SYBASE®
An SAP Company

参考手册：命令

Adaptive Server® Enterprise

15.7 ESD #2

文档 ID：DC37418-01-1572-01

最后修订日期：2012 年 7 月

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

目录

第 1 章

命令

本卷介绍用来构造 Transact-SQL 语句的命令、子句以及其它元素。

对命令的权限检查因您的细化权限设置而异。有关各命令的详细信息，请参见 “权限”部分。有关细化权限的详细信息，请参见《安全性管理指南》中的 “使用细化权限”。

alter database

说明

增加分配给数据库以及存档数据库的修改页部分的空间量。改变一个或多个数据库范围的属性，如 DML 日志记录级别、压缩的缺省值、行内大对象 (LOB) 存储等。

语法

```
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]
```

参数

database_name

是数据库的名称。数据库名称可以是文字、变量或存储过程参数。

on

指示数据库扩展的大小和位置。如果您的日志和数据在不同的设备段上，请为数据设备使用 on，并为日志设备使用 log on。

default

指示 alter database 可将数据库扩展放到任何缺省数据库设备上（如《参考手册：过程》的“系统过程”中的 sp_helpdevice 存储过程所示）。若要指定数据库扩展的大小而不指定其确切位置，请使用：

```
on default = size
```

若要将数据库设备的状态更改为缺省状态，请使用 sp_diskdefault。

database_device

是用于放置数据库扩展的数据库设备的名称。一个数据库可以存储于多个数据库设备上，每个设备都可有不同的空间量。若要在 Adaptive Server® 中添加数据库设备，请使用 disk init。

size

是分配给数据库扩展的空间量。以下是示例单位指示符，交替使用了大写、小写、单引号和双引号：“k”或“K”（千字节）、“m”或“M”（兆字节）、“g”或“G”（千兆字节）以及“t”或“T”（兆兆字节）。 Sybase® 建议始终包括单位指示符。如果不包括单位指示符，则引号是可选的。不过，如果包括单位指示符，则必须使用引号。如果不提供单位指示符，则假定提供的值的单位为兆字节。

如果不指定值，则 alter database 会将数据库扩展 1MB 或 4 个分配单元 （选择两者中的较大者）。最小量为：

服务器的逻辑页大小	数据库扩展大小
2K	1MB
4K	1MB
8K	2MB
16K	4MB

log on

指示您希望为数据库的事务日志指定额外空间。log on 子句使用与 on 子句相同的缺省值。

durability

确定数据库的持久性级别，取以下值之一：

- full – 将所有事务都写入磁盘。如果创建数据库时不指定持久性级别，此值将为缺省值，以确保从服务器故障中完全恢复。除 tempdb 之外的所有系统数据库都使用此持久性级别 （用于磁盘驻留式数据库的传统持久性级别）。 tempdb 使用的持久性级别为 no_recovery。
- no_recovery – 只有服务器处于运行状态时，事务才是持久性的。如果服务器出现故障或正常关闭，那么所有持久性都将丢失。对于持久性设置为 no_recovery 的磁盘驻留式数据库， Adaptive Server 会在运行时以不可控方式定期向磁盘设备中写入数据。用 no_recovery 创建的数据库以任何方式 （正常、非正常或服务器故障及重新启动）关闭后均无法恢复，但会基于 model 或模板数据库 （如果已定义）重新创建该数据库。
- at_shutdown – 在服务器运行时以及正常关闭后，事务是持久性的。如果服务器出现故障，所有持久性都将丢失。

[no] async_init

启用或禁用异步数据库初始化。

compression

指示 alter database 对此数据库中的新建表应用的压缩级别。

- none – 不压缩数据。
- row – 压缩单个行中的一个或多个数据项。只有在压缩形式比未压缩形式更节省空间时， Adaptive Server 才会用 row 压缩形式存储数据。
- page – 当页面已满时，则会使用页级压缩来压缩现有的采用行压缩形式的数据行，从而创建页级字典、索引和字符编码条目。

  Adaptive Server 只有已经在行级压缩数据后才会在页级压缩数据，因此，将压缩设置为 page 意味着 page 和 row 压缩。

lob_compression = off | *compression_level*

确定新创建的表的压缩级别。选择 off 意味着表不使用 LOB 压缩。

compression_level

表压缩级别：

- 0 – 不压缩 lob 列。
- 1 到 9 – Adaptive Server 使用 ZLib 压缩。通常，压缩级别数字越大， Adaptive Server 压缩 LOB 数据的程度就更大，压缩和非压缩数据之间的比率就越大 （也就是说，压缩数据与非压缩数据大小相比，节省的空间量就越大 （以字节为单位））。

  但是，压缩量取决于 LOB 内容，压缩级别越高，进程的 CPU 占用率就越高。也就是说，级别 9 提供的压缩率最高，但 CPU 使用率也最高。
- 100 – Adaptive Server 使用 FastLZ 压缩。这是 CPU 使用率最低的压缩率；通常用于较少的数据量。
- 101 – Adaptive Server 使用 FastLZ 压缩。与值为 100 时相比，值为 101 时的 CPU 占用率略高，但压缩率也略高。

inrow_lob_length = *value*

指定行内 text 或 image LOB 列的字节数，以及行内 Unitext LOB 列的字符数。

log off *database_device*

从指定的数据库设备中删除不想要的数据库日志部分。使用 log off 可减少分配给数据库日志以及存档数据库已修改页部分的空间量。

不能将 log off 与任何其它 alter database 参数（包括 log on、 for load 或 with override）一起使用。

= *size*

指定此命令应影响的设备的结尾处的空间量。为此，设备结尾处是此数据库在该设备上使用的编号最大的逻辑页。此命令指定物理存储，从数据库中删除指定设备上的每个日志页：

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

指定的大小值会进行舍入，以恰好符合分配单位数。在使用 16KB 逻辑页大小的安装中，您将会删除 52MB，而非 50，因为该服务器上的每个分配单位都是 4MB。只有在数据库的日志段使用的空间量比该设备上的空间少时， Adaptive Server 才会删除比指定空间量少的空间。

from *logical_page_number*

标识此命令影响的第一个页码。 Adaptive Server 会自动调整该页码以指示分配页的页 ID。缺省的 from 位置是设备上的最小编号逻辑页。

to *logical_page_number*

标识此命令影响的最后一个页码。只能删除整个分配单位 （256 页的倍数）因此， Adaptive Server 会自动调整页码，使其达到分配单位的最后一页。如果 *logical_page_number* 恰好是分配页的页码（即，可被 256 整除），则该分配单位不会受影响。例如，to 512 影响第 512 页以前的页 （但不包括第 512 页）。

缺省的 to 位置是设备上的最大编号逻辑页。

dml_logging {minimal | default}

指示 insert、 update 和 delete 命令的记录级别。 full （缺省值）会将所有更改都记录到日志中以保留所有事务的完整记录。如果数据库使用 minimal logging，Adaptive Server 会尝试不将行或页更改记录到 syslogs 中。但 Adaptive Server 可能会生成某种内存日志记录活动，以支持回退事务等运行时操作。

template

确定数据库使用的模板。可以是以下各项之一：

- *database_name* – 具有完全持久性且处于可用状态的磁盘驻留式用户数据库。
- NULL – 删除与当前模板数据库的绑定。在以后重新启动服务器的过程中，该数据库将使用 model 作为模板数据库。

with override

强制 Adaptive Server 接受设备规范，即使在同一设备上混合有数据和事务日志也要接受，因此会危及数据库的最新数据的可恢复性。如果试图不使用此子句在同一设备上混合日志和数据，则 alter database 命令会失败。如果混合了日志和数据并且使用 with override，将会收到警告，但命令会成功执行。

for load

仅在 create database for load 之后并必须重新创建要从转储装载的数据库的空间分配和段使用情况时才使用。

for proxy_update

强制重新同步代理数据库内的代理表。

示例

示例 1 向缺省数据库设备上为 2K 逻辑页配置的用户数据库中添加 3MB（1,536 页）：

```
alter database mydb
```

示例 2 为名为 newdata 的数据库设备上的 pubs2 数据库的已分配空间添加 3MB 空间：

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

示例 3 针对为 2K 逻辑页配置的服务器，为 userdata1 中的数据添加 10MB 空间，为 logdev 中的日志添加 2MB 空间：

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

示例 4 更改宽松持久性数据库 pubs5_rddb 的持久性级别，使其成为具有完全持久性的常规数据库：

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

示例 5 改变 pubs3 数据库的模板：

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

示例 6 更改宽松持久性磁盘驻留式数据库的持久性级别：

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

示例 7 更改 model 数据库的 DML 日志记录级别，该数据库的持久性级别设置为 full。在完成此更改后基于 model 创建的所有数据库都将继承最少日志记录级别属性：

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

示例 8 将 pubs2 数据更改为页级压缩：

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

示例 9 将 pubs2 数据库更改为使用 LOB 压缩：

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

示例 10 此示例修改 pubs 数据库以使其行内 LOB 列的长度更改为 400 字节：

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

示例 11 从设备 mylogdev 中删除 50MB 的数据库 sales_db：

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

此示例删除 mylogdev 上 sales_db 中编号最大的逻辑页 （最大值可达 50 MB）。

示例 12 从设备 mylogdev 中删除数据库 sales_db 的空间，并指定要删除的确切页：

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

由于逻辑页 15360 是分配页，因此，此示例影响 mydev 上从 7168 到 15359 的所有逻辑页，但不影响第 15360 页，也不影响指定范围内不在 mylogdev 上实际存在的任何页。

用法

限制

- 如果不包括单位指示符，则引号是可选的。不过，如果包括单位指示符，则必须使用引号。
- 如果所有固定长度列的总大小加上行开销大于表的锁定方案和页大小允许的值，则 Adaptive Server 会报告错误。
- 因为 Adaptive Server 以包含 256 个逻辑页的块为单位为 create database 和 alter database 分配数据库空间，所以这些命令会将指定大小向下取整为最接近的分配单元的整数倍。
- 您可以将 *size* 指定为 float 数据类型，但它会向下取整为分配单元的整数倍。
- 虽然 Adaptive Server 的确在以下情况下创建表，但当您执行数据操纵语言操作时，将会收到关于大小限制的错误：
  - 单个可变长度列的长度超出最大列大小。
  - 对于仅数据锁定表，除初始列之外的任何可变长度列的偏移量超出了 8191 字节的限制。
- 如果 Adaptive Server 无法分配请求的空间，它会为每个设备分配尽可能相近的空间，并显示一条消息，告知已经为每个数据库设备分配了多少空间。
- 要使用 alter database，您必须正在使用 master 数据库，或正在执行 master 数据库中的存储过程。

- 只能在主设备上扩展 master 数据库。试图使用 alter database 将 master 数据库扩展到其它任何数据库设备都将导致错误消息。例如，使用：

  ```
  alter database master on master = 1
  ```

- 每次您在数据库设备上用 create database 或 alter database 分配空间时，分配均代表一个设备段，且分配是作为一行在 sysusages 中输入的。
- 如果您对一个正在进行转储的数据库使用 alter database，则除非转储完成，否则 alter database 不能完成。 Adaptive Server 可锁定转储期间数据库空间使用的内存映射。如果在此内存映射锁定期间发出 alter database，则 Adaptive Server 可在转储完成后从磁盘更新此映射。如果中断 alter database，则 Adaptive Server 会指示您运行 sp_dbremap。如果不运行 sp_dbremap，则直到重新启动服务器，Adaptive Server 才可以使用添加的空间。
- 您可以在脱机数据库上 database_device 上使用 alter database 。

将 *alter database* 用于存档数据库

任何时候（而不是仅仅在空间不足时）都可以使用 alter database 为存档数据库的修改页面区域增加空间。增加修改页面区域中的空间可让挂起的命令继续执行。语法为：

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

改变内存数据库和宽松持久性数据库

- 不能将 model、 master 或 sybsystemdb 指定为模板数据库。
- 如果将 use template 子句中的数据库名称设置为 NULL，则会删除与现有模板数据库的绑定，并将 model 定义为模板数据库。
- 改变在数据库恢复顺序序列中比其模板数据库先出现的数据库的模板定义，将自动对恢复顺序重新排序，因此当您重新启动服务器时，新模板数据库将在数据库恢复顺序中出现在其相关数据库之前。
- 如果更改 durability 或 dml_logging 的设置， alter database 会在执行命令前自动尝试将数据库设置为单用户模式。您也可以在发出 alter table 之前手动将数据库设置为单用户模式。
- 数据库必须处于单用户模式，才能更改其持久性级别设置。
- 只能在已包含内存数据库的内存存储高速缓存中增加内存数据库的大小。
- 不能更改系统数据库、模板数据库或本地临时数据库的持久性级别。

- 在将数据库的持久性级别更改为 full 时，装载序列将中断。例如，对于使用完全持久性的磁盘驻留式数据库，如果您：

  a　转储数据库。

  b　执行 dump transaction。

  c　第二次执行 dump transaction。

  d　将持久性更改为 no_recovery。

  e　将持久性更改为 full。

  您将不能第三次执行 dump transaction，而必须执行完全 dump database。

在分配更多空间后备份 *master*

- 每次使用 alter database 之后，请使用 dump database 备份 master 数据库。这样做可使得在 master 破坏后恢复起来更加容易和安全。
- 如果您使用 alter database 而不备份 master，则您可以使用 disk refit 恢复更改。

把日志存放在单独的设备上

- 当您对 create database 使用了 log on 扩展时，若要增加为事务日志分配的存储空间量，请在发出 alter database 时在 log on 子句中给出日志设备的名称。
- 如果您不使用 create database 的 log on 扩展将日志放置在单独的设备上，则万一发生硬盘故障，您将无法完全恢复。在这种情况下，您可以通过使用 alter database 及 log on 子句，然后使用 sp_logdevice 将日志移动到它自己的设备，从而扩展日志。

改变数据库的压缩

- set compression 指定数据库范围的压缩级别，这仅适用于新建的表。
- 可以将 set lob_compression 单独使用，也可以将其与其它 set 子子句一起使用。但是，其它 set 子子句要求数据库处于单用户模式（例如，如果更改数据库的持久性级别）。

行内 LOB 列

- 使用 inrow_lob_length 可在数据库范围增大或减小行内 LOB 长度。
- 更改 inrow_lob_length 会影响在以后的 create table 或 alter table add 列命令中创建 LOB 列。有效值范围为 0 到数据库逻辑页大小。

缩减日志空间

有关 alter database 的 log off 变体：

- 虽然 log off 选项指定要作为逻辑页删除的页范围，但它是实际已删除的关联物理页。逻辑页仍保留在数据库中，但因形成了空洞而无法可用。空洞是一个或多个没有物理存储与之关联的分配单位。
- 有关在 master.dbo.sysusages 表中提供的设备上存在哪些分配单位（当您创建数据库或增加数据库空间时，以 256 个数据页为单位进行划分的空间）的信息，该表中按数据库 ID、起始逻辑页码、逻辑页中的大小、设备 ID 以及设备中的起始偏移量列出磁盘区段。
- 如果指定的 to 页小于 from 页，则页会进行互换，也就是说， to 页变成 from 页，反之亦然。如果 from 和 to 指定相同的页，则该命令仅会影响含有该页的分配单位。该命令不会因调整 to 页而导致命令错误。
- 如果您不提供任何子句，则整个设备都会受影响。这等同于 log off*device* from 0。此命令指定物理存储，从数据库中删除指定设备上的每个日志页：

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

- 如果 alter database 检测到错误，它不会执行，并且会返回一条消息指示原因，如：
  - The database log becomes too small.（数据库日志变得太小。）
  - 要删除的片段中包含分配给 syslogs 的页。也就是说，活动日志占用了要删除的日志片段的空间。
  - The amount of log free space after the fragment is removed is too small to accommodate the last chance threshold.（删除该片段后的可用日志空间量太小，无法容纳最后机会阈值。）

获得关于空间使用情况的帮助

- 若要查看数据库已使用的设备段的名称、大小和使用情况，请执行 sp_helpdb *dbname*。
- 若要查看当前数据库使用了多少空间，请执行 sp_spaceused。

***system* 和 *default* 段**

- system 和 default 段会映射到 alter database 命令的 on 子句中包括的每个新数据库设备。若要解除这些段的映射，请使用 sp_dropsegment。
- 当您使用 alter database （不带 override）来扩展正由此数据库使用的设备上的数据库时，映射到此设备的段会同时得到扩展。如果您使用 override 子句，则所有在 on 子句中命名的设备段都成为系统/缺省段，且所有在 log on 子句中命名的设备段成为日志段。

使用 *alter database* 唤醒休眠进程

- 如果用户进程因达到日志段的最后机会阈值而挂起，请使用 alter database 增加日志段空间。如果可用空间量超过最后机会阈值，则进程被唤醒。

使用 *for proxy_update*

- 如果输入 for proxy_update 子句时没有指定其它选项，则数据库的大小不会扩展；而代理表 （如果有）将从代理数据库中删除并通过元数据进行重新创建，该元数据是从执行 create database ... with default_location = 'pathname' 时所指定的路径名中获取的。
- 如果使用 alter database 和其它选项来扩展数据库的大小，则在扩展大小后将执行代理表同步。
- for proxy_update 为数据库管理员提供易于使用的单步操作，通过该操作可在单个远程站点获得所有表精确的最新代理表示。
- 所有外部数据源都支持重新同步，而不仅仅是高可用性集群环境中的主服务器。而且，不需要用 for proxy_update 子句来创建数据库。如果通过 create database 或使用 sp_defaultloc 指定了缺省存储位置，则数据库中包含的元数据可与远程存储位置上的元数据同步。
- 若要确认数据库得到正确同步，以使所有代理表拥有正确的方案可访问您刚刚重新装载的主数据库的内容，您可能需要在承载代理数据库的服务器上运行 for proxy_update 子句。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 alter database 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，对于 sybsecurity，您必须是数据库所有者或者具有以下特权之一： own database (on sybsecurity) 或 manage auditing。 对于所有其它数据库，您必须是数据库所有者或者具有 own database 特权（对数据库）。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者，或者是具有 sa_role 或 sso_role 的用户 （对于 sybsecurity）。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
2	alter	alter database	• *角色* – 当前活动角色 • *关键字或选项* – alter size • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 create database, disk init, drop database, load database.

系统过程 sp_addsegment, sp_dropsegment, sp_helpdb, sp_helpsegment, sp_logdevice, sp_renamedb, sp_spaceused.

alter encryption key

说明

更改当前口令、添加和删除密钥副本、重新生成加密密钥。

有关加密列的详细信息，请参见 《加密列用户指南》。

语法

改变主密钥：

```
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 }
```

改变 syb_extpasswdkey 服务密钥：

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

改变列加密密钥：

```
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}
```

参数

keyname

是列加密密钥的名称。

as [not] default
指示数据库缺省属性应该分配给此密钥，或者从此密钥取消分配。

[dual] master
数据库级密钥，用于对在其中定义它们的数据库内的其它密钥进行加密。这些密钥不用于加密数据。

static key | master key
with {static key | master key} 子句的第一个实例只是有关 syb_extpasswdkey 当前加密方式的断言。由于 Adaptive Server 知道 syb_extpasswdkey 当前是如何加密的，因此，此子句是可选的。

with {static key | master key} 子句的第二个实例位于 regenerate key 操作之后，能让管理员将重新生成的密钥的加密从静态更改为动态，反之亦然。如果省略该子句，重新生成的密钥就会像在发出此命令之前那样被加密。

with {static key | master key} 子句的第三个实例位于 modify encryption 操作之后，用于将现有密钥的保护更改为使用静态密钥或主密钥 （按照指定）。如果省略该子句，缺省情况下就会使用静态密钥。

[no] dual_control
指示是否使用双控制来创建主密钥。

regenerate key
指示您正在重新生成密钥。

with passwd ['*password*' | *system_encr_passwd* | *login_password*
'*base_key_password*']

指定 Adaptive Server 用于解密列加密密钥的当前口令和用于下列用途之一的新口令：

- 修改密钥或密钥副本的加密。
- 加密新添加的密钥副本。密钥所有者可为单个用户添加能够通过私有口令或登录口令访问的密钥副本。
- 在丢失口令后恢复加密密钥。

Adaptive Server 支持以下密钥口令：

- *password* – 最大长度为 255 字节的字符串。
- *login_passwd* – 通知 Adaptive Server 使用会话的登录口令。
- *system_encr_passwd* – 当前数据库的系统加密口令。
- '*base_key_password*' – 用于加密基本密钥的口令，可能只有密钥管理者知道。该口令长度最多可以是 255 个字节。 Adaptive Server 使用第一个口令来解密基本列加密密钥。

如果未指定 with passwd，则缺省值为 *system_encr_passwd*。

modify encryption

指示您要修改加密密钥或密钥副本。

for automatic startup

指示密钥副本将用于在启用了自动主密钥访问的服务器重新启动后访问主密钥或双主密钥。

add encryption

为指定用户添加加密密钥副本。

key_copy_password - 用于加密密钥副本的口令。该口令长度不能大于 255 个字节。 Adaptive Server 会创建解密基本密钥的副本，使用从 *key_copy_password* 派生的密钥加密密钥对该副本进行加密，并在 sysencryptkeys 中将加密基本密钥副本保存为新行。

for user *user_name*

指定要为其添加或删除密钥副本的用户。

for login_association

指示要添加的密钥副本以后将由在用户首次访问该密钥时分配的用户登录口令加密。

for recovery

指示是否在口令丢失的情况下使用密钥副本恢复主密钥。

drop encryption
指示您要为指定用户删除密钥副本。

recover encryption
使新口令可以访问基本密钥。不适用于密钥副本。

modify owner
将密钥所有者更改为指定用户。

示例

示例 1 将 my_key 更改为缺省加密密钥：

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

您必须拥有 sso_role 或 keycustodian_role，才能更改密钥的缺省属性。如果上述命令：

- 由系统安全员 (SSO) 执行， Adaptive Server 将无条件地从以前的缺省密钥 （如果有）中删除缺省属性。
- 由密钥管理者执行，则此人必须拥有 my_key。密钥管理者必须拥有以前的缺省密钥 （如果存在）。

