセスが終了したかを確認するには、`sp_who` を使用して `sysprocesses` テーブルをチェックしてください。
- `waitfor time` or `waitfor delay` によって指定する時刻には、時間、分、および秒を含めることができます。『リファレンス・マニュアル：ビルディング・ブロック』の「第 1 章 システム・データ型とユーザ定義データ型」の「日付と時刻のデータ型」(20 ページ)に記載されている説明に従って、“hh:mm:ss” フォーマットを使用してください。

次の例は、Adaptive Server に対し、午後 4 時 23 分まで待機するように指示します。

```
waitfor time "16:23"
```

次の文は、Adaptive Server に対し、1 時間 30 分待機するように指示します。

```
waitfor delay "01:30"
```

- システム時刻を変更 (夏時間への時計の時刻の繰り上げなど) すると、`waitfor` コマンドの実行が遅れることがあります。
- DB-Library プログラム内で `waitfor mirrorexit` を使用すると、ミラー障害をユーザに通知できます。

#### 標準規格

ANSI SQL – 準拠レベル：Transact-SQL 拡張機能

#### パーミッション

`waitfor` の使用にパーミッションは必要ありません。

#### 参照

コマンド [begin...end](#)

データ型 [日付と時刻のデータ型](#)

システム・プロシージャ [sp\\_who](#)

## where 句

説明

select、insert、update、または delete の各文で探索条件を設定します。

構文

探索条件は、select、insert、update、または delete 文のキーワード where のすぐ後に指定します。1 つの文中で複数の探索条件を指定する場合は、それらの条件を and または or で連結してください。

```

where [not] expression comparison_operator expression
where {[not] expression comparison_operator expression} | {...}
where [not] expression [not] like "match_string"
      [escape "escape_character "]
where [not] expression is [not] null
where [not] expression [not] between expression and expression
where [not] expression [not] in ({value_list | subquery})
where [not] exists (subquery)
where [not] expression comparison_operator {any | all} (subquery)
where [not] column_name join_operator column_name
where [not] logical_expression
where [not] expression {and | or} [not] expression
where column_name is [not] null

```

パラメータ

not

like、null、between、in、exists などの論理式またはキーワードを否定します。

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、『リファレンス・マニュアル：ビルディング・ブロック』の「[第 4 章 式、識別子、およびワイルドカード文字](#)」にある「[式](#)」(363 ページ)を参照してください。

comparison\_operator

次のいずれかを指定します。

演算子	意味
=	等しい
>	より大きい
<	より小さい
>=	以上
<=	以下

演算子	意味
!=	等しくない
<>	等しくない
>	より大きくない
<	より小さくない

char、nchar、unichar、varchar、univarchar、および nvarchar データを比較する場合、< はアルファベットの最初に近いことを、> はアルファベットの最後に近いことを示します。

大文字と小文字および特殊文字の評価は、Adaptive Server のあるマシンのオペレーティング・システムの照合順によって異なります。たとえば、小文字が大文字よりも大きい場合と、大文字が数字よりも大きい場合があります。

比較を行う場合には、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。

日付の比較では、< はより前を意味し、> はより後を意味します。比較演算子で 사용되는文字および日付データはすべて、引用符で囲んでください。次に例を示します。

```
= "Bennet"  
> "94609"
```

データ入力規則の詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の「[第 1 章 システム・データ型とユーザ定義データ型](#)」の「[ユーザ定義データ型](#)」(49 ページ)を参照してください。

#### like

後続の文字列（一重引用符か二重引用符で囲まれた文字列）がパターン一致文字であることを示すキーワードです。like は、char、varchar、unichar、univarchar、nchar、nvarchar、datetime、date、time、text、unitext カラムで使用することができますが、秒またはミリ秒の検索には使用できません。

キーワード like とワイルドカード文字は、char や varchar だけでなく、datetime や date データでも使用できます。datetime や date と time 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。

`date/time` 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、`date/time` 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` という名前のカラムに値 "9:20" を挿入すると、`Adaptive Server` ではこのエントリが "Jan 1, 1900 9:20AM." に変換されるため、次の句ではこの値を検出できません。

```
where arrival_time = '9:20'
```

しかし、次の句ではこの値が検出されます。

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

### *match\_string*

引用符で囲まれた文字列とワイルドカード文字です。表 1-37 に、ワイルドカード文字をリストします。

表 1-37: ワイルドカード文字

ワイルドカード文字	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
[^ ]	指定の範囲外 ([^a-f]) またはセット以外 ([^abcdef]) の任意の 1 文字

### *escape*

ワイルドカード文字のリテラルを検索するためのエスケープ文字を指定します。

### *escape\_character*

任意の 1 文字です。『リファレンス・マニュアル：ビルディング・ブロック』の「第 4 章 式、識別子、およびワイルドカード文字」にある「[escape 句の使用](#)」(392 ページ)を参照してください。

### *is null*

`null` 値を検索します。

### *between*

範囲開始キーワードです。範囲終了値には `and` を使用します。次の範囲は包括的範囲です。

```
where @val between x and y
```

次の範囲は包括的範囲ではありません。

```
x and @val < y
```

指定した 1 番目の値が 2 番目の値より大きいと、`between` を使用したクエリではローは返されません。

**and**

2つの条件を結合し、2つの条件の両方が真である場合に結果を返します。

1つの文で複数の論理演算子を使用する場合、通常 **and** 演算子が最初に評価されます。ただし、カッコを使用すれば、評価する順序を変えることができます。

**in**

値リストのいずれかに一致する値を選択できるようにします。比較子には定数またはカラム名を指定できます。また、リストには定数の集合、またはサブクエリを指定できます。通常は、サブクエリを指定します。サブクエリでの **in** の使用方法については、『**Transact-SQL ユーザーズ・ガイド**』を参照してください。値のリストはカッコで囲んでください。

**value\_list**

値のリストです。文字値を一重引用符または二重引用符で囲んで、各値をカンマで区切ってください(例7を参照)。リストには、変数のリストを指定することができます。次に例を示します。

```
in (@a, @b, @c)
```

ただし、次のようなリストを含む変数は、値のリストとして使用できません。

```
@a = "'1', '2', '3'"
```

**exists**

サブクエリで、サブクエリによる結果が存在するかどうかをテストするために使用されます。『**Transact-SQL ユーザーズ・ガイド**』を参照してください。

**subquery**

**select**、**insert**、**delete**、または **update** 文あるいはサブクエリの **where** 句または **having** 句内の制限された **select** 文です (**order by** 句および **compute** 句とキーワード **into** は使用できません)。『**Transact-SQL ユーザーズ・ガイド**』を参照してください。

**any**

サブクエリと、**>**、**<**、または **=** 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。『**Transact-SQL ユーザーズ・ガイド**』を参照してください。



**all**

サブクエリと、> または < とともに使用します。サブクエリ内で検索された値がすべて、外側の文の **where** 句または **having** 句の中の値と一致する場合に、その結果を返します。『Transact-SQL ユーザーズ・ガイド』を参照してください。

**column\_name**

比較で使用されるカラムの名前です。カラム名が一意でない場合は、テーブル名またはビュー名で修飾してください。**IDENTITY** プロパティを持つカラムの場合は、**syb\_identity** キーワードを指定し、必要であれば実際のカラム名ではなくテーブル名によって修飾することができます。

**column\_name** には、**text**、**unitext**、または **image** データ型を使用できます。

**join\_operator**

比較演算子、あるいはジョイン演算子 **=\*** または **\*=** のいずれかです。『Transact-SQL ユーザーズ・ガイド』を参照してください。

**logical\_expression**

**TRUE** または **FALSE** を返す式です。

**or**

2 つの条件を結合し、2 つの条件のいずれかが真である場合に結果を返します。

1 つの文で複数の論理演算子を使用すると、通常 **or** 演算子よりも先に **and** が評価されます。ただし、カッコを使用すれば、評価する順序を変えることができます。

**例****例 1**

```
where advance * $2 > total_sales * price
```

**例 2** 電話番号の先頭が 415 でないローをすべて検索します。

```
where phone not like '415%'
```

**例 3** Carson、Carsen、Karsen、および Karson という名前の著者のローをすべて検索します。

```
where au_lname like "[CK]ars[eo]n"
```

**例 4** **sales\_east** テーブルで、**IDENTITY** カラムの値が 4 のローを検索します。

```
where sales_east.syb_identity = 4
```

**例 5**

```
where advance < $5000 or advance is null
```

**例 6**

```
where (type = "business" or type = "psychology") and advance > $5500
```

**例 7**

```
where total_sales between 4095 and 12000
```

**例 8** リスト内の 3 つの州のどれかが入っているローを検索します。

```
where state in ('CA', 'IN', 'MD')
```

**例 9** text、unitext、image カラムの null 値に基づいてデータを選択します。

```
create table temp1(c1 int, c2 text null, c3 unitext null, c4 image null)
insert into temp1 values(1, null, replicate("u",5), null)
insert into temp1 values(2, replicate("x",3), null, null)
```

```
go
select * from temp1 where c2 is null
go
```

c1	c2	c3	c4
1	NULL	0x75007500750075007500	NULL

(1 row affected)

```
select * from temp1 where c2 is not null and c3 is null and c4 is null
go
```

c1	c2	c3	c4
2	xxx	NULL	NULL

**例 10** text カラムの非 null 値に基づいてデータを更新します。

```
insert into temp1 values(3, replicate("y", 3), null, 0x858585847474)
insert into temp1 values(4, replicate("z",3),"aaa", 0x75)
```

```
go
update temp1 set c2 = "updated" where c2 is not null
select * from temp1 where c2 is not null
go
```

(3 rows affected)

c1	c2	c3	c4
2	updated	NULL	NULL
3	updated	NULL	0x858585847474

```
4          updated          0x610061006100    0x75
```

**例 11** temp1 の text カラムの null 値に基づいて、テーブル temp2 にデータを選択します。

```
select c1, c2 into temp2 from temp1 where c2 is null
select * from temp2
go
(1 row affected)
c1      c2
-----
1      NULL
```

**例 12** temp1 の text カラムの非 null 値に基づいて、テーブル temp1 から選択して、テーブル temp2 にデータを挿入します。

```
insert into temp2 select c1, c2 from temp1 where c2 is not null
select * from temp2
go
(3 rows affected)
c1      c2
-----
1      NULL
2      updated
3      updated
4      updated
(4 rows affected)
```

**例 13** text カラムの null 値に基づいて、サブクエリにデータを選択します。

```
select count(*) from temp2
where c1 in (select c1 from temp1 where c2 is null and c3 is not null)
-----
1
(1 row affected)
```

**例 14** unitext カラムの null 値に基づいてデータを削除します。

```
delete from temp1 where c3 is null
go
(2 rows affected)
```

#### 使用法

- where および having 探索条件は、where 句では集合関数が使用できない点を除いて同じです。たとえば、次の句は有効です。

```
having avg (price) > 20
```

次の句は無効です。

```
where avg (price) > 20
```

集合関数の使用方法については、『リファレンス・マニュアル：ビルディング・ブロック』の「第2章 Transact-SQL 関数」および「group by 句と having 句」(527 ページ)を参照してください。

- ジョインとサブクエリは探索条件で指定されます。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。
- like キーワードを使用すると、unitext カラムを特定のパターンで検索できます。ただし、like 句は unitext カラムで使用すると最適化されません。like による unitext のパターン検索は、デフォルトの Unicode ソート順に影響されます。このソート順は unichar と univarchar データ型の like を使用したパターン検索でも使用されます。
- 変数 *expression* と *match\_string* で、where 句は text と unitext LOB ロケータを受け入れますが、image LOB ロケータは受け入れられません。

```
...
where expression like 'match_string'
...
```

*match\_string* がロケータの場合、Adaptive Server は対応する LOB の最大 16KB のみを使用します。

- null 条件を指定すると、指定した LOB カラムに null 値があるローだけが選択されます。null 値が明示的に割り当てられたか、LOB が初期化されなかった場合は、LOB 値は null になることがあります。
- where 句で指定できる and および or 条件の数は、クエリの実行に使用可能なメモリの量によってのみ制限されます。
- like 述部に含まれるパターン文字列は、varchar に配置可能な文字列のサイズによってのみ制限されます。
- 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、2つの引用符を使用する方法です。たとえば、一重引用符で文字の入力を開始していて、入力の一部に一重引用符を使用する場合には、一重引用符を2つ使用します。

```
'I don't understand.'
```

二重引用符の場合は、次のようになります。

```
"He said, "It's not really confusing.""
```

2つ目は、もう一方の種類の引用符で、引用符を囲む方法です。つまり、二重引用符が含まれているものを入力する場合は、それを一重引用符で囲みます（またはその逆の方法で囲みます）。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?"'
```

- 画面の幅よりも長い文字列を入力する場合は、改行する前に円記号(¥)を入力してください。
- カラムを **where** 句の定数または変数と比較すると、この定数または変数がカラムのデータ型に変換されます。これで、オプティマイザがデータ検索時にインデックスを使用できます。たとえば、**float** 式が **int** カラムと比較される場合、この式は **int** へ変換されます。次に例を示します。

```
where int_column = 2
```

*int\_column* = 2 のローを選択します。

- **Adaptive Server** は、クエリを最適化するとき **where** 句と **having** 句内の探索条件を評価して、最適なインデックスとクエリ・プランの選択に使用可能な探索指数 (**SARG**) の条件を決定します。すべての探索条件を使用して、ローが限定されます。探索指数の詳細については、『パフォーマンス&チューニング・ガイド』を参照してください。

#### 標準規格

ANSI SQL — 準拠レベル：初級レベル

#### 参照

コマンド [delete](#), [execute](#), [group by 句](#) と [having 句](#), [insert](#), [select](#), [update](#)

データ型 [日付と時刻のデータ型](#)

システム・プロシージャ [sp\\_helpjoins](#)

## while

**説明** 文または文ブロックの繰り返し実行の条件を設定します。設定した条件が `true` である場合には、文は反復実行されます。

**構文** `while logical_expression [plan "abstract plan"] statement`

**パラメータ** `logical_expression`

TRUE、FALSE、または NULL を返す式です。

`plan "abstract plan"`

クエリを最適化するために使用する抽象プランを指定します。抽象プラン言語で指定された完全プランまたは部分プランを指定できます。プランは、最適化可能な SQL 文、つまりテーブルにアクセスするクエリにのみ指定できます。詳細については、『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「抽象プランの作成と使用」を参照してください。

**statement**

通常は `begin` と `end` で区切られている SQL 文のブロックです。ただし、単一 SQL 文も指定できます。

**例** 平均価格が 30 ドル未満の場合は、`titles` テーブル内のすべての本の価格を 2 倍にします。それでも価格が 30 ドル未満である間は、`while` ループで価格を 2 倍にする操作を続行します。`while` ループ内の `select` は、価格が 20 ドルを超える本のタイトルを特定するだけでなく、ループした回数も示します (Adaptive Server から返される各平均結果が 1 回のループを示します)。

```
while (select avg (price) from titles) < $30
begin
    select title_id, price
        from titles
        where price > $20
    update titles
        set price = price * 2
end
```

**使用法**

- `while` ループ内の文の実行を制御するには、ループ内で `break` コマンドと `continue` コマンドを使用します。
- `continue` コマンドを使用すると、`while` ループが再起動され、`continue` の後にあるすべての文が省略されます。`break` コマンドを使用すると、`while` ループが終了します。次に、ループの終わりを示すキーワード `end` の後にある文がすべて実行されます。`break` および `continue` コマンドは、通常 `if` テストによってアクティブ化されます。

次に例を示します。

```

while (select avg (price) from titles) < $30
begin
    update titles
    set price = price * 2
    if (select max (price) from titles) > $50
        break
    else
        if (select avg (price) from titles) > $30
            continue
        print "Average price still under $30"
end

select title_id, price from titles
    where price > $30

```

このバッチは、本の平均価格が 30 ドル未満であるかぎり、`titles` テーブル内のすべての本の価格を 2 倍にする操作を続行します。ただし、いずれかの本の価格が 50 ドルを超えると、`break` コマンドによって `while` ループが停止します。平均価格が 30 ドルを超えると、`continue` コマンドによって `print` 文の実行が抑止されます。`while` ループの終了方法 (通常どおりに終了するか、または `break` コマンドによって終了するか) に関係なく、最後のクエリでは、価格が 30 ドルを超えた本が示されます。

- 複数の `while` ループがネストされている場合は、`break` コマンドによって、次に外側にあるループに移行します。内側のループの終わりの後にあるすべての文が実行されてから、次に外側にあるループが再開されます。

---

**警告!** `create view` コマンドが `while` ループ内で発生すると、Adaptive Server によって、条件が `true` かどうかを判断する前にテーブルまたはビューのスキーマが作成されます。このため、そのテーブルまたはビューがすでに存在している場合には、エラーが発生する可能性があります。

---

標準規格

ANSI SQL — 準拠レベル : Transact-SQL 拡張機能

パーミッション

`while` の使用にパーミッションは必要ありません。

参照

コマンド [begin...end](#), [break](#), [continue](#), [goto label](#)

## writetext

**説明** 既存の `text` カラム、`unitext` カラム、または `image` カラムで、最小限のログを取る対話型の更新を可能にします。

**構文** `writetext [[database.]owner.]table_name.column_name  
text_pointer [readpast] [with log] data`

**パラメータ** `table_name.column_name`  
更新するテーブルと `text`、`unitext`、または `image` カラムの名前を示します。テーブルが別のデータベース内にある場合はデータベース名を指定し、データベース内にその名前のテーブルが複数ある場合は所有者名を指定します。`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。

`text_pointer`  
`text`、`unitext`、または `image` データへのポインタを格納する `varbinary(16)` 値です。この値を確認するには、`textptr` 関数を使用してください。`text` データ、`unitext` データ、または `image` データは、他のテーブルとは異なり、リンクされた同じセットのページには格納されません。このデータは、リンク・ページの個別のセットに格納されます。実際のロケーションへのポインタはデータとともに格納されます。`textptr` はこのポインタを返します。

`readpast`  
このコマンドが、ロックが解除されているローだけを修正するように指定します。`writetext` コマンドは、ロックされたローを検出すると、ロックが解除されるまで待機せずに、これらのローを省略します。

`with log`  
挿入された `text`、`unitext`、または `image` データのログを取ります。このオプションはメディアのリカバリに役立ちますが、大きなデータ・ブロックのログを取ると、トランザクション・ログのサイズが急激に増加します。トランザクション・ログは必ず別のデータベース・デバイス上に格納してください。詳細については、「`create database`」、「`sp_logdevice`」、『システム管理ガイド』を参照してください。

`data`  
`text`、`unitext`、または `image` カラムへ書き込まれるデータです。`text` および `unitext` データはカッコで囲みます。`image` データの前には、“0x”を指定します。使用しているクライアント・ソフトウェアに関する情報をチェックして、クライアントで処理できる `text`、`unitext`、または `image` データの最大長を判断してください。



例

**例 1** テキスト・ポインタをローカル変数 `@val` に格納します。次に `writetext` が、`@val` が指すテキスト・フィールドに、テキスト文字列 "hello world" を挿入します。