若要从 my_key 中删除缺省属性，并且 SSO 或密钥管理者为密钥所有者，请执行：

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

如果 my_key 不是缺省密钥，则此命令将返回错误。

示例 2 更改 important_key 加密密钥的口令：

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

如果此命令：

- 由密钥所有者执行 – 则命令将对基本密钥重新加密
- 由分配了密钥副本的用户执行 – 则命令将对该密钥副本重新加密。

示例 3 将密钥副本的口令更改为当前会话的登录口令 （只能由已分配密钥副本的用户执行）：

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

使用登录口令只能加密密钥副本。如果您尝试使用登录口令加密基本密钥， Adaptive Server 将返回错误。

示例 4 将 important_key 加密密钥的口令更改为系统口令：

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

此命令只能由密钥所有者或者具有 sso_role 的用户执行，并且只允许在密钥没有密钥副本时执行。（有副本的基本密钥必须由用户指定的口令进行加密。）此示例将修改基本密钥的加密。

示例 5 将 important_key 加密密钥的口令由系统加密口令更改为新口令。由于系统加密口令是缺省口令，因此无需在语句中指定：

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

示例 6 使用口令 “just4now”为用户 “ted”的 important_key 加密密钥添加加密：

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

只有密钥所有者或具有 sso_role 的用户才能执行此命令。 Adaptive Server 使用 “TopSecret”解密基本密钥，同时创建原始密钥的副本并使用口令 “just4now”为 Ted 加密该副本。

示例 7 修改 Ted 的加密以使用新口令。只有 Ted 才能执行此命令：

```
alter encryption key important_key
     with passwd 'just4now'
     modify encryption
     with passwd 'TedsOwnPassword'
```

示例 8 为用户 “ted”删除 important_key 加密密钥的加密 （只有拥有 sso_role 的用户或密钥所有者才能执行此命令）：

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

示例 9 将 important_key 的所有者修改为新所有者 “tinnap”（只有拥有 sso_role 的用户或密钥所有者才能执行此命令）：

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

示例 10 使用主密钥来加密现有 CEK “k2”：

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

示例 11 重新加密当前用主密钥加密的现有 CEK “k3”，以便使用双控制：

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

示例 12

示例 13 设置恢复密钥副本，并在丢失口令后将其用于密钥恢复。

1 密钥管理者最初创建由口令保护的新加密密钥：

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

2 密钥管理者为 Charlie 添加 key1 的特殊加密密钥恢复副本。

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

3 Charlie 分配了不同的口令以恢复副本，并将此口令存放在锁住的抽屉中：

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

4 如果密钥管理者丢失了基本密钥的口令，则他可以从 Charlie 处获得口令，并使用以下命令从恢复副本恢复基本密钥：

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

用法

- 如果 SSO 发出 alter encryption key，将密钥设置为数据库缺省值，则指定密钥会将任何现有密钥替换为缺省值。
- 如果密钥管理者发出 alter encryption key，将密钥设置为数据库缺省值，则指定密钥和当前缺省密钥（如果存在）必须由密钥管理者拥有。
- 密钥由具有 keycustodian_role、 sso_role 的用户或被显式授予 create encryption key 命令执行权限的用户拥有和管理。有权处理和查看加密列中数据的所有用户均可使用密钥。 Adaptive Server 对密钥的保护方式将影响密钥的访问方式：

a　密钥所有者通过系统加密口令创建加密密钥 – 当用户访问加密数据时， Adaptive Server 将使用系统加密口令解密基本密钥。密钥所有者不为用户创建单个密钥副本。

b　密钥管理者使用显式口令加密基本密钥 – 密钥管理者不创建密钥副本，而是与处理加密数据的所有用户共享此口令。用户或应用程序必须使用 set encryption passwd 命令提供此口令，才能访问数据。请参见 set encryption passwrd。

c　密钥管理者为最终用户添加密钥副本，因此用户不必共享口令。用户必须使用 set encryption passwd 输入其密钥副本的口令，才能访问加密列。或者，密钥管理者也可以通过密钥被授予人的登录口令来设置用于加密的密钥副本。不必通过 set encryption passwd 输入此口令。

- 在使用 create encryption key 创建密钥时，Adaptive Server 会以加密形式将密钥与密钥的属性一起作为 sysencryptkeys 中的一行进行保存。此行表示基本密钥。密钥所有者可以选择仅允许通过基本密钥访问加密数据，或使用 alter encryption key 为单个用户添加密钥副本。
- 如果 alter encryption 不包括 with passwd 参数，Adaptive Server 将使用系统加密口令。
- 使用系统加密口令不能改变具有副本的密钥的基本密钥，也不能加密密钥副本。
- 用户分配的密钥副本只能修改这些用户自己的密钥副本。
- 如果指定 *login_association*，则 Adaptive Server 会使用系统加密口令临时加密密钥副本。当副本所有者使用该密钥加密或解密数据时，该所有者的登录口令将对密钥副本重新加密。
- 无法为相同的密钥副本指定 for recovery 和 *login_association*。

权限

对 alter encryption key 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage column encryption key 特权的用户，才能执行 alter encryption key as default 或 not default。 您必须是密钥所有者或具有以下取决于密钥类型的特权： • 列加密密钥 – manage column encryption key • 主密钥 – manage master key • 服务密钥 – manage service key 更改为： • 使用 alter encryption key 添加或删除密钥副本、恢复密钥和修改密钥所有者。 • 执行 alter encryption key 修改基本密钥的口令。 **注释** 您必须是分配有密钥副本的用户，才能修改密钥副本口令。您隐式拥有修改自己的密钥副本口令的权限。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sso_role 或 keycustodian_role 的用户，才能执行 alter encryption key as default 或 not default。 只有系统安全员或密钥所有者才能执行以下操作： • 使用 alter encryption key 添加或删除密钥副本、恢复密钥和修改密钥所有者。 • 执行 alter encryption key 修改基本密钥的口令。 **注释** 您必须是分配有密钥副本的用户，才能修改密钥副本口令。您隐式拥有修改自己的密钥副本口令的权限。
审计	有关审计加密列的信息，请参见 《加密列用户指南》中的 “审计加密列”。
另请参见	create encryption key、 drop encryption key 和sp_encryption。

alter login

说明　　更改登录帐户的属性。

语法　　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*] }

参数　　*login_name*

指定要更改的登录帐户的名称。

modify

如果属性存在，则将属性值更改为指定的新值。如果属性不存在，则指定的属性列表和相应的值会添加到登录帐户中。 *attribute_value_pair_list* 是属性名和指定的值。指定以下各项中的一个或多个：

参数	参数值	说明
login profile	有效值： • *login_profile_name* • ignore • default	• *login_profile_name* 将指定的登录配置文件绑定到指定的登录帐户。如果已经存在登录配置文件绑定，则会将它替换为指定的登录配置文件。 • ignore 消除所有登录配置文件绑定。如果存在登录配置文件绑定，则将其删除。缺省的登录配置文件将不适用，而是会按照 15.7 之前的版本那样应用属性。 • default 会删除现有登录配置文件绑定，并将缺省登录配置文件与登录帐户相关联。 如果未指定登录配置文件 *login_profile_name*，而存在缺省登录配置文件，则缺省登录配置文件将与登录帐户相关联。
fullname	*name_value*	拥有登录帐户的用户的全名。可添加全名或修改现有名称。 缺省值为 NULL。
password expiration	有效范围：0 到 32767 天	口令有效期。 缺省值为 0，意思是口令永不过期。
min password length	有效范围：0 到 30。	要求的最小口令长度。 缺省值为 6。
max failed attempts	有效范围：-1 到 32767。	允许的登录尝试次数，此后登录帐户将被锁定。 -1 指示将会记录失败次数，但不锁定。 缺省值为 0，意味着不会记录失败次数，而且不会由于登录尝试失败而锁定帐户。

参数	参数值	说明
authenticate with	有效值：ASE、 LDAP、 PAM、 KERBEROS、 ANY	指定用于鉴定登录帐户的机制。 当使用 ANY 时， Adaptive Server 会检查有无定义的外部鉴定机制。如果定义了外部鉴定机制，Adaptive Server 会使用它，否则会使用 ASE 机制。 如果未指定 authenticate with *authentication mechanism*，则 ANY 将用于登录帐户。
default database	*default_database_name*	指定用作缺省数据库的数据库。 缺省值为 Master。
default language	*default_language*	指定用作缺省语言的语言。 缺省值为 us_english
login script	*login_script_name*	指定有效的存储过程。登录脚本仅限于 120 个字符。
exempt inactive lock	有效值：TRUE 或 FALSE。	指定是否不让登录帐户因非活动而被锁定。 缺省值为 FALSE，意味着不让帐户锁定。

add auto activated roles

指定在已经授予的非口令保护的用户定义角色中，哪些必须在登录时自动激活。

drop auto activated roles

指定在已经授予的用户定义角色中，哪些不得在登录时自动激活。ALL 指定所有授予的用户定义角色。

drop

从登录帐户中删除指定的属性。从以下属性中指定一个或多个要删除的属性：

参数	
login profile	从指定的登录帐户中删除登录配置文件绑定。如果指定了 login profile ignore 参数，则删除该参数，而且不再忽略现有的缺省登录配置文件。
fullname	删除与登录帐户关联的名称。
password expiration	删除口令有效期值。
min password length	删除对最小口令长度的所有限制。
max failed attempts	删除对允许的失败尝试次数的限制。
authenticate with	删除对鉴定机制的指定。
default database	删除对缺省数据库的指定。
default language	删除对缺省语言的指定。
login script	删除对应用登录脚本的指定。
exempt inactive lock	删除有关是否因非活动而锁定登录帐户的指定。如果不想锁定登录帐户，则设置缺省值 FALSE。

with password *caller_password* modify *new_loginName_password*
将登录口令更改为新指定的口令。

immediately
指定口令是否对登录的用户立即生效。如果您：

- 指定 immediately – 口令更改在 syslogins 表中立即生效，并且登录的用户将在其仍处于登录状态的情况下更新其口令。
- 不指定 immediately – 所有登录的用户 （调用方除外）在其重新连接前保持其旧口令。

示例

示例 1 将登录配置文件 emp_lp 绑定到登录帐户 ravi。

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

示例 2 当指定 ignore 时，会忽略所有登录配置文件，无论它是绑定到 users_1 登录帐户的登录配置文件还是定义的缺省登录配置文件都是如此。

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

示例 3 创建两个登录配置文件；第一个是 general_lp，它是缺省登录配置文件，第二个是名为 emp_lp 的登录配置文件，它是针对特定的员工组定义的。创建登录配置文件后，这两个登录配置文件中的属性都会应用到一个帐户帐户。有关属性的应用顺序的信息，请参见 《安全性管理指南》中的 “应用登录配置文件和口令策略属性”。

```
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 autheticate with
LDAP
```

以下示例将登录配置文件 emp_lp 绑定到登录帐户 users_2。 default language 和 track lastlogin 不是在登录配置文件 emp_lp 中定义的，而是在缺省登录配置文件中定义的。因此，default language 和 track lastlogin 的值获取自 general_lp。

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

示例 4 创建两个登录配置文件；第一个是针对新员工的 newEmployee_lp，第二个是针对现有员工的 default_lp。

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

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

以下示例在登录时将登录脚本 newEmp_script 应用到 employee_new。

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

```
"newEmployee_lp"
```

登录配置文件 default_lp 在登录时会应用到没有通过登录配置文件指定登录脚本的现有帐户。

示例 5 演示如何针对签约员工和全职员工实现授予且自动激活的各种不同角色：

```
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"
```

用法

优先规则决定在从不同登录配置文件中获得属性时或使用 sp_passwordpolicy 指定了值时如何应用登录帐户属性。

有关优先规则，请参见 《安全性管理指南》中的 “应用登录配置文件和口令策略属性”。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 alter login 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您通常必须具有 manage any login 特权才能执行 alter login 帐户。要修改登录帐户口令，您必须具有 change password 特权或者是帐户的所有者。帐户所有者可以修改帐户的全名。
细化权限已禁用	在禁用细化权限的情况下，您通常必须具有 sso_role 才能执行 alter login 帐户。帐户所有者可以修改帐户口令和全名。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
138	login_admin	alter login	关键字包含： • MODIFY *attribute_value_pair_list* • DROP *attribute_name_list* • MODIFY PASSWORD • ADD AUTO ACTIVATED ROLES *role1* [*role2*][...[*roleN*]...]] • DROP AUTO ACTIVATED ROLES {ALL | *role1* [, *role2* [... [, *roleN*] ...]] }

另请参见

命令 create login、 create login profile、 alter login profile、 drop login、 drop login profile.

文档 有关改变登录帐户的信息，请参见 《安全性管理指南》。

函数 lprofile_id、 lprofile_name.

系统过程 sp_passwordpolicy、sp_displaylogin、sp_displayroles、sp_locklogin.

alter login profile

说明

更改登录配置文件的属性。

语法

```
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] }
```

参数

as [not] default

as default 将登录配置文件修改为缺省登录配置文件。 as not default 删除指定登录配置文件的缺省属性。

login_profile_name

指定要更改的登录配置文件的名称。

modify

如果属性存在，则将属性值更改为指定的新值。如果属性不存在，则指定的属性列表和相应的值会添加到指定的登录配置文件中。

attribute_value_ pair_list 是属性名和指定的值。指定以下属性中的一个或多个：

参数	参数值	说明
default database	*default_database_name*	指定 Adaptive Server 中的数据库。 缺省值为 master。
default language	*default_language*	指定语言。 缺省值为 us_english
login script	*login_script_name*	指定有效的存储过程。登录脚本仅限于 120 个字符。
authenticate with	有效值：ASE、 LDAP、 PAM、 KERBEROS、 ANY	指定用于鉴定登录帐户的机制。 当使用 ANY 时， Adaptive Server 会检查有无定义的外部鉴定机制。如果定义了外部鉴定机制，Adaptive Server 会使用它，否则会使用 ASE 机制。 如果未指定 authenticate with *authentication mechanism*，则 ANY 将用于登录帐户。
track lastlogin	有效值：TRUE、 FALSE。	启用上次登录更新。 缺省值为 TRUE，表示更新。
stale period	1 .. 32767 天。 持续时间：D （天）、 W （周）、 M （月）、 Y （年）	指示允许登录帐户在因不活动而被锁定前保持不活动状态的持续时间。缺省值为 D （天）。

add auto activated roles
指定在已经授予的非口令保护的用户定义角色中，哪些必须在登录时自动激活。如果指定的角色未授予登录名，则会生成错误。缺省情况下，用户定义的角色不会在登录时自动激活。

drop auto activated roles
指定在已经授予的用户定义角色中，哪些不得在登录时自动激活。ALL 指定所有授予的用户定义角色。

drop *attribute_name_list*
删除以下内容：

- default database – 删除缺省数据库的指定。
- default language – 删除缺省语言的指定。
- login script – 删除应用登录脚本的指定。
- authenticate with – 删除与帐户关联的鉴定机制的指定。
- track last login – 删除启用上次登录更新的指定。
- stale period – 删除针对登录帐户在锁定前保持非活动状态而指定的任何限制。

示例

示例 1 将 eng_lp 配置为缺省登录配置文件。如果存在现有的缺省登录配置文件，则会删除其缺省属性。

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

示例 2 改变登录配置文件 mgr_lp 以便在登录时自动激活已经授予的角色 program_role、 product_role 和 admin_role （如果它们不受口令保护）。

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

示例 3 改变登录配置文件 mgr_lp 以便删除已经授予的角色 admin_role 在登录时的自动激活。

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

示例 4 改变登录配置文件 mgr_lp 以便删除 login script 属性。删除后，与 mgr_lp 关联的登录帐户将会使用缺省登录脚本 （如果已经定义）的值。如果未定义，则登录脚本属性将被设置为缺省值，即在登录时不调用登录脚本。

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

用法

- 优先规则决定在从不同登录配置文件中获得属性时或使用 sp_passwordpolicy 指定了值时如何应用登录帐户属性。有关优先规则，请参见 《安全性管理指南》中的 “应用登录配置文件和口令策略属性”。
- 可以指定在登录时调用的登录脚本。有关详细信息，请参见 《安全性管理指南》中的 “调用登录脚本”。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 alter login profile 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 manage any login profile 特权才能执行 alter login profile。
细化权限已禁用	在禁用细化权限的情况下，您必须具有 sso_role 才能执行 alter login profile。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
140	security_profile	alter login profile	关键字包含： • DEFAULT • NOT DEFAULT • DROP *attribute_name_list* • MODIFY *attribute_value_pair_list* • ADD AUTO ACTIVATED ROLES *role1* [*role2*][...[*roleN*]...]] • DROP AUTO ACTIVATED ROLES {ALL | *role1* [, *role2* [... [, *roleN*] ...]] }

另请参见

命令 create login、 create login profile、 alter login、 drop login、 drop login profile.

文档 有关改变登录配置文件的信息，请参见 《安全性管理指南》。

函数 lprofile_id、 lprofile_name.

系统过程 sp_passwordpolicy、sp_displaylogin、sp_displayroles、sp_locklogin.

alter precomputed result set

说明　更改预先计算结果集的属性或策略

语法

```
alter {precomputed result set | materialized view}
        [owner_name.]prs_name
    [{immediate | manual } refresh]
    [enable | disable]
    [{enable | disable } use in optimization]
```

参数

precomputed result set | materialized view
更改物化视图或预先计算结果集。

prs_name
预先计算结果集的名称。完全限定的 *prs_name* 不得含有服务器或数据库名称。

{immediate | manual} refresh
确定刷新策略：

- immediate – （缺省值）预先计算结果集在用于更新基表的事务中进行更新。
- manual – 显式更新预先计算结果集。如果使用 manual 参数，则只有在显式发出 refresh 后，基表的更新内容才会反映在预先计算结果集中。由于预先计算结果集设为 manual 的状态并不会持续，因此， Adaptive Server 会认为它们已失效（即使在发出 refresh 参数后），因而只有在查询接受失效数据时，查询处理器才会选择该数据进行查询重写。

enable | disable
指定预先计算结果集是否可用于操作。此选项会覆盖所有其它预先计算结果集选项。

- enable – （缺省值）预先计算结果集可用于操作。只有配置为 enable 的预先计算结果集才按刷新策略进行维护。
- disable – 预先计算结果集不可用于操作。维护或查询重写不考虑禁用的预先计算结果集。如果预先计算的结果集配置为 disable，则不：
  - 在优化期间用于查询重写，无论是否指定 use in optimization。
  - 进行填充，无论是否指定 with populate。

{enable | disable} use in optimization

指定是否包含预先计算结果集以在优化期间进行查询重写。 use in optimization 缺省状态下已启用。查询重写是否考虑预先计算结果集取决于 refresh 参数的设置：

- immediate – 所有查询都考虑使用。
- manual – 只在查询接受失效数据时才考虑使用。

示例

将 authors_prs 从 manual 更改为 immediate 刷新策略：

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

用法

- 将预先计算结果集从 manual 更改为 immediate 刷新策略后，预先计算结果集将自动刷新。
- 删除或更改预先计算结果集所依据的基表或视图后，该预先计算结果集将自动更改为 disable。

权限

只有预先计算结果集所有者才可以执行 alter 命令。

alter...modify owner

说明

将数据库对象的所有权从一个所有者移交给另一个所有者。

语法

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

参数

object_type

所有权要显式移交的对象类型。可指定以下对象类型之一：

- table – 用户表和代理表
- view – 视图
- procedure – 存储过程
- function – 用户定义的函数
- default – 与创建表无关的、单独定义的缺省值
- rule – 规则
- type – 用户定义的数据类型
- encryption key – 加密密钥

all

所有准许的对象类型。当指定为 all *owner*.* 时，指定所有者拥有的所有准许对象的所有权都将被移交。当指定为 all *owner.object_name* 时，指定所有者拥有的名为 *object_name* 的准许对象的所有权将被移交。

owner.

指示所有者数据库用户 ID (uid) 所确定的对象的当前所有者。如果用户正在移交自己所拥有的对象的所有权，指定 *owner* 就是可选的。sysobjects 表中的对象的所有权与所有者的登录名和 uid 相关联。

object_name

指示所有权将要移交的对象的名称。如果尝试将名为 *object_name* 的对象的所有权移交给同一个所有者，则会引发错误消息。

*

由 *owner* 拥有且由 *object_type* 指定的所有对象。当 *object_type* 为 all 时，由 *owner* 拥有的所有对象都会被移交。当 *owner* 是数据库所有者时，* 是不被允许的。

name_in_db

所有权移交到的新所有者的数据库用户名。*name_in_db* 指定的用户必须是现有用户，不能是、 guest、角色 组或别名。

loginame only *login_name*

仅将相关对象的 sysobjects 中的 loginame 字段移交给 *login_name*。*login_name* 必须是 syslogins 表中的有效登录名。

preserve permissions

指示是否为其所有权将被移交的对象保留显式授予或撤消的权限：

- 如果指定 – 针对这些对象的所有显式授予或撤消的权限都将被保留，并且权限的 *grantor* 将更改为新的所有者。

  例如， bill 通过 grant 选项向 mark 授予了表 `bill_table` 的 select 权限。然后， mark 向 john 授予了针对表 `bill_table` 的 select 权限。如果随后通过指定的 preserve permissions 将该表的所有权移交给 eric，则 mark 和 john 仍将具有其对 `bill_table` 的权限。

- 如果不指定 – 针对这些对象的所有现有的显式授予或撤消的权限都将从系统中删除，结果是， sysprotects 表中与该对象对应的行将被删除。

不会为以前的所有者保留对对象拥有的隐式权限。新所有者将获得所有隐式权限。

例如，bill 是 bill_table 的所有者，对 bill_table 拥有隐式 alter、delete、insert、references、select 和 update 权限和显式 decrypt 权限。在通过指定的 preserve permission 将所有权移交给 eric 后，bill 将仅对 bill_table 具有 decrypt 权限。

示例

示例 1 将名为 bill.author 的表的所有权移交给 eric：

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

示例 2 将名为 bill.vw_author_in_ca 的视图的所有权移交给 eric 而不删除所有现有的显式授予权限：

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

示例 3 将 bill 拥有的所有表的所有权都移交给 eric：

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

示例 4 将 bill 拥有的所有对象的所有权都移交给 eric：

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

示例 5 如果新所有者 cindy 已经拥有一个名为 cindy.publisher 的表，则此命令会失败。

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

示例 6 如果尝试将 bill.publisher 的所有权移交给 cindy，则会引发错误，因为 bill.publisher 不是存储过程。

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

用法

对象的隐式移交

以下依赖对象的所有权在其所依赖的对象的所有权被移交后，也会被隐式移交：

- trigger – 当所有者相同时，触发器的所有权随同依赖表一起更新。如果 DBO 拥有的触发器是针对非 DBO 拥有的表或视图创建的，则该触发器的所有权是无法改变的。
- 在创建表或视图的过程中定义的声明性对象。
  - 缺省值
  - 解密缺省值
  - 检查约束
  - 引用约束
  - 分区约束
  - 计算列

系统数据库中的对象

在移交以下系统数据库中的对象的所有权时，应十分谨慎：sybsecurity、sybsystemdb、 model、 sybsystemprocs、 sybsyntax、 dbccdb 和 tempdb。

不要移交由 Sybase 提供或管理的系统对象的所有权，如 （但不限于）带有 spt_ 前缀的用户表、带有 sp_ 前缀的系统存储过程，以及监控表。这样做可能会导致系统数据库不可使用。

加密密钥

不允许将加密密钥移交给一个拥有密钥副本的所有者，否则该命令会失败。

更改加密密钥的所有者不影响加密密钥副本的被授予人。

预先计算结果集

不能使用 alter ... modify owner 命令更改预先计算结果集的所有者。要更改所有者，请删除预先计算结果集，然后以新所有者重新创建。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 alter... modify owner 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有对象的 alter any object owner 特权，而不是加密密钥。 只有加密密钥所有者或具有以下特权 （基于加密密钥的类型）的用户可以移交加密密钥所有权： • 列加密密钥 – manage column encryption key • 主密钥 – manage master key • 服务密钥 – manage service key 显式或隐式别名为数据库所有者的数据库用户 （拥有 alter any object owner 特权）不可以移交其具体拥有的对象的所有权。如果对象将数据库所有者用户 ID 作为 sysobjects.uid 值，而将 NULL 或数据库所有者的登录名作为 sysobjects.loginame 值，则该对象会被标识为由数据库所有者具体拥有。

细化权限已禁用	在禁用细化权限的情况下：
	• 允许系统安全员 (SSO) 用户使用此命令移交对象的所有权。
	• 只有 SSO 用户和加密密钥所有者可以移交加密密钥所有权。
	• 允许数据库所有者 (DBO) 用户以及显式或隐式别名成 DBO 的用户以及类型不是加密密钥的对象的所有权，但有以下限制：
	  • 不允许数据库所有者移交其具体拥有的对象的所有权。如果对象将 DBO_UID 作为 sysobjects.uid 值，而将 NULL 或数据库所有者的登录名作为 sysobjects.loginame 值，则该对象会被标识为由数据库所有者具体拥有。
	  • 为安全起见，禁止在一个命令中移交多个对象的所有权，而且， DBO 对象的移交必须采用 dbo.*object_name* 形式。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息

124	alter	alter .. modify owner	• *角色* – 当前活动角色
			• *关键字或选项* – 以下各项之一：
			  • USER TYPE *owner.obj_name* – 如果您要更改用户定义类型的所有权
			  • NEW OWNER – *name_in_db*
			  • PRESERVE PERMISSIONS – 如果指定该选项
			  • NEW LOGINAME – *login_name* （如果指定 LOGINAME ONLY *login_name*）
			• *先前值* – NULL
			• *当前值* – NULL
			• *其它信息* – NULL
			• *代理信息* – set proxy 有效时的初始登录名

alter role

说明

定义角色之间的互斥关系；为角色添加、删除和更改口令；指定口令有效期、最小口令长度，以及指定的角色允许的最大登录尝试失败次数。alter role 也用于锁定和解锁角色。

语法

```
alter role role1 {add | drop} exclusive
        {membership | activation} role2

alter role role_name [add passwd "password" |
        drop passwd] [lock | unlock]

alter role {role_name | "all overrides"}
        set {passwd expiration | min passwd length |
        max failed_logins} option_value
```

参数

role1

是互斥关系中的一个角色。

add

添加一个互斥关系的角色；添加口令到角色。

drop

删除一个互斥关系的角色；从角色删除一个口令。

exclusive

使两个命名角色互斥。

membership

不允许同时向用户授予这两个角色。

activation

允许您向用户同时授予这两个角色，但不允许用户同时激活这两个角色。

role2

是在互斥关系中的另一个角色。

role_name

是您要为之添加、删除或更改口令的角色的名称。使用 *role_name* 可指定口令有效期、最小口令长度和最大登录尝试失败次数。

passwd

为角色添加或删除口令。

password

是添加给角色的口令。口令不能使用变量。有关口令规则，请参见《系统管理指南，卷 1》中的 “管理 Adaptive Server 登录名、数据库用户和客户端连接”。

lock
锁定指定角色。

unlock
解锁指定角色。

all overrides
将后面的设置应用于整个服务器而不是特定角色。

set
激活它后面的选项。

passwd expiration
指定口令的有效期（以天为单位）。可以是介于 0 和 32767 之间（包含这两个值）的任何值。

min passwd length
指定所指定口令允许的最小长度。

max failed_logins
指定所指定口令允许的最大登录尝试失败次数。

option_value
指定 passwd expiration、 min passwd length 或 max failed_logins 的值。若要设置 all overrides，请将 *option_value* 的值设置为 -1。

示例

示例 1 将 intern_role 和 specialist_role 定义为在成员资格级别互斥：

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

示例 2 将角色定义为在成员资格级别和激活级别互斥：

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

示例 3 将口令到添加现有角色：

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

示例 4 从现有角色删除口令：

```
alter role doctor_role drop passwd
```

示例 5 锁定 physician_role：

```
alter role physician_role lock
```

示例 6 解锁 physician_role：

```
alter role physician_role unlock
```

示例 7 将 physician_role 允许的最大登录尝试失败次数更改为 5：

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

示例 8 将现有角色 physician_role 的最短口令长度设置为五个字符：

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

示例 9 替换所有角色的最短口令长度：

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

示例 10 删除所有角色的最大登录失败次数的替换值：

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

用法

- alter role 命令定义角色之间的互斥关系，并为角色添加、删除和更改口令。
- all overrides 参数删除使用 sp_configure 及下列任意参数设置的系统替换值：
  - passwd expiration
  - max failed_logins
  - min passwd length

  删除角色口令可删除口令有效期的替换值及最大登录尝试失败次数选项。
- 当您使用 alter role 锁定或解锁角色时，您会设置 （或取消设置）添加到 syssrvroles 中的 locksuid、 lockdate 和 lockreason 列。

互斥角色

- 不需要使用任何特定的次序来指定互斥关系的角色或角色层次。
- 您可以使用互斥关系和角色层次对用户定义角色加以约束。
- 互斥成员资格与互斥激活相比是更为强烈的约束。如果将两个角色定义为成员资格互斥，则二者为激活隐式互斥。
- 如果将两个角色定义为成员资格互斥，则定义二者为激活互斥对其成员资格定义毫无影响。激活互斥相对于成员关系互斥来说是单独添加和删除的。
- 将两个角色授予给用户或角色后就不能将二者定义为互斥的。在试图将角色定义为在成员资格级别互斥前，请从现有的被授权者中撤消已授予的角色。
- 如果将两个角色定义为在激活时互斥，系统安全员可将这两个角色分配至同一用户，但此用户无法同时激活这两个角色。

- 如果系统安全员将两个角色定义为激活互斥，而且用户已经激活了这两个角色，或者缺省情况下在登录的时候激活了这两个角色，Adaptive Server 将使这两个角色互斥，但会发出一个警告消息，指出特定用户具有冲突的角色。用户激活的角色并不发生更改。

为角色更改口令

若要为角色更改口令，请首先删除现有的口令，然后添加新口令，如下所示：

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

注释 低于 12.x 的 Adaptive Server 版本的口令以及用户定义角色的口令不会到期。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 alter role 的权限检查因细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须拥有 manage roles 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须为拥有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
85	roles	create role、drop role、alter role、 grant role 或 revoke role	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 create role, drop role, grant, revoke, set.

文档 有关改变角色的详细信息，请参见 《系统管理指南》。

函数 mut_excl_roles, proc_role, role_contain, role_id, role_name.

系统过程 sp_activeroles, sp_displaylogin, sp_displayroles, sp_modifylogin.

alter table

说明

- 向表中添加新列；删除或修改现有列；添加、更改或删除约束；更改现有表的属性；启用或禁用表的触发器；更改表的压缩级别。
- 支持添加、删除和修改计算列，并允许更改现有计算列的实现属性、可为空性或定义。
- 按照指定分区策略对表进行分区或重新分区，向具有现有分区的表中添加分区，并拆分或合并现有分区。

语法

```
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
```

分区的 alter table 语法：

```
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]] ...)}
```

计算列的 alter table 语法：

```
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]}
```

用于删除、拆分、合并和移动分区的 alter table 语法：

```
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
```

参数

table_name

是要更改的表的名称。如果该表位于另一数据库中，请指定数据库名称，如果此数据库含有多个具有该名称的表，请指定所有者的名称。*owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

add

指定添加到表的列或约束的名称。如果已经启用 CIS，则不能对远程服务器使用 add。

column_name

是表中列的名称。如果在数据库中启用了 Java，则此列可能为 Java-SQL 列。

datatype

除 bit 之外的任何系统数据类型或除基于 bit 的数据类型之外的用户定义的任何数据类型。

如果在数据库中启用了 Java， *datatype* 可以是数据库中安装的 Java 类的名称，此 Java 类可以是系统类或用户定义类。请参见 《Adaptive Server Enterpris 中的 Java》。

default

指定列的缺省值。如果您指定了一个缺省值，而且在插入数据时用户没有为该列提供值，则 Adaptive Server 会插入此值。缺省值可以是 *constant_expression*、 user （插入正在插入数据的用户的名称）或 null （插入空值）。

Adaptive Server 以 *tabname_colname_objid* 为格式生成缺省名称，其中 *tabname* 是表名的前 10 个字符， *colname* 是列名的前 5 个字符，而 *objid* 是缺省对象 ID 号。将缺省值设置为 null 会删除缺省值。

如果已经启用 CIS，则不能对远程服务器使用 default。

constant_expression

是用作列的缺省值的常量表达式。它不能包含全局变量、任何列名或其它数据库对象，但可以包含内置函数。该缺省值必须与列的数据类型兼容。

user

指定如果用户不提供值，则 Adaptive Server 插入用户名作为缺省值。列的数据类型必须是 char(30)、varchar(30) 或 Adaptive Server 隐式转换为 char 的类型；但如果数据类型不是 char(30) 或 varchar(30)，可能出现截断。

null | not null

指定在无缺省值的情况下，Adaptive Server 在数据插入过程中的行为。

null 指定添加一个允许空值的列。如果用户不提供值，则 Adaptive Server 在插入的时候分配一个空值。

Bit 类型列的属性必须始终为 not null。

not null 指定添加一个不允许空值的列。如果没有缺省值，则用户必须在插入的时候提供一个非空值。

如果不指定 null 或 not null，缺省情况下 Adaptive Server 将使用 not null。不过，为了使此缺省值与 SQL 标准兼容，可以使用 sp_dboption 对它进行切换。如果为新添加的列指定（或表示）not null，则需要一个缺省子句。缺省值可用于新添加的列的所有现有行，也适用于以后插入的内容。

materialized | not materialized

指示您是在创建实现列还是未实现列。

encrypt [with *keyname*]

指定加密列和用于对其加密的密钥。

keyname 标识使用 create encryption key 创建的密钥。表所有者必须对 *keyname* 拥有 select 权限。如果未提供 *keyname*，服务器将查找使用 create encryption key 或 alter encryption key 创建的缺省密钥。

有关支持的数据类型列表，请参见 《加密列用户指南》中的 “加密数据”。

decrypt_default *constant_expression*

指定此列为没有解密权限的用户返回缺省值， *constant_expression* 是 Adaptive Server 在 select 语句中返回的值，而不是解密值。该值只能在可空列中为 NULL。如果 decrypt_value 无法转换为列的数据类型，则仅当查询时， Adaptive Server 才会捕获转换错误。

decrypt

解密加密列。

compressed = *compression_level* | not compressed

指示是否压缩行中数据以及压缩级别。

compression_level

压缩级别。压缩级别包括：

- 0 – 不压缩行。
- 1 到 9 – Adaptive Server 使用 ZLib 压缩。通常，压缩级别数字越大， Adaptive Server 压缩 LOB 数据的程度就更大，压缩和未压缩数据之间的比率就越大 （也就是说，压缩数据与未压缩数据大小相比，节省的空间量就越大 （以字节为单位））。

  但是，压缩量取决于 LOB 内容，压缩级别越高，进程的 CPU 占用率就越高。也就是说，级别 9 的压缩率最高，但其 CPU 占用率也最高。
- 100 – Adaptive Server 使用 FastLZ 压缩。此压缩率的 CPU 占用率最低；通常在数据较少时使用。
- 101 – Adaptive Server 使用 FastLZ 压缩。与值为 100 时相比，值为 101 时的 CPU 占用率略高，但压缩率也略高。

压缩算法忽略不使用 LOB 数据的行。

identity

表示该列具有 IDENTITY 属性。数据库中的每个表都可以具有一个以下数据类型的 IDENTITY 列：

- 精确 numeric 类型 （标度为 0），或
- 任意整数数据类型，包括有符号或无符号 bigint、 int、 smallint 或 tinyint。

IDENTITY 列不能更新，也不允许有空值。

IDENTITY 列存储由 Adaptive Server 自动生成的序列号，例如发票编号或职员编号。 IDENTITY 列的值唯一地标识表的每一行。

off row | in row

指定 Java-SQL 列是与行分开存储，还是存储在行中直接分配的存储区。

in row 列的存储区不可超过 16K 字节，其大小根据数据库服务器的页大小以及其它变量而异。缺省值是 off row。

constraint

引入完整性约束的名称。如果已经启用 CIS，则不能对远程服务器使用 constraint。

constraint_name

是约束的名称，必须符合标识符规则并且在数据库中唯一。如果没有指定表级约束的名称， Adaptive Server 则生成格式为 *tabname_colname_objectid* 的名称，其中， *tabname* 是表名的前 10 个字符，*colname* 是列名的前 5 个字符，而 *objectid* 则是约束的对象 ID 号。如果没有为唯一约束或主键约束指定名称， Adaptive Server 会生成格式为 *tabname_colname_tabindid* 的名称，其中 *tabindid* 是表 ID 和索引 ID 的字符串并置。

约束并不适用于添加约束时就已经存在于表中的数据。

unique

约束指定列的值，以使任两行都不具有相同的非空值。此约束将创建一个唯一索引，该索引只有在删除了约束后才能被删除。此选项不能与 null 选项一起使用。

primary key

约束指定列的值，以使任两行都不具有相同的值，而且值都不能为空。此约束将创建一个唯一索引，该索引只有在删除了约束后才能被删除。

clustered | nonclustered

指定 unique 或 primary key 约束创建的索引为聚簇索引或非聚簇索引。clustered 是主键约束的缺省值 （除非表中已存在聚簇索引）；nonclustered 是唯一约束的缺省值。每个表只能有一个聚簇索引。有关详细信息，请参见 create index。

asc | desc

指定索引是按升序 (asc) 的顺序还是按降序 (desc) 的顺序创建的。缺省设置是按升序排列。

with fillfactor=*pct*

指定 Adaptive Server 对现有数据创建新索引时使每一页达到的填满程度。“pct”代表百分比。 fillfactor 百分比仅在创建索引时使用。随着数据变化，页的填充程度不会维持任何特定水平。

警告！ 使用 fillfactor 创建聚簇索引会影响数据占用的存储空间，因为创建聚簇索引时 Adaptive Server 会重新分配数据。

---

fillfactor 的缺省值是 0；此值在 create index 语句不包含 with fillfactor 时使用 （除非使用 sp_configure 更改了此值）。指定 fillfactor 时，请使用介于 1 和 100 之间的值。

fillfactor 值为 0 可创建具有完全填充页的聚簇索引及具有完全填充叶页的非聚簇索引。它在聚簇索引和非聚簇索引的索引 B 树中都保留适量空间。通常不需要更改 fillfactor。

如果将 fillfactor 设置为 100，Adaptive Server 在创建聚簇索引和非聚簇索引时完全填满每页。将 fillfactor 的值设置为 100 仅对只读表 （将不再向其中添加数据的表）有意义。

如果 fillfactor 的值小于 100（0 除外，0 是特殊情况），Adaptive Server 在创建新索引时不会填满每页。创建表的索引时，如果表最终将包含大量数据，那么将 fillfactor 的值设置为 10 可能是一种明智的选择，但较小的 fillfactor 值会导致每个索引 （或索引和数据）占用较多的存储空间。

transfer table [on | off]

改变表进行增量传输的合格性。无论是否将表标记为用于传输，缺省值均为不进行任何更改。如果 alter table 命令指定 set transfer table，同时对 on 或 off 的选择与当前值不同，则会更改表的合格性。

max_rows_per_page = *num_rows*

限制数据页和索引的叶级页上的行数。与 fillfactor 不同，max_rows_per_page 的值将保持不变，直到使用 sp_chgattribute 更改它。

如果不指定 max_rows_per_page 的值， Adaptive Server 将在创建索引时使用值 0。当为数据页指定 max_rows_per_page 时，请使用 0 到 256 之间的值。非聚簇索引的每页最大行数因索引键的大小而异；如果指定的值过高， Adaptive Server 则返回错误消息。

对于由约束创建的索引，将 max_rows_per_page 设置为 0 将创建充满页的聚簇索引和充满叶页的非聚簇索引。设置为 0 时在聚簇索引和非聚簇索引的索引 B 树内都保留了适当数量的空间。

如果 max_rows_per_page 设置为 1，Adaptive Server 会创建叶级为每页一行的聚簇和非聚簇叶索引页。使用这个方法可以减少经常访问的数据的锁争用。

如果 max_rows_per_page 值较小， Adaptive Server 会创建具有不完全填充页的新索引，使用更多的存储空间，并可能导致更多的页面拆分。

警告！ 使用 max_rows_per_page 创建聚簇索引会影响数据占用的存储空间，因为创建聚簇索引时 Adaptive Server 会重新分配数据。

---

reservepagegap = *num_pages*

指定对约束创建的索引进行扩充 I/O 分配操作时填充页与留下的空白页的比率。对于每个指定的 *num_pages*，都留出空白页以便在将来扩展表。有效值是 0 到 255。缺省值 0 表示不留下任何空白页。

immediate_allocation

启用 sp_dboption 'deferred table allocation' 后，将显式创建表。

on *segment_name*

指定其上存在索引或将在其上创建索引的段。使用 on *segment_name* 时，必须已经采用 create database 或 alter database 将逻辑设备分配至数据库，且必须已经采用 sp_addsegment 在数据库中创建了段。要获得数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。

如果指定 clustered 并使用 on *segment_name* 选项，则整个表都将迁移到您指定的段，因为索引的叶级包含实际的数据页。

对于分区， on *segment_name* 指定用于放置分区的段。

references

为参照完整性约束指定列列表。只能为一个列约束指定一个列值。通过将此约束包括到引用其它表的表中，插入到*引用*表中的任何数据都必须已经存在于*被引用*表中。

若要使用此约束，必须对被引用表拥有 references 权限。被引用表中的指定列必须由唯一索引（由 unique 约束或 create index 语句创建）约束。如果未指定任何列，则被引用表的相应列必须包含 primary key 约束。而且，引用表列的数据类型必须和被引用表列的数据类型完全匹配。

如果已经启用 CIS，则不能对远程服务器使用 references。

foreign key

指定列出的列是该表中的外键，这些外键对应的主键是 references 子句中列出的列。

ref_table

是包含被引用列的表的名称。您可以引用其它数据库中的表。约束可引用最多 192 个用户表和内部生成的工作表。使用 sp_helpconstraint 检查表的参照约束。

ref_column

是被引用表中的列的名称。

match full

指定如果引用行的引用列中的所有值均为：

- 空值 – 参照完整性条件为真。
- 非空值 – 如果被引用行中每个对应列在被引用表中都相等，则参照完整性条件为真。

如果不属于上述任何一种情况，则参照完整性条件在以下条件成立时为假：

- 所有值都是非空值且不相等，或者
- 引用行的引用列中的某些值是非空值，而其它值为空。

check

指定 *search_condition* 约束， Adaptive Server 将强制表中的所有行都遵守这一约束。如果已经启用 CIS，则不能对远程服务器使用 check。

search_condition

是一个布尔表达式，用于定义列值的 check 约束。这些约束可以包括：

- 用 in 引入的一系列常量表达式
- 由 like 引入的一组条件 （可以包含通配符）

表达式可以提供算术运算和 Transact-SQL 函数。 *search_condition* 不能包含子查询、集合函数、参数和主变量。

next_column

使用与列定义相同的语法提供附加列定义 （用逗号隔开）。

set dml_logging

确定 insert、update 和 delete (DML) 操作的记录量。可以是以下各项之一：

- full – Adaptive Server 记录所有事务，
- minimal – Adaptive Server 不记录行或页更改，
- default – 日志记录设置为表缺省值。