```
declare @val varbinary (16)
select @val = textptr (copy) from blurbs
       where au_id = "409-56-7008"
writetext blurbs.copy @val with log "hello world"
```

**例 2**

```
declare @val varbinary (16)
select @val = textptr (copy)
from blurbs readpast
       where au_id = "409-56-7008"
writetext blurbs.copy @val readpast with log "hello
world"
```

**例 3** `writetext` は、`unitext` データ型に関する情報を追加し、`@val` が指す `unitext` 型フィールドに文字列 "Hello world" を配置します。

```
declare @val varbinary (16)
select @val = textptr (ut) from unitable
       where i = 100
writetext unitable.ut @val with log "Hello world"
```

カラムが更新される前に、`varchar` 定数は `unitext` に暗黙に変換されます。

使用法

- `writetext` を使用して対話型で挿入できるテキストの最大長は、`text`、`unitext`、または `image` データでは約 120K バイトです。
- デフォルトでは、`writetext` は最小のログを取るオペレーションであり、ページの割り付けと割り付け解除だけがログに取られます。`text`、`unitext`、または `image` データがデータベースに書き込まれてもログは取られません。`writetext` をデフォルトの最小のログが取られる状態で使用するために、システム管理者は、`sp_dboption` を使用して、`select into/bulkcopy/pllort` を `true` に設定する必要があります。
- `writetext` は、既存のローの `text` データを更新します。この更新では、既存のテキストすべてが完全に置換されます。
- `writetext` オペレーションは、`insert` または `update` トリガによって起動することはできません。
- `writetext` には、`text`、`unitext`、または `image` カラムへの有効なテキスト・ポインタが必要です。有効なテキスト・ポインタを存在させるには、`update` によって明示的に入力された実際のデータまたは `null` 値のいずれかを、`text` カラムまたは `unitext` カラムに含める必要があります。

カラム `textid` および `x` が含まれるテーブル `textnull` があるとします。`x` が `null` 値が許容される `text` カラムである場合、次の `update` ではすべての `text` 値が `NULL` に設定され、`text` カラムに有効なテキスト・ポインタが割り当てられます。

```
update textnull
set x = null
```

次のように `null` 値を明示的に指定した `insert` 文を実行しても、テキスト・ポインタは割り当てられません。

```
insert textnull values (2,null)
```

暗黙的 `null` の `insert` によってテキスト・ポインタが割り当てられることもありません。

```
insert textnull (textid)
values (2)
```

- `text` カラムに対して実行される `insert` 文および `update` 文は、ログに取られます。
- ビューの中の `text` カラムや `image` カラムには、`writetext` を使用できません。
- マルチバイト文字セットに変更した `text` 値に `dbcc fix_text` を実行せずに `writetext` を実行しようとする、このコマンドは失敗し、テーブルに対して `dbcc fix_text` を実行するように指示するエラー・メッセージが表示されます。
- `writetext` を、デフォルトのログを記録しないモードで実行すると、`dump database` が実行されている間は低速になります。
- Client-Library 関数 `dbwritetext` と `dbmoretext` は、`writetext` よりも処理速度が速く、動的メモリの使用量が少ない関数です。これらの関数は、最大 2GB の `text` データを挿入できます。

#### *readpast* オプションの使用

- `readpast` オプションは、データオンリーロック・テーブルのみに適用されます。全ページロック・テーブルに指定されている `readpast` は無視されます。
- セッション-ワイドな独立性レベルが 3 の場合、`readpast` オプションは暗黙的に無視されます。
- セッションのトランザクション独立性レベルが 0 の場合、`readpast` を使用する `writetext` コマンドは、警告メッセージを発行しません。セッション独立性レベル 0 のこれらのコマンドは、`text` カラムが互換性のないロックによってロックされていない場合、指定の `text` カラムを修正します。

---

標準規格	ANSI SQL – 準拠レベル : Transact-SQL 拡張機能
パーミッション	<code>writetext</code> を実行するには、テーブル所有者であるか、テーブルに対する <code>update</code> パーミッションを持つユーザであることが必要です。
参照	コマンド <a href="#">readtext</a> データ型 <a href="#">text</a> および <a href="#">image</a> データ型の変換



この章では、Interactive SQL コマンドについて説明します。これらのコマンドは、Interactive SQL 画面の上部ウィンドウ枠で入力できます。これらのコマンドは Interactive SQL のみで使用されるもので、Adaptive Server に送信され、実行されることはありません。Interactive SQL の詳細については、『ユーティリティ・ガイド』の「第 3 章 グラフィック・モードでの Interactive SQL の使用」および Adaptive Server プラグインのオンライン・ヘルプを参照してください。

表 2-1: DBISQL コマンド

コマンド	説明
「clear」(856 ページ)	Interactive SQL ウィンドウ枠をクリアする。
「configure」(857 ページ)	Interactive SQL の [ オプション ] ダイアログを開く。
「connect」(858 ページ)	データベースへの接続を確立する。
「disconnect」(861 ページ)	データベースへの現在の接続を切断する。
「exit」(862 ページ)	Interactive SQL を終了する。
「input」(863 ページ)	外部ファイルまたはキーボードからデータベース・テーブルにデータをインポートする。
「output」(869 ページ)	外部ファイルまたはキーボードからデータベース・テーブルにデータをインポートする。
「parameters」(874 ページ)	Interactive SQL コマンド・ファイルに対するパラメータを指定する。
「read」(875 ページ)	Interactive SQL 文をファイルから読み込む。
「set connection」(877 ページ)	現在のデータベース接続を別のサーバに変更する。
「start logging」(878 ページ)	実行された SQL 文のログ・ファイルへの記録の開始に使用される。
「stop logging」(879 ページ)	現在のセッションで SQL 文のロギングの停止に使用される。
「system」(880 ページ)	Interactive SQL 内からの実行ファイルの起動に使用される。

## clear

説明	Interactive SQL ウィンドウ枠をクリアします。
構文	clear
使用法	<ul style="list-style-type: none"><li>• clear 文を使用して、[SQL 文] ウィンドウ枠と [メッセージ] ウィンドウ枠、および [結果] ウィンドウ枠内のタブ [結果]、[メッセージ]、[プラン] をクリアします。</li><li>• clear を実行すると、クリアしたデータに関連付けられているカーソルが閉じます。</li></ul>
パーミッション	このコマンドは、すべてのユーザが実行できます。

## configure

説明	Interactive SQL の [ オプション ] ダイアログを開きます。
構文	configure
使用法	<ul style="list-style-type: none"><li>• configure 文は、[Interactive SQL Options] ダイアログを開き、すべての Interactive SQL オプションの現在の設定を表示できます。データベース・オプションは表示されず、変更を加えることもできません。</li><li>• このダイアログを使用して、Interactive SQL の設定を変更できます。[Make Permanent] を選択すると、オプションは保存され、以後の Interactive SQL セッションに適用されます。[Make Permanent] を選択しないで [OK] をクリックすると、オプションは一時的に設定され、現在のデータベース接続についてのみ有効になります。</li></ul>
パーミッション	すべてのユーザが configure を実行できます。
参照	<a href="#">set</a>

## connect

説明

データベースへの接続を確立します。

構文

```
connect
    [to engine_name]
    [database database_name]
    [as connection_name]
    [user] user_id identified by password
    engine_name, database_name, connection_name, user_id,
    password :{identifier | string | hostvar}

connect using connect_string : {identifier | string | hostvar}
```

パラメータ

**engine\_name**

接続先のエンジンの名前です。

**database\_name**

接続先のデータベースの名前です。名前は識別子の規則に従っている必要があります。また、変数を使うことはできません。

**as**

**as** 句を指定して、接続に名前を付けることができます。接続に名前を付けると、同じデータベースへの複数の接続、あるいは同じまたは異なるデータベース・サーバへの複数の接続が、すべて同時に行えるようになります。それぞれの接続には、固有のトランザクションがあります。ただし、2つの異なる接続から同じデータベース内の同じレコードを修正した場合などに、トランザクション間でロックの競合が起こることがあります。

**connection\_name**

接続に使用するログイン名です。

**user**

Adaptive Server にユーザとして接続することを示します。

**user\_id**

接続するユーザの ID です。

**identified by password**

接続時にユーザがパスワードを指定する必要があることを示します。

**password**

Adaptive Server への接続に使用するパスワードです。

**identifier**

接続情報に使用する識別子です。

**string**

接続情報に使用する文字列です。



**hostvar**

ホスト名とポートを示す変数情報です。

**connect\_string**

セミコロンで区切られた **keyword = value** 形式のパラメータ設定リストであり、一重引用符で囲む必要があります。

例

**例 1** Interactive SQL からデータベースへ接続します。Interactive SQL によって、ユーザ ID とパスワードの入力が要求されます。

```
connect
```

**例 2** Interactive SQL からデフォルトのデータベースへ DBA として接続します。Interactive SQL によって、パスワードの入力が要求されます。

```
connect user "DBA"
```

**例 3** ユーザ dba として、パスワード sql を使用して、ホスト "tribble" のポート番号 5000 で実行されている Adaptive Server データベース pubs2 に接続します。

```
connect to "tribble:5000"
database pubs2
user dba
identified by sql
```

**例 4** ユーザ dba として、パスワード sql を使用し、"tribble" という名前の Adaptive Server interfaces ファイル内に定義されている ) に接続します。

```
connect to tribble
user dba
identified by sql
```

使用法

- **connect** は、**engine\_name** で識別されるサーバ上で実行している、**database\_name** で識別されるデータベースへの接続を確立します。
- **connect** 文が正しく実行されるまで、他の文を使用できません。

- **Interactive SQL の動作** – `connect` 文にデータベースやサーバを指定しない場合、**Interactive SQL** は、デフォルトのサーバとデータベースではなく現在のデータベースへの接続を維持します。サーバ名を指定しないでデータベース名を指定した場合、**Interactive SQL** は現在のサーバ上で指定したデータベースに継続しようとします。データベース名を指定せずにサーバ名を指定した場合、**Interactive SQL** は、指定したサーバ上のデフォルト・データベースに接続します。
- ユーザ・インタフェースでパスワード、またはユーザ ID とパスワードを指定しない場合、不足している情報の入力を要求されます。
- **Interactive SQL** がコマンド・プロンプト・モード (コマンド・プロンプトで `-nogui` を指定して **Interactive SQL** を起動した) またはバッチ・モードで実行されている場合、または `as` 句を指定しないで `connect` 文を実行した場合、名前のない接続が開かれます。名前のない別の接続がすでにオープンしている場合、古い接続は自動的にクローズします。それ以外の場合、`connect` の実行時に既存の接続はクローズしません。
- 複数の接続を管理するときにも、現在の接続という概念が使用されます。`connect` 文が正常に実行されると、その新しい接続が現在の接続になります。別の接続に切り替えるには、`set connection` 文を使用します。接続を削除するには `disconnect` 文を使用します。
- **Interactive SQL** では、接続情報 (データベース名、ユーザ ID、データベース・サーバなど) が [SQL 文] ウィンドウ枠の上のタイトル・バーに表示されます。データベースに接続していないときは、タイトル・バーに「Not Connected」と表示されます。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照

[disconnect](#), [set connection](#)

## disconnect

説明	データベースへの現在の接続を切断します。
構文	<code>disconnect [(identifier   string   hostvar)   current   all]</code>
パラメータ	<code>{identifier   string   hostvar}</code> 接続に使用するログイン名です。 <ul style="list-style-type: none"><li>• <code>identifier</code> — 接続情報に使用する識別子です。</li><li>• <code>string</code> — 接続情報に使用する文字列です。</li><li>• <code>hostvar</code> — ホスト名とポートを示す変数情報です。</li></ul> <code>current</code> 現在の接続を切断することを示します。 <code>all</code> すべての接続を切断することを示します。
例	すべての接続を切断します。 <pre>disconnect all</pre>
使用法	<ul style="list-style-type: none"><li>• <code>disconnect</code> 文はデータベース・サーバとの接続を削除し、その接続で使用したすべてのリソースを解放します。削除しようとする接続の名前が <code>connect</code> 文上で指定されている場合、その名前を使用して接続を指定できます。<code>all</code> を指定すると、すべてのデータベース環境へのアプリケーションの接続すべてが削除されます。<code>current</code> はデフォルトであり、現在の接続を削除します。</li><li>• 接続が削除されると、暗黙の <code>rollback</code> が実行されます。</li></ul>
パーミッション	このコマンドは、すべてのユーザが実行できます。
参照	<a href="#">connect</a> , <a href="#">set connection</a>

## exit

説明	Interactive SQL を終了します。
構文	{exit   quit   bye} [{number   connection_variable}]
パラメータ	<p>exit   quit   bye</p> <p>データベースとの接続を閉じてから、Interactive SQL 環境を閉じます。</p> <p>{number   connection_variable}</p> <p>Interactive SQL コマンド・ファイル内のコマンドが実行に成功したか失敗したかを示すため、バッチ・ファイルで使用できます。デフォルトのリターン・コードは 0 です。</p> <ul style="list-style-type: none"><li>• number — リターン・コードの番号です。</li><li>• connection_variable — 特定の接続を示す変数です。</li></ul>
使用法	commit_on_exit オプションをオンにすると、データベース接続をクローズする前に commit 文を自動的に実行します。このオプションをオフにすると、暗黙的に rollback が実行されます。デフォルトでは、commit_on_exit オプションはオンに設定されます。
パーミッション	このコマンドは、すべてのユーザが実行できます。

# input

**説明** 外部ファイルまたはキーボードからデータベース・テーブルにデータをインポートします。

**構文**

```
input into [ owner.]table_name
  [ from filename | prompt]
  [ format { ascii | dbase | dbaseII | dbaseIII | excel | fixed | foxpro |
lotus }}]
  [ escape character character]
  [ escapes { on | off}]
  [ by order | by name ]
  [ delimited by string ]
  [ column widths (integer , ... ) ]
  [ nostrip ]
  [ ( column_name, ... ) ]
  [ encoding {identifier | string}]
```

**パラメータ** **from 句**  
引用符で囲まれた文字列としてサーバに渡されるファイル名です。このため、文字列は他の SQL 文字列と同じフォーマット要件に従います。特に、次の点に注意が必要です。

- ディレクトリ・パスを示すには、円記号(¥)を2つの円記号で表す必要があります。ファイル c:¥emp¥input.dat から employee テーブルにデータをロードするには、次の文を使用します。

```
input into employee
from 'c:\¥emp¥input.dat'
```

- パス名は、Interactive SQL を実行しているマシンでの相対パスです。

## prompt

ロー内の各カラムにユーザが値を入力できるようにします。ウィンドウを使用するモードで動作している場合は、ダイアログが表示され、そこから新しいローの値を入力できます。Interactive SQL をコマンド・ラインで使用している場合は、コマンド・ラインで各カラムの値を入力するように要求されます。

## format

各値セットは、format 句に指定されたフォーマットで表す必要があります。format 句を指定しない場合は set option input\_format 文に指定されたフォーマットで表す必要があります。ユーザが input と入力するとダイアログが表示され、この入力フォーマットを使用して1行に1つのローを入力できます。

ファイル・フォーマットによっては、カラム名と種類に関する情報が含まれていることがあります。

input 文を実行したときにデータベース・テーブルがまだ存在していない場合は、この情報を使用してデータベース・テーブルが作成されます。この方法を使用すると、簡単にデータをデータベースにロードできます。テーブルを作成するための情報が含まれるフォーマットは、**dbasell**、**dbaselll**、**foxpro**、**lotus** です。

コマンド・ファイルからの入力、**end** が含まれる行で終了します。ファイルからの入力は、ファイルの末尾で終了します。

input では次のフォーマットを指定できます。

- **ascii** — 入力行は、1 行が 1 ローを表す、値がカンマで区切られた ASCII 文字として扱われます。アルファベット文字列は、アポストロフィ (一重引用符) または引用符 (二重引用符) で囲むことができます。カンマを含む文字列は、一重引用符または二重引用符で囲む必要があります。文字列そのものに一重引用符または二重引用符が含まれる場合は、引用符を文字列内で使用するために引用符を 2 つ続けます。**delimited by** 句を使用して、デフォルト (カンマ) 以外のデリミタを指定することもできます。

この他に 3 つの特別なシーケンスが認識されます。2 文字は改行文字を表し、"**¥**" は単一の円記号 (¥) を示し、**¥xDD** のようなシーケンスは 16 進コード **DD** の付いた文字を示します。

- **dbase** — ファイルは **DBASEII** フォーマットまたは **DBASEIII** フォーマットです。**Interactive SQL** は、ファイル内の情報に基づいてファイル・フォーマットを確認しようとします。テーブルが存在しない場合には、テーブルを作成します。
- **dbasell** — ファイルは **DBASEII** フォーマットです。テーブルが存在しない場合には、テーブルを作成します。
- **dbaselll** — ファイルは **DBASEIII** フォーマットです。テーブルが存在しない場合には、テーブルを作成します。
- **excel** — 入力ファイルは **Microsoft Excel 2.1** フォーマットです。テーブルが存在しない場合には、テーブルを作成します。
- **fixed** — 入力行は固定フォーマットです。**column widths** 句を使用して、カラム幅を指定します。カラム幅を指定しない場合、ファイル内のカラム幅は、対応するデータベース・カラムの種類に必要な最大文字数と同じである必要があります。

改行文字と **End of File** 文字シーケンスが埋め込まれているバイナリ・カラムには、**fixed** フォーマットを使用できません。

- **foxpro** – ファイルは FoxPro フォーマットです。テーブルが存在しない場合には、テーブルを作成します。
- **lotus** – ファイルは Lotus WKS フォーマットのワークシートです。**input** は、Lotus WKS ワークシートの最初のローがカラム名であるとみなします。テーブルが存在しない場合には、テーブルを作成します。この場合、ファイル内の情報はカラムではなくセルに関するものなので、作成されるカラムの種類とサイズが正しくない可能性があります。

#### escape character

16 進数のコードに使用されるデフォルトのエスケープ文字です。記号は円記号 (¥) で、たとえば改行文字は ¥x0A と表記します。

**escape character** 句を使用してエスケープ文字を変更できます。たとえば、感嘆符 (!) をエスケープ文字として使用するには、次のように入力します。

```
... escape character '|'
```

エスケープ文字として使用できるのは、シングルバイトの文字のみです。

#### escapes

**escapes** を有効 (デフォルト) にすると、データベース・サーバは、円記号に続く文字を特別な文字として認識および解釈します。改行文字は ¥n という文字組み合わせで、それ以外の文字は 16 進数の ASCII コード (タブ文字は ¥x09 など) でデータに含めることができます。2 つの円記号 (¥¥) は 1 つの円記号として解釈されます。円記号の後に n、x、X、¥ 以外の文字がある場合、この文字と円記号は無関係の 2 つの文字として解釈されます。たとえば、¥q は "¥" と文字 "q" として挿入されます。

#### by

入力ファイルのカラムをテーブル・カラムと対応付けるときに、入力ファイルのカラムの順序に基づくか (**order**)、またはカラム名に基づくか (**name**) を指定できます。デフォルトは **order** です。すべての入力フォーマットで、ファイルにカラム名が情報として含まれているわけではありません。**name** は、カラム名があるフォーマットにのみ使用できます。**dbaselI**、**dbaselIII**、**foxpro**、**lotus** です。

#### delimited

ASCII 入力フォーマットのデリミタとして使用される文字列を指定します。

### column widths

入力ファイル内のカラム幅を示します。この値は、**fixed** フォーマットの使用時にのみ指定できます。**column widths** を指定しない場合、カラム幅はデータベース・カラムの種類をもとに決定されます。**fixed** フォーマットで **long varchar** または **binary** のデータを挿入するときは、この句を使用しないでください。

### nostrip

通常、**ASCII** 入力フォーマットでは、値を挿入する前に、引用符で囲まれていない文字列から後続ブランクが削除されます。**nostrip** を指定すると、後続ブランクの削除は行われません。このオプションを使用してもしなくても、引用符で囲まれた文字列の後続ブランクは削除されません。**nostrip** オプションの設定に関係なく、引用符で囲まれていない文字列の先行ブランクは削除されます。

**null** と解釈されるカラムなどのエントリが **ASCII** ファイルにある場合、これらは **null** として扱われます。その位置にあるカラムの値を **null** にできないときは、数値カラムには **0** が挿入され、文字カラムには空の文字列が挿入されます。

### encoding

ファイルの読み取りに使用するエンコーディングを指定できます。**encoding** 句を使用できるのは **ASCII** フォーマットだけです。

**encoding** を指定しない場合、ファイルの読み取りに使用するコード・ページは次のリストに従って決定されます。このリストで上位に記載されているコード・ページの値は、下位の値よりも優先されます。

- **default\_isql\_encoding** オプションで指定されたコード・ページ (このオプションが設定された場合)
- **Interactive SQL** を開始するとき **-codepage** オプションで指定されたコード・ページ
- **Interactive SQL** を実行しているコンピュータのデフォルトのコード・ページ

### 例

以下は、**ASCII** テキスト・ファイルを使用する **input** 文です。

```
input into employee
from new_emp.inp
format ASCII
```



## 使用法

- `input` 文を使用して、指定のデータベース・テーブルに大量の挿入を効率よく行うことができます。入力する行は、入力ウィンドウを介してユーザが入力するか (`prompt` の指定時)、ファイルから読み取られます (`from file_name` の指定時)。どちらも指定しない場合、入力は `input` 文を含むコマンド・ファイルから読み取られません。Interactive SQL では、この読み取りを [SQL Statements] ウィンドウ枠から直接行うこともできます。その場合、文字列 `end` のみがある行で入力は終了します。
- カラム・リストが入力フォーマットに指定されていると、データは指定のテーブルの指定のカラムに挿入されます。デフォルトで、`input` 文は、入力ファイル内のカラム値の順序が、データベース・テーブル定義内での値と同じ順序であるとみなします。入力ファイルのカラムが異なる順序で配置されているときは、入力ファイル内の実際のカラム順のリストを `input` 文の最後に指定する必要があります。

次の例では、`inventory` という名前のテーブルを作成します。入力ファイルでは `name` 値が `quantity` 値の前に配置されているため、この入力ファイルから ASCII データをインポートするには、入力ファイル内の実際のカラム順のリストを `input` 文の最後に指定して、データが正しく挿入されるようにします。

```
create table inventory (  
  quantity int,  
  item varchar(60)  
)
```

次の入力ファイル `stock.txt` の ASCII データには、`name` 値が `quantity` 値の前にあります。

```
'Shirts', 100  
'Shorts', 60
```

入力ファイル内の実際のカラム順のリストを `input` 文の最後に追加し、データを正しく挿入します。

```
input into inventory  
from stock.txt  
FORMAT ASCII  
(item, quantity)
```

- `input` は、エラーの原因となるローを挿入しようとする、デフォルトで停止します。エラーの扱いは、`on_error` オプションと `conversion_error` オプションの設定によって異なります。**Interactive SQL** は、`input` で文字列値がトランケートされた場合に警告を [メッセージ] ウィンドウ枠に表示します。**NOT NULL** カラムで失われた値は、数値型では 0 に設定され、数値でない型では空の文字列に設定されます。`input` 文が `null` ローを挿入しようとするのは、入力ファイルに空のローが含まれているときです。

### パーミッション

テーブルまたはビューへの `insert` パーミッションが必要です。

## output

**説明** 外部ファイルまたはキーボードからデータベース・テーブルにデータをインポートします。

**構文**

```
output to filename
  [ append ]
  [ verbose ]
  [ format {ascii | dbase | dbasel|| dbaselll
           | excel | fixed | foxpro | lotus | sql | xml}]
  [ escape character character ]
  [ escapes { on | off}
  [ delimited by string ]
  [ quote string [ all ] ]
  [ column widths (integer , ... ) ]
  [ hexadecimal { on | off | asis } ]
  [ encoding {string | identifier}]
```

**パラメータ** **append**  
クエリの結果を、既存の出力ファイルに記述されている内容を上書きするのではなく、ファイルの末尾に追加します。**append** 句を使用しない場合、**output** 文はデフォルトで出力ファイルの内容を上書きします。**append** キーワードは、出力フォーマットが **ASCII**、**fixed**、または **SQL** の場合に使用できます。

**verbose**  
クエリに関するエラー・メッセージ、データの選択に使用した **SQL** 文、およびデータそのものを出力ファイルに書き込みます。データが含まれない行には、先頭に 2 つのハイフンが記入されます。**verbose** を省略すると（つまり、デフォルトでは）、データだけがファイルに書き込まれます。**verbose** キーワードは、出力フォーマットが **ASCII**、**fixed**、または **SQL** の場合に使用できます。次の出力フォーマットを指定できます。

- **ascii** — **ASCII** フォーマットでファイルに出力され、1 行につき 1 ローが書き込まれます。すべての値がカンマで区切られ、文字列はアポストロフィ（一重引用符）で囲まれます。デリミタと引用符文字列は、**delimited by** 句と **quote** 句を使用して変更できます。**quote** 句に **all** が指定されている場合は、文字列だけでなく、すべての値に引用符が付きます。

この他に 3 つの特別なシーケンスが認識されます。2 文字は改行文字を表し、"**¥**" は単一の円記号 (¥) を示し、**¥xDD** のようなシーケンスは 16 進コード **DD** の付いた文字を示します。これがデフォルトの出力フォーマットです。

- **dbasell** — カラム定義を含む **DBASEII** フォーマットで出力されます。最大 32 カラムまで出力できます。カラム名は 11 文字にトランケートされ、それぞれのローの各カラムのデータは 255 文字にトランケートされます。テーブルが存在しない場合には、テーブルを作成します。
- **dbaselll** — カラム定義を含む **DBASE III** フォーマットのファイルで出力されます。最大 128 カラムまで出力できます。カラム名は 11 文字にトランケートされ、それぞれのローの各カラムのデータは 255 文字にトランケートされます。
- **excel** — Excel 2.1 のワークシートに出力されます。ワークシートの最初のローには、カラムのラベル (ラベルが定義されていない場合はカラム名) が含まれます。実際のテーブル・データは、ワークシートの 2 番目以降のローに書き込まれます。
- **fixed** — 各カラムは、固定幅を持つ固定フォーマットで出力されます。各カラムの幅は、**column widths** を使用して指定できます。このフォーマットでは、カラム見出しは出力されません。

**column widths** 句を省略した場合、各カラム幅はそのカラムのデータ型をもとに、そのデータ型の値を保持するのに十分な大きさになるように計算されます。ただし、**long varchar** と **long binary** のデータのみ、デフォルトで 32KB になります。
- **foxpro** — カラム定義を含む **FoxPro** フォーマットのファイルで出力されます。最大 128 カラムまで出力できます。カラム名は 11 文字にトランケートされます。カラム名は 11 文字にトランケートされ、それぞれのローの各カラムのデータは 255 文字にトランケートされます。
- **html** — ハイパーテキスト・マークアップ言語 (**HTML**) フォーマットで出力されます。
- **lotus** — **Lotus WKS** フォーマットのワークシートとして出力されます。カラム名はワークシートの最初のローに配置されます。(Lotus 1-2-3 などの) 他のソフトウェアがロードできる **Lotus WKS** フォーマットのワークシートには、最大サイズに一定の制限があります。**Interactive SQL** が生成するファイルのサイズに制限はありません。
- **SQL** — テーブル内の情報を再作成するのに必要な **Interactive SQL input** 文が出力されます。

- XML – UTF-8 でエンコードされ、DTD が埋め込まれた XML ファイルに出力されます。バイナリ値は、バイナリ・データを 2 桁の 16 進文字列でエンコードし、CDATA ブロックに保存します。input 文のファイル・フォーマットには XML を指定できません。

#### escape character

16 進数のコードに使用されるデフォルトのエスケープ文字です。記号は円記号 (¥) で、たとえば、改行文字は ¥x0A と表記します。

デフォルトのエスケープ文字は、escape character 句を使用して変更できます。たとえば、感嘆符 (!) をエスケープ文字として使用するには、次のように入力します。

```
... escape character '!'
```

#### escapes

有効 (デフォルト) にすると、データベース・サーバは、円記号に続く文字を特別な文字として認識および解釈します。改行文字は ¥n という組み合わせで、それ以外の文字は 16 進数の ASCII コード (例: タブ文字は ¥x09) でデータに含めることができます。2 つの円記号 (¥¥) は 1 つの円記号として解釈されます。円記号の後に n、x、X、¥ 以外の文字がある場合、この文字と円記号は無関係の 2 つの文字として解釈されます。たとえば、¥q は "¥" と文字 "q" として挿入されます。

#### delimited by

ASCII 出力フォーマットのみで使用します。デリミタ文字列 (デフォルトはカンマ) は、カラムの間に置かれます。

#### quote

ASCII 出力フォーマットのみで使用します。文字列の値の前後を引用符文字列で囲みます。デフォルトは一重引用符です。quote 句に all を指定すると、引用符文字列は文字列だけでなく、すべての値の前後に置かれます。

#### column width

fixed フォーマットの出力にカラム幅を指定します。

### hexidecimal

バイナリ・データをアンロードして ASCII フォーマットのみ统一到する方法を指定します。オンに設定すると、バイナリ・データは 0xabcd のようなフォーマットでアンロードされます。オフに設定すると、バイナリ・データはアンロード時に (¥xab¥xcd のように) エスケープされます。asis を設定すると、値が制御文字を含んでいても、そのまま (エスケープなしで) 書き込まれます。asis は、テキストにタブや復帰改行などのフォーマット文字列が含まれる場合に使用します。

### encoding

ファイルの書き込みに使用するエンコーディングを指定できます。encoding 句を使用できるのは ASCII フォーマットだけです。

encoding を指定しない場合、Interactive SQL はファイルの書き込みに使用するコード・ページは次のリストに従って決定されます。このリストで上位に記載されているコード・ページの値は、下位の値よりも優先されます。

- default\_isql\_encoding オプションで指定されたコード・ページ (このオプションが設定された場合)
- Interactive SQL を開始するとき -codepage オプションで指定されたコード・ページ
- Interactive SQL を実行しているコンピュータのデフォルトのコード・ページ

### 例

**例 1** employee テーブルの内容を ASCII フォーマットでファイルに書き込みます。

```
select *
  from employee
go
output to employee.txt
  format ASCII
```

**例 2** 既存のファイルの末尾に employee テーブルの内容を追加します。また、クエリに関するすべてのメッセージも同様に、このファイルに含めます。

```
select *
  from employee
go
output to employee.txt append verbose
```

**例 3** 次の例は、改行文字を含む値をエクスポートする場合です。改行文字は数値 10 であり、SQL 文では文字列 "¥x0a" と表現されます。`hexadecimal` に `on` を設定して、次の文を実行します。

```
select 'line1 n x0aline2'
go
output to file.txt hexadecimal on
```

次のテキストを含む 1 行のファイルが作成されます。

```
line10x0aline2
```

しかし、同じ文で `hexadecimal` に `off` を設定した場合、次の行が出力されます。

```
line1 n x0aline2
```

最後に、`hexadecimal` に `asis` を設定すると、次の 2 行を含むファイルが作成されます。

```
line1
line2
```

`asis` を指定すると出力が 2 行になるのは、行間の改行文字を 2 桁の 16 進表現に変換したり、改行文字の先頭に他の文字を追加したりせず、そのままエクスポートされるためです。

#### 使用法

- `output` 文は、現在のクエリで取得された情報をファイルにコピーします。
- 出力フォーマットは、オプションの `format` 句を使用して指定できます。`format` 句を指定しないと、Interactive SQL の `output_format` オプションの設定が使用されます。
- 現在のクエリとは、[Results] タブや [Results] ウィンドウ枠に表示される情報を生成した、`select` 文や `input` 文のことです。現在のクエリが存在しない場合、`output` 文はエラーをレポートします。
- Interactive SQL では、現在のクエリの結果のみが [Results] タブに表示されます。前回のクエリの結果はすべて、現在のクエリの結果に置き換えられます。

#### パーミッション

このコマンドは、すべてのユーザが実行できます。

## parameters

**説明** Interactive SQL コマンド・ファイルに対するパラメータを指定します。

**構文** `parameters parameter1, parameter2, ...`

**例** 次の Interactive SQL コマンド・ファイルは、2つのパラメータを使用しています。

```
parameters department_id, file;
select emp_lname
  from employee
  where dept_id = {department_id}
 >#{file}.dat
```

このスクリプトを `test.sql` という名前のファイルに保存すると、次のコマンドを使用して Interactive SQL からこのファイルを実行できます。