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

添加具有指定数据类型的 LOB 列。

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

指定 LOB 列的最大长度仍为行内长度。如果不指定长度， Adaptive Server 则应用对行内长度有效的数据库范围设置。

如果不用 in row (*length*) 并且数据库范围设置无效，则向 LOB 列中添加在行外存储的数据。

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

仅将 LOB 列的属性更改为 in-row，直至达到指定长度。当运行此命令时，没有任何数据移动。

您还可以使用此选项增加行内 LOB 列的长度。

注释　您不能使用此选项来减小 LOB 列的长度，也不能指定 0 作为长度。根据页的可用空间量，在此次修改后的更新过程中，将行外 LOB 数据移入行内，直至达到指定行内长度。

---

set compression

表示要用于表或分区的压缩级别。新压缩级别用于新插入或更新的数据：

- default – 将指定分区的压缩级别重置为表的压缩级别。
- none – 不压缩该表或分区中的数据。对于分区， none 表示始终不压缩该分区中的数据，即使将表压缩修改为 row 或 page 压缩也是如此。
- page – 当页面已满时，则会使用页级压缩来压缩现有的采用行压缩形式的数据行，从而创建页级字典、索引和字符编码条目。将 page 压缩设置为分区或表级别。

  Adaptive Server 只有已经在行级压缩数据后才在页级压缩数据，因此，将压缩设置为 page 意味着 page 和 row 压缩。
- row – 压缩各行中的一个或多个数据项。只有在压缩形式比未压缩形式更节省空间时， Adaptive Server 才会用 row 压缩形式存储数据。设置分区级或表级的 row 压缩。

set lob_compression = *compression_level*

更改使用 LOB 数据类型的表的压缩级别。

drop

指定从表中删除的列或约束的名称。如果已经启用 CIS，则不能对远程服务器使用 drop。

modify

指定您要更改其数据类型或可为空性的列的名称。

[not] compressed

指示是否压缩修改的列。

replace

指定要用新值来替换缺省值的列，新值由下面的 default 子句指定。如果已经启用 CIS，则不能对远程服务器使用 replace。

enable | disable trigger

启用或禁用触发器。有关触发器的信息，请参见 《系统管理指南》。

lock datarows | datapages | allpages

更改将要对表使用的锁定方案。

with exp_row_size=*num_bytes*

指定期望行宽。只能将此参数应用于：

- 数据行和数据页锁定方案。
- 具有可变长度行的表。
- 当 alter table 执行数据复制时，例如使用 alter table add 或 modify。不能将 with exp_row_size=*num_bytes* 用于 alter table lock change 操作。

有效值可以是 0、1 以及介于表的最小和最大行长度之间的任何值。缺省值为 0，表示应用服务器范围的设置。

no datacopy

表示 alter table 在不复制数据的情况下删除列，同时在发生 alter table 操作时防止 alter table... drop 命令阻止对表运行其它命令。

partition *number_of_partitions*

向未分区表（带单个分区的循环分区表）中添加 (*number_of_partitions* – 1) 空分区。这样，表的总分区数变为 *number_of_partitions*。如果已经启用组件集成服务 (CIS)，则不能对远程服务器使用 partition。

unpartition

将无索引的循环分区表更改为未分区表。如果已经启用 CIS，则不能对远程服务器使用 unpartition。

partition by range

指定将根据一个或多个分区列中的值对记录分区。将每个分区列值与用户提供的几组上下限相比较，以确定分区分配。

column_name

在 *partition_clause* 中使用时，指定一个分区键列。分区键列不能是加密列。

partition_name

指定要在其上存储表记录的新分区的名称。分区名称在表或索引上的分区组内必须唯一。如果启用 set quoted_identifier，则分区名称可以是分隔标识符。否则，分区名称必须是有效的标识符。

如果省略 *partition_name*，则 Adaptive Server 将创建一个格式为 *table_name_partition_id* 的名称。 Adaptive Server 将截断超过允许的最大长度的分区名称。

values <= *constant* | MAX

指定命名分区的上限值 （包括边界值）。为最高分区上限指定常量值将在表上施加隐式完整性约束。关键字 MAX 指定给定数据类型的最大值。

on *segment_name*

在 *partition_clause* 中使用时，指定将在其上放置分区的段。使用 on *segment_name* 时，必须已经采用 create database 或 alter database 将逻辑设备分配至数据库，且必须已经采用 sp_addsegment 在数据库中创建了段。要获得数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。

partition by hash

指定将按系统提供的散列函数对记录分区。该函数计算分区键的散列值，这些分区键指定将记录分配到的分区。

partition by list

指定将按命名列中指定的实际值对记录分区。分区键只包含一个列。最多可以列出 250 个常量作为各列表分区的分区值。

partition by round-robin

指定将按顺序方式对记录分区。循环分区表没有分区键。用户和优化程序都不知道特定记录位于哪个分区中。

add partition

仅适用于域分区表或列表分区表：

- 对于域分区表 – 向域分区表的上端添加一个或多个分区。
- 对于列表分区表 – 添加带有一组新值的一个或多个分区。

modify partition

指定要对其修改压缩级别的分区。

compute | as

添加或删除新的计算列。遵循为 create table 命令定义的相同规则和 alter table add 规则。

computed_column_expression

任意有效的 Transact-SQL 表达式，该表达式不包含来自其它表、局部变量、聚合函数或子查询的列。它可以是由一个或多个运算符连接起来的列名、常量、函数、全局变量、 case 表达式或它们的组合。除虚拟计算列引用实现计算列时以外，不能在计算列之间进行交叉引用。不能在 *computed_column_expression* 中引用加密列。

materialized | not materialized

指定计算列是否为实现列。它们是 modify 子句中的保留关键字，用于指定是否实现计算列 （即，以物理方式存储在表中）。缺省情况下，计算列使用 not materialized （即，不以物理方式存储在表中）。也可以使用此参数更改现有虚拟计算列的定义；即，实现这些列。

table_name drop partition *partition_name* [, *partition_name*]...

删除一个或多个列表或域分区。不能使用 alter table 删除散列或循环分区。

对于删除的每个分区， Adaptive Server：

- 删除此分区上的所有数据
- 从系统目录中删除分区定义
- 删除引用此数据分区的所有相应本地索引分区
- 重新生成基表和每个本地索引的分区条件对象
- 删除此分区的所有统计信息
- 重建所有全局索引

注释　如果尝试从其它表引用的表中删除分区，并且要删除的分区和引用表不为空，则此命令会因可能的外键约束冲突而失败，并且 Adaptive Server 将显示错误消息 13971。

---

split partition *partition_name* into *partition_condition_clause*

将分区数据重新分配至两个或多个分区。

partition_condition_clause

表示指定如何拆分源分区数据的条件。通常，这些条件是数值范围或数据范围。分区条件应仅涵盖源分区中的全部数据。

partition_condition_clause 可能与源分区处于同一段，也可以处于新段上。如果不指定目标分区段， Adaptive Server 则在源分区所在段上创建新分区。

merge partition

将两个或多个可合并分区的数据并入一个分区中。

destination_partition_name

新分区或现有分区。如果 *destination_partition_name* 是现有分区，它不能是您正在合并的任何源分区。如果不指定目标分区名，请选择系统生成的名称。

move partition

将分区 （及其索引）移至指定段。

destination_segment_name

您正将分区移向的新段或现有段。不能指定 “default”作为 *destination_segment_name*。

示例

示例 1 为表添加列。 Adaptive Server 为表中每个现有行分配一个 NULL 列值：

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

示例 2 向表中添加 IDENTITY 列。 Adaptive Server 为表中的每个现有行分配一个唯一的顺序列值。IDENTITY 列可以是 numeric 或 integer 类型，标度为零。精度确定可插入该列的最大值 （10^5 -1 或 99,999）：

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

示例 3 向 authors 表添加主键约束。如果表上现在有一个主键或唯一约束，则必须首先删除现有的约束 （请参见下面的示例）：

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

示例 4 删除 au_identification 约束：

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

示例 5 对 authors 创建索引；该索引的 reservepagegap 值为 16，在索引中为每 15 个分配的页留一个空白页：

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

示例 6 删除 authors 表中 phone 列的缺省约束。如果列允许空值，则没有指定值时会插入空值。如果列不允许空值，则不指定列值的插入操作将失败：

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

示例 7 修改 emp 表以加密 ssn 列，并指定解密缺省值：

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

示例 8 解密不再敏感的信用卡数据：

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

如果已经通过受用户定义的口令保护的密钥对卡加密，请在执行此命令之前执行 set encryption key 命令。

示例 9 向现有表中添加加密列。由于省略了 keyname，因此 Adaptive Server 将查找数据库的缺省加密密钥：

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

示例 10 为 ssn_key 加密密钥设置口令，并对现有 employee 表中的 ssn 列进行加密。

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

如果此示例中的 ssn 是通过 “key1” 加密的现有加密列，则 alter table 将导致 Adaptive Server 使用 “key1” 解密 ssn，并使用 “ssn_key” 重新加密 ssn。

示例 11 向已加密的 salary 列中添加解密缺省值：

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

示例 12 删除 salary 的解密缺省值，而不删除加密属性：

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

示例 13 将未分区表更改为具有三个分区的域分区表，这三个分区分别位于不同段上：

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

示例 14 向 titles 表中添加另一个域分区：

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

示例 15 将 pubs2 数据库中的 titles 表改为使用行级压缩：

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

示例 16 将 sales 表的 Y2009 分区更改为使用页级压缩：

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

示例 17 将 titles 表的锁定方案更改为数据行锁定：

```
alter table titles lock datarows
```

示例 18 将非空列 author_type 添加到缺省值为 primary_author 的 authors 表中：

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

示例 19 从 titles 表删除 advance、 notes 和 contract 列：

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

示例 20 将 authors 表的 city 列更改为具有缺省值 NULL 的 varchar(30)：

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

示例 21 将 stores 表的 stor_name 列更改为 NOT NULL。其数据类型 varchar(40) 保持不变：

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

示例 22 修改 titles 表的 type 列，并将 titles 表的锁定方案从所有页锁定更改为数据行锁定：

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

示例 23 将 titles 表的 notes 列从 varchar(200) 更改为 varchar(150)，将缺省值从 NULL 更改为 NOT NULL，并指定一个值为 40 的 exp_row_size：

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

示例 24 向 mytable 中添加增量传输属性：

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

示例 25 从 mytable 中删除增量传输属性：

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

示例 26 添加、修改和删除一列，然后在同一查询中添加另一列。改变锁定方案，并指定新列的 exp_row_size 值：

```
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
```

示例 27 修改 mymsgs 表的 description 列，以便支持 400 字节的行内 LOB 长度：

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

示例 28 添加一个虚拟计算列：

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

示例 29 将一个虚拟计算列更改为实现计算列：

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

示例 30 将包含 orders 表的分区拆成两个分区：

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

示例 31 将包含 sales 表的分区合并成一个分区：

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

示例 32 将 orders 表移至 seg4 段：

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

示例 33 从 titles 表中删除 total_sales 列并复制数据：

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

用法

- 您不能在包括虚拟散列表的段上使用 alter table。
- 您不能在包括 VHASH 表的段上使用 alter table，因为虚拟散列表必须仅占用一个排它段，其它表或数据库不能共享此排它段。
- 在表中添加、修改或删除列之前，可通过运行 sp_depends 来查看是否有任何存储过程依赖于您要更改的表。如果存在此类存储过程，则将其删除，然后在更改表模式之后根据需要重新创建这些存储过程。

- 如果使用 select * 的存储过程引用一个已更改的表，则即使使用 with recompile 选项，结果集中也不会出现新列。为了包括这些新列，必须删除过程并重新创建。否则，在已经修改表并在表中添加新列后，此过程中的 insert into table1 select * from table2 可能导致错误结果。
- 当表所有者使用 alter table 时， Adaptive Server 在命令执行期间禁用访问规则，并在命令完成时启用访问规则。禁用访问规则是为了避免在 alter table 期间过滤表数据。
- 如果指定 clustered 并使用 on *segment_name* 选项，则整个表都将迁移到您指定的段，因为索引的叶级包含实际的数据页。
- alter table.. .transfer table 涉及数据复制 （与添加或删除列类似）：此命令会对性能产生极大的影响。
- alter table 在改变表之前会针对检查约束进行错误检查。
- 对分区使用 on *segment_name* 时，必须已经用 create database 或 alter database 将逻辑设备指派给了数据库，且必须已经用 sp_addsegment 在数据库中创建了段。有关数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。

限制

警告！ 不要改变系统表。

- 如果指定了缺省值，则不能向现有表添加数据类型为 bit 的列。该缺省值必须是 0 或 1。
- 表中的最大列数为：
  - 对于所有页锁定 (APL) 表和仅数据锁定 (DOL) 表中的固定长度列，为 1024
  - 对于 APL 表中的可变长度列，为 254
  - 对于 DOL 表中的可变长度列，为 1024
- 如果 APL 表中的可变长度列的数目超过 254，则 alter table 将引发错误。
- 在更改表的锁定方案后，删除已编译对象，然后重新创建。
- 不能对以下对象使用 no datacopy 参数：
  - 已实现或虚拟的计算列
  - 加密列
  - XML 列

  - Java 列
  - 使用 timestamp 或 bit 数据类型的列
- 行内 Java 列的最大长度取决于可变长度列的最大长度，而后者则取决于表的模式、锁定风格和页大小。
- 将表转换为不同的锁定方案时，源表中的数据不能违反目标表的限制。例如，如果尝试将可变长度列数超过 254 的 DOL 表转换为 APL 表，alter table 将失败，因为 APL 表的可变长度列数被限制为不超过 254。
- 具有固定长度数据 （char、 binary 等）的列的最大长度如表 1-1 所示：

表 1-1: 行和列的最大长度 – APL 和 DOL 表

锁定方案	页大小	最大行长度	最大列长度
APL 表	2KB （2048 字节）	1962	1960 字节
	4KB （4096 字节）	4010	4008 字节
	8KB （8192 字节）	8106	8104 字节
	16KB （16384 字节）	16298	16296 字节
DOL 表	2KB （2048 字节）	1964	1958 字节
	4KB （4096 字节）	4012	4006 字节
	8KB （8192 字节）	8108	8102 字节
	16KB （16384 字节）	16300	16294 字节 （如果表不包含任何可变长度列）。
	16KB （16384 字节）	16300 （取决于 varlen 的最大起始偏移 = 8191）	8191-6-2 = 8183 字节 （如果表至少包含一个可变长度列）。 *

* 此大小包括 6 个字节的行开销和 2 个字节的行长度字段。

- 每行的可变长度数据的最大字节数取决于表的锁定方案。下面描述了 APL 锁定表的最大大小列：

页大小	最大行长度	最大列长度
2KB （2048 字节）	1960	1960
4KB （4096 字节）	4008	4008
8KB （8192 字节）	8104	8157
16KB （16384 字节）	16296	16227

下面描述了 DOL 表的最大大小列：

页大小	最大行长度	最大列长度
2KB （2048 字节）	1960	1958
4KB （4096 字节）	4008	4006

页大小	最大行长度	最大列长度
8KB （8192 字节）	8157	8102
16KB （16384 字节）	16294	16294

- 不能使用 alter table 添加声明或检查约束，然后将数据插入到同一批处理或过程的表中。可以将 alter 和 insert 语句分放在两个不同的批处理或过程中，也可以使用 execute 分别执行操作。
- 不能在包含缺省值的 alter table 语句中使用下面的变量：

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

  这样做将出现错误消息 154：`Variable is not allowed in default.`
- create proxy table、create table at remote server 或 alter table 当前不支持用户定义的 SQL 函数。

注释 执行 SQL 函数时，需要使用语法 username.functionname()。

alter table 和加密列

- 在使用 alter table 添加或修改加密列时，如果表中包含大量的行，则此操作可能会占用相当长的时间。
- 修改要加密的列可能导致表的行宽增大。
- 在下列情况下，不能使用 alter table 对列进行加密或解密：
  - 如果列属于聚簇索引或位置索引。要对此类列进行加密或解密，应删除索引，修改列，然后重新创建索引。
  - 如果表定义了触发器。在修改列之前，先删除触发器。然后，重新创建触发器：
- 如果修改属于聚簇索引或位置索引的加密列的数据类型，索引会顺序混乱， alter table 则显示错误。在修改类型之前，先删除索引。然后，重新创建索引。
- 可以对以下数据类型进行加密：
  - int、 smallint、 tinyint
  - unsigned int、 unsigned smallint、 unsigned tinyint
  - bigint、 unsigned bigint

  - decimal 和 numeric
  - float4 和 float8
  - money、 smallmoney
  - date、 time、 smalldatetime、 datetime、 bigdatetime
  - char 和 varchar
  - unichar、 univarchar
  - binary 和 varbinary
  - bit
- 磁盘上的加密数据的基本数据类型为 varbinary。不会对空值加密。
- 如果修改属于聚簇索引或位置索引的加密列的数据类型，索引会顺序混乱， alter table 将显示错误。在修改类型之前删除索引，然后重新创建索引。
- 如果执行以下操作， alter table 将报告错误：
  - 将计算列更改为加密列或将加密列更改为计算列
  - 启用某个要加密的列，计算列使用的表达式中引用该列
  - 将计算列修改为引用加密列
  - 加密属于函数索引成员的列
  - 将加密列指定为分区键
  - 启用一个已经用作分区键的列进行加密

注释当　使用同一密钥对列加密时，支持加密列之间的参照完整性。有关详细信息，请参见 《加密列用户指南》中的 “加密数据”。

改变表的压缩

- 可使用 set compression 更改表的压缩级别，以便将来进行数据插入或更新。 set compression 不影响现有的已经压缩的数据行和页，但需要对表具有独占访问权限。
- 不能在同一命令中修改表的压缩级别和分区的压缩级别。必须采用单独命令执行这些操作。
- 可以将 set compression 与其它 set 参数一起使用。

- 更改表的压缩级别仅影响尚未显式定义压缩级别的分区。所有未显式定义压缩级别的分区都会隐式继承表的压缩级别。例如，如果将表的压缩级别从未压缩改为行级压缩，则所有压缩级别为 none 的分区都不会发生更改，但未定义压缩级别的分区会更改为行级压缩。
- 修变表的压缩级别不会更改现有列的压缩级别。例如，如果 my_table 及其列尚未压缩，则当您改变 my_table 的压缩级别时，它的列最初仍然尚未压缩。但是，向这些列中填充了足够数据从而触发压缩机制时， Adaptive Server 将分别压缩它们。
- 新添加列的缺省行为取决于表的压缩设置。对于已压缩表，列的数据类型确定其压缩级别。对于未压缩表，新列尚未压缩。
- 可以向表中添加压缩的已实现计算列，也可以以后压缩它们。

压缩和其它 alter table 参数之间的交互

当一个命令需要移动数据时，如果目标分区已压缩， Adaptive Server 会压缩源分区中的所有未压缩数据行。当包括 compression 子句时， alter table 包括参数之间的下列交互：

- set – 您将：
  - 不能将 set 与 add、 drop 或 modify 子句合并。
  - 不能将 modify partition set 子句与其它 modify *column_name* 参数合并。
  - 不能将 all 关键字和 modify partition 一起使用并在子句中包括分区名。
- add：
  - 可以在现有表中添加可空和不可空的压缩列。添加不可空列时，需要移动数据。
  - 如果添加到压缩表中的列配置了适当的数据类型，它们会使用行压缩。
  - 在压缩表中将列的数据类型修改为符合行压缩资格的数据类型不会更改列的压缩级别。
  - 如果您不对新列指定 not compressed，它会继承表的压缩级别。例如，如果表是行级压缩，则您添加到表中的任何列都也使用行级压缩。
- drop：
  - 删除压缩列会导致数据移动。
  - 如果表中的其它列或分区未压缩，或者无法压缩，您必须在删除最后一个压缩列之前将压缩状态更改为 none。

- modify：
  - 可以将现有表中的压缩列修改为 not compressed，反之亦然。
  - 可以同时更改列的压缩级别和数据类型。但目标数据类型必须符合行压缩级别的资格。如果目标数据类型不符合压缩资格，Adaptive Server 在 modify 操作期间忽略压缩它的请求。
  - 如果您尝试修改一个无法创建为压缩列的列的压缩级别，Adaptive Server 会发出错误消息。例如，如果您尝试压缩虚拟计算列或行内 Java 列， Adaptive Server 会发出错误消息。
- 将 add、 drop 和 modify 合并：
  - 可以在单个 alter table 语句中发出包括一个或多个压缩列的多个 add、drop 或 modify 参数。此 alter table 命令的数据移动取决于参数的数据移动限制。
  - 如果 alter table 需要移动数据，则对于不受 add、 drop 或 modify 参数影响的列，列的压缩级别会保持不变。
- 重新对表分区 – 如果没有为新分区指定 compression 子句，Adaptive Server 则将新分区的压缩级别设置为：
  - 未压缩，如果源表及其所有分区都未压缩。
  - 和源表相同的压缩级别，如果采用相同压缩级别压缩了所有分区。
  - 未压缩，如果：
    - 表或各个分区以不同方式压缩，或者
    - 源表尚未压缩，但它的某些分区经过压缩