```
read test.SQL [100] [data]
```

- 使用法**
- `parameters` 文は、コマンド・ファイルのパラメータに名前を付けて、コマンド・ファイル内で後でパラメータを参照できるようにします。
  - パラメータの参照は、ファイル内で名前付きパラメータを置き換える位置に `{parameter1}` を配置することにより行います。大カッコとパラメータ名の間スペースを入れることはできません。
  - コマンド・ファイルを呼び出すときに、必要なパラメータの数が不足していると、不足しているパラメータの値を要求するメッセージを表示します。

**パーミッション** このコマンドは、すべてのユーザが実行できます。

**参照** [read](#)



## read

説明	ファイルから Interactive SQL 文を読み込みます。
構文	<code>read [ encoding {identifier   string}] file_name [ parameters ]</code>
パラメータ	<p><code>encoding {identifier   string}</code>          ファイルの書き込みに使用するエンコーディングを指定できます。  <code>encoding</code> 句を使用できるのは ASCII フォーマットだけです。</p> <ul style="list-style-type: none"> <li>• <code>identifier</code> — 読み取るファイルを示すために使用する識別子です。</li> <li>• <code>string</code> — 読み取るファイルを示すために使用する文字列です。</li> </ul> <p><code>file_name</code>          読み取るファイルの名前です。</p> <p><code>parameters</code>          文ファイルにリストされているパラメータに対応します。</p>
例	<p>次は、<code>read</code> 文の例です。</p> <pre>READ status.rpt '160'</pre> <pre>READ birthday.SQL [ &gt;= '1988-1-1' ] [ &lt;= '1988-1-30' ]</pre>
使用法	<ul style="list-style-type: none"> <li>• <code>read</code> 文は、指定したファイルから一連の Interactive SQL 文を読み込みます。このファイルには、別の <code>read</code> 文も含め、有効な Interactive SQL 文を含めることができます。<code>read</code> 文のネストの深さに制限はありません。ファイル名に絶対パスが含まれていない場合、Interactive SQL はファイルを検索します。コマンド・ファイルを検索する場合、最初に現在のディレクトリを検索し、次に環境変数 <code>SQLPATH</code> で指定したディレクトリを検索し、次に環境変数 <code>PATH</code> で指定したディレクトリを検索します。指定したファイルにファイルの拡張子がない場合、Interactive SQL は、拡張子 <code>.SQL</code> 付きの同名のファイルも各ディレクトリで検索します。</li> <li>• <code>encoding</code> 引数を使用して、ファイルの読み取りに使用するエンコーディングを指定できます。<code>read</code> 文は、ファイルの読み取り時にエスケープ文字を処理せず、ファイル全体が指定のエンコード方式であるとみなします。<code>encoding</code> を指定しない場合、Interactive SQL はファイルの読み取りに使用するコード・ページは次のリストに従って決定されます。このリストで上位に記載されているコード・ページの値は、下位の値よりも優先されます。</li> <li>• <code>default_isql_encoding</code> オプションで指定されたコード・ページ (このオプションが設定された場合)</li> </ul>

- **Interactive SQL** を開始するとき `-codepage` オプションで指定されたコード・ページ
- **Interactive SQL** を実行しているコンピュータのデフォルトのコード・ページ
- パラメータは、コマンド・ファイル名の後にリストできます。これらのパラメータは、文ファイルの先頭の `parameters` 文で指定したパラメータと対応します。**Interactive SQL** は、ソース・ファイル内の `{parameter_name}` を適切なパラメータに置き換えます。この `parameter_name` は、対応するパラメータ名です。
- コマンド・ファイルに渡すパラメータは、識別子、数、引用符付きの識別子、または文字列です。パラメータの前後を引用符で囲むときは、置き換えるテキストの中に引用符を配置します。識別子、数、または文字列 (スペースまたはタブを含む) ではないパラメータは、角カッコ (`[ ]`) で囲む必要があります。これにより、コマンド・ファイル内で任意のテキストを置き換えることができます。
- コマンド・ファイルに渡されるパラメータが不足している場合、不足しているパラメータの値を要求するメッセージを表示します。

#### パーミッション

このコマンドは、すべてのユーザが実行できます。

## set connection

説明	現在のデータベース接続を別のサーバに変更します。
構文	<code>set connection {identifier   string   hostvar}</code>
パラメータ	<b>identifier</b> 接続情報に使用するログイン名識別子です。 <b>string</b> 接続情報に使用する文字列です。 <b>hostvar</b> ホスト名とポートを示す変数情報です。
使用法	<code>set connection</code> 文は、アクティブなデータベース接続を別のサーバへ変更します。現在の接続状態は保存され、再びアクティブな接続になるときに再開されます。 <code>connection_name</code> を省略し、名前のない接続がある場合は、この接続がアクティブな接続になります。
パーミッション	このコマンドは、すべてのユーザが実行できます。
参照	<a href="#">connect</a> , <a href="#">disconnect</a>

## start logging

説明	実行された SQL 文のログ・ファイルへの記録を開始します。
構文	<code>start logging file_name</code>
パラメータ	<code>file_name</code> セッションのログを保管するファイルです。
例	<code>c:</code> ディレクトリにある <code>filename.sql</code> というファイルへのロギングを開始します。  <code>start logging 'c:¥¥n filename.sql'</code>
使用法	<code>start logging</code> 文は、それ以降に実行されたすべての SQL 文を指定されたログ・ファイルにコピーします。指定されたファイルが存在しない場合は、 <b>Interactive SQL</b> によってファイルが作成されます。ロギングは、 <code>stop logging</code> 文を使用して明示的にロギング・プロセスを停止するか、現在の <b>Interactive SQL</b> セッションを終了するまで継続します。 <b>[SQL] - [Start Logging]</b> または <b>[SQL] - [Stop Logging]</b> を選択して、ロギングを開始または停止することもできます。
パーミッション	このコマンドは、すべてのユーザが実行できます。
参照	<a href="#">stop logging</a>

## stop logging

説明	現在のセッションで実行された SQL 文のロギングを停止します。
構文	<code>stop logging</code>
例	現在のロギング・セッションを停止します。 <code>stop logging</code>
使用法	<code>stop logging</code> 文は、実行された各 SQL 文のログ・ファイルへの書き込みを停止します。ロギングを開始するには、 <code>start logging</code> 文を使用します。 <code>[SQL] - [Start Logging]</code> または <code>[SQL] - [Stop Logging]</code> を選択して、ロギングを開始または停止することもできます。
パーミッション	このコマンドは、すべてのユーザが実行できます。
参照	<a href="#">start logging</a>

## system

説明	Interactive SQL 内から実行ファイルを起動します。
構文	<code>system '[path] file_name'</code>
パラメータ	<b>path</b> メモ帳プログラムへのパスです。 <b>file_name</b> 起動するプログラムのファイル名です。
例	メモ帳プログラムを起動します。メモ帳の実行ファイルへのパスが有効であることが前提です。 <code>system 'notepad.exe'</code>
使用法	指定されたリソース・ファイルを起動します。 <ul style="list-style-type: none"><li>• <code>system</code> 文は、全体を 1 行に記述する必要があります。</li><li>• <code>system</code> 文の末尾にコメントを付けることはできません。</li><li>• ファイル名とパスを一重引用符で囲みます。</li></ul>
パーミッション	このコマンドは、すべてのユーザが実行できます。
参照	<a href="#">connect</a>

# 索引

## 記号

¥! (感嘆符) エラー・メッセージのプレースホルダ 617

¥ (円記号)  
文字列の改行 847

" " (引用符) リテラル指定 846

# (シャープ記号)、テンポラリ・テーブル識別子のプレフィクス 235

% (パーセント記号)  
エラー・メッセージのプレースホルダ 617  
エラー・メッセージのリテラル 619

%nn! (プレースホルダの形式) 617

\* (アスタリスク)  
**select** 309

= (等号)  
カラム見出し名の変更 678  
変数の割り当て 679

@ (アットマーク)  
プロシージャ・パラメータ 470  
ルールの引数 222  
ローカル変数名 344-345

@@char\_convert グローバル変数 766

@@clientexpansion グローバル変数 766

@@cursor\_rows グローバル変数 766

@@datefirst グローバル変数 766

@@error グローバル変数  
**select into** 701  
ストアド・プロシージャ 201  
ユーザ定義エラー・メッセージ 619, 629

@@identity グローバル変数 550

@@isolation グローバル変数 766

@@langid グローバル変数 625

@@lock\_timeout グローバル変数 767

@@nestlevel グローバル変数 474  
ネストされたトリガ 300  
ネストされたプロシージャ 204

@@options グローバル変数 767

@@parallel\_degree グローバル変数 767  
**set parallel\_degree** 730

@@rowcount グローバル変数 767  
**set nocount** 767  
トリガ 298

@@scan\_parallel\_degree グローバル変数 767  
**set scan\_parallel\_degree** 738

@@textsize グローバル変数 767  
**readtext** 632  
**set textsize** 742

@@tranchained グローバル変数 767

@@version グローバル変数 618

## 数字

“0x”  
デフォルト内 138  
**writetext** コマンドと *image* データ 850

1 テーブルあたりのカラム数 71

16 進数  
"0x" プレフィクス 138

2 独立性レベル (繰り返し読み出し) 693

## A

**add** キーワード  
**alter role** 43  
**alter table** 51, 61

**all** キーワード  
**grant** 487, 513, 651  
**select** 677, 698  
**union** 793, 797  
**group by** 527  
**having** 句による否定 528  
**where** 843

**allow nested triggers** 設定パラメータ 300

**allow\_dup\_row** オプション、**create index** 163

## 索引

- alter database** コマンド 2–15
    - default** キーワード 2
    - for load** キーワード 7
    - for proxy\_update** キーワード 7
    - log on** キーワード 3
    - on** キーワード 2
    - with override** キーワード 7
  - オフライン・データベース 10
  - データベースのダンプ 10
  - alter encryption key** コマンド 16–24
  - alter object modify owner** コマンド 37–42
  - alter role** コマンド 43–47
    - activation** キーワード 43
    - add** キーワード 43
    - drop** キーワード 43
    - exclusive** キーワード 43
    - membership** キーワード 43
    - passwd** キーワード 43
  - alter table** コマンド 48–96
    - add** キーワード 51, 61
    - asc** オプション 55
    - check** オプション 59
    - clustered** 制約 55
    - constraint** キーワード 54
    - default** キーワード 51
    - desc** オプション 55
    - drop** キーワード 61
    - exp\_row\_size** オプション 61
    - fillfactor** オプション 55
    - foreign key** 制約 58
    - identity** キーワード 54
    - lock allpages** オプション 61
    - lock datapages** オプション 61
    - lock datarows** オプション 61
    - max\_rows\_per\_page** オプション 56
    - nonclustered** 制約 55
    - on** キーワード 57, 247
    - partition** 句 62
    - primary key** 制約 55
    - references** 制約 58
    - replace** キーワード 61
    - reservepagegap** オプション 57
  - sp\_dboption** およびロック・スキームの変更 92
  - unique** 制約 55
  - unpartition** 句 62
  - user keyword** 52
  - データ・コピーが必要な場合 88
  - ロック・スキーム 48
  - and** キーワード
    - 探索条件 842
    - 範囲終了 841
  - ANSI テープ・ラベル
    - load database** の **listonly** オプション 562
    - dump database** の **dumpvolume** 428
    - dump transaction** に対する **dumpvolume** 453
    - load transaction** の **listonly** 582
  - arithabort** オプション、**set**
    - arith\_overflow** および 715
  - arithignore** オプション、**set**
    - arith\_overflow** および 716
  - asc** インデックス・オプション
    - alter table** コマンド 55, 80
    - create index** コマンド 161
    - create table** コマンド 239
  - at** オプション
    - create existing table** 147
    - create proxy\_table** 215
    - dump database** 426
    - dump transaction** 451
    - load transaction** 580
    - create table** 248
    - load database** 560
- ## B
- B ツリー、インデックスおよび fillfactor 161
  - bcp** (バルク・コピー・ユーティリティ)
    - ロック・スキームの変更 94
  - begin transaction** コマンド 100
    - commit** 106
    - rollback** 先 670
  - begin...end** コマンド 99
    - if...else** 541
    - トリガ 291



- between** キーワード
    - check** 制約 275
    - where** 841
  - binary データ型
    - “0x” プレフィクス 138, 222
  - blocksize** オプション
    - dump database** 427
    - dump transaction** 452
    - load transaction** 581
    - load database** 560
  - break** コマンド 101, 848–849
  - bulk array size** オプション、**set**
    - bulk array size** 716
  - bulk batch size** オプション、**set**
    - bulk batch size** 717
  - by** ロー集合サブグループ 107
- C**
- capacity** オプション
    - dump database** 427
    - dump transaction** 452
  - char* データ型
    - ローのソート順 614
  - check** オプション
    - alter table** 59
    - create table** 243
  - checkpoint** コマンド 102–103
  - checktable** オプション、**dbcc** 317–338
  - cis\_rpc\_handling** オプション、**set** コマンド 720
  - CIS。 116
  - clear** Interactive SQL コマンド 856
  - clientapplname** オプション、**set** コマンド 720
  - clienthostname** オプション、**set** コマンド 720
  - clientname** オプション、**set** コマンド 720
  - close** コマンド 104
  - clustered** 制約
    - alter table** 55
    - create table** 239
  - cntrltype** オプション
    - disk init** 367
    - disk reinit** 380
  - commit work** コマンド。 105
  - commit** コマンド 105–106
    - begin transaction** 100, 106
  - rollback** 106, 670
  - compact** オプション、**reorg** コマンド 640
  - compute** 句 107–115
    - select** 691
    - order by** 612, 691
    - by** 111
  - configure** Interactive SQL コマンド 857
  - connect** Interactive SQL コマンド 858
  - connect to** コマンド 116–118
  - constraint** キーワード
    - alter table** 54
    - create table** 190, 238
  - consumers** オプション、**update statistics**
    - コマンド 813, 818, 819, 825
  - continue** コマンド 119
    - while** ループ 848
  - create archive database** コマンド 120–121
  - create database** コマンド 122–137
    - default** オプション 122
    - disk init** 370
    - for load** キーワード 125
    - for proxy\_update** キーワード 126
    - log on** キーワード 123
    - on** キーワード 122
    - with dbid** キーワード 123
    - with default\_location** キーワード 123
    - with override** キーワード 123
    - パーミッション 513
  - create default** コマンド 138–141
    - バッチ 139
  - create encryption key** コマンド 142–145
  - create existing table** コマンド 146–152
    - サーバ・クラスの変換 150
    - データ型の変換 150
    - リモート・テーブルへのマッピング 146
    - リモート・プロシージャの定義 150
  - create function (SQLJ)** コマンド 156–158
  - create function** コマンド 153–155
  - create index** コマンド 159–181
    - insert** 546
    - インデックスのオプションと
      - ロック・モード 177
    - 記憶領域管理プロパティ 176
  - create plan** コマンド 188–189
  - create precomputed result set** コマンド 190–192

**create procedure** (SQLJ) コマンド 210–214  
**create procedure** コマンド 193–209  
**select \*** 202  
「ストアド・プロシージャ」「拡張ストアド・プロシージャ (ESP)」参照  
パラメータの順序 470, 473  
リターン・ステータス 205–206  
**create proxy\_table** コマンド 215–218  
リモート・テーブルへのプロキシ・テーブルのマッピング 215  
**create role** コマンド 219–221  
**grant all** 364  
**create rule** コマンド 222–225  
**create scheme** コマンド 226–227  
**create service** コマンド 228–232  
構文 228  
パラメータ 228  
例 229  
**create table** コマンド 233–287  
null 値 52, 237  
カラム順 613  
記憶領域管理プロパティ 277  
リモート・テーブルへのプロキシ・テーブルのマッピング 259  
ロック・スキームの指定 277  
**create trigger** コマンド 290–304, 512, 660  
**create view** コマンド 305–313  
SQL 抽出テーブル 308

## D

**dataserver** ユーティリティ 376  
「ASE ユーティリティ・ガイド」参照  
**disk mirror** 376  
**disk remirror** 385  
**dateformat** オプション、**set** 722  
**datetime** データ型  
「**set** コマンド」参照  
**dbcc** (データベース一貫性チェッカ)  
**readtext** 632  
**dbcc checktable**、仮想ハッシュ・テーブル 317  
**dbcc complete\_xact 1pc** コマンド 335  
**dbcc pravailbletempdbs** と **tempdbs** 321  
**dbcc** コマンド 314–341

個々の **dbcc** オプションを参照  
DB-Library のプログラム  
**prepare transaction** 616  
**dbwritetext** と **dbmoretext**、**writetext**  
との比較 852  
**set** オプション 728, 756  
**waitfor mirrorexit** 838  
ブラウザ・モード 693  
**deallocate cursor** コマンド 342  
**declare cursor** コマンド 346–353  
スクロール可能カーソル 348  
**declare** コマンド 344–345  
**default database size** 設定パラメータ  
*sysconfigures* 130  
**default** オプション  
**create database** コマンド 122  
**default** キーワード  
**alter database** 2  
**alter table** 51  
**create table** 236  
**default** セグメント  
拡張 13  
**delete statistics** コマンド 363–364  
**delete** コマンド 354–362  
**readpast** オプション 354  
**truncate table** との比較 791  
トリガ 296  
**deleted** テーブル  
トリガ 295, 297  
**density** オプション  
**dump database** 426  
**dump transaction** 452  
**load transaction** 580  
**load database** 560  
**desc** インデックス・オプション  
**alter table** コマンド 80  
**create index** コマンド 161  
**create table** コマンド 239  
**desc** オプション  
**alter table** 55  
**disconnect** Interactive SQL コマンド 861  
**disconnect** コマンド 116–118  
**disk init** コマンド 365–372  
その後の *master* データベースのバックアップ 369  
**disk mirror** コマンド 373–377

- disk refit** コマンド 378
  - disk reinit** コマンド 379–383
    - 「**disk init** コマンド」参照
  - disk remirror** コマンド 384–386
    - 「ディスク・ミラーリング」参照
  - disk resize** コマンド 387–388
  - disk unmirror** コマンド 389–392
    - 「ディスク・ミラーリング」参照
  - dismount** オプション
    - dump database** 429
    - dump transaction** 453
    - load transaction** 582
    - load database** 561
  - distinct** キーワード
    - create view** 305
    - select** 678, 698
  - drop database** コマンド 393–395
    - 損傷したデータベース 319
  - drop default** コマンド 396–397
  - drop encryption key** コマンド 398
  - drop function (SQLJ)** コマンド 401
  - drop function** コマンド 400
  - drop index** コマンド 402–403
  - drop procedure** コマンド 409–410
    - グループ化されたプロシージャ 409, 470
  - drop role** コマンド 411
  - drop rule** コマンド 413–414
  - drop service** コマンド 415
  - drop table** コマンド 416–418
  - drop trigger** コマンド 420
  - drop view** コマンド 421
  - drop** キーワード
    - alter role** 43
    - alter table** 61
  - dump** 445
  - dump configuration** コマンド 422–423
  - dump database**
    - compress** オプション 426
    - プラットフォーム間 435
  - dump database** コマンド 424–447
    - dump transaction** 435
    - master* データベース 436
    - select into** 702
    - 「ダンプ」、「データベース」参照
    - create database** 135
    - disk init** 369
    - dump transaction with no\_log** 451
    - dump transaction** コマンド 448–468
      - compress** オプション 450
      - select into/bulkcopy/pllsort** 458
      - standby\_access** オプション 455
      - trunc log on chkpt** 458
      - with no\_log** オプション 461
      - with no\_truncate** オプション 455, 459
      - with truncate\_only** オプション 460
    - 「ダンプ」、「トランザクション・ログ」参照
    - disk init** 369
  - dumpvolume** オプション
    - dump database** 428
    - dump transaction** 453
    - load transaction** 581
    - load database** 561
- ## E
- else** キーワード。「**if...else** 条件」参照
  - enable xact coordination** 設定パラメータ 745
  - end** キーワード 99
  - escape** キーワード
    - where** 841
  - exclusive** キーワード
    - alter role** 43
  - execute** コマンド 469–476
    - create procedure** 201
  - exists** キーワード
    - where** 842
  - 終了
    - 無条件、**return** コマンド 649
  - exit** Interactive SQL コマンド 862
  - exp\_row\_size** オプション
    - select into** 683
    - 指定、**create table** 244
    - 指定、**select into** 683
    - create table** 244, 278
    - 設定、**alter table...lock** 84
  - external** オプション
    - create existing table** 146
    - create proxy\_table** 215
    - create table** 246

## F

- fast** オプション
  - dbcc indexalloc** 320
  - dbcc tablealloc** 321, 324
- fetch** コマンド 477–484
  - 複数ロー 480
- file** オプション
  - dump database** 430
  - dump transaction** 454
  - load transaction** 581
  - load database** 561
- fillfactor** オプション
  - alter table** 55
  - create index** 161, 176
  - create table** 239, 277
- fillfactor** 値
  - alter table...lock** 82
- FIPS フラガ
  - set** オプション 724
  - 検出されない **update** 拡張機能 810
- fix** オプション
  - dbcc indexalloc** 320
  - dbcc tablealloc** 316
  - dbcc** 316, 321, 324
- fix\_text** オプション、**dbcc** 319, 334
- for browse** オプション、**select**
  - 使用不可の **union** 796
- for load** オプション
  - create database** 135
- for load** キーワード
  - alter database** 7
  - create database** コマンド 125
- for load** を使用して作成したデータベースの "don't recover" (リカバリ禁止) ステータス 136
- for proxy\_update** キーワード
  - alter database** 7
  - create database** コマンド 126
- foreign key** 制約
  - alter table** 58
  - create table** 244
- forwarded\_rows** オプション、**reorg** コマンド 640
- from** キーワード
  - delete** 354
  - load transaction** 580
  - select** 685
  - grant** 500

- load database** 560
- update** 801
- full** オプション
  - dbcc indexalloc** 320
  - dbcc tablealloc** 320, 323

## G

- goto** キーワード 485
- grant dbcc**
  - 構文 486
  - 使用 512
  - 説明 486
  - パラメータ 491
- grant** コマンド 486–523
  - all** キーワード 487, 651
  - public** グループ 490
  - 役割 525
- group by** clause
  - select** 689
- group by** 句 527–540
  - select** 689
  - having** 句 527–540
  - having** 句を使用しない 538
  - having** 句、Transact-SQL 内 532
  - having** 句、ソート順 539
  - having** 句、標準の SQL 532
  - 集合関数 527, 530
  - ビュー 310
- guest ユーザ
  - パーミッション 516

## H

- having** 句 527–540
  - group by** 527–540
  - select** 690
  - Transact-SQL 内の **group by** の拡張機能 532
  - 集合関数 528, 530
  - all** 528
- holdlock** キーワード
  - readtext** 630
  - select** 687

## I

## I/O

- prefetch および **delete** 355
- prefetch および **select** 686
- prefetch および **update** 801
- 実際の全コストの表示 (**statistics io**) 740
- デバイス、ディスクのミラーリング先 373

## ID

- sa\_role** およびデータベース所有者 761
- session authorization** 764
- set proxy** 764
- setuser** コマンド 773

**identity burning set factor** 設定パラメータ 549

## IDENTITY カラム

- 追加、削除、または変更、**alter table** 90
- null 値 550
- 値のギャップ 725
- 値の挿入 544
- 更新不可 808
- 最大値 549
- 選択 550, 702–703
- テーブルへの挿入 549
- テーブルを作成 276
- デフォルト 87
- ビュー 310

IDENTITY カラムの明示的な値 549, 725

## identity キーワード

- alter table** 54
- create table** 237

identity ギャップ  
設定 277

identity ギャップの設定 277

**if update** 句、**create trigger** 290, 291, 299

**if...else** 条件 541–543

- continue** 119
- ローカル変数 345

**ignore\_dup\_row** オプション、**create index** 163

## image データ型

- order by** 使用不可 612
- writetext** 850
- 返されるデータの長さ 699, 742
- トリガ 296
- readtext** 630

別のデバイスの記憶領域 630

## in キーワード

- alter table** 59
- check** 制約 275
- where** 842

## init オプション

- dump database** 430
- dump transaction** 454

**input** Interactive SQL コマンド 863

**insert** コマンド 544–553

- create default** 138
- IDENTITY カラム 549
- null/not null カラム 309
- update** 546
- トリガ 296, 299
- ビュー 310, 550–551

## inserted テーブル

- トリガ 295, 297

Interactive SQL コマンド 855–880

- clear** 856
- configure** 857
- connect** 858
- disconnect** 861
- exit** 862
- input** 863
- output** 869
- parameters** 874
- read** 875
- set connection** 877
- start logging** 878
- stop logging** 879
- system** 880

## into キーワード

- insert** 544
- select** 679, 700
- union** 793, 797
- fetch** 477

## is null キーワード

- where** 841

## isnull システム関数

- insert** 548
- print** 619
- select** 699

## 索引

### J

- Java カラム、追加 89
- Java 項目
  - remove java** コマンド 637

### K

- kill** コマンド 554–556

### L

- like** キーワード
  - alter table** 59
  - check** 制約 275
  - where** 840
- listonly** オプション
  - load transaction** 582
  - load database** 562
- load** 445
- load database**
  - compress** オプション 559
  - プラットフォーム間 570
- load database** コマンド 557–577
- load transaction**
  - compress** オプション 580
- load transaction** コマンド 578–592
- lock allpages** オプション
  - alter table** 61
  - create table** コマンド 244
  - select into** コマンド 683
- lock datapages** オプション
  - alter table** 61
  - create table** コマンド 244
  - select into** コマンド 683
- lock datarows** オプション
  - alter table** 61
  - alter table** コマンド 92
  - create table** コマンド 244
  - select into** コマンド 683
- lock nowait** オプション、**set lock** コマンド 727
- lock table** コマンド 593
- lock wait** オプション、**set** コマンド 727

**log on** キーワード

- alter database** 3
- create database** 123

### M

- master** データベース
  - alter database** 9
  - create database** 134
  - disk init** 369
  - disk mirror** 374
  - disk refit** 378
  - disk reinit** 379
  - disk remirror** 384
  - disk unmirror** 390
  - 「**master** データベースのリカバリ」
    - 「データベース」参照
  - データベースの削除 393
  - トランザクション・ログの消去 437, 460
  - バックアップ 460
- master**<データベース: システム・テーブルのデフォルト・パーミッションの取り消し; 取り消し: システム・テーブルのデフォルト・パーミッション 517
- master** データベースのリカバリ 437
  - create database** 134
  - disk init** 369
- max\_rows\_per\_page** オプション
  - select into** 683
  - alter table** 56, 82
  - create index** 162, 176
  - create table** 240, 278
- membership** キーワード
  - alter role** 43
- model** データベース、コピー 131
- mount** コマンド 599–604
  - 「**quiesce database**」参照
  - 「**unmount**」参照
- N**
  - name** オプション
    - disk init** 365
    - disk reinit** 379

**nowait** オプション  
   **lock table** コマンド 593  
   **set lock** コマンド 727  
**no\_truncate** オプション、**dump transaction** 455  
**nodismount** オプション  
   **dump database** 429  
   **dump transaction** 453  
   **load transaction** 582  
   **load database** 561  
**nofix** オプション、**dbcc**  
   **checkalloc** 316  
   **indexalloc** 321  
   **tablealloc** 324  
**noinit** オプション  
   **dump database** 430  
   **dump transaction** 454  
**nonclustered** 制約  
   **alter table** 55  
   **create table** 239  
**not null** キーワード  
   **create table** 52, 237  
**not null** 値  
   **insert** 548  
   **select** 文 699  
   デフォルトの削除 396  
   ビュー 309  
**not** キーワード  
   **where** 839  
**notify** オプション  
   **dump database** 430  
   **dump transaction** 455  
   **load transaction** 583  
   **load database** 563  
**nounload** オプション  
   **dump database** 429  
   **dump transaction** 454  
   **load transaction** 582  
   **load database** 561  
**null** カラムの内部データ型 265  
**null** キーワード  
   **create table** 52, 237  
**null** 値  
   **group by** 529  
   **null** デフォルト 140, 224  
   **select** 文 699  
   **text** カラムと **image** カラム 547

新しいカラム 140  
 新しいルールとカラム定義 224  
 カラムのデフォルト 140, 224  
 検査制約 276  
 スタアド・プロシージャで返せない 649  
 ソート順 612  
 代入値の挿入 548  
 定義 140, 265  
 デフォルトの削除 396  
 トリガ 299

## O

**OAM** (オブジェクト・アロケーション・マップ)  
   ページ  
     **dbcc indexalloc** 321  
     テーブルに関する **dbcc** レポート 324  
**on** キーワード  
   **alter database** 2  
   **alter table** 57, 247  
   **create database** コマンド 122  
   **create index** 164, 170  
   **create table** 63, 242, 682  
**online database** コマンド 572, 605–607  
   **dump transaction** 585  
   **load transaction** 585  
   アップグレード 588  
   データベースをオンラインにする 572  
**Open Client** アプリケーション  
   **set** オプション 728, 756  
   キーワード 728  
**open** コマンド 608  
**optdiag** ユーティリティ  
   統計の上書き、**create index** 176  
   シミュレートした統計情報のロード 364, 766  
**optimized** レポート  
   **dbcc indexalloc** 320, 321  
   **dbcc tablealloc** 324  
**or** キーワード  
   **where** 843  
**order by** 句 609–615  
   **compute by** 111, 612, 691  
   **select** 690

**output** Interactive SQL コマンド 869

**output** オプション

**create procedure** 195, 470

**execute** 470

リターン・パラメータ 470

## P

**parallel** キーワード、**select** コマンド 686

**parallel\_degree** オプション、**set** コマンド 730

**parameters** Interactive SQL コマンド 874

**partition** 句、**alter table** コマンド 62

**physname** オプション

**disk init** 365

**disk reinit** 379

**prefetch** キーワード

**delete** 355

**select** 686

**set** 733

**update** 801

**primary key** 制約

**alter table** 55

**create table** 239

**print** コマンド 617–619

**raiserror** の使用 619

ローカル変数 345

**procedure** オプション

**create existing table** 146

**proxy** オプション、**set**

付与 522

**public** キーワード

**grant** 490

**revoke** 653

"public" グループ

**grant** 490

**revoke** 653

「グループ」参照

## Q

**quiesce database**

暗号化 623

**quiesce database** コマンド 620–624

## R

**raiserror** コマンド 625–629

**print** の使用 619

**restricted\_select\_list** パラメータ 626–627

**print** 629

ローカル変数 345

**read** Interactive SQL コマンド 875

**readpast** オプション

**delete** コマンド 355

**readtext** コマンド 631

**select** コマンド 683

**update** コマンド 801

**writetext ÉRÉ)ÉiÈh** 850

独立性レベル 708

**readtext** コマンド 630–634

**rebuild** オプション、**reorg** コマンド 641

**reclaim\_space** オプション、**reorg** コマンド 641

**reconfigure** コマンド 635

**references** 制約

**alter table** 58

**create table** 242

**reindex** オプション、**dbcc** 322–323

**remove java** コマンド 637–639

**reorg** コマンド 640–646

**reservepagegap** オプション

**select into** 683

**alter table** 57, 82

**create index** 163, 176

**create table** 241, 278

**retaindays** オプション

**dump database** 429

**dump transaction** 454

**return** コマンド 647–649

**revoke dbcc**

構文 650

説明 650

パラメータ 654

例 655

**revoke** コマンド 650–668

**public** グループ 653

オブジェクト・パーミッションおよびコマンド・パーミッション 502

**role** オプション

**revoke** 667

**set** コマンド 737



**rollback transaction** コマンド。  
「**rollback** コマンド」参照  
**rollback trigger** コマンド 672  
**rollback work** コマンド。「**rollback** コマンド」参照  
**rollback** コマンド 669–671  
  **begin transaction** 100  
  **commit** 106  
  トリガ 297, 300

## S

**save transaction** コマンド 673–674  
**select into** コマンド 679–701  
  **compute** 112, 691  
**select into/bulkcopy/pllsort**  
  データベース・オプション  
  **select into** 701  
  トランザクション・ログのダンプ 458  
**select** コマンド 675–711  
  **create procedure** 202  
  **create view** 306  
  **group by** 句と **having** 句 527  
  **insert** 547  
  **select \*** 構文の機能 709  
  **top n** 678  
  返される *text* データのサイズ 742  
  変更されたロー 70, 88  
  変数 344  
  ローカル変数 345  
**select** リスト 626–627, 678–679  
  **order by** 690  
  **union** 文 795  
**session authorization** オプション、**set**  
  取り消し 522  
**set connection** Interactive SQL コマンド 877  
**set** オプション  
  エクスポート可能 770  
**set** オプションのエクスポート 770  
**set** コマンド 712–772  
  **lock wait** 727  
  **statistics simulate** 740  
  **strict\_dtm\_enforcement** 741  
  **transaction isolation level** 744  
  個別の **set** オプション参照

  ストアド・プロシージャ内 207  
  デフォルト設定 755  
  トリガ内 297  
  役割 737  
  **update** 801  
**setuser** コマンド 773–774  
**shared** キーワード  
  **select** 688  
**shutdown** コマンド 775–779  
**size**  
  データベースの拡張 3  
**size** オプション  
  **disk init** 366, 380  
**sp\_bindefault** システム・プロシージャ  
  **create default** 138  
**sp\_bindrule** システム・プロシージャ  
  **create rule** 223  
**sp\_dboption** システム・プロシージャ  
  チェックポイント 102  