    Adaptive Server 之所以不压缩新分区，是因为可能很难唯一地将压缩属性从原始压缩分区映射和迁移到新分区。必须将压缩级别显式声明为 alter table 命令的一部分，才能指定必须在数据复制期间压缩的目标分区。
- add partition – 如果不在 compression 子句中指定分区级别，新添加的分区会继承表的压缩级别。
- drop partition – 如果表中包含多个分区，则仅删除表的压缩分区不会更改表的压缩级别。

  如果已经对表定义了压缩，则在删除分区后，表仍是压缩的，而且 Adaptive Server 会自动将以后的分区配置为压缩。

- 更改锁定方案 – 如果将表的锁定方案从所有页锁定更改为仅数据锁定 （反之亦然）， Adaptive Server 需要移动数据。您不能在更改锁定方案的同时更改表或单个分区的压缩级别，而是必须在更改锁定方案之前运行 set compression 命令来指定压缩级别。
- 取消对表进行分区 – 如果在取消对表进行分区时至少一个源分区是压缩的，则在运行 alter table 后，会将整个表标记为压缩。如果表在最初尚未压缩， Adaptive Server 则发出警告消息。
- 其它命令 – 您不能将用于指定列的缺省值、启用或禁用触发器、添加或删除列级或表级约束等的参数，与用于指定列级或分区级压缩或复制数据的命令组合。
- 如果 alter table 不包括数据移动，则现有数据不会受到影响，一旦向表中插入这些数据， Adaptive Server 会向它们应用表的压缩级别。如果 alter table 包括数据移动，则现有的数据会根据表的压缩级别来决定是压缩还是解压缩。
- 以下 alter table 事件包括数据移动：
  - 添加非空列
  - 删除列
  - 修改一个列以增大其长度 （例如，从 smallint 到 int，或从 char(3) 到 varchar(45)）
- 以下 alter table 事件不包括数据移动：
  - 添加空列
  - 添加空文本列 （添加非空文本列是有限制的）
  - 修改一个可变长度列以增大其长度 （例如，从 varchar(5) 到 varchar(20) 或从 varchar(20) 到 varchar(40)）
- 您不能压缩代理表或代理表上的分区或列。

改变使用大对象的表的压缩级别

- 更改表的大对象 (LOB) 压缩级别仅会影响未显式定义压缩级别的 LOB 列。未显式定义压缩级别的列会隐式继承表的压缩级别。
- 如果还没为新添加的 LOB 列指定 LOB 压缩子句，则其缺省行为取决于表的 LOB 压缩级别。对于 LOB 压缩表， Adaptive Server 对列使用表的 LOB 压缩级别。对于 LOB 未压缩表，新添加的 LOB 列仍然尚未压缩。

压缩与含有 LOB 数据的表的其它 alter table 参数之间的交互：

- drop column – 如果在删除列后表中不包括任何压缩 LOB 列，则表使用表级 LOB 列压缩级别。
- add column
  - 您可以添加可为空的压缩 LOB 列，但不能添加不可为空的压缩 LOB 列。
  - 对于未设置为 LOB 压缩的表，缺省情况下，新添加的 LOB 列尚未压缩。新添加的具有 LOB 压缩子子句的 LOB 列可以是压缩的，也可以是未压缩的，具体取决于指定内容。
- modify column
  - 可以解压缩现有的压缩 LOB 列。虽然新插入的数据是未压缩的，但现有数据仍是压缩的。
  - 可以更改现有 LOB 列的压缩级别。虽然新插入的数据采用新的压缩级别，但现有数据仍保留原来的压缩级别。
  - 可以将未压缩的 LOB 列更改为 compressed。
  - 不能将常规列修改为 LOB 列 （压缩或未压缩）。
  - 可以将压缩的 LOB 列修改为：
    - 压缩的 text 列 （使用 *n*char、 *n*varchar、 unichar 和 univarchar）
    - 压缩的 image 列 （使用 varbinary 和 binary）
    - 压缩的 unitext 列 （使用 *n*char、 *n*varchar、 unichar、 univarchar、 varbinary 和 binary）

    不能将压缩的行外 java 列修改为常规列。

    Adaptive Server 会解压缩 LOB 数据，在需要时截断数据以适合常规列长度，并将其转换为常规数据类型。常规列的最大长度由 Adaptive Server 页大小控制。
- add、 drop、 modify 和 set lob_compression 的组合：
  - 可以在涉及一个或多个压缩列的单个 alter table 命令 （或 set lob_compression 和 set compression 子子句）中发出多个 add、 drop 或 modify 子命令。
  - 如果向 LOB 压缩表中添加列并在命令中包括 set lob_compression = 0，则不会压缩新添加的列。
  - 如果向常规的未压缩表中添加列并在命令中包括 set lob_compression = *compression_level*，则会压缩新添加的列。

现有的 LOB 数据不受 alter table 命令影响，只有以后的 DML 会受更改的 LOB 压缩属性的影响。使用 update 和 select into 压缩或解压缩现有 LOB 数据。

获取有关表的信息

- 有关表及其列的信息，请使用 sp_help。
- 若要重命名表，请执行 sp_rename （不可重命名系统表）。
- 有关完整性约束 （unique、 primary key、 references 和 check）或 default 子句的信息，请参见本章的 create table。

指定索引是按升序还是按降序排列

- 在索引列名后使用 asc 和 desc 关键字指定索引的排序顺序。通过创建索引使各列按照在查询的 order by 子句中指定的顺序排列，可以避免在查询过程中进行排序。请参见 《性能和调优系列：基础知识》中的 “为改善性能建立索引”。

使用跨数据库参照完整性约束

- 创建跨数据库约束时， Adaptive Server 会在每个数据库的 sysreferences 表中存储以下信息：

表 1-2: 存储的有关参照完整性约束的信息

存储在 sysreferences 中的信息	包含有关被引用表的信息的列	包含有关引用表的信息的列
键列 ID	refkey1 到 refkey16	fokey1 到 fokey16
表 ID	reftabid	tableid
数据库 ID	pmrydbid	frgndbid
数据库名称	pmrydbname	frgndbname

- 删除引用表或其数据库时， Adaptive Server 会从被引用数据库中删除外键信息。
- 由于引用表依赖于来自被引用表的信息， Adaptive Server 不允许您：
  - 删除被引用表，
  - 删除包含被引用表的外部数据库，或者
  - 用 sp_renamedb 对其中任何一个数据库进行重命名。

  您必须先用 alter table 删除跨数据库约束
- 每次添加或删除跨数据库约束或者删除包含跨数据库约束的表时，都请转储*上述两个*受影响的数据库。

警告！ 装载这些数据库的早期转储可能会导致数据库损坏。

- sysreferences 系统表存储外部数据库的 name 和 ID 号。如果您使用 load database 命令更改数据库名称或将其装载到其它服务器上，则 Adaptive Server 无法保证参照完整性。

  警告！ 在转储数据库以便采用其它名称装载它或将其移至另一个 Adaptive Server 之前，请使用 alter table 删除所有外部参照完整性约束。

更改缺省值

- 可以用两种方式创建列缺省值：通过在 create table 或 alter table 语句中将缺省值声明为列约束，或者通过使用 create default 语句创建缺省值然后使用 sp_bindefault 将它绑定到列。
- 不能使用 sp_bindefault 替换绑定至列的用户定义的缺省值。首先使用 sp_unbindefault 解除缺省值的绑定。
- 如果使用 create table 或 alter table 声明缺省列值，则不能使用 sp_bindefault 在此列上绑定缺省值。通过将缺省值更改为 NULL 来删除缺省值，然后绑定用户定义的缺省值。将缺省值更改为 NULL 会解除绑定缺省值并会从 sysobjects 表中删除缺省值。

设置索引的空间管理属性

- alter table 语句中的空间管理属性 fillfactor、 max_rows_per_page 和 reservepagegap 适用于为 primary key 或 unique 约束创建的索引。如果约束在所有页锁定表上创建聚簇索引，则空间管理属性将影响该表的数据页。
- 使用 sp_chgattribute 为表或索引更改 max_rows_per_page 或 reservepagegap、为表更改 exp_row_size 值或存储 fillfactor 值。
- 在索引为以下情形时应用索引的空间管理属性：
  - 作为 alter table 命令的结果而重新创建时，此命令将表的锁定方案从所有页锁定更改为仅数据锁定 （反之亦然）。请参见 第 74 页的 “更改锁定方案”。
  - 作为 reorg rebuild 命令的一部分被自动重建时。
- 若要查看表的当前生效的空间管理属性，请使用 sp_help。若要查看索引的当前生效的空间管理属性，请使用 sp_helpindex。
- 空间管理属性 fillfactor、max_rows_per_page 和 reservepagegap 以下列方式帮助管理表和索引的空间使用：
  - fillfactor 在创建索引时在页上留出额外空间，但不会不断对 fillfactor 进行维护。它适用于所有锁定方案。

- max_rows_per_page 限制数据或索引页的行数。它的主要用途是改善所有页锁定表的并发性。
- reservepagegap 指定空白页与整页的比率以供执行扩展分配的命令使用。它适用于所有锁定方案。

可以为表或索引存储空间管理属性以便在执行 alter table 和 reorg rebuild 命令时应用。

- 下表显示了空间管理属性和锁定方案的有效组合。如果 alter table 命令更改了表致使组合不兼容，则存储在系统表中的值仍然存在，但不会在表的操作过程中被应用。如果表的锁定方案的更改使这些属性成为有效的，随后将使用它们。

参数	Allpages	Datapages	Datarows
max_rows_per_page	可用	不可用	不可用
reservepagegap	可用	可用	可用
fillfactor	可用	可用	可用
exp_row_size	不可用	可用	可用

- 下表显示了缺省值以及对空间管理属性使用缺省值的效果。

参数	缺省值	使用缺省值的效果
max_rows_per_page	0	使每页容纳尽可能多的行，最多可容纳 256 行
reservepagegap	0	不留间距
fillfactor	0	完全填满叶页

max_rows_per_page 到 *exp_row_size* 的转换

- 如果表设置了 max_rows_per_page，且表由所有页锁定转换为仅数据锁定，则在 alter table...lock 命令将表复制到新位置之前，该值将被转换为 exp_row_size 值。复制期间将强制使用 exp_row_size。下表显示了这些值的转换方式。

如果 max_rows_per_page 设置为	将 exp_row_size 设置为
0	通过 default exp_row_size percent 设置的百分比值
255	1，即完全充满页
1 – 254	以下两项中的较小值： • 最大行宽 • 2002/max_rows_per_page 值

使用 *reservepagegap*

- 使用大量空间的命令通过分配扩展而不是分配单页来分配新空间。reservepagegap 关键字会导致这些命令留出空白页，以使将来的页分配紧邻被拆分的页或执行转移行所在的页进行。
- 表的 reservepagegap 值存储在 sysindexes 中，并在表的锁定方案从所有页锁定更改为仅数据锁定 （反之亦然）时应用。若要更改存储的值，请在运行 alter table 前使用 sp_chgattribute。
- 在所有页锁定表上使用 clustered 关键字指定的 reservepagegap 将覆盖以前使用 create table 或 alter table 指定的任何值。

对表进行分区以提高性能

- 可使用 partition by 子句对未分区的表进行分区，或者对已经分区的表重新分区。该任务需要数据复制；所有数据行都将根据指定的分区条件重新分配。如果 Adaptive Server 配置为并行处理，则可以并行运行该任务。必须将 select into/bulkcopy/pllsort 选项设置为 true。如果表有索引，则必须首先删除索引，然后才能：
  - 将未分区表更改为语义分区表。
  - 更改分区类型。
  - 更改分区键 – 不需要删除索引即可更改分区的其它属性，例如分区数、分区边界或分区位置；会自动建立索引。有关分区键和边界约束的详细信息，请参见第 193 页的 “create table”。
- 可以使用 add partition 子句向列表分区表或域分区表中添加空分区，但是不能向散列分区表或循环分区表中添加空分区。

  对于域分区表，只能将新分区添加到分区条件的上端。如果最后一个现有分区具有最大边界 (values <= (MAX))，则不能添加新分区。
- 提供 partition *number_of_partition* 和 unpartition 子句是为了与早于 15.0 版的 Adaptive Server 兼容。只能对未分区表使用 partition *number_of_partition* 来添加 (*number_of_partition*-1) 个空循环分区；现有数据位于第一个分区上，后续数据在所有分区之间分布。如果表有全局聚簇索引， Adaptive Server 会将后续数据行放在第一个分区中。要重新分布数据，请删除索引，然后重新创建。

  注释　这些命令不需要移动数据。但由于 Adaptive Server 要执行许多内部步骤，因此这些命令不会立即执行，对大型表执行这些命令时尤其如此。为了避免数据损坏，在对表进行分区或取消分区时不要中断操作。

  只能对不包含索引的循环分区表使用 unpartition 子句。

- 不能对系统表进行分区。
- 不能对远程代理表进行分区。
- 不能在用户定义的事务中发出与分区相关的 alter table 命令。
- 如果表上有活动的打开游标，则不能使用 partition by 子句更改表的分区属性。
- 使用 partition by 子句后，必须执行整个数据库转储，然后才能使用 dump transaction。
- 不能删除属于分区键的一部分的列。
- 变更键列时要小心。在某些情况下，修改键列的数据类型可能会在分区之间重新分布数据。请参见 《Transact-SQL 用户指南》。
- 更改表的分区属性会增加模式计数，使访问该表的现有存储过程在下次执行时重新编译。

使用计算列

- 添加新计算列而不指定可为空性和实现属性时，缺省选项可为空并且不实现。
- 添加新的实现计算列时，会对表中的每个现有行进行 *computed_column_expression* 求值，并将结果存储在表中。
- 不能在添加新的计算列的同时添加或修改它们的基列。
- 可以修改现有计算列的整个定义。这是删除计算列并添加具有相同名称的新计算列的快捷方法。这种列的行为与新计算列相似：如果不指定这些选项，则其缺省值不实现且可为空。
- 可以修改现有计算列的实现属性而不更该列的其它属性，例如定义该列或定义其可为空性的表达式。
- 将非空实现计算列修改为虚拟列时，必须在 modify 子句中指定“null”。
- 修改未实现的计算列以实现它时，会对表中的每个现有行进行 *computed_column_expression* 求值，并将结果存储在表中。
- 如果修改属于索引键的现有列，则会重新创建索引。
- 不能将已用作索引键的实现计算列修改为虚拟列；必须首先删除索引。
- 不能将普通列修改为计算列，或者将计算列修改为普通列。
- 不能修改或删除由计算列引用的基列。
- 不能删除被用作索引键的计算列。

添加 IDENTITY 列

- 向表中添加 numeric 或整数 IDENTITY 列时，请确保列精度足够大以容纳现有的行数。如果行数超过 10 精度 - 1， Adaptive Server 输出错误消息并且不会添加列。
- 当向表中添加 IDENTITY 列时， Adaptive Server 会：
  - 锁定表直到已生成所有 IDENTITY 列值。如果表包含大量行，在此过程可能会花费大量时间。
  - 为每个现有行指定一个唯一的、连续的 IDENTITY 列值 （从 1 开始）。
  - 记录对该表的每个插入操作。在向有大量行的表中添加 IDENTITY 列之前，使用 dump transaction 清除数据库的事务日志。
- 每次向表中插入行时， Adaptive Server 都会生成一个比该值大 1 的 IDENTITY 列值。该值优先于任何在 alter table 语句中为该列声明的或使用 sp_bindefault 绑定到该列的缺省值。

更改表模式

- add、 drop、 modify 和 lock 子子句在更改现有表的模式时非常有用。单个语句可以按照任何顺序包含任何数目的此类子子句，前提是不在一个语句中多次引用同一列名。
- 如果使用 select * 的存储过程引用一个已更改的表，则即使使用 with recompile 选项，结果集中也不会出现新列。为了包括这些新列，必须删除过程并重新创建。
- 为了确保触发器正确触发，必须在执行 add、drop、modify 或 lock 操作后删除并重新创建已修改表的所有触发器。
- 如果用 alter table 添加一个 not null 列，Adaptive Server 将发出错误消息。
- 不能删除表中的所有列。同样，不能删除表中剩余的最后一列 （例如，如果从含有五个列的表中删除四个列，则不能删除最后一列）。若要从数据库中删除表，请使用 drop table。
- 在以下情形需要数据复制：
  - 删除一列
  - 添加 NOT NULL 列
  - 对于大多数 alter table ... modify 命令

  使用 set noexec on 和 showplan on 选项确定特定的 alter table 命令是否需要数据复制。

- 当一个 alter table 命令需要数据复制时，可以使用其它的 alter table 命令 （add、 drop 或 modify）为修改的表指定锁定方案的更改。
- 如果 alter table 执行数据复制，包括正在更改模式的表的数据库的 select into /bulkcopy/pllsort 必须打开。
- 修改的表保持现有的空间管理属性 （max_rows_per_page、 fillfactor 等等）和表索引。
- 需要数据复制的 alter table 不引发任何触发器。
- 可以使用 alter table 更改由 CIS 创建和维护的远程代理表的模式。请参见 《组件集成服务用户指南》。
- 不能在同一语句中执行数据复制和添加表级约束或参照完整性约束。
- 不能在同一语句中执行数据复制并且创建聚簇索引。
- 如果添加 not null 列，还必须指定缺省子句。此规则有一个例外：如果添加用户定义类型的列，并且该类型具有绑定的缺省值，则不必指定缺省子句。
- 始终可以在所有页锁定表中添加、删除或修改列。然而，在仅数据锁定表中添加、删除或修改列存在一些限制，如下表所述：

索引类型	分区的、所有页锁定表	未分区的、所有页锁定表	已分区的、仅数据锁定表	未分区的、仅数据锁定表
聚簇	是	是	否	是
非聚簇索引	是	是	是	是

  如果需要在具有聚簇索引的、已分区的仅数据锁定表中添加、删除或修改列，您可以：

  a 删除聚簇索引。

  b 更改仅数据锁定表。

  c 重新创建聚簇索引。

- 不能将 NOT NULL Java 对象作为列添加。缺省情况下，所有的 Java 列始终有缺省值 NULL，并且以 varbinary 字符串或 image 数据类型存储。
- 如果修改需要数据复制，则不能修改包含 Java 列的分区表。而是应先取消表的分区，执行 alter table，然后重新对表进行分区。
- 不能从索引或参照完整性约束中删除键列。若要删除键列，请先删除索引或参照完整性约束，然后删除键列。请参见 《Transact-SQL 用户指南》。

- 可以删除有缺省值或绑定了规则的列。删除列时，还会删除任何列特定的缺省值。不能删除绑定有检查约束或参照约束的列。而要先删除检查约束或参照约束，然后删除列。使用 sp_helpconstraint 确定表中的任何约束，并使用 sp_depends 确定任何列级依赖性。
- 不能从系统表中删除列。同样，不能从由 Sybase 提供的工具和存储过程创建和使用的用户表中删除列。
- 如果表为空，通常可以将现有列的数据类型修改为任何其它数据类型。如果表不为空，可以将数据类型修改为任何能显式地转换为原有数据类型的数据类型。
- 您可以：
  - 添加新的 IDENTITY 列。
  - 删除现有 IDENTITY 列。
  - 修改现有 IDENTITY 列的大小。

  有关详细信息，请参见 《Transact-SQL 用户指南》。
- 更改表模式会增加模式计数，使访问该表的现有存储过程在下次执行时被重新规范化。依赖数据类型的存储过程或视图中的更改可能会由于数据类型规范化错误而失败。更新这些依赖对象以使它们引用修改后的表模式。

修改表模式的限制

- 不能在事务内运行 alter table。
- 改变表的模式可使用 bcp 所作的备份失效。这些备份可能使用与表的当前模式不再兼容的表模式。
- 可以添加具有检查约束的 NOT NULL 列，不过 Adaptive Server 不对现有数据验证该约束。
- 如果表有聚簇索引并且操作要求数据复制，就不能使用 alter table...add、 drop 或 modify 命令更改表的锁定方案。但您可以

  a　删除聚簇索引。

  b　改变表的模式。

  c　重新创建聚簇索引。
- 如果表上存在任何活动的打开游标，则不能改变表的模式。

修改 *text* 和 *image* 列的限制

- 可以只添加接受空值的 text 或 image 列。

若要添加 text 或 image 列以使其只包含非空值，请先添加只接受空值的列，然后更新为非空值。

- 只能将列从 text 数据类型修改为以下数据类型：
  - [n]char
  - [n]varchar
  - unichar
  - univarchar
  - nchar
  - nvarchar
- 只能将列从 image 数据类型修改为 binary 或 varbinary。
- 不能在同一语句中添加新的 text 或 image 列然后删除现有的 text 或 image 列。
- 不能将列修改为 text 或 image 数据类型。

修改包含 unitext 列的表

使用 alter table 修改 unitext 列时，以下限制将适用：

- 可以添加接受空值的新的 unitext 列。
- 只能将列从 unitext 数据类型修改为以下数据类型：
  - [n]char
  - [n]varchar
  - unichar
  - univarchar
  - binary
  - varbinary
- 不能将列修改为 unitext 数据类型。
- 不能在同一语句中添加一个 unitext 列并删除一个现有 unitext 列。

更改锁定方案

- alter table 支持从一种锁定方案更改到任何其它锁定方案。您可以更改：
  - 从 allpages 到 datapages （或者反向转换）
  - 从 allpages 到 datarows （或者反向转换）