**sp\_depends** システム・プロシージャ 266  
**sp\_transactions** システム・プロシージャ 318  
**sp\_unbindefault** システム・プロシージャ 396  
**sp\_unbindrule** システム・プロシージャ  
  **create rule** 223  
  **drop rule** 413  
SQL 規格  
  **session authorization** 738  
  **set** オプション 771  
SQL 抽出テーブル  
  **create view** コマンド 308  
  ビューの作成 312  
SQLJ ストアド・プロシージャ  
  作成 210–214  
**standby\_access** オプション  
  **dump transaction** 455  
  **online database** 605  
**start logging** Interactive SQL コマンド 878  
**startserver** ユーティリティ・コマンド  
  「ASE ユーティリティ・ガイド」参照  
  **disk remirror** 385  
  ディスク・ミラーリング 376  
**statistics simulate** オプション、**set** コマンド 740  
**statistics** 句、**create index** コマンド 164  
**stop logging** Interactive SQL コマンド 879  
**strict dtm enforcement** 設定パラメータ 741

## 索引

**strict\_dtm\_enforcement** オプション、  
**set** コマンド 741

**string\_truncation** オプション、**set** 741  
**insert** 547  
**update** 807

**stripe on** オプション  
**dump database** 428  
**dump transaction** 453  
**load transaction** 581  
**load database** 561

suspect (疑わしい) インデックス。  
「**reindex** オプション」、**「dbcc」** 参照

**syb\_identity** キーワード  
**select** 702

**sybsecurity** データベース、削除 394

**syscolumns** テーブル 316

**syscomments** テーブル  
デフォルトの定義 139  
トリガ定義 302, 311  
プロシージャ定義 207  
ルールの定義 224

**sysconfigures** テーブル  
**database size** パラメータ 130

**sysdevices** テーブル  
**disk init** 369  
ミラー名 389

**sysindexes** テーブル  
複合インデックス 178

**syslogs** テーブル  
**dbcc checktable** の実行 317  
別のデバイスへの配置 375, 385  
「リカバリ」「トランザクション・ログ」参照

**sysmessages** テーブル  
**raiserror** 625

**sysobjects** テーブル  
トリガ ID 302

**sysprocedures** テーブル  
トリガの実行プラン 302

**sysprotects** テーブル  
**grant/revoke** 文 511, 659  
**sp\_changegroup** 516

**sys.servers** テーブル  
Backup Server 438, 463  
**load database** 572

**sysstatistics** テーブル、**delete statistics**  
による統計の削除 363

**system** Interactive SQL コマンド 880

**sys.transactions** テーブル 318

**sys.usermessages** テーブル  
**raiserror** 625

## T

**tempdb**  
**dbcc pravailabletempdbs** 321

**tempdb** データベース  
**sysobjects** テーブル 266  
**systypes** テーブル 267  
オブジェクトの追加 267

**tempdbs**  
**create database** の使用法 131

**text** データ型  
**order by** 使用不可 612  
**textsize** 設定 742  
返されるデータの長さ 699, 742  
トリガ 296  
別のデバイスの記憶領域 630  
**update** 807

**textptr** 関数 630, 632

**time** オプション  
**reorg** 641  
**waitfor** 837

**to** オプション  
**dump database** 426  
**dump transaction** 451  
**revoke** 659

Transact-SQL コマンド  
拡張機能 532  
実行 469

Transact-SQL コマンドの動的な実行 469

**truncate table** コマンド 791-792  
**delete** コマンドより高速 357  
**delete** トリガ 297

## U

- union** 演算子 793–796
  - 使用上の制限 796
  - テーブルの最大数 794
- union** コマンド、変更 794
- unique** キーワード
  - alter table** 55
  - create index** 159
  - create table** 190, 239
- unload** オプション
  - dump database** 429
  - dump transaction** 454
  - load transaction** 582
  - load database** 561
- unmount**
  - 「**mount**」参照
  - 「**quiesce database**」参照
  - 暗号化 798
- update all statistics** コマンド 812–816
- update index statistics** コマンド 817–822
- update statistics** コマンド 823–833
  - create index** 169
  - スキャン・タイプ 829
  - ソート条件 829
  - ロック 829
- update table statistics** コマンド 834–835
- update** コマンド 800–811
  - トリガおよび **if update** 299
  - ignore\_dup\_key** 163
  - ignore\_dup\_row** 173
  - insert** 546
  - readpast** オプション 801
  - トリガ 296
  - ビュー 310, 809
- update** コマンドと **delete** コマンドのワーク・  
テーブル 357, 804
- update**、スクロール可能カ  
ーソルで使用不可 807
- us\_english** 言語、曜日設定 756
- use** コマンド 836
- user keyword**
  - alter table** 52
  - create table** 237
- using...values** オプション、**update statistics**  
コマンド 812, 818, 824

## V

- values** オプション、**insert** 544
- varchar** データ型
  - スペースおよび **insert** 547
- vdevno** オプション
  - disk init** 366, 379

## W

- wait** オプション、**lock table** コマンド 593
- waitfor** コマンド 837–838
- waitfor** コマンドでの **mirrorexit** キーワード 837
- where current of** 句
  - delete** 356
  - update** 802
- where** 句 839–847
  - delete** 354
  - group by** 句 533
  - having** 845
  - 繰り返し 535
  - 使用できない集合関数 845
- where** 句の **any** キーワード 842
- while** キーワード 848–849
  - ループを終了、**break** 101
  - continue** 119
  - ループ 848
- with check option** オプション
  - create view** 306
  - ビュー 311
- with consumers** オプション、**update statistics**  
コマンド 813, 818, 819, 825
- with dbid** キーワード
  - create database** コマンド 123
- with default\_location** キーワード
  - create database** コマンド 123
- with no\_truncate** オプション、  
**dump transaction** 455
- with override** オプション 411
- with override** キーワード
  - alter database** 7
  - create database** コマンド 123
- with recompile** オプション
  - create procedure** 195
  - execute** 471

## 索引

**with standby\_access** オプション  
  **dump transaction** 455  
**with statistics** 句、**create index** コマンド 164  
**with** キーワード  
  **rollback trigger** 672  
  **set role** コマンド 737  
**writetext** コマンド 850–853  
  トリガ 297

## X

X/Open XA 319

## あ

アーカイブ・データベースの互換性 445  
アーカイブ・データベースのマテリアライズ 574  
アーカイブ・データベースへのアクセス  
  **create archive database** コマンド、使用 120  
  アーカイブ・データベースのマテリアライズ 574  
  圧縮ダンプ 444  
  互換性 445  
  リカバリなし 574  
  論理デバイス 574  
アクセス、オブジェクト。「パーミッション」  
  「ユーザ」参照  
アクセス、テープの ANSI 制限 465  
アクセントの区別  
  **compute** 114  
  **group by** 539  
  辞書ソート順 613  
アスタリスク (\*)  
  **select** 309  
値  
  IDENTITY カラム 549  
  プロシージャのパラメータまたは引数 470  
値の比較  
  **where** 句 847  
  ソート順 613–614  
  データ型の変換 847  
新しいデータベースのロケーション 122  
圧縮ダンプ

  アーカイブされたデータベースを使用 444  
圧縮バックアップ  
  アンロード 559, 580  
  作成 426, 450  
圧縮バックアップのアンロード 559, 580  
圧縮バックアップの作成 426, 450  
アットマーク (@)  
  プロシージャ・パラメータ 470  
  ルールの引数 222  
  ローカル変数名 344–345  
誤り、ユーザ。「エラー」参照  
アロケーション・マップ。「OAM (オブジェクト・アロケーション・マップ)」参照  
暗号化  
  **unmount** 798  
  **quiesce database** 623  
暗号化キー  
  作成 142–145

## い

異常なプロセス  
  **waitfor errorexit** 838  
依存性、データベース・オブジェクト  
  **sp\_depends** システム・プロシージャ 266  
一意性制約 271  
一貫性の検査。「dbcc コマンド」参照  
インクリメンタル・バックアップ。  
インデックス  
  ソート順の指定、**alter table** 80  
  ソート順の指定、**create index** 172  
  ソート順の指定、**create table** 259  
  **dbcc indexalloc** 320  
  **max\_rows\_per\_page** 241  
  **truncate table** 791  
  **update index statistics** 817–822  
  **update statistics** 169  
  オブジェクト・アロケーション・マップ 321  
  キー値 820, 827  
  降順 55  
  削除 402–403  
  作成 159–181  
  種類 160

順序の指定 48  
 ジョイン 169  
 昇順 55  
 整合性の検査 (**dbcc**) 322  
 ノンクラスタード 160  
 複合 178  
 ページ割り付けの検査 320  
 命名 160  
 リスト作成 402  
 インデックス・キー  
   **asc** オプション、順序 172  
   **desc** オプション、順序 172  
   順序 172  
 インデックスのリーフ・レベル  
   クラスタード・インデッ  
   クス 56, 160, 161, 240  
 インデックス・ページ  
   **fillfactor** の効果 55, 161, 239  
   リーフ・レベル 56, 160, 161, 240  
 インメモリ・マップ 10  
 引用符 (" ")  
   リテラル指定 846

## う

疑わしいパーティション、プラットフォーム間  
   のダンプとロード 570  
 埋め込み、データおよびブランク 547  
 上書き。 7

## え

エイリアス、コラム  
   **compute** 句で使用 112  
   **group by** 後の禁止 528, 529  
 エクステンツ 169  
   **create table** 259  
   **dbcc indexalloc** レポート、インデックス 321  
   テーブルに関する **dbcc** レポート 324

エラー  
   「エラー・メッセージ」「SQLSTATE コード」  
   参照  
   データ型変換 237  
   ユーザ定義の番号 625  
   リターン・ステータス値 648  
   割り付け 316, 321, 324  
 エラー処理  
   **dbcc** 334  
   文字セット変換 719  
 エラー・メッセージ  
   12207 593, 595  
   文字セット変換 719  
   ユーザ定義 625-629  
   ユーザ定義の出力 619  
 円記号 (¥)、文字列の改行 847

## お

オーバフロー・エラー  
   **set arithabort** 715  
 オーバヘッド  
   トリガ 296  
 大文字と小文字の区別  
   **compute** 113  
   **group by** 539  
   ソート順 613  
 大文字の優先度を付けた順位 613  
 オブジェクト所有者。「データベース・オブジェ  
   クト所有者」参照  
 オブジェクト・パーミッション  
   **grant** 486-523  
   **grant all** 514  
   「コマンド・パーミッション」「パーミッシ  
   ョン」参照  
 オブジェクト名、データベース  
   ストアド・プロシージャ内 204, 206  
   パラメータとして使用 194  
 オフセット位置、**readtext** コマンド 630  
 オフライン・データベースと **alter database**  
   コマンド 10  
 オフライン・ページのオンライン強制 435  
 オリジナル ID、(**setuser** コマンド)の再開 773

## か

- カーソル
  - compute** 句 111
  - grant** 511
  - group by** 530
  - order by** 612
  - select** 699
  - オープン 608
  - クローズ 104
  - 更新可能 351
  - 更新可能カーソルで使用できない **union** 794
  - スキャン 350
  - スコープ 348
  - 宣言 346-353
  - データ型の互換性 478
  - ハロウィーン問題 352
  - フェッチ 477-484
  - 読み取り専用 351
  - ローの更新 807
  - ローの削除 359
  - 割り付け解除 342
- カーソル結果セット 350
  - データ型 478
  - ローの返送 477
- カーソルのオープン 608
- カーソルのクローズ 104
- カーソルのフェッチ 477-484
- カーソルの割り付け解除 342
- 改行、文字列 847
- 概念(論理)テーブル 295, 297
- 外部キー 271
- カウンタ、**while** ループ。「**while** ループ」参照
- 書き込み操作
  - text* または *image* 850
- 拡張
  - データベース記憶領域 2, 37
- 拡張カラム、Transact-SQL 532, 534
- 拡張機能、Transact-SQL 532
- 拡張ストアド・プロシージャ
  - C ランタイム・シグナルは使用不可 203
  - 削除 409
  - 作成 193-209
  - 実行 469
- 数(量)
  - 1 データベースあたりのテーブル数 259
  - 1 ローあたりのバイト数 71
  - having** 句の探索引数 528
  - アクティブなダンプまたはロード 438, 463, 572, 589
  - 返されるテキスト内のバイト数 632
  - クエリで使用できるテーブル数 685
  - クラスタード・インデックス 160
  - 更新 300
  - 指定セグメント 131
  - スキャン (**statistics io**) 740
  - ストアド・プロシージャのパラメータ 201
  - データベース・サーバで管理可能 131
  - デバイス・フラグメント 131
  - ネストのレベル、トリガ 300
  - ネスト・レベル 204
  - ノンクラスタード・インデックス 160
  - 引数およびプレースホルダ 618
  - 引数、**where** 句 847
  - フォーマット文字列内のプレースホルダ 619
  - 複数のトリガ 296
  - 物理読み込み (**statistics io**) 740
  - プロシージャ内のパラメータ 345
  - 分布ヒストグラムのステップ 164
  - ユーザ定義の役割 220
  - 論理読み込み (**statistics io**) 740
- 仮想デバイス番号 366, 379
- 可変長カラム
  - 空文字列 547
  - 保管順序 613
- 空の文字列(“ ”)または(¥' ¥' )
  - 更新 806
  - シングル・スペース 547
- 空ページに対する満杯率 48
- カラム
  - order by** 690
  - リストおよび **insert** 544
  - 1 テーブルあたり 71
  - 1 テーブルあたりの最大数 71
  - group by** 528
  - IDENTITY 値のギャップ 725
  - null 値と検査制約 276

null 値とデフォルト 140, 224  
**union** 795  
 インデックスの作成 159–181  
 可変長、ソート順 613  
 検査制約、定義との競合 276  
 定義と対立するルール 224  
 テーブルへの追加 48  
**insert** 546  
 デフォルト 138–141, 547  
 取り消されたパーミッション 651  
 パーミッション 488  
 ビュー 305  
 ルール 546  
 カラム数  
   1 テーブルあたり 71, 259  
   **order by** 句内 612  
   ビュー内 308  
 カラムの最大数 71  
 カラム見出し名変更の **as** キーワード 678  
 カラム名  
   **grouping by** 528, 529  
   **union** 結果セット 795  
   エイリアス 626, 678  
   ビュー 305  
 間隔、自動チェックポイント 102  
 感嘆符 (¥!)  
   エラー・メッセージのプレースホルダ 617

## き

キー・カラム  
   **alter table** 89  
 キー値 820, 827  
**alter role** の **passwd** 43  
 キー、テーブル 271  
   「共通キー」「インデックス」参照  
 記憶領域管理プロパティ  
   **create index** 176  
   **create table** 277  
 記憶領域の断片化の減少 48  
 記憶領域の断片化、減少 48  
 基本テーブル。「テーブル」参照  
 疑問符 (??)  
   不完全な文字 632

キャンセル  
   「**rollback** コマンド」参照  
   算術エラーのあるトランザクション 715  
   重複する更新または挿入 163  
**rowcount** 737  
   トリガ 672  
   プランが調整されたクエリ 734  
 共通キー 239  
   「外部キー」「ジョイン」「プライマリ・キー」  
   参照  
 共有の統計、削除 363

## く

クエリ  
   **group by** および **having**  
     を使用する／使用しない 531  
   **union** 793–796  
   キーワード・リスト 728  
   構文チェック (**set parseonly**) 718  
   実行なしのコンパイル 718  
   実行の設定 712–772  
   トリガの起動 295  
   ビュー 308  
 クエリ処理  
   **set** オプション 712  
 クエリ・プラン  
   **set showplan on** 718  
 クエリ分析  
   **set noexec** 718  
   **set statistics io** 740  
   **set statistics time** 740  
 クライアント、文字セット変換 719  
 クラスタード・インデックス  
   **fillfactor** 161  
   「インデックス」参照  
   作成 160  
   セグメント 164, 170  
   テーブルのマイグレーション 170, 267  
 繰り返し実行。「**while** ループ」参照  
 繰り返し読み出し独立性レベル 693

## 索引

### グループ

「"public" グループ」参照

**grant** 515

**revoke** 662

テーブルのロー 527

### グループ化

同じ名前のプロシージャ 193, 409, 470

テーブルのロー 531

複数のトリガ動作 291

### グローバル変数

**@@clientexpansion** 766

**@@cursor\_rows** 766

**@@datefirst** 766

**@@lock\_timeout** 767

## け

### 計算値

生成、**compute** 111

### 結果

**order by** およびソート 609–615

カーソル結果セット 350, 477

集合演算 530

「出力」参照

結合、ソート順の規制 613–614

権限。「パーミッション」参照

### 言語、代替

構造および変換 617

システム・メッセージ 722

曜日順 756

### 現在のデータベース

変更 836

現在のプロセス。「プロセス(サーバのタスク)」参照

現在のロック、**sp\_lock** システム

・プロシージャ 556

### 検索

エラー・メッセージ・テキスト 617

### 検査制約

**insert** 546

カラム定義の矛盾 276

## こ

合計、**compute** コマンドを使用した取得 612

降順(**desc** キーワード) 609, 690

降順スキャン 614

オーバーフロー・ページ 615

デッドロック 614

降順のインデックス 55

降順のインデックス、指定 48

### 更新

**writetext** 850

**ignore\_dup\_key** 163

ダーティ・ページ 102–103

トリガの起動 301

ビューのデータ 309

プライマリ・キー 292

ロック解除されたロー 800

更新可能なカーソル 351

### 構造体

**clustered** インデックスと **nonclustered**

インデックス 160

「順序」参照

### 構文

**grant dbcc** 486

**revoke dbcc** 650

**set parseonly** 718

### 互換性、データ

**create default** 139

ルールとカラムのデータ型 223

### 固定長カラム

保管順序 613

### コピー

**model** データベース 131

データベース、**create database**

を使用 134–136

**select into** 700

**insert...select** 545

### コマンド

**alter database** 2–15

**alter encryption key** 16–24

**alter ownership** 37–42

**alter role** 43–47

**begin transaction** 100

**begin...end** 99

**break** 101



**close** 104  
**compute** 107-115  
**connect to** 116-118  
**continue** 119  
**create archive database** 120-121  
**create encryption key** 142-145  
**create existing table** 146-152  
**create function** 153-155  
**create plan** 188-189  
**create procedure** 193-209  
**create proxy\_table** 215-218  
**create role** 219-221  
**create scheme** 226-227  
**create service** 228-232  
**deallocate cursor** 342  
**declare** 344-345  
**declare cursor** 346-353  
**delete** 354-362  
**disconnect** 116-118  
**disk init** 365-372  
**disk mirror** 373-377  
**disk refit** 378  
**disk reinit** 379-383  
**disk remirror** 384-386  
**disk resize** 387-388  
**disk unmirror** 389-392  
**drop database** 393-395  
**set proxy** 765  
**checkpoint** 102-103  
**commit** 105-106  
**create function (SQLJ)** 156-158  
**create index** 159-181  
**create precomputed result set** 190-192  
**create procedure (SQLJ)** 210-214  
**create rule** 222-225  
**create table** 233-287  
**create trigger** 290-304  
**dbcc** 314-341  
**declare cursor**、スクロール可能カーソルで使用不可 348  
**delete statistics** 363-364  
**fetch**、複数ロー 480  
**rowcount** 範囲 737  
**statistics io** 740  
**statistics time** 情報 740  
**update**、スクロール可能カーソルで使用不可 807

順序の区別 510, 659  