  - 从 datapages 到 datarows （或者反向转换）
- 将所有页锁定更改为仅数据锁定方案 （或者反向转换）之前，如果所有表都已分区并且索引排序要求并行排序，则使用 sp_dboption 将数据库选项 select into/bulkcopy/pllsort 设置为 true，然后在数据库中运行 checkpoint。
- 将锁定方案从所有页锁定更改为仅数据锁定 （或相反）之后，不能使用 dump transaction 命令备份事务日志；必须先执行完整数据库转储。
- 当使用 alter table...lock 将表的锁定方案从所有页锁定更改为仅数据锁定 （或者反向转换）时， Adaptive Server 会制作表的数据页的副本。在表所驻留的段上必须有足够的空间以容纳数据页的完整副本。在索引所驻留的段上必须有空间来重建索引。

  DOL 锁定表的聚簇索引在数据页上层有一叶级。如果将带有聚簇索引的表从所有页锁定更改为仅数据锁定，最终的聚簇索引将需要更多的空间。所需的额外空间取决于索引键的大小。

  使用 sp_spaceused 确定表当前占用了多大空间，并且使用 sp_helpsegment 查看可用于存储表的空间。
- 当将表的锁定方案从所有页锁定更改为数据页锁定 （或者反向转换）时，空间管理属性会于复制数据行时应用到表，而且会于创建索引时，应用到索引。当从一种仅数据锁定方案更改为其它方案时，不会复制数据页，并且不会应用空间管理属性。
- 如果表已分区，更改锁定方案将执行分区到分区的行复制。复制过程中不会平衡分区上的数据。
- 当更改表的锁定方案时， alter table...lock 命令获取表上的排它锁直到命令结束。
- 当使用 alter table...lock 从数据页锁定更改为数据行锁定时，命令不会复制数据页或重新创建索引。它只更新系统表。
- 当其他用户在系统中处于活动状态时更改锁定方案可能会对用户活动产生如下影响：
  - 访问该表的过程高速缓存中的查询计划在下次运行时将重新编译。
  - 使用该表的活动多语句过程在继续进行下一步之前将重新编译。

- 使用该表的即席批处理事务被终止。

警告！批量复制操作处于活动状态时更改表的锁定方案可能会导致表损坏。批量复制操作先获取表信息并且在读取表信息和开始发送行的时间间隔内不持有锁，这样就会为 alter table...lock 命令开始运行留下一小段时间。

添加 Java-SQL 列

- 如果数据库中启用了 Java，则可以在表中添加 Java-SQL 列。请参见《Adaptive Server Enterprise 中的 Java》。
- 新 Java-SQL 列的声明类 (*datatype*) 必须使用 Serializable 或 Externalizable 接口。
- 当向表中添加 Java-SQL 列时， Java-SQL 列不能：
  - 指定为外键
  - 在 references 子句中指定
  - 指定为具有 UNIQUE 属性
  - 指定为主键
- 如果指定了 in row，则存储的值不能超过 16KB，具体取决于数据服务器的页大小。
- 如果指定了 off row，则列不能：
  - 在检查约束中被引用
  - 在指定了 distinct 的 select 中被引用
  - 在比较运算符、谓词或 group by 子句中指定

对共享磁盘集群的限制

- 除了从同一本地临时数据库中的表引用之外，参照完整性约束不能引用本地临时数据库中的列。当 alter table 尝试从其它数据库的表中创建对此本地临时数据库中列的引用时会失败。
- 除非包含列的表驻留在同一本地临时数据库中，否则您不能用本地临时数据库中存储的加密密钥对该列加密。如果 alter table 尝试用本地临时数据库中的加密密钥对列加密，而表位于其它数据库中，该命令将失败。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

请参见 《参考手册：过程》中的 第 1 章 “系统数据类型和用户定义的数据类型”。有关数据类型兼容的信息，请参见 《参考手册：构件块》中的。

权限

对 alter table 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或拥有 alter any table 特权的用户。具有 setuser 特权的用户可通过执行 setuser 命令来充当表所有者。
细化权限已禁用	在禁用细化权限的情况下，您必须是表所有者或具有 sa_role 的用户。数据库所有者可通过运行 setuser 命令来充当表所有者

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息

3	alter	alter table	• *角色* - 当前活动角色
			• *关键字或选项* - add column、 drop column、 modify column、replace column、add constraint 或 drop constraint
			• *先前值* - NULL
			• *当前值* - NULL
			• *其它信息* - NULL
			• *代理信息* - set proxy 有效时的初始登录名
			• 如果 set transfer table [on | off] 的 set 选项为：
			  • on - Adaptive Server 会在审计记录的额外信息中输出 `SET TRANSFER TABLE ON`。
			  • off - Adaptive Server 会输出 `SET TRANSFER TABLE OFF`。

另请参见

命令 create index, create table, dbcc, drop database, dump transaction, insert, setuser.

系统过程 sp_chgattribute, sp_help, sp_helpartition, sp_rename.

alter thread pool

说明

改变线程池。

有关进程模式的考虑事项

alter thread pool 在进程模式中不受支持。

语法

```
alter thread pool pool_name with { pool name = "new_name"
        thread count = thread_count,
        [pool description = "description"]}
        [idle timeout = time_period]
```

参数

pool_name
要改变的线程池的名称。

pool name = "*new_name*"
要改变的池的新名称。

thread count = *thread_count*
线程池中的新线程数。必须大于或等于 1。

pool description = "*description*"
说明池的用途。必须少于 256 个字符。

idle timeout = *time_period*
线程在进入休眠状态之前查找工作的时间（以毫秒为单位）。值为 -1 表示线程永不进入休眠状态，即使没有工作可做，也继续耗用 CPU。值为 0 表示如果找不到工作，线程会立即进入休眠状态。

示例

示例 1 将order_pool 线程池重命名为 sales_pool：

```
alter thread pool order_pool with pool name = 'sales_pool'
```

示例 2 将 sales_pool 线程池修改为包含 7 个线程：

```
alter thread pool sales_pool with thread count = 7
```

示例 3 修改 sales_pool 的名称和说明：

```
alter thread pool sales_pool with pool name = "larger_sales_pool", pool
    description = 'thread pool exclusive to sales group'
```

示例 4 修改 sales_pool 以使线程在 500 毫秒后仍找不到工作时进入休眠状态：

```
alter thread pool order_pool with idle timeout = 500
```

示例 5 修改 sales_pool 以使线程即使找不到工作也永不进入休眠状态：

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

用法

- *thread_count* 必须大于或等于 1。

- 当您减少线程计数时，指定的线程池必须等待当前的运行任务交出控制权，然后才能减少线程数，这可能导致 Adaptive Server 在缩减线程池时略有延迟。
- 不能重命名系统创建的线程池 （以 syb_ 开头）。但是，可以使用 alter thread pool 更改系统创建的线程池中的线程数或空闲超时。
- 不能将 Transact-SQL 变量用作 alter thread pool 的参数。
- 只能为引擎线程池设置 idle timeout。
- 可以连同 alter thread pool 一起发出 execute immediate。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展

权限

对 alter thread pool 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 manage any thread pool 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须具有 sa_role。 alter thread pool 权限不包括在 grant all 命令中。

审计

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
143			旧名称、新名称和线程计数

begin...end

说明

包含一系列 SQL 语句，以使控制流语言 （例如 if...else）可以影响整个组的性能。

语法

```
begin
        语句块
end
```

参数

语句块

是包含在 begin 和 end 内的一系列语句。

示例

示例 1 如果没有 begin 和 end， if 条件将导致只执行一条 SQL 语句：

```
if (select avg (price) from titles) < $15
begin
    update titles
    set price = price * $2
    select title, price
    from titles
    where price > $28
end
```

示例 2 如果没有 begin 和 end， print 语句将不会执行：

```
create trigger deltitle
on titles
for delete
as
if (select count (*) from deleted, salesdetail
 where salesdetail.title_id = deleted.title_id) > 0
    begin
        rollback transaction
        print "You can’ t delete a title with sales."
    end
else
    print "Deletion successful--no sales for this
        title."
```

用法

- begin...end 块可以嵌套在其它 begin...end 块中。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 begin...end 无需任何权限。

另请参见

命令 if...else.

begin transaction

说明

标记用户定义事务的起点。

语法

begin tran[saction] [*transaction_name*]

参数

transaction_name

是指派给事务的名称。事务名必须符合标识符的规则。仅在最外面的一对嵌套的 begin transaction/commit 或 begin transaction/rollback 语句中使用事务名称。

示例

为 insert 语句显式地开始一个事务：

```
begin transaction
    insert into publishers (pub_id) values ("9999")
commit transaction
```

用法

- 通过将 SQL 语句和系统过程放在短语 begin transaction 和 commit 内来定义事务。如果设置了链式事务模式， Adaptive Server 会在以下语句之前隐式调用一个 begin transaction：delete、 insert、 open、 fetch、 select 和 update。仍然必须使用 commit 显式地关闭事务。

- 若要取消所有或部分事务，请使用 rollback 命令。 rollback 命令必须出现在事务内；不能在提交事务后再回退事务。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 begin transaction 无需任何权限。

另请参见

命令 commit, rollback, save transaction.

break

说明

导致从 while 循环中退出。 break 经常通过 if 测试激活。

语法

```
while logical_expression
        statement
break
        statement
continue
```

参数

logical_expression

是返回 TRUE、 FALSE 或 NULL 的表达式 （列名、常量、任何由算术运算符或逐位运算符连接起来的列名和常量的组合，也可以是子查询）。如果逻辑表达式包含 select 语句，则用小括号将 select 语句括起来。

示例

如果平均价格低于 $30，则将价格翻番。然后选择最高价格；如果小于或等于 $50，则重新启动 while 循环并且再次将价格翻番。如果最高价格高于 $50，则退出 while 循环并且输出消息：

```
while (select avg (price) from titles) < $30
 begin
    update titles
    set price = price * 2
    select max (price) from titles
  if (select max (price) from titles) > $50
      break
    else
      continue
 end
 begin
  print "Too much for the market to bear"
 end
```

用法

- break 导致从 while 循环中退出。接着执行出现在标识循环结束的关键字 end 之后的语句。
- 如果嵌套有两个或多个 while 循环，则内层 break 退出到下一外层循环。首先，运行内层循环结束之后的所有语句；然后，重新开始下一外层循环。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 break 无需任何权限。

另请参见

命令 continue, while.

checkpoint

说明　将所有**脏**页 （自上次写入后已更新的页）写入数据库设备。

语法　checkpoint [all | [*dbname*[, *dbname*, *dbname*, ...]]

示例　将当前数据库中的所有脏页写入数据库设备，不考虑系统检查点日程表：

```
checkpoint
```

用法

- 您可以将 checkpoint 用于存档数据库，但是，检查点进程不会自动对存档数据库设置检查点。
- checkpoint 仅用作特殊情况下的预防措施。
- 更改数据库选项时， sp_dboption 自动缺省设置为使用 checkpoint。
- 可以指定一个或多个数据库运行 checkpoint。

自动检查点

- 由 checkpoint 命令产生的检查点补充了自动检查点，自动检查点按 Adaptive Server 根据最大可接受恢复时间的可配置值计算出的间隔发生。
- checkpoint 通过确定一个用于保证所有已完成事务已写入数据库设备的点来缩短自动恢复过程。虽然检查点时间因 Adaptive Server 上的活动量而异，但典型的 checkpoint 历时约一秒钟。
- 自动 checkpoint 间隔由 Adaptive Server 根据系统活动和系统表 syscurconfigs 中的恢复间隔值计算得出。恢复间隔通过指定系统恢复所需时间的最大值来确定 checkpoint 频率。通过执行 sp_configure 重新设置此值。
- 可以为 Adaptive Server 配置多个检查点进程。这允许具有多个引擎的 Adaptive Server 更频繁地对任务执行检查点操作，从而缩短自动恢复进程。
- 如果管家任务在服务器空闲时间内可以刷新所有已配置高速缓存中的全部活动缓冲池，它将唤醒检查点任务。检查点任务决定了它是否能够对数据库执行检查点操作。

  作为管家任务的结果发生的检查点称为**自由检查点**。自由检查点不包括将许多脏页写入数据库设备的工作，因为该工作已由管家任务完成。它们可以提高数据库的恢复速度。

标准　符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 checkpoint 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下： • 要执行特定数据库上的 checkpoint，您必须是数据库所有者或具有 checkpoint 或 own database （在数据库上）特权。 • 要执行 checkpoint all，您必须是所有适用数据库的数据库所有者或者具有 checkpoint any database 特权或 own and database 特权。否则，checkpoint all 会针对您具有 checkpoint 运行权限的数据库运行。

细化权限已禁用	在禁用细化权限的情况下：
	• 要执行 checkpoint *database*，您必须是数据库所有者或具有以下角色的用户：
	  • sa_role 或
	  • replication_role 或
	  • oper_role
	• 要执行 checkpoint all，您必须是所有适用数据库的所有者或具有以下角色的用户：
	  • sa_role 或
	  • replication_role
	否则， checkpoint all 仅针对您所具有的数据库运行。

另请参见

系统过程 sp_configure, sp_dboption.

close

说明

使游标失效。

语法

close *cursor_name*

参数

cursor_name
是要关闭的游标的名称。

示例

关闭名为 authors_crsr 的游标：

```
close authors_crsr
```

用法

- 此 close 命令实质上删除游标的结果集。对于关闭的游标，未定义结果集中的游标位置。
- 如果游标已关闭或不存在， Adaptive Server 将返回错误消息。

标准

符合 ANSI SQL 的级别符合初级标准。

权限

使用 close 无需任何权限。

另请参见

命令 deallocate cursor, declare cursor, fetch, open.

commit

说明

标记用户定义的事务的终点。

语法

commit [tran | transaction | work] [*transaction_name*]

参数

tran | transaction | work

指定您想提交事务或工作。如果指定 tran、 transaction 或 work，则还可指定 *transaction_name*。

transaction_name

是指派给事务的名称。它必须符合标识符的规则。仅在最外面的一对嵌套的 begin transaction/commit 或 begin transaction/rollback 语句中使用事务名称。

示例

更新两位作者的 royaltyper 条目后，插入保存点 percentchanged，然后确定书籍涨价 10% 对作者的版税收入有何影响。使用 rollback transaction 命令使事务回退到保存点：

```
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
```

用法

- 通过将 SQL 语句和系统过程放在短语 begin transaction 和 commit 内来定义事务。如果设置了链式事务模式， Adaptive Server 会在以下语句之前隐式调用一个 begin transaction：delete、 insert、 open、 fetch、 select 和 update。仍然必须使用 commit 显式地关闭事务。
- 若要取消所有或部分整个事务，请使用 rollback 命令。 rollback 命令必须出现在事务中。执行 commit 后，不能再回退事务。
- 如果当前没有事务处于活动状态，则 commit 或 rollback 语句对 Adaptive Server 不起作用。

标准

符合 ANSI SQL 的级别符合初级标准。

commit transaction 和 commit tran 形式的语句是 Transact-SQL 的扩展。

权限

使用 commit 无需任何权限。

另请参见

命令 begin transaction, rollback, save transaction.

compute clause

说明

生成在查询结果中显示为附加行的摘要值。

语法

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

参数

row_aggregate

是以下值之一：

- sum – 是 （数值）列中的值的总和。
- avg – 是 （数值）列中的值的平均值。
- min – 是列中的最小值。
- max – 是列中的最大值。
- count – 是列中 integer 形式的值的数量。
- count – 是列中 bigint 形式的值的数量。

column_name

是列名，必须用小括号括起。sum 和 avg 只能用于数值列。sum 和 avg 只能用于 integer、 numeric 和 decimal 列。

by

计算子群的行集合值。只要 by 项的值发生变化，就会生成行集合值。如果使用 by，就必须使用 order by。

在 by 后列出多个项可将一个组分为多个子群并在每个分组级别应用一个函数。

示例

示例 1 计算价格超过 $12 的每种烹饪书的总价格：

```
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)
```

示例 2 使用一个 compute 子句并向同一组分组列应用几个集合函数，计算价格超过 $12 的每种烹饪书的总价格和总预付款：

```
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          sum
           ---------  ------------
               19.99          0.00

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

示例 3 使用一个 compute 子句并向同一组分组列应用几个集合函数，计算价格超过 $12 的每种烹饪书的总价格和最高预付款：

```
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)
```

示例 4 按 type 和 pub_id 分组，并通过类型和出版社 ID 的组合来计算心理学书籍的总价格：

```
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)
```

示例 5 使用多个 compute 子句创建多个组，除按 type 和 pub_id 计算总和外，还计算价格超过 $10 的心理学书籍的总价格：

```
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)
```

示例 6 计算价格超过 $10 的烹饪书的总价格和总预付款：

```
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)
```

示例 7 计算烹饪书的总价格以及在表达式中使用的总价格：

```
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
```

用法

- compute 子句允许您在一个结果集中查看明细行和摘要行。可计算子群的摘要值，还可计算同一组的多个集合值。
- 不带 by 的 compute 可用于生成总和、总计数等。如果使用不带 by 的 compute 关键字， order by 就是可选的。请参见示例 6。
- 如果使用 compute by，必须同时使用 order by 子句。在 compute by 后面列出的列必须与 order by 后面列出的列相同或是其子集，它们从左向右的顺序必须一致，以同一表达式开始且不跳过任何表达式。例如，如果 order by 子句为 order by a, b, c，则 compute by 子句可以是以下任意 （或所有）形式：

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

限制

- 不能在 compute 子句中使用多于 127 个集合列。
- 不能在游标声明中使用 compute 子句。
- 可以计算表达式和列的摘要值。任何在 compute 子句中出现的表达式或列都必须在 select 列表中出现。
- 列名的别名不可以作为 compute 子句中行集合的参数，尽管可以在 select 列表、 order by 子句和 compute 的 by 子句中使用它们。
- 在带有 compute 子句的 select 语句中，选择列表中的列顺序将替换 compute 子句中的集合的顺序。Open Client™ 、JDBC 和 DB-Library™ 程序员必须了解这一点才能将集合结果放在正确的位置。
- 在同一语句中不能将 select into 用作 compute 子句，因为包括 compute 的语句不能生成常规表。
- 如果 compute 子句包括一个 group by 子句：
  - compute 子句不能包括 255 个以上的集合。

  - group by 子句包含的列不能超过 255 个。
- compute 子句中包含的列不能超过 255 个字节。

compute 的结果作为一个或多个新行出现。

- 集合函数通常为表中的所有选定行或每个组生成一个值，这些摘要值将显示为新列。例如：

```
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 子句允许使用一个命令检索明细行和摘要行。例如：

```
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)
```

- 不同类型的 compute 子句的输出和分组为：

子句和分组	输出	示例
一个 compute 子句，相同函数	一个明细行	1, 2, 4, 6, 7
一个 compute 子句，不同函数	每种函数一个明细行	3
多个 compute 子句，相同的分组列	每个 compute 子句一个明细行；明细行全部在输出中	与一个有不同函数的 compute 子句的结果相同
多个 compute 子句；不同的分组列	每个 compute 子句一个明细行；明细行的位置取决于分组	5

区分大小写

如果您的服务器安装了不区分大小写的排序顺序， compute 会忽略所指定列中的数据的大小写。例如，给定数据：

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

对 lname 执行 compute by 会产生以下结果：

```
select lname, amount from groupdemo
order by lname
compute sum (amount) by lname
lname      amount
---------- ------------------------
Levi                           9.00

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

lname      amount
---------- ------------------------
Lé vi                         20.00

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

lname      amount
---------- ------------------------
smith                          5.00
```

```
SMITH                          7.00
Smith                         10.00

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

相同的查询在不区分大小写和变音的服务器上产生以下结果：

```
lname      amount
---------- ------------------------
Levi                           9.00
Lé vi                         20.00

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

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

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

标准　符合 ANSI SQL 的级别：Transact-SQL 扩展。

另请参见　**命令** group by 和 having 子句, select.

函数 avg, count, max, min, sum.

connect to...disconnect

说明

（仅限组件集成服务）连接到指定服务器和断开已连接的服务器。

语法

此语法将被一字不差地发送到 Adaptive Server。将此语法与 CIS 一起使用，可创建一个至另一服务器的直通连接：

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

此语法打开一个与 Adaptive Server 的新的 JDBC 级连接，且不使用 CIS。可按任意顺序指定参数。如果不包括参数， Adaptive Server 会提示您提供连接参数：

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

此语法打开一个与 Adaptive Server 的新的 JDBC 级连接。此语法不使用 CIS：

```
connect using connect_string
```

参数

server_name
是一个要求直通连接的服务器。

from ASE
断开与当前 Adaptive Server 的连接。

all
断开与所有 Adaptive Server 的连接。

connection_name
断开指定的连接。

engine_name
连接到指定的引擎。

database_name
连接到指定的数据库。

connection_name
连接到已配置的连接。

user_id
连接到具有该 ID 的用户。

connection_string
使用预先确定的连接字符串进行连接。

示例

示例 1 建立到名为 SYBASE 的服务器的直通连接：

```
connect to SYBASE
```

示例 2 断开连接的服务器：

```
disconnect
```

示例 3 断开与所有服务器的连接：

```
disconnect all
```

用法

- connect to 用于指定直通连接所需的服务器。采用直通模式可在远程服务器上执行本地操作。
- *server_name* 必须是 sysservers 表中的服务器名，并且定义了其服务器类和网络名。
- 代表用户与 *server_name* 建立连接时， CIS 使用下列标识符之一：
  - sysattributes 中描述的远程登录别名 （如果存在）
  - 用户名和口令

  任一情况下，如果无法连接到指定的服务器， Adaptive Server 都将返回错误消息。
- 完成直通连接后， CIS 在接收后续语言文本时会绕过 Transact-SQL 语法分析程序和编译器。它将语句直接传递到指定服务器，并将结果转换为 Open Client 接口可以识别的形式，然后返回给客户程序。
- 要关闭 connect to 命令创建的连接，请使用 disconnect 命令。只有在使用 connect to 建立连接后，才能使用此命令。
- 可以将 disconnect 命令缩写为 disc。
- 除非先前已发出 connect to 命令，而且服务器连接到远程服务器，否则 disconnect 命令会返回错误。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

必须具有 connect 特权才能使用 connect to 命令。

审计　　sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
90	security	connect to	• *角色* – 当前活动角色 • *关键字或选项* – connect to • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见　　**命令** create existing table, grant.

系统过程 sp_addserver, sp_autoconnect, sp_helpserver, sp_passthru, sp_remotesql, sp_serveroption.

continue

说明

重新启动 while 循环。 continue 经常通过 if 测试激活。

语法

```
while boolean_expression
        statement
    break
        statement
continue
```

示例

如果平均价格低于 $30，则将价格翻番。然后选择最高价格。如果最高价格小于或等于 $50，则重新启动 while 循环并且再次将价格翻番。如果最高价格高于 $50，则退出 while 循环并且输出消息：

```
while (select avg (price) from titles) < $30
begin
  update titles
  set price = price * 2
  select max (price) from titles

  if (select max (price) from titles) > $50
     break
  else
     continue
end

begin
print "Too much for the market to bear"
end
```

用法

continue 会重新启动 while 循环，并跳过 continue 之后的任何语句。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 continue 无需任何权限。

另请参见

命令 break, while

create archive database

说明

创建存档数据库。

语法

```
create archive database db_name
        [on db_device [= size]
            [, db_device [= size] ] ...]
        with scratch_database = db_name
```

参数

on

指定修改的页面区域。 Adaptive Server 需要使用传统数据库存储来存储修改的页面。使用 on 子句来指定修改页面区域的位置和大小。

db device

指定您要在其上创建修改页面区域的数据库设备。

size

指定您要创建的修改页面区域的大小。若忽略 *size*，则会分配 5120 页。

with scratch_database

（如果尚不存在空数据库，则为必需）指定要在其中维护存档数据库信息的现有数据库的名称。 sysaltusages 系统表将存档数据库中的逻辑页映射到物理页上，该表存储在空数据库中。

示例

这是典型的存档数据库命令序列。

1 根据需要创建空数据库：

```
create database scratchdb
    on datadev1 = 100
    log on logdev1 = 50
```

这将创建一个名为 scratchdb 的 150MB 的传统数据库。

2 将刚创建的数据库指定为空数据库：

```
sp_dboption "scratchdb", "scratch database", "true"
```

3 创建存档数据库：

```
create archive database archivedb
    on datadev2 = 20
    with scratch_database = scratchdb
```

此命令会创建一个名为 archivedb 的存档数据库，此数据库带有 20MB 的修改页面区域。

4 实现存档数据库：

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

5　使数据库联机：

```
online database archivedb
```

6　使用 dbcc 命令检查存档数据库的一致性。例如：

```
dbcc checkdb(archivedb)
```

7　装载事务日志转储，并从存档数据库中恢复对象：

```
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
```

用法

- 可以将 master 数据库的转储装载到存档数据库中。
- 不能将内存数据库用作存档数据库。 Sybase 建议您不要将内存数据库用作空数据库。

权限

对 create archive database 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 create database 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须为系统管理员或具有 create database 特权。

create database

说明

创建新数据库。

语法

用于非集群环境的语法：

```
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}]
```

用于集群环境的语法：

```
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}]
```

参数

temporary

指示您正在创建一个临时数据库。

inmemory

内存数据库的必选参数。

database_name

是新数据库的名称，必须符合标识符规则，并且不能是变量。

on

指示数据库的位置和大小。

default

指示 create database 可将新数据库放到任何缺省数据库设备上，如 sysdevices.status 中所示。若要指定数据库的大小而不指定位置，请使用：

```
on default = size
```

若要将数据库设备的状态更改为 “default”，请使用 sp_diskdefault。

database_device

是用于定位数据库的设备的逻辑名。一个数据库可以在每个不同的数据库设备上占用不同的空间量。若要在 Adaptive Server 中添加数据库设备，请使用 disk init。

size

是分配给数据库扩展的空间量。可以使用以下单位指示符，并交替使用大写、小写、单引号和双引号：“k”或“K”（千字节）、“m”或“M”（兆字节）、“g”或“G”（千兆字节）以及“t”或“T”（兆兆字节）。 Sybase 建议始终包括单位指示符。如果不包括单位指示符，则引号是可选的。不过，如果包括单位指示符，则必须使用引号。如果不提供单位指示符，则假定提供的值的单位为兆字节。

log on

指定数据库日志设备的逻辑名。您可以在 log on 子句中指定多个设备。

with

可按任意顺序指定。在使用 with 子句时，必须至少指定下列选项之一：

- with dbid = *number* – 为新数据库指定 dbid。如果不显式指定 dbid，服务器将指派一个未使用的 dbid。
- with default_location – 指定新表的存储位置。如果还指定了 for proxy_update 子句，则会从指定位置为每个远程表或视图自动创建一个代理表。
- with override – 强制 Adaptive Server 接受设备规范，即使同一设备上混合有数据和事务日志也是如此，因此会危及数据库的最新数据的可恢复性。如果试图不使用此子句在同一设备上混合日志和数据，则 create database 命令会失败。如果混合了日志和数据并且使用 with override，将会收到警告，但命令会成功执行。

durability =

确定数据库的持久性级别：

- full – 将所有事务都写入磁盘。如果创建数据库时不指定持久性级别，此值将为缺省值，以确保从服务器故障中完全恢复。所有系统数据库都使用此持久性级别 （用于磁盘驻留式数据库的传统持久性级别）。
- no_recovery – 事务不在磁盘中持久存在，如果服务器出现故障或关闭，则所有更改都将丢失。对于基于磁盘的数据库， Adaptive Server 会在运行时以不可控方式定期向磁盘设备中写入数据。用 no_recovery 创建的数据库以任意方式 （正常、非正常或服务器故障及重新启动）关闭后均无法恢复，但可以基于 model 或模板（如果已定义）数据库重新创建该数据库。
- at_shutdown – 在服务器运行时以及正常关闭后，事务是持久性的。如果服务器出现故障，所有持久性都将丢失。

[no] async_init

启用或禁用异步数据库初始化。

compression

指示要对新创建的表或分区应用的压缩级别：

- none – 不压缩数据。
- row – 压缩单个行中的一个或多个数据项。只有在压缩形式比非压缩形式更节省空间时， Adaptive Server 才会用 row 压缩形式存储数据。
- page – 当页面已满时，则会使用页级压缩来压缩现有的采用行压缩形式的数据行，从而创建页级字典、索引和字符编码条目。

  Adaptive Server 只有在已经在行级压缩数据后才会在页级压缩数据，因此，将压缩设置为 page 意味着 page 和 row 压缩。

lob_compression = off | *compression_level*

确定新创建的表的压缩级别。选择 off 意味着表不使用 LOB 压缩。

压缩算法忽略不使用 LOB 数据的行。

表压缩级别。压缩级别包括：

- 0 – 不压缩行。
- 1 到 9 – Adaptive Server 使用 ZLib 压缩。通常，压缩级别数字越大，Adaptive Server 压缩 LOB 数据的程度就更大，压缩和非压缩数据之间的比率就越大 （也就是说，压缩数据与非压缩数据大小相比，节省的空间量就越大 （以字节为单位））。

  但是，压缩量取决于 LOB 内容，压缩级别越高，进程的 CPU 占用率就越高。也就是说，级别 9 提供最高压缩率，但其 CPU 使用率也最高。
- 100 – Adaptive Server 使用 FastLZ 压缩。此压缩率的 CPU 占用率最低；通常在数据较少时使用。
- 101 – Adaptive Server 使用 FastLZ 压缩。值 101 使用的 CPU 比值 100 略多，但压缩率比值 100 高。

inrow_lob_length = *value*

指定字节数。 inrow_lob_length 的有效值范围是 0 到数据库的逻辑页大小。值为 0 会在数据库范围关闭 LOB 指定，而且，所有不带特定 in row 子句的 LOB 列都将创建为行外 LOB 列。

dml_logging

指定 DML 操作的日志记录级别。

full | minimal

指定 DML 操作的完全日志记录或最少日志记录。

for load

调用只能用于装载数据库转储的 create database 命令的改进版本。请参见第 113 页的 “使用 for load 选项”。

for proxy_update

从远程位置自动获取元数据并且创建代理表。除非也指定了 with default_location，否则不能使用 for proxy_update。

global temporary

指示您正在创建一个全局临时数据库。

system temporary

指示您正在创建一个本地系统临时数据库。

temporary
指示您正在创建一个临时数据库。

for instance *instance_name*
指定将拥有您正在创建的本地系统临时数据库或本地临时数据库的实例。在创建全局临时数据库时，不使用此参数。

注释 您必须从将拥有该数据库的实例中创建本地用户临时数据库。您可以从任何实例中创建本地系统临时数据库。

示例

示例 1 创建名为 pubs 的数据库：

```
create database pubs
```

示例 2 创建一个名为 pubs 的 4MB 的数据库：

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

如果不为 *size* 提供单位指示符，则假定为 pubs 提供的值的单位为兆字节。

示例 3 创建一个名为 pubs 的数据库并且在 datadev 设备上有 3MB，在 moredatadev 设备上有 2MB 空间：

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

示例 4 创建一个名为 pubs 的数据库并且在 datadev 设备上有 3MB 的数据，在 logdev 设备上有 0.5GB 的日志：

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

示例 5 创建名为 proxydb 的代理数据库但不自动创建代理表：

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

示例 6 创建名为 proxydb 的代理数据库并且自动创建代理表：

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

示例 7 创建名为 proxydb 的代理数据库，并且从一个远程数据库检索所有远程表的元数据：

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

示例 8 创建一个名为 pubs、 dbid 为 15 的数据库：

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

示例 9 创建一个名为 mytempdb1 的临时数据库，它在 datadev 设备上有 3MB 的数据，在 logdev 设备上有 1MB 的日志：

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

示例 10 在集群环境中，在 “ase1” 上创建一个本地用户临时数据库。从所有者实例 （“ase1”）中执行以下命令：

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

或：

```
create temporary database local_tempdb1
```

示例 11 在集群环境中，在 “ase1” 上创建一个本地系统临时数据库。从集群中的任何实例执行此命令：

```
create system temporary database local_systempdb1 for
instance ase1
```

示例 12 在集群环境中，创建一个全局临时数据库：

```
create global temporary database global_tempdb1
```

示例 13 在两个不同的内存存储设备 （用于数据的 imdb_data_dev1 和用于日志的 imdb_logdev）上创建一个内存数据库：

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

示例 14 在多个内存-存储设备上创建一个内-存数据库。 imdb_data_dev1 和 imdb_data_dev2 包含所有数据， inmem_logdev 包含日志：

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

示例 15 使用 pubs2 数据库作为模板创建 pubs5 数据库：

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

示例 16 创建一个名为 pubs5_rddb 的宽松持久性数据库：

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

示例 17 创建一个供内存临时数据库专用的内存存储高速缓存：

1 创建内存存储高速缓存：

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

2 创建内存设备以创建临时数据库：

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

3 创建内存数据库：

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

示例 18 在现有磁盘设备上创建一个临时数据库，持久性设置为 no_recovery：

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

示例 19 创建 emaildb 数据库，并为其配置页级压缩：

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

示例 20 创建 email_lob_db 数据库，并为其配置 101 LOB 压缩级：

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

示例 21 创建一个名为 pubs 的数据库，允许长度为 300 字节的行内 LOB 数据：

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

用法

- 从 master 数据库使用 create database。
- 您可以将 *size* 指定为 float 数据类型，但它会向下取整为分配单元的整数倍。
- 如果不显式声明数据库的大小，则其大小由 model 数据库的大小决定。能创建的数据库的最小大小为四个分配单元。
- 因为 Adaptive Server 以包含 256 个逻辑页的块为单位为 create database 和 alter database 分配数据库空间，这些命令会将指定大小向下取整为最接近的分配单元的整数倍。
- 如果不包括单位指示符，Adaptive Server 将按照磁盘空间使用的兆字节单位来解释大小，此数值会被转换为服务器使用的逻辑页大小。
- 如果不为数据库指定位置和大小，则缺省位置为 master..sysdevices 中指示的任何缺省数据库设备。缺省大小是 sysconfigures 中的参数 default database size 与 model 数据库大小二者之中的较大值。

  系统管理员可以使用 sp_configure 更改 default database size 的值并重新启动 Adaptive Server 来增加缺省大小。default database size 参数的值必须至少与 model 数据库的大小一样大。如果增加 model 数据库的大小，则还必须增加缺省大小。

  如果 Adaptive Server 不能在您要求的位置提供您需要的空间大小，它会在每个设备的基础上分配尽可能相近的空间，并显示一条消息告诉您分配的空间大小和分配的位置。数据库的最大大小取决于系统。
- 如果使用下面的语句创建代理数据库：

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

  设备名的存在足以绕过大小计算，并且如果缺省数据库的大小（model 数据库的大小）不足以容纳所有的代理表，则此命令可能会失败。

  要想允许 CIS 估计数据库大小，则不要在该命令中包括任何设备名或其它选项：

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

限制

- Adaptive Server 可以管理多达 32,767 个数据库。
- dbid 应始终大于零，小于最大 dbid 32,767。
- Adaptive Server 一次只能创建一个数据库。如果两个数据库创建请求发生冲突，一位用户会看到以下消息：

  ```
  model database in use:cannot create new database
  ```

- 每次您在数据库设备上用 create database 或 alter database 分配空间时，分配代表一个设备段，并且分配是作为一行在 sysusages 中输入的。
- 数据库最多可有 32 个命名段。段是特定 Adaptive Server 的可用数据库设备的命名子集。有关段的详细信息，请参见 《系统管理指南》。

临时数据库

- 不能将 with default_location 或 for proxy_update 参数和 create temporary database 命令一起使用。这样做会生成错误，如以下两个示例：

```
1> create temporary database tb1 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 tb1 with default_location
     for proxy_update

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

- 数据库的临时状态是在创建临时数据库时设置的，由 sysdatabases 条目的 status3 字段的值 0x00000100 （十进制 256）表示。
- 除了从 model 继承来的所有选项，临时数据库 （如系统 tempdb）有以下的数据库选项集：
  - select into/bulkcopy
  - trunc log on chkpt
- 对于系统 tempdb， guest 用户被添加到临时数据库中，并且将 create table 权限授予给 PUBLIC。
- 由于服务器每次重新启动时都会重新创建临时数据库，因此在临时数据库创建过程中不会清除未用页。

创建压缩数据库

- create table . . .with compression 的压缩设置会覆盖 create database 压缩设置。
- 用 select into 创建的临时数据库不会继承数据库中的压缩级别。
- model 数据库中的缺省压缩设置是 none （对所有基于 model 数据库都关闭数据压缩）。
- 当您在 tempdb 或其它临时数据库中启用数据压缩后，在某个会话中或在存储过程内创建的临时表不会继承 tempdb 的压缩级别。
- 任务绑定到的数据库决定它所创建的临时数据库的压缩级别。

创建具有行内 LOB 的数据库

- 行内大小可以和数据库中允许的最大行大小一样大。 Adaptive Server 降低了插入和更新期间各个列的行内 LOB 存储大小限制，具体依据是单个页中的可用存储空间减去所有页或行开销。
- 当您在已经指定了数据库范围的有效行内 LOB 长度的数据库中创建表时，表中的所有 LOB 列都将被创建为行内，除非您在列定义的语法中指定了 off row。列的行内长度 （以字节为单位）是由该数据库范围的设置指定的。
- inrow_lob_length 设置的缺省值是 0 字节，当您升级到 Adaptive Server 15.7 版后，该值不会导致现有数据库发生任何行为变化。更改该缺省值能让对行内存储有着不同要求的应用程序控制在行内存储多少 LOB 数据。
- 数据库范围的设置仅适用于新创建的表或在应用或更改数据库范围的设置后添加到现有表中的 LOB 列。最初创建表时每个 LOB 列继承的行内 LOB 长度保持不变，即使数据库范围的设置发生了变化也是如此。要更改各个 LOB 列的行内长度或行内属性，请使用 alter table modify column。
- 若要更改缺省长度，请使用 alter database 的 inrow_lob_length 参数。

创建内存数据库和宽松持久性数据库

- 为内存数据库列出的 *database_device* 必须为内存存储设备。
- for load 参数指示最初创建数据库时，数据库处于等待使用 load database 命令装载的状态。
- 您不能：
  - 在缺省设备上创建内存数据库。
  - 将内存数据库用于系统数据库 （tempdb 除外）。

  - 将内存数据库用作存档数据库。 Sybase 建议您不要将内存数据库用作空数据库。
  - 对您创建的数据库及其模板数据库使用同一名称。
  - 将系统数据库 （包括 model）指定为模板数据库。
  - 将基于磁盘和基于高速缓存的存储设备混合。 Adaptive Server 将完全在内存存储高速缓存中创建的数据库视为内存数据库。您不能：
    - 将在不同内存存储高速缓存中创建的内存存储设备用于一个内存数据库
    - 将在一个内存存储高速缓存中创建的 （完全或部分）内存存储设备用于不同的内存数据库
  - 将 use as template 参数与 with default_location = 参数一起使用。
  - 将 for load 和 for proxy update 参数与 use as template 参数一起使用。
- 您可以：
  - 在驻留于指定高速缓存上的内存存储设备中创建内存数据库，前提是所有内存存储设备都由内存存储高速缓存托管。
  - 在同一内存存储设备中创建混合日志和数据的内存数据库。但是，混合日志和数据的内存数据库必须位于单个高速缓存中。
- 创建内存数据库时需要使用 durability = no_recovery 参数。此参数可加强在重新启动服务器时始终重新创建内存数据库的行为。

从 *model* 创建的新数据库

- Adaptive Server 通过复制 model 数据库创建新的数据库。
- 可以通过添加表、存储过程、用户定义数据类型和其它对象以及更改数据库选项设置来自定义 model。新的数据库从 model 继承这些对象和设置。
- 为了保证可恢复性，create database 必须清除在复制 model 数据库时没有初始化的每一页。这可能会需要几分钟，取决于数据库的大小和您系统的速度。

  如果正在创建一个数据库来装载数据库转储，可以用 for load 选项跳过页清除步骤。这样会极大地加快数据库创建过程。

确保数据库可恢复性

- 每次创建新数据库时，备份 master 数据库。如果 master 损坏，这样做可使恢复过程更加容易和安全。

  注释　注意：如果您创建一个数据库但 master 备份失败，则可以用 disk reinit 来恢复所做的更改。

- with override 子句允许您在单个设备上混合日志和数据段。然而，为了完全恢复，该设备和 log on 中指定的多个设备应该与存储数据的物理设备不同。在硬盘发生故障的情况下，可以从数据库转储和事务日志恢复数据库。

  您可以在用于存储事务日志和数据的单个设备上创建小数据库，但是您*必须*依赖 dump database 命令来备份。

- 事务日志所需的设备大小因更新活动量和事务日志转储频率不同而有所不同。作为经验法则，一般为日志设备分配的空间应为数据库空间的 10-25 %。最好从小的值开始，因为给事务日志设备分配的空间不能回收并且不能用来存储数据。

使用 *for load* 选项

如果还没有使用 sp_addsegment 添加到数据库，您可以使用 for load 选项从介质故障中恢复或是将数据库从一台计算机移到另一台计算机上。使用 alter database for load 创建一个新数据库，该数据库与从中创建了要装载的数据库转储的数据库结构相同。有关将转储装载到新数据库时重复进行空间分配的讨论，请参见 《系统管理指南》。

- 当使用 for load 选项创建数据库时，装载数据库转储之前在新数据库中只能运行如下命令：
  - alter database for load
  - drop database
  - load database

  在将数据库转储装载到新数据库之后，您还可以在数据库中使用一些 dbcc 诊断命令。发出 online database 命令之后，对您可以使用哪些命令没有限制。

- 使用 for load 选项创建的数据库在 sp_helpdb 输出中有状态 “don’ t recover”。

获取有关数据库的信息

- 若要获取有关数据库的报告，请执行 sp_helpdb。
- 若要获取数据库使用空间的报告，请使用 sp_spaceused。

使用 *with default_location* 和 *for proxy_update*

如果无 for proxy_update 子句，则 with default_location 子句的行为与 sp_defaultloc 所提供的行为相同 建立缺省的存储位置来创建新表和现有表，但在处理 create database 的过程中不会自动导入代理表定义。

- 如果指定 for proxy_update 时不指定 default_location，则报告错误。
- 使用 for proxy_update 选项创建代理数据库后，将调用 CIS 来执行以下操作：
  - 提供包含所有代理表所需的数据库大小估计值，代理表表示在主服务器的数据库中找到的实际表和视图。此估计值是包含所有代理表和索引所需的数据库页数。如果未指定大小和数据库设备，则使用此估计值。
  - 创建表示在协同服务器的数据库中找到的实际表和视图的所有代理表。
  - 将代理表的所有权限授予 public。
  - 在代理数据库中添加 guest 用户。
  - 设置数据库的状态以指示该数据库 “Is_A_Proxy”。该状态包含在 master.dbo.sysdatabases.status3 中。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 create database 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 create database 特权。如果要创建 sybsecurity 数据库，您必须具有 manage auditing 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须为系统管理员或具有 create database 特权。如果要创建 sybsecurity 数据库，则您必须是系统安全员。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
9	create	create database	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter database, disk init, drop database, dump database, load database, online database.

系统过程 sp_changedbowner, sp_diskdefault, sp_helpdb, sp_logdevice, sp_renamedb, sp_spaceused.

create default

说明

如果插入时没有显式地提供要插入列 （或用户定义的数据类型的所有列）中的值，则指定一个值。

语法

create default [*owner.*]*default_name*
as *constant_expression*

参数

default_name

是缺省值名称，必须符合标识符规则，并且不能是变量。指定所有者的名称，以创建由当前数据库中的其他用户拥有的另一个同名缺省值。 *owner* 的缺省值是当前用户。

constant_expression

是不包括任何列或其它数据库对象的名称的表达式。它可以包括全局变量和不引用数据库对象的内置函数。用引号将字符和日期常量括起来并使用 “0x”作为二进制常量的前缀。

示例

示例 1 创建一个名为 D1 并使用 @@*spid* 全局变量的缺省值：