 コマンド実行の遅延。「**waitfor** コマンド」参照  
 コマンドの順序 510, 659  
 コマンド・パーミッション  
   **grant all** 513  
   **grant** 割り当て 486-523  
     「オブジェクト・パーミッション」  
     「パーミッション」参照  
     レベル 500  
 小文字、ソート順 613  
 コンシューマ・プロセス 163  
 コンパイル  
   **exec with recompile** 471  
   時間 (**statistics time**) 740  
   実行なし (**noexec**) 718  
 コンポーネント統合サービス  
   リモート・サーバの制約 54, 59  
 コンポーネント統合サービスのコマンド  
   **connect to** 116  
   **create existing table** 146  
   **create proxy\_table** 215

## さ

サーバ  
   データベースの容量 131  
   「プロセス (サーバのタスク)」 「リモート・サーバ」参照  
 サーバの起動  
   マスタ・デバイスのディスク再ミラーリング 385  
   マスタ・デバイスのディスク・ミラーリング 376  
 サーバの停止 775  
 サーバ・プロセスの ID 番号。  
   「プロセス (サーバのタスク)」参照  
 サービス  
   削除 415  
   新規作成 228-232  
 再開、**while** ループ 119  
 再起動、サーバ  
   **dataserver** ユーティリティの使用 376, 385  
   **disk refit** 378  
   **create database** 130

## 索引

- 再帰、制限 300
- 再構築
  - text および image データ 321
  - インデックス 322
  - システム・テーブル 321, 324
  - 自動、ノンクラスタード・インデックス 170
- 再コンパイル
  - create procedure with recompile**
    - オプション 195, 202
  - execute with recompile** オプション 471
  - ストアド・プロシージャ 202
- 再作成
  - text および image データ 321
  - インデックス 322
  - テーブル 416
  - プロシージャ 206
- 再初期化、**disk reinit** 379–383
- 最初のカラムのパラメータ。 239
- サイズ
  - readtext** データ 630, 632
  - set textsize** 関数 742
  - 新しいデータベース 123
  - コンパイルされたストアド・プロシージャ 202
  - コンパイルされたストアド・プロシージャの見積もり 202
  - 再コンパイルされたストアド・プロシージャ 202
  - select** で返される *text* 742
  - writetext** で返される *image* 851
  - writetext** で返される *text* 851
  - テーブル 259
  - テーブル内のカラム 71
  - トランザクション・ログ・デバイス 135
  - 複合インデックス 161
  - ロー 71
- サイズの制限
  - 1 データベースあたりのテーブル数 259
  - 1 テーブルあたりの許容カラム数 259
  - print** コマンド 618
- 再ミラーリング。「ディスク・ミラーリング」参照
- 作業セッション、**set** オプション 712–772
- 削除
  - dbcc dbrepair** データベース 319
  - インデックス 402–403
  - 共有の統計 363
  - グループ化されたプロシージャ 193
  - サービス 415
  - 「削除」参照
  - 相互排他的な関係の役割 43
  - 損傷したデータベース 319
  - データベース 393–395
  - テーブル 416–418
  - テーブルからのローの削除 354–362, 417
  - テーブルに対する制約 48
  - デフォルト 139, 396
  - トリガ 297, 420
  - トリガのあるテーブル 297
  - 破壊されたインデックス 322
  - ビュー 421
  - プロシージャ 409–410
  - 役割からのパスワードの削除 43
  - ユーザ定義の役割 411
  - ルール 413
  - ロック解除されたロー 354
  - truncate table** 791
- 削除。「削除」を参照
- 作成
  - SQL 抽出テーブルからのビュー 312
  - SQLJ ストアド・プロシージャ 210–214
  - アーカイブ・データベース 120–121
  - 暗号化キー 142–145
  - インデックス 159–181
  - 拡張ストアド・プロシージャ 193–209
  - 仮想ハッシュ・テーブル 263
  - サービス 228–232
  - スキーマ 226–227
  - データベース 122–137
  - テーブル 233–287, 679
  - テーブル、identity カラム 277
  - デフォルト 138–141
  - トリガ 290–304, 512, 660
  - ビュー 305–313
  - ユーザ定義の役割 219
  - ルール 222–225
  - サスペンド、データベース 620

サブクエリ

**order by** 612

使用不可の **union** 796

サブグループ、合計値 111

参照整合性制約 48, 272, 434

**create table** 269

データベース間 274, 417

参照整合性、トリガ 290–304

参照、オブジェクト。 268

## し

シーケンス。「**order by** 句」「ソート順」参照

時間間隔

**reorg** 641

**waitfor** 837

経過実行時間 (**statistics time**) 740

自動チェックポイント 102

「タイミング」参照

トリガの実行 296

式

grouping by 529

**insert** 544

合計値 111

評価順序 794

識別子

**select** 698

辞書ソート順 613

システム・アクティビティ

**shutdown** 775

クエリ処理オプションの設定 712–772

システム・データベース、ダンプ 437

システム・テーブル

**dbcc checkcatalog** 316

**drop table** の影響 416

**lock table** 禁止 595

検出された割り付けエラーの修正 321, 324

再構築 321, 324

「テーブル」、「個々のテーブル名」参照

デフォルトの定義 139

トリガ 296

**drop view** 421

ルール情報 223

システムの論理名。「論理デバイス名」参照

システム標準の役割

ストアド・プロシージャ 525

取り消し 667

システム・プロシージャ

「テーブル」、個々のテーブル名

参照プロシージャ

**create procedure** 193–209

**create procedure (SQLJ)** 210–214

**create procedure (SQLJ)** コマンド、

「個々のプロシージャ名」参照

ユーザ定義の削除 409–410

システム・メッセージ

「エラー・メッセージ」「メッセージ」参照

言語設定 722

実行

Transact-SQL コマンド 469

拡張ストアド・プロシージャ 469

プロシージャ 469

ユーザ定義プロシージャ 469

実行の遅延。「**waitfor** コマンド」参照

実行、時間の指定 837

自動操作

チェックポイント 102

データ型変換 265

トリガ 290

自発的完了 318

シャープ記号(#)、テンポラリ・テー

ブル名のプレフィクス 235

集合関数

**group by** 句 527, 530

**having** 句 528, 530

スカラ集合 530

ベクトル集合、**group by** 530

集合関数を含まない式、グループ分け 528

重大度レベル、エラー、ユーザ定義メ

ッセージ 628

終了

**waitfor** コマンド 837

無条件、**return** コマンド 647

出力

**dbcc** 334

長さがゼロの文字列 619

順序

null 値 612

## 索引

- インデックス作成 170
  - エラー・メッセージの引数 617
  - カラム (固定長と可変長) 613
  - カラムとロー集合関数 112
  - カラム・リストおよびデータの挿入 544
  - グループ内の名前 539
  - 降順ソート 609, 690
  - 昇順ソート 609, 690
  - 日付要素 722
  - 評価 794
  - 変換文字列の引数 617
  - ルールのバインド解除 223
  - create procedure** 470, 473
  - 準備状態でのトランザクションのコミット 335
  - ジョイン
    - インデックス 169
    - テーブル・グループ 534
  - 障害、メディア
    - disk remirror** 385
    - 自動フェールオーバー 389
    - 「リカバリ」参照
  - 照合順。「ソート順」参照
  - 昇順のインデックス 55
  - 昇順のインデックス、指定 48
  - 昇順、**asc** キーワード 609, 690
  - 使用法
    - grant dbcc** 512
  - 情報 (サーバ)
    - text** 207
    - プロシージャの表示 197
    - 領域の使用状況 178
  - 情報メッセージ (サーバ)。「エラー・メッセージ」  
「重大度レベル」、「エラー」参照
  - 省略形
    - out** と **output** 195, 470
    - tran** は **transaction** の略 (**rollback** コマンド) 669
    - chars** および **characters** (**readtext** 631
    - exec** は **execute** 469
  - 初期化
    - disk reinit** 369, 379–383
    - ディスク領域 365–372
  - 所有権
    - コマンド・パーミッションおよびオブジェクト・  
パーミッション 500
    - ストアド・プロシージャ 208, 213, 232
    - トリガ 303
    - 「パーミッション」「**setuser** コマンド」参照
    - ビュー 313
    - ルール 224
  - 所有者。「データベース所有者」「データベース・  
オブジェクト所有者」参照
- ## す
- スカラ集合
    - group by** 530
  - スキーマ
    - 新規作成 226–227
    - パーミッション 227
  - スキャン
    - (**statistics io**) の数 740
    - カーソル 350
  - スクロール可能カーソル
    - 更新不可 348, 807
  - スコープ、カーソル 348
  - ステータス
    - ストアド・プロシージャの実行 474
  - ストアド・プロシージャ
    - parseonly** を使用しない 718
    - set** コマンド 712
    - グループ化 193, 470
    - グループの削除 409
    - 最大記憶サイズ 201
    - 削除 193, 409–410
    - 作成 193–209
    - 実行 469
    - 取り消されたパーミッション 651
    - 名前の変更 202
    - ネスト 201, 474
    - パーミッションの付与 488
    - 命名 193
    - リターン・ステータス 205–206, 469, 474, 647
  - ストアド・プロシージャの 0 リターン
    - ・ステータス 205
  - ストアド・プロシージャのトリガ。 290
  - スペース、文字
    - update** 807

スレッシュホールド  
 データベース・ダンプ 437  
 トランザクション・ログのダンプ 462

## せ

制限、仮想ハッシュ・テーブル 263  
 整合性。 314  
 制約

作成されたインデックスおよび

**max\_rows\_per\_page** 56

**create table** 268

エラー・メッセージ 270

参照整合性 272

データベース間 274, 417

テーブルの削除 48

テーブルの追加 48

テーブルの変更 48

ユニーク 271

セーブポイント

**rollback** 669

「チェックポイント処理」参照

**save transaction** 674

セキュリティ

コマンド・パーミッションおよびオブジェクト・パーミッション 500

「パーミッション」参照

ビュー 308

セグメント

**dbcc checktable** レポート 317

**dbcc indexalloc** レポート 320

新しいデバイスへのマップ 13

インデックスの作成 57, 164, 170, 242, 247, 682

オブジェクトの配置 164

クラスタード・インデックス 170

指定された数 131

「データベース・デバイス」、「ログ・セグメント」、「領域の割り付け」参照

テーブルとインデックスの分離 169, 267

テーブルのロック・スキームの変更 92

名前 57, 63, 242, 247, 682

設定

ダンプ 422

設定パラメータ 635  
 説明

**grant dbcc** 486

**revoke dbcc** 650

宣言

パラメータ 194

ローカル変数 344

選択

ロック解除されたロー 706

## そ

関連名とテーブル名 685

総計

**compute** 111

**order by** 612

相互排他的な役割 43

ソート・オペレーション (**order by**)、  
 ソート・プラン 719

ソート順

インデックスに指定、**create table** 259

インデックスの指定、**alter table** 80

インデックスの指定、**create index** 172

**group by** および **having** 539

**order by** 612

降順 609

「順序」参照

昇順 609

選択と影響 612

名前のグループ 539

変更後のインデックスの再構築 322

ソート順の結合の規制 613-614

即時停止 775

属性

リモート・テーブル 149

速度(サーバ)

**create database for load** 134

**sorted\_data** を使用した **create index** 164

**execute** 474

**dump transaction** と **dump database** 462

**truncate table** と **delete** 791

**writetext** と **dbwritetext** および **dbmoretext** 852

損傷したデータベースの修復 319

損傷したデータベース、削除と修復 319

## た

ダーティ・ページ

更新 102-103

タイミング

「時間間隔」参照

自動チェックポイント 102

タイムスタンプ、トランザクション・

ログ・ダンプの順序 571

探索条件

**select** 689

**group by** クエリと **having** クエリ 528, 533

**where** 句 839-847

ダンプ・ストライピング

データベース・ダンプ 428

トランザクション・ダンプ 453

ダンプ・デバイス

ダンプ、データベース 426

ダンプ、トランザクション・ログ 451

「データベース・デバイス」

「ログ・デバイス」参照

必要な数 572

命名 426, 451, 462-463

ダンプ、データベース

Backup Server 438

Backup Server、リモート 426

*master* データベース 437

新しいデータベース 436

一連のダンプ 441, 465

上書き 429, 442-443

システム・データベース 437

使用するコマンド 458

初期化/追加 430

スケジューリング 436-437

スレッシュホールド 437

ダンプ・ストライピング 428

ダンプ・デバイス 426, 437

テープのマウント解除 429

テープの巻き戻し 429

テープ密度 426

テープ容量 427

動的 436

ネットワーク間 436

ファイル名 430, 439

ブロック・サイズ 427

ボリュームの変更 441

ボリュームへの接続 442-443

ボリューム名 428, 441

メッセージ送信先 430

有効期限 429

リモート 439

ロード 135, 557-577

ダンプ、トランザクション・ログ

Backup Server、リモート 463

使用するコマンド 458

スケジューリング 462

スレッシュホールド 462

ダンプ・ストライピング 453

ダンプの追加 454

テープの初期化 454

テープのマウント解除 453

テープの巻き戻し 454

テープ容量 452

ネットワーク間 462

パーミッションの問題 458

ファイル名 454, 463-464

ボリュームの初期化 465-466

ボリュームへの接続 465-466

ボリューム名 453, 464

メッセージ送信先 455

有効期限 454

リモート 463, 464

ロード 578-592

ログ領域不足のオプション 461

断片化、減少 48

## ち

チェック、一貫性。 314

チェックポイント・プロセス 102-103

「リカバリ」「セーブポイント」参照

遅延実行 (**waitfor**) 837

重複するロー

インデックス 159, 163

**union** 793

## つ

## 追加

- オブジェクト、*tempdb* 267
- sysusermessages* へのメッセージの追加 619
- 相互排他的なユーザ定義の役割 43
- データベースに領域を追加する 2-15
- テーブルに対する制約 48
- テーブルへのカラムの追加 48
- テーブルまたはビューへのローの追加 544-553
- ミラーリング・デバイス 373-377
- 役割 220
- 役割へのパスワードの追加 43
- ユーザ定義の役割 220

## て

## 停止

- プロシージャ。「**return** コマンド」参照
- 定数、代わりにパラメータを返す 473
- ディスク・コントロール 367, 380
- ディスク・デバイス追加 365-372
  - ミラーリング 373-377
  - ミラーリングの解除 389-392
- ディスク・ミラーリング 373-377
  - waitfor mirrorexist** 837
  - load database 573
  - 再起動 384-386
  - データベース・ダンプ 443
    - トランザクション・ログのダンプ 466
    - トランザクション・ログのロード 590
  - ミラーリングの解除 389-392
- ディスク・ミラーリングの解除 389-392
- データオンリー・ロック・テーブルカラムの追加、削除、変更の制限 89
- データ型
  - group by** 句と **having** 句内で無効 530
  - union** 演算での比較 795
  - カーソル結果セット 478
  - カラムとデフォルトの互換性 139
  - ローカル変数 344

## データ型変換

- カラム定義 265

## データ辞書。「システム・テーブル」参照

## データ整合性 547

- 「参照整合性制約」参照

## データの整合性

- 制約 268
- メソッド 269

## データベース

- checkalloc** オプション (**dbcc**) 316
- checkdb** オプション (**dbcc**) 316
- checkstorage** オプション (**dbcc**) 317, 318
- use** コマンド 836

## オフライン、変更 10

## サーバの数 131

## サイズの増加 2, 37

## 削除 394

## 作成 122

## 選択 836

## 損傷したデータベースの削除と修復 319

## ダンプ 424-447

## 中断 620

## データベース・ダンプのアップグレード 571, 588

## デフォルト・サイズ 130

## 独立したログ・セグメントを持つデータベースの作成 460

## バックアップ 424-447

## リカバリ 557-577

## ロード 557-577

データベース一貫性チェック。「**dbcc** コマンド」参照

## データベース・オブジェクト

*select\_list* 678-679追加先、*tempdb* 266参照、**create procedure** 202

## プロシージャ作成時のパーミッション 208, 213

## プロシージャ実行時のパーミッション 208, 213

## データベース・オブジェクト所有者

## 「データベース所有者」「所有権」参照

## データベース・サイズの変更 387-388

## データベース作成要求の衝突 131

## 索引

- データベース所有者
    - 「データベース・オブジェクト所有者」
    - 「パーミッション」参照
  - データベース所有者によるパーミッションの付与 487, 651
  - setuser** 500
  - データベース・デバイス
    - alter database** 3
    - 新しいデータベース 123
    - トランザクション・ログの分離 375, 385
  - データベースのダンプ。「ダンプ」、「データベース」
  - 「ダンプ・デバイス」参照
  - データベースのロード 574
  - データ変更
    - update** 800
    - writetext** での *text* および *image* 850
  - テーブル・ラベル
    - load database** の **listonly** オプション 562
    - load transaction** の **listonly** 582
  - テーブル
    - dbcc checkdb** 316
    - create schema** による作成 226–227
    - external 215
    - from** 句で使用可能 685
    - identity カラムあり作成 277
    - Transact-SQL 拡張機能の効果とクエリ 532
    - update statistics** 834–835
    - インデックス・ロケーション 402, 820, 827
    - オブジェクト・アロケーション・マップ 324
    - 仮想ハッシュ、制限 263
    - クラスタード・インデックスへのマイグレーション 170, 267
    - 削除 416–418
    - 新規作成 233–287, 679
    - シングル・グループ 531
    - データなしで 702
    - 取り消されたパーミッション 651
    - パーミッション 488
    - 複製の作成 702
    - プロキシ 146
    - 分割 48, 62, 84–86
    - 分割の解除 48, 62
    - 分類、**group by** 句と **having** 句を使用 527–540
    - 変更 48–96
  - テーブル関連名のエイリアス 685
  - テーブルのグループへの分類。
    - 「**group by** 句」参照
  - テーブルの分割 48
  - テーブルのページ
    - dbcc tablealloc** 323
  - デッドロック
    - 降順スキャン 614
  - デバイス
    - 採番 366, 379
    - セカンダリ 375
    - ディスク・ミラーリング 373–377
    - マスタ 9
  - デバイスの障害
    - トランザクション・ログのダンプ 455, 459
  - デバイスの初期化。「初期化」参照
  - デバイスのミラーリングの解除。
    - 「ディスク・ミラーリング」参照
  - デバイス・フラグメント
    - 数 131
  - デバイス名
    - ダンプ・デバイス 426, 451
    - ディスクの再ミラーリング 384
    - ディスク・ミラーリング 373
    - ディスク・ミラーリングの解除 389
    - 物理的、**disk reinit** 379
    - リモート・ダンプ・デバイス 572
  - デバッグのためのツール
    - set showplan on** 718
    - set sort\_resources on** 719
    - set statistics io on** 740
  - デフォルト 547
  - IDENTITY カラム 87
  - カラム 51
  - 削除 396
  - 作成 138–141
  - 定義および **create default** 138–141
  - ルール 140, 224
- デフォルト設定
    - set** コマンド・オプション 755
    - ストアド・プロシージャのパラメータ 195
    - 曜日順 756



- デフォルト値
  - データ型、長さを指定しない場合 194
- テンポラリ・テーブル
  - create procedure** 207
  - create table** 235, 266
  - lock table** 禁止 595
  - 識別子プレフィクス (#) 235
  - 命名 266
- と**
- ドイツ語の print メッセージ・サンプル 617
- 同義語
  - out** と **output** 195, 470
  - tran**、**transaction**、**work**、**commit**  
コマンド 105
  - tran**、**transaction**、**work**、**rollback**  
コマンド 669
  - chars** および **characters (readtext)** 631
- 統計
  - update all statistics** 812–816
  - インデックスなしのカラムの生成 820, 827
  - シミュレート、ロード 364, 766
  - delete statistics** 363
- 動的ダンプ 436, 462
- 独立性レベル
  - readpast** オプション 708
  - 繰り返し読み出し 693
- ドメインまたは整合性ルールの違反 547
- ドメイン・ルール 546
  - create rule** コマンド 222
  - 違反 547
- トランケーション
  - insert** 547
  - set string\_rtruncation** 741
  - データ型、長さ指定なし 194
  - デフォルト値 139
  - トランザクション・ログ 448
  - 複数スペースのシングル・スペースへのトランケーション 807
  - ログ、混在デバイスでの禁止 123
- トランザクション
  - begin** 100
  - 終了、**commit** 105
  - dump transaction** コマンド 448–468
  - fetch** 478
  - save transaction** 673–674
  - キャンセル。「**rollback** コマンド」参照
  - 構成部分ではないパラメータ 474
  - 指定内での **update** の反復 806
  - 準備 616
  - 準備状態でのコミット 335
  - 独立性レベル 744
  - 「バッチ処理」「**rollback** コマンド」  
「ユーザ定義トランザクション」参照
  - 連鎖 106
- トランザクション独立性レベル
  - readpast** オプション 708
- トランザクション・ログ
  - syslogs** テーブル **trunc log on chkpt** 458
  - dump database** 424
  - 「**dump transaction** コマンド」  
「**syslogs** テーブル」参照
  - master** データベース 437, 460
  - writetext with log** 850
  - 削除されたロー 358
  - 消去 437
  - ダンプ 449
  - 独立したセグメントへの配置 460
  - バックアップ 424
  - 非アクティブな領域 451
  - 別のデバイス 370, 375, 385, 458
  - 領域の拡張 13
  - 領域、モニタ 462
  - ロード 578–592
- トリガ
  - @@nestlevel** 300
  - @@rowcount** 298
  - delete** 359
  - image** カラム 296
  - insert** 547
  - parseonly** を使用しない 718
  - rollback** 297, 298, 670
  - set** コマンド 712
  - text** カラム 296
  - truncate table** コマンド 792

**update** 804  
 再帰 301  
 削除 420  
 作成 290–304, 512, 660  
 時間間隔 296  
 自己再帰 301  
 自己再帰の有効化 301  
 システム・テーブル 296  
 ストアド・プロシージャ 300  
 名前の変更 297  
 ネスト 300–301  
 ネスト、**rollback trigger** 672  
 ロールバック 672  
 トリガ・テーブル 297  
 トリガの上書き 296, 420  
 取り消し  
   **create trigger** パーミッション 304, 512, 660  
   **with override** 411

## な

長さがゼロの文字列出力 619  
 名前  
   **setuser** 773  
   パラメータ、**create procedure** 194  
   カラム、ビュー内 305  
   グループのソート 539  
   セグメント 57, 63, 164, 242, 247, 682  
   テーブルのエイリアス 685  
   ビュー 421  
 名前の変更  
   ストアド・プロシージャ 202  
   トリガ 297  
   ビュー 308

## に

日本語文字セット  
   **print** メッセージ・サンプル 617, 626  
**dbcc complete\_xact 1pc** 335  
**execute** 469

## ね

ネスト  
   **begin...end** ブロック 99  
   **if...else** 条件 543  
   **while** ループ 849  
   **while** ループ、**break** 101  
   ストアド・プロシージャ 201, 474  
   トリガ 300  
   トリガのレベル 300  
   レベル 204  
 ネスト **select** 文。「**select** コマンド」  
   「サブクエリ」参照  
**select** 693  
 ノンクラスタード・インデックス 160

## は

パーセント記号 (%)  
   エラー・メッセージ内のリテラル 619  
   エラー・メッセージのプレースホルダ 617  
 パーミッション  
   **grant** 486–523  
   **create schema** による作成 226–227  
   **revoke** コマンド 650–668  
   データベース所有者による割り当て 487, 651  
   トリガの作成 304, 512, 660  
   割り当て 487, 651  
   **setuser** 773  
 パーミッションの階層。「パーミッション」参照  
 バイト  
   1 ローあたり 71  
   「サイズ」参照  
 バインド  
   デフォルト 138  
   バインド解除 396  
   ルール 224  
 バインド解除  
   デフォルト 139, 396  
   ルール 413  
 破壊されたインデックス。  
   「**reindex** オプション」、「**dbcc**」参照  
 パススルー・モード  
   **connect to** コマンド 117

- パス名
    - DLL および拡張ストアド・
      - プロシージャ 196
    - ミラーリング・デバイス 373
    - リモート・ダンプ・デバイス 572
  - パスワード
    - 役割 43
    - 役割からの削除 43
    - 役割への追加 43
    - ユーザ定義の役割 219, 737
    - ユーザ定義の役割からの削除 45
    - ユーザ定義の役割での変更 46
    - ユーザ定義の役割への追加 45
  - バックアップ
    - master データベース 11
    - 「ダンプ」、「データベース」「ダンプ」、「トランザクション・ログ」「ロード」、「データベース」「ロード」、「トランザクション・ログ」参照
    - インクリメンタル。「ダンプ」、「トランザクション・ログ」参照
    - ディスクの再ミラーリング 384
    - ディスク・ミラーリング 374, 390
  - バッチ処理
    - create default** 139
    - execute** 469, 473
    - set** オプション 754
    - リターン・ステータス 647–649
  - パフォーマンス
    - dump database** 実行中の **writetext** 852
    - select into** 702
    - showplan** と診断 718
    - sort\_resources** と診断 719
    - トリガ 296
  - パラメータ
    - grant dbcc** 491
    - revoke dbcc** 654
  - パラメータ、プロシージャ
    - execute** 470
    - 指定する方法 470, 473
    - データ型 194
    - デフォルト 195
    - トランザクションの一部でない 474
    - 命名 194
  - ハロウィーン問題 352
  - 番号
    - statistics io** 740
    - select** リスト 690
    - エラー戻り値(サーバ) 648
    - 同じ名前のグループのプ
      - ロシージャ 193, 409, 470
    - 仮想デバイス 366, 379
    - プレースホルダ(%nn!) 617
    - 曜日名 721
- ## ひ
- 非アクティブなトランザクション・ログ領域 451
  - 比較演算子
    - where** 句 839
  - 引数
    - where** 句、使用できる数 847
    - 番号付けされたプレースホルダ、**print** コマンド 617, 618
    - ユーザ定義のエラー・メッセージ 626
    - 「論理式」参照
  - 引数の変換 617
  - ヒストグラム
    - ステップの指定、**create index** 175
    - update statistics** 812, 818, 824
  - 日付
    - 表示フォーマット 721
    - 表示フォーマット、**waitfor** コマンド 838
  - 日付要素
    - 順序 722
  - ビュー
    - with check option** 310, 550–551
    - check option** 808–809
    - create schema** による作成 226–227
    - from** 句で使用可能 685
    - readtext** 632
    - update** 310, 808–809
    - 基本となるテーブルの変更 309
    - 更新の制限 809
    - 削除 421
    - 作成 305–313

## 索引

「データベース・オブジェクト」、「複数のテーブルから構成されるビュー」参照  
取り消されたパーミッション 651  
名前の変更 308  
パーミッション 488  
ビューを介したデータの挿入 550  
評価順序 794  
表示  
**create procedure** 文のテキスト 207  
コマンドの影響を受けるローの設定 718  
情報のプロシージャ 197

## ふ

ファイル  
「テーブル」「トランザクション・ログ」参照  
ミラーリング・デバイス 373  
ファイル名  
DLL 196  
データベースのダンプ 439  
トランザクション・ログのダンプ 454, 581  
**listonly** 562, 582  
ブール (論理) 式、**select** 文 542  
フォーマット文字列  
**print** 617  
**raiserror** 625  
ユーザ定義のエラー・メッセージ 625  
不完全な文字の ?? (疑問符) 632  
不完全な文字、読み込み 632  
複合インデックス 160, 178  
複数カラムのインデックス。  
「複合インデックス」参照  
複数のテーブルから構成されるビュー 809  
**delete** 309, 357  
「ビュー」参照  
複数のトリガ動作 291  
複数ローのフェッチ 480  
複製  
新しいデータベースの領域 135  
データを持たないテーブル 702  
含まれるグループ、**group by** クエリ 533  
2つの一重引用符の使用  
文字列 846  
物理読み込み (**statistics io**) 740  
付与  
**create trigger** パーミッション 304, 512, 660  
プライマリ・キー 271  
更新 292  
プラットフォーム間のダンプとロード、疑わしい  
パーティションの処理 570  
プラン  
作成、**create plan** 188  
**create procedure** 195  
**set showplan on** 718  
**set sort\_resources on** 719  
ブランク、文字データ型 547, 807  
ブレースホルダ  
**print** メッセージ 617  
フロー制御言語  
**begin...end** 99  
**create procedure** 196  
プロキシ・テーブル  
リモート・テーブルへのマッピング、  
**create table** 259  
**create proxy\_table** 215  
リモート・テーブルへのマッピング 146  
プロシージャ・グループ 409, 470  
プロシージャの中止。「**return** コマンド」参照  
プロシージャ・プラン、**create procedure** 195  
プロシージャ。193  
プロセス (サーバのタスク)  
影響、**waitfor errorexit** 838  
ID 番号 554  
**sp\_who** レポート 554  
強制終了 554-556  
「サーバ」参照  
プロセスのブロック 555  
プロセスのロールバック  
**checkpoint** 102  
パラメータ値 474  
プロセスの論理名。「論理デバイス名」  
参照  
文  
**create procedure** 196  
**create trigger** 291  
分割の解除  
テーブル 48  
分岐 485  
分散トランザクション処理 (DTP) 319

文の順序および番号付けされた  
 プレースホルダ 617  
 文法構造、番号付けされたプレースホルダ 617  
 分離、物理的  
 テーブルとインデックスの  
 セグメント 169, 267  
 トランザクション・ログ・デバイス 375, 385

## へ

ペア、ミラーリング済み 389  
 並列度 (degree of parallelism)  
**select** および **parallel** 686  
 ページ数  
**statistics io** 740  
 エクステント内 169, 259  
 書き込まれたページ数 (**statistics io**) 740  
 ページのチェーン  
 パーティション 62, 84  
 分割の解除 62  
 ページ分割 57, 162, 241  
 ページ、OAM (オブジェクト・アロケ  
 ーション・マップ)  
**dbcc indexalloc** レポート 321  
 テーブルに関する **dbcc** レポート 324  
 ページ、オーバフロー  
 降順スキャン 615  
 ページ、空白に対する満杯率 48  
 ページ、データ  
 エクステントと **dbcc tablealloc** 324  
 エクステントのレポート、  
**dbcc indexalloc** 321  
**statistics io** 740  
 「インデックス・ページ」、  
 「テーブル・ページ」も参照  
 エクステント 172, 259  
 チェーン 62, 84-86  
 マルチバイト文字 320  
 ベクトル集合  
**group by** 530  
 変換  
**like** キーワードを使用した日付 840  
**null** 値と自動的な変換 265

**where** 句とデータ型 847  
 カラム 265  
 変更  
 「更新」参照  
 所有権 37-42  
 データベース 2, 37  
 データベース・サイズ 2-15  
 テーブル 48-96  
 テーブルに対する制約 48  
 ビュー定義 309  
 役割 43  
 ユーザ定義の役割 43  
 ユーザ定義の役割のパスワード 46  
 ロック・スキーム 48, 61  
 変更のカスケード (トリガ) 295  
 変更の取り消し。「**rollback** コマンド」参照  
 変更、キャンセル。「**rollback** コマンド」参照  
 変数  
**print** メッセージ 618  
**select** リストの一部として割り当て 679  
**update** 文 804  
 戻り値 473  
 ローカル 344-345

## ほ

ポインタ  
*text* または *image* カラム 630  
 ポインタ、デバイス。「セグメント」参照  
 保護システム  
 コマンド・パーミッションおよびオブジェク  
 ト・パーミッション 500  
 スタアド・プロシージャ 208, 213  
 役割、グループ、ユーザの階層 513, 520  
 ユーザ定義の役割 220  
 ボリューム名、データベースのダンプ 441

## ま

マーク、ユーザ定義。「プレースホルダ」「セーブ  
 ポイント」参照  
 マイグレーション

## 索引

システム・ログの他のデバイスへの移行 370  
テーブルをクラスタード・  
インデックスに 170, 267  
マスタ・デバイス 9  
マッピング  
system と default セグメント 13  
マルチバイト文字セット  
**fix\_text** アップグレード 319, 334  
**readtext** 632  
**readtext using characters** 632  
**writetext** 852  
変更 319

## み

見出し、カラム 528  
ビュー内 305  
ミラーリングの無効化。「ディスク・ミラーリング」  
参照

## め

命名  
インデックス 160  
カーソル 348  
ストアド・プロシージャ 202  
データベース・デバイス 365  
テーブル 235  
テンポラリ・テーブル 266  
トリガ 290  
ビュー 305  
ビュー内のカラム 305  
ファイル 365  
メッセージ  
**revoke** 660  
画面 617-619  
言語設定 722  
トリガ 296, 420  
ユーザ定義の出力 617-619  
メモリ  
**deallocate cursor** 342  
「領域」参照

## も

文字  
**char\_convert** 719  
“0x”  
222  
文字セット  
**set char\_convert** 719  
クライアント/サーバ間の変換 719  
変更後の **fix\_text** アップグレード 319  
マルチバイト、変更 319  
文字セットのバイナリ・ソート順  
**order by** 613  
文字列  
**print** メッセージ 617  
空 547  
トランケーション 547, 741  
トランケート 547, 807

## や

役割  
ストアド・プロシージャ  
・パーミッション 525  
相互排他的 43  
パーミッション 513, 520  
パスワードの削除 43  
パスワードの追加 43  
作成(ユーザ定義) 219  
**set role** 737  
役割の競合 46  
役割、システム  
取り消し 667  
役割、ユーザ定義  
制限事項 220  
取り消し 667  
有効/無効の切り替え 737

## ゆ

ユーザ  
**guest** パーミッション 516  
システム・プロシージャのパ  
ーミッション 510

同一化 (**setuser**) 500  
 ユーザ ID。「エイリアス」「ログイン」「ユーザ」参照  
 ユーザ・エラー。「エラー」「重大度レベル」参照  
 ユーザ・グループ。「グループ」「"public" グループ」参照  
 ユーザ定義 SQLJ プロシージャ作成 210-214  
 ユーザ定義トランザクション  
   **begin transaction** 100  
   終了、**commit** 105  
   「トランザクション」参照  
 ユーザ定義の役割  
   競合 46  
   作成 219  
   システム・プロシージャ 525  
   取り消し 667  
   パスワードの追加 43  
   有効/無効の切り替え 737  
 ユーザ定義プロシージャ  
   作成 193-209  
   実行 469  
 ユーザ定義メッセージの出力 617-619  
 ユーザ定義のラベルへの無条件分岐 485  
 ユーザとの同一化。「**setuser** コマンド」参照  
 ユーザ・パーミッション。「データベース所有者」「パーミッション」参照  
 優先度  
   順序を区別するコマンド 510, 659  
   ユーザ定義の戻り値 649  
   ルールのバインド 224  
 優先度を付けた順位、大文字のソート順 613

## よ

曜日の日付値  
 名前と番号 721  
 読み取り専用カーソル 351  
 予約リターン・ステータス値 648

## ら

ラベル  
   **goto label** 485  
   ダンプ・ボリューム 441, 572, 589

## り

リカバリ  
   時間および **checkpoint** 102  
   **dump transaction** 462  
   トランザクション・ログの指定時点まで 586  
 リカバリなし 574  
 リカバリなしのデータベースのロード 574  
 リスティング  
   ユーザ・グループのメンバ 516  
 リスト  
   エラー戻り値 648  
   ソート順の選択と影響 613  
   予約リターン・ステータス値 648  
 リストア  
   **load database**  
     を使用したデータベース 557-577  
   損傷した *master* でデータベース 378  
   損傷した *master* データベース 378, 379  
   「リカバリ」参照  
 リスト作成  
   既存のデフォルト 396  
 リターン・ステータス  
   ストアド・プロシージャ 469, 647  
 リターン・パラメータ  
   **output** キーワード 195, 470  
 リモート・サーバ 699  
   制約 54, 59  
 リモート・プロシージャ・コール 699  
   **execute** 474  
   **rollback** 670  
 リモート・プロシージャ、定義 150  
 領域  
   クラスタード・インデックスおよび  
     **max\_rows\_per\_page** 163  
   **dbcc checktable** でレポートされた空き領域  
     317  
   **max\_rows\_per\_page** 162, 241  
   新しいデータベース 123

## 索引

インデックスのエクステント 321  
インデックス・ページ 56, 161–162, 240  
エクステント 169, 259  
クラスタード・インデックス 55, 162, 170, 240  
再コンパイルされたストアド・プロシージャ  
202  
「サイズ」、「領域の割り付け」参照  
ストアド・プロシージャ 201  
データベース記憶領域 55, 162, 170, 240  
データベースへの追加 2–15  
**alter table...lock** 92  
**reorg rebuild** 644  
非アクティブなログの検索 451  
不足 451  
ログ・セグメントでの使用 317, 451  
領域の再利用 (space reclamation)  
**reorg reclaim\_space** for 640  
領域の割り付け  
**dbcc** コマンド、検査 316  
テーブル 259, 316  
ページ 323  
ログ・デバイス 135

## る

### ループ

**while** 101, 848  
**break** 101  
**continue** 119  
**goto** label 485  
トリガの無限連鎖 300

### ルール

**insert** 546  
カラム定義の矛盾 224  
新規作成 222–225  
デフォルトの違反 140  
バインド 224  
ユーザが作成したルールの命名 222  
ユーザ定義の削除 413

## れ

### 例

**revoke dbcc** 655

### レベル

@**@nestlevel** 204  
トリガのネスト 300  
ネストされたプロシージャ 204, 474  
パーミッションの割り当て 500

### レポート

**sp\_who** 554

**dbcc** 323

### 連鎖トランザクション・モード

**commit** 106  
**delete** 358  
**fetch** 478  
**insert** 548  
**open** 608  
**update** 805

## ろ

### ローカル変数

**declare** (名前とデータ型) 344  
**raiserror** 626  
画面メッセージ 618  
ユーザ定義のエラー・メッセージ 626

### ローカル変数の定義 344–345

### ロー・サイズ 71

### ロー集合演算子

**compute** 107

### ロード、データベース 557–577

**Backup Server** 572

新しいデータベース 135

サポートされないプラットフォームの組み合わせ 568

使用上の制限 571, 588

ディスク・ミラーリング 573

テープのマウント解除 561

テープの巻き戻し 561

ネットワーク間 572

必要なサイズ 569

ファイル名、リスト 562

ブロック・サイズ 560



- ヘッダ、リスト 563
- ボリューム名 561
- メッセージ送信先 563, 589
- リモート 572
- ロード・ストライピング 561
- ロード中の更新の禁止 569
- ロード、トランザクション・ログ 578–592
  - until\_time** 583
  - ダンプ・デバイス 580
  - ディスク・ミラーリング 590
  - テープのマウント解除 582
  - テープの巻き戻し 582
  - ファイル名、リスト 582
  - ヘッダ、リスト 583
  - ポイントインタイム・リカバリ 583
  - ボリューム名 581
  - メッセージ送信先 583
  - ロード・ストライピング 580
- ローの最大サイズ 71
- ローの長さ 71
- ロー、テーブル
  - insert** 546
  - create index** および重複 163
  - rowcount** 設定 737
  - 「**select** コマンド」参照
  - update** 800
  - グループ化 527
  - グループ分けする方法 530
  - コマンドの影響を受けるものの表示 718
  - 集合関数の適用先 530
  - スカラ集合関数の適用先 530
  - 比較順序 613
  - ロック解除されたものの選択 706
  - ロック解除されたものを削除 354
  - ロック解除されたローの更新 800
  - truncate table** 791
- ロギング
  - select into** 700
  - text* または *image* データ 850
  - writetext** コマンド 850
  - トリガおよびロギングされないオペレーション 297
- ログイン
  - char\_convert** 設定 719
  - 無効化 776
  - 「リモート・ログイン」「ユーザ」参照
- ログイン・トリガ
  - set** オプション 770
- ログ・セグメント
  - dbcc checktable** レポート 317
  - 専用のデバイス上にないログ・セグメント 338
- ログ・デバイス
  - 消去 437
  - 「トランザクション・ログ」参照
  - 領域の割り付け 135, 334
- ログ。「セグメント」「トランザクション・ログ」参照
- ロック
  - lock table** コマンドを使用したテーブル 593
  - 読み込み用テキスト 630
  - ロックされたローのスキップの選択 706, 800
  - ロックされたローをスキップして削除 354
  - ロック解除されたローの選択 706
  - ロック・スキーム
    - 指定、**select into** 683
    - alter table** によるへんこう 48
    - create table** 277
    - 変更 48, 61
- 論理 (概念) テーブル 295, 297
- 論理式
  - if...else** 541
  - 構文 101
- 論理デバイスおよびアーカイブ・データベースへのアクセス 574
- 論理デバイス名
  - 新しいデータベース 123
  - ディスクの再ミラーリング 384
  - ディスク・ミラーリング 373
  - ディスク・ミラーリングの解除 389
- 論理読み込み (**statistics io**) 740

## わ

- ワーク・テーブルを使用する
  - update** と **delete** 357, 804