```
create default D1 as @@spid
```

示例 2 定义一个缺省值，然后将其绑定到适当的列或用户定义数据类型：

```
create default phonedflt as "UNKNOWN"

sp_bindefault phonedflt, "authors.phone"
```

缺省值只有在 authors 表的 phone 列中没有条目时起作用。无条目与空值条目是不同的。若要获取缺省值，请对列列表 （不含具有缺省值的列）发出 insert 命令。

示例 3 创建缺省值 todays_date，它将当前日期插入到缺省值要绑定的列：

```
create default todays_date as getdate ()
```

用法

- 使用 sp_bindefault 将缺省值绑定到列或用户定义的数据类型 - 但不能绑定到 Adaptive Server 提供的数据类型。
- 可以在不解除绑定旧缺省值的情况下将新的缺省值绑定到数据类型。新的缺省值会覆盖并解除绑定旧的缺省值。
- create default 在创建缺省值之前会针对检查约束执行错误检查。
- 若要隐藏缺省值的源文本，请使用 sp_hidetext。

限制

- 只能在当前数据库中创建缺省值。
- 不能将 create default 语句与其它语句组合在同一批处理中。

- 在创建一个同名的新缺省值之前，必须使用 drop default 删除旧缺省值；删除缺省值之前，必须使用 sp_unbindefault 解除绑定该缺省值。

数据类型兼容性

- 当试图向列中插入与列的数据类型不兼容的缺省值时，Adaptive Server 会产生一条错误消息。例如，如果将一个字符表达式（如 “N/A”）绑定到一个 integer 列，任何没有指定列值的 insert 都会失败。
- 如果缺省值对字符列来说太长，Adaptive Server 会根据 string_rtruncation 选项的设置来截断字符串或者产生例外。请参见 set 命令。

获取有关缺省值的信息

- 缺省值定义存储在 syscomments 中。
- 将缺省值绑定到列之后，它的对象 ID 会存储于 syscolumns 中。将缺省值绑定到用户定义数据类型之后，它的对象 ID 存储在 systypes 中。
- 若要重命名缺省值，请使用 sp_rename。
- 若要获取缺省值文本的报告，请使用 sp_helptext。

缺省值和规则

如果列既有缺省值又有规则与其关联，则缺省值一定不能违反规则。不能插入与规则冲突的缺省值。每次插入这种缺省值时， Adaptive Server 都会产生一个错误消息。

缺省值和空值

如果一列不允许为空，并且没有为该列创建缺省值，那么当用户试图插入一行但该行不包括此列值时，插入失败，同时 Adaptive Server 产生错误消息。

表 1-3 说明缺省值的存在和 NULL 或 NOT NULL 列定义之间的关系。

表 1-3: 空值与列缺省值之间的关系

列空值类型	无输入，无缺省值	无输入，有缺省值	输入为空，无缺省值	输入为空，存在缺省值
NULL	插入空值	插入缺省值	插入空值	插入空值
NOT NULL	错误，命令失败	插入缺省值	错误，命令失败	错误，命令失败

在 *create table* 中指定缺省值

可以使用 create table 语句的 default 子句定义列缺省值作为使用 create default 的替代方法。但是，这些列缺省值是专用于此表的；不能将它们绑定到其它表。有关完整性约束的信息，请参见 create table 和 alter table。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

使用 create table 语句的 default 子句创建符合 ANSI SQL 的缺省值。

权限

对 create default 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 create default 特权。要为另一用户创建缺省值，您必须具有 create any default 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者，或者是具有 sa_role 或 create default 特权的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
14	create	create default	• *角色* – 当前活动角色 • *关键字或选项* – NULL • *先前值* – NULL • *其它信息* – NULL • *当前值* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 alter table, create rule, create table, drop default, drop rule.

系统过程 sp_bindefault, sp_help, sp_helptext, sp_rename, sp_unbindefault.

create encryption key

说明

创建加密密钥。所有与密钥和加密相关的信息都由 create encryption key 封装，从而允许您在加密过程中指定加密算法和密钥大小、密钥的缺省属性、用于加密密钥的可选的用户指定口令，以及是使用初始化矢量还是填充。

Adaptive Server 使用 Security Builder Crypto 来生成密钥并进行加密。

语法

创建主密钥：

```
create encryption key [dual] master
    [for AES] with passwd char_literal
```

创建服务密钥：

```
create encryption key syb_extpasswdkey
    [ with { static key | master key }]
create encryption key syb_syscommkey
        [ with { static key | master key }]
```

创建列加密密钥：

```
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]}]
```

参数

keyname

在当前数据库的用户表、视图和过程名称空间中必须是唯一的。如果密钥位于其它数据库中，需指定 *database* 名称；如果您是为其他用户创建密钥，则需指定 *owner* 名称。*owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。只有系统安全员才能为其他用户创建密钥。

as default

允许系统安全员或密钥管理者创建用于加密的数据库缺省密钥。由于存在数据库缺省加密密钥，表创建者无需在 create table、alter table 和 select into 中使用 keyname 便可指定加密。Adaptive Server 使用同一数据库中的缺省密钥。可以更改缺省密钥。请参见第 13 页的 “alter encryption key”。

for *algorithm_name*

指定您正在使用的算法。高级加密标准 (AES) 是唯一受支持的算法。AES 支持的密钥大小为 128 位、 192 位和 256 位，支持的块大小为 16 字节。

for AES

使用高级加密标准 (AES) 加密算法来加密数据。

syb_extpasswdkey | syb_syscommkey

- syb_extpasswdkey – sysattributes 中的所有外部口令都重新用使用强加密的新密钥加密
- syb_syscommkey – sp_hidetext 的所有后续执行都使用具有强加密的新密钥。必须在现有数据库对象上执行 sp_hidetext，该对象才能用新密钥加密

static key | master key

指示您正在使用静态密钥或主密钥创建加密密钥。

keylength *num_bits*

要创建的密钥的大小（以位为单位）。对于 AES，有效的密钥长度为 128、 192 和 256 位。缺省密钥长度为 128 位。

password_phrase

是带引号的字母数字字符串，其长度最多为 255 个字节，Adaptive Server 将使用它来生成用于对列加密密钥进行加密的密钥 （密钥加密密钥）。

init_vector random

指定在加密过程中使用初始化矢量。当加密算法使用初始化矢量时，两段相同纯文本的密文是不同的，这样可防止 cryptanalyst 检测数据模式。使用初始化矢量可提高数据的安全性。

初始化矢量对性能还有一些影响。只能对其加密密钥未指定初始化矢量的列执行索引创建、优化连接和搜索。

缺省设置为使用初始化矢量 （即 init_vector random）。使用初始化矢量意味着使用加密的密码块链 (CBC) 模式；设置 init_vector null 则意味着将使用电码本 (ECB) 模式。

init_vector null

在加密时不使用初始化矢量。这样可使列支持索引。

pad null

缺省值，它将省略数据随机填充。如果列必须支持索引，则不能使用填充。

pad random

在加密前，自动使用随机字节来填充数据。通过使用填充代替初始化矢量，可以使密文随机化。填充仅适用于明文长度少于块长度一半的列。对于 AES 算法，块长度为 16 个字节。

[no] dual_control

指示是否使用双控制来创建主密钥。

示例

示例 1 指定一个名为 “safe_key”的 256 位密钥作为数据库缺省密钥。系统安全员输入：

```
create encryption key safe_key as default for AES with keylength 256
```

示例 2 创建一个名为 “salary_key”的 128 位密钥，以使用随机填充来加密列：

```
create encryption key salary_key for AES with init_vector null pad random
```

示例 3 创建一个名为 “mykey”的 192 位密钥，以使用初始化矢量来加密列：

```
create encryption key mykey for AES with keylength 192 init_vector random
```

示例 4 创建一个受用户指定的口令保护的密钥：

```
create encryption key key1 with passwd 'Worlds1Biggest6Secret'
```

必须输入用于保护密钥的用户指定口令，才能访问由该密钥加密的列。请参见第 589 页的 “set”。

示例 5 指定一个名为 “safe_key”的 256 位密钥作为数据库缺省密钥。由于该密钥不指定口令，因此， Adaptive Server 使用数据库级的主密钥作为 safe_key 的密钥加密密钥。如果没有主密钥， Adaptive Server 会使用系统加密口令：

```
create encryption key safe_key as default for AES with keylength 256
```

示例 6 使用主密钥和 “Whybother”的组合来加密 CEK k3：

```
create encryption key k3 with passwd 'Whybother' dual_control
create encryption key k1 with keylength 192
```

用法

Adaptive Server 不会保存用户指定的口令。它会将称为 “salt”的一组验证字节保存在 sysencryptkeys.eksalt 中，这使 Adaptive Server 能够识别在后续加密或解密操作中所使用的口令对密钥是否合法。您必须向 Adaptive Server 提供口令，然后才能访问由 keyname 加密的任何列。

权限

对 create encryption key 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有以下特权或基于加密密钥类型的特权： • 列加密密钥 – create encryption key 或 manage column encryption key • 主密钥 – manage master key • 服务密钥 – manage service key
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sso_role、keycustodian_role 的用户或具有 create encryption key 特权。

另请参见

命令 alter encryption key, drop encryption key, grant, revoke.

文档 有关审计的信息，请参见 《加密列用户指南》中的 “审计加密列”。

create existing table

说明

（仅限组件集成服务）创建代理表，然后检索和存储远程表中的元数据，并将数据放到该代理表中。允许将代理表映射到远程位置上的表、视图或过程上。

创建代理表的首选方法是使用 create proxy_table 命令，此命令无需定义列定义。

语法

create existing table *table_name* (*column_list*)
[on *segment_name*]
[[external {table | procedure | file | *connection_type*}] at *pathname*
[column delimiter "*string*"]]

参数

table_name
指定要为其创建代理表的表的名称。

column_list
指定存储远程表的有关信息的列的列表的名称。

on *segment_name*
指定包含远程表的段。

external
指定对象是远程对象。

table
指定远程对象是表或视图。缺省为 external table。

procedure
指定远程对象是存储过程。

file
指定远程对象是文件。

connection_type
确定远程过程调用使用当前连接还是单独连接。有效值为：

- non_transactional – 用于执行 RPC 的单独连接。
- transactional – 是用于执行 RPC 的现有连接。

缺省行为是事务型行为。

at *pathname*

指定远程对象的位置。*pathname* 的形式如下：*server_name.dbname.owner.object*，其中：

- *server_name* （必需） - 是包含远程对象的服务器的名称。
- *dbname* （可选）是包含此对象的远程服务器所管理的数据库的名称。
- *owner* （可选）是拥有远程对象的远程服务器用户的名称。
- *object* （必需） – 是远程表、视图或过程的名称。

column delimiter

用于在访问平面文件时分隔每个记录中的字段。列分隔符的最大长度可以为 16 个字节。

string

列分隔符字符串可以是任何字符序列，但如果字符串的长度超出了 16 个字节，则只使用前 16 个字节。对映射到除文件之外的任何对象的代理表使用列分隔符会导致语法错误。

示例

示例 1 创建代理表 authors：

```
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"
```

示例 2 创建代理表 syb_columns：

```
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 "remote1.master.dbo.columns"
```

示例 3 为远程服务器 SERVER_A 上的 blurbs 表创建名为 blurbs 的代理表：

```
create existing table blurbs
(
author_id      id      not null,
copy           text    not null
)
at "SERVER_A.db1.joe.blurbs"
```

示例 4 为名为 p1 的远程过程创建名为 rpc1 的代理表：

```
create existing table rpc1
(
column_1       int,
column_2       int
)
external procedure
at "SERVER_A.db1.joe.p1"
```

用法

- 除非远程对象是文件，否则 create existing table 将不创建新表。而是由 CIS 检查表映射，以确认 *column_list* 中的信息与远程表匹配，检验是否存在基础对象，并检索和存储有关该远程表的元数据。
- 如果主机数据文件或远程服务器对象不存在，该命令将被拒绝，并给出错误消息。
- 如果对象存在，将更新系统表 sysobjects、syscolumns 和 sysindexes。检验操作需要如下步骤：
  a　确定现有对象的性质。对于主机数据文件，这需要确定文件结构和记录格式。对于远程服务器对象，这需要确定对象是否为表、视图或 RPC。
  b　对于远程服务器对象 （不包括 RPC），将获得的表或视图的列属性与 column_list 中定义的属性相比较。

c 提取主机数据文件或远程服务器表中的索引信息，并将其用于创建系统表 sysindexes 的行。这将用 Adaptive Server 的术语定义索引和键，并启用查询优化程序来考虑表中任何可能存在的索引。

- on *segment_name* 子句在本地处理且不传递给远程服务器。
- 成功定义现有表后，对该表发出 update statistics。这将允许查询优化程序在考虑选择索引和连接顺序时做出正确选择。
- CIS 允许创建带有一个定义为 NOT NULL 的列的代理表，即使远程列被定义为 NULL。会显示一个警告，通知您不匹配。
- at 关键字所提供的位置信息与 sp_addobjectdef 所提供的信息相同。该信息存储在 sysattributes 表中。
- CIS 在 systabstats 目录中为远程表的每个索引插入或更新一条记录。由于详细的结构统计信息与远程索引毫不相关，因此在 systabstats 记录中只设置最少数量的列：id、 indid 和 rowcnt。
- 外部文件的数据类型不能是 text, image 或 Java ADT。

数据类型转换

- 使用 create existing table 时，必须使用所识别的 Adaptive Server 数据类型指定所有数据类型。如果远程服务器表驻留在异构服务器类中，则在检索数据时，远程表的数据类型将自动转换成指定的 Adaptive Server 类型。如果不能进行转换，则 CIS 不允许定义表。
- 《组件集成服务用户指南》包含对应于每个支持的服务器类的节，并标识 CIS 隐式执行的所有可能的数据类型转换。

服务器类进行的更改

- 所有服务器类都允许指定比远程服务器上的表中所含的列更少的列。
- 所有服务器类都按名称匹配列。
- 所有服务器类都允许列类型为可以转换成远程表中的列的类型的任何类型或可以从远程表中的列的类型转换成的任何类型。

远程过程

- 当代理表是过程类型的表时，必须提供与远程过程结果集的说明相匹配的列列表。 create existing table *并不*检验该列列表的精确性。
- 未为过程创建索引。
- CIS 将远程过程的结果集视为可以存储、可与其它表连接或用 insert 或 select 插入另一个表的虚拟表。然而，过程类型的表是只读的，这意味着不能对该表发出以下命令：
  - alter table

  - create index
  - delete
  - insert
  - truncate table
  - update
- 列名称以下划线 (_) 开头，以表明列不是远程过程结果集的一部分。这些列将作为参数列被引用。例如：

  ```
  create existing table rpc1
   (
      a         int,
      b         int,
      c         int,
      _p1       int null,
      _p2       int null
  )
  external procedure
  at "SYBASE.sybsystemprocs.dbo.myproc"
  ```

  在此示例中，参数列 _p1 和 _p2 是输入参数。它们不会出现在结果集中，但可以在查询中引用：

  ```
  select a, b, c from t1
  where _p1 = 10 and _p2 = 20
  ```

  CIS 通过名为 *@p1* 和 *@p2* 的参数将搜索参数作为参数传递给远程过程。
- create existing table 语句中的参数列定义：
  - 必须允许空值。
  - 不能位于常规结果列之前，它们必须出现在列列表的末尾。
- 如果参数列包含在 select 列表中，并且是作为参数传递给远程过程的，那么返回值将由 where 子句指派。
- 如果参数列包含在 select 列表中，但未出现在 where 子句中或不能作为参数传递给远程过程，则其值为 NULL。
- 如果 Adaptive Server 查询处理器认为某个参数列是可搜索的参数，则可将该参数列作为参数传递给远程过程。如果某个参数列未包含在任何 or 谓词中，则认为该参数列是可搜索的参数。例如，以下查询的第二行中的 or 谓词禁止将参数列用作参数：

  ```
  select a, b, c from t1
  where _p1 = 10 or _p2 = 20
  ```

加密列

create existing table 可自动使用远程表中的任何加密列元数据更新 syscolumns。create existing table 命令的列列表中不能包含 encrypt 关键字。

标准	符合 ANSI SQL 的级别Transact-SQL 扩展。
权限	create existing table 权限缺省情况下授予表所有者，并且不能移交。
另请参见	**命令** alter table, create table, create proxy_table, drop index, insert, order by 子句, set, update.

create function

说明

创建一个用户定义的函数，该函数是返回指定值的已保存 Transact-SQL 例程。

语法

```
create function [ owner_name.]function_name
        [ ( @parameter_name [as] parameter_datatype [ = default ]
            [ ,...n ] ) ]
        returns return_datatype
        [ with recompile]
        as
        [begin]
        function_body
        return scalar_expression
        [end]
```

参数

owner_name

是拥有该用户定义函数的用户 ID 的名称。必须是现有用户 ID。

function_name

是用户定义的函数的名称。函数名必须符合标识符规则，且必须在数据库中以及对于其所有者来说都具有唯一性。函数名不能与其它 Adaptive Server 函数名相同。

注释　要引用或调用用户定义的函数，请指定 *owner_name.function_name*，后跟小括号（请参见下面的“示例”部分中的 BONUS 函数）。在小括号中为所有参数指定参数表达式。调用函数时，不能指定参数列表中的参数名称。必须为所有参数提供参数值，且这些参数值的序列必须与在 create function 语句中定义参数时所采用的序列相同。如果函数的参数具有缺省值，则在调用该函数以获取缺省值时，必须指定关键字“default”。

@parameter_name

是用户定义的函数中的参数。可以在 create function 语句中声明一个或多个参数。一个函数最多可以包含 2,047 个参数。在执行函数时，除非定义了参数的缺省值，否则用户必须提供每个已声明参数的值。

使用“at”符号 (@) 作为第一个字符来指定参数名。参数名必须符合标识符规则。参数是函数的局部参数。您可以在其它函数中使用相同的参数名。

如果参数具有缺省值，则当用户调用函数以获取缺省值时，他们必须指定关键字“default”。

parameter_datatype

是参数的数据类型。所有标量数据类型和 Java 抽象数据类型 (ADT) 都可用作用户定义的函数的参数。但是，用户定义的函数不支持 timestamp、 text、 image 和 unitext。

with recompile

指示 Adaptive Server 从不保存此函数的计划；而是每次在 SQL 语句中引用该函数时创建一个新计划。如果您希望以不规则方式执行此函数，请使用 *with recompile*，它将需要一个新的计划。

return_datatype

是用户定义的标量函数的返回值。 *return_datatype* 可以是除 text、image、 unitext 和 timestamp 之外的任意标量数据类型和 Java ADT。

scalar_expression

指定标量函数返回的标量值。

可以调用使用了标量表达式 （包括计算列和 check 约束定义）的标量值函数。

function_body

指定一系列 Transact-SQL 语句，结合使用这些语句时不会产生副作用，但会定义该函数的值。 *function_body* 只用在标量函数以及多语句表值函数中。在标量函数中， *function_body* 是一系列计算结果为标量值的 Transact-SQL 语句。

示例

创建一个名为 bonus 的用户定义函数：

```
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
```

用法

- 如果此用户定义函数的所有者还具有所有内部引用的数据库对象，则其他所有具有此函数 execute 权限的用户执行该函数时，都会被自动授予对所有引用对象的访问权限。
- 创建函数时， Adaptive Server 会通过检查确定该函数是 SQL 用户定义的函数还是 SQLJ 用户定义的函数。如果是 SQLJ 函数，Adaptive Server 将检查 “sa” 权限。如果是 SQL 函数， Adaptive Server 将检查 create function 权限。

权限

对 create function 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 create function 特权。您必须具有 create any function 特权才能为其他用户运行 create function。
细化权限已禁用	在禁用细化权限的情况下，您必须为数据库所有者或具有 create function 特权。

create function (SQLJ)

说明

通过将 SQL 包装加入到 Java 静态方法中，创建用户定义的函数。可返回一由该方法定义的值。

语法

```
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
        ...]])] '
```

参数

sql_function_name

是函数的 Transact-SQL 名称，必须符合标识符规则，并且不能是变量。

sql_parameter_name

是该函数的参数名。函数执行时，提供每个输入参数的值。参数是可选的， SQLJ 函数不需要带参数。

参数名必须符合标识符的规则。如果参数的值包含非字母数字字符，则必须用引号将其引起来。这包括数据库名或所有者名限定的对象名，因为它们包含一个句点。如果参数值以数字字符开头，则还必须用引号将其引起来。

sql_datatype [(*length*) | (*precision* [, *scale*])]

是参数的 Transact-SQL 数据类型。有关这些参数的详细信息，请参见第 160 页的 “create procedure”。

sql_datatype 是 SQL 过程签名。

returns *sql_datatype*

指定函数的结果数据类型。

modifies sql data

表示 Java 方法调用 SQL 操作，读取并修改数据库中的 SQL 数据。这是缺省的、同时也是唯一的实现方法。将它包含进来是为了在语法上与 ANSI 标准兼容。

deterministic | not deterministic
包含进来是为了在语法上与 ANSI 标准兼容。当前尚未实现。

可导出
指定过程将在使用 Adaptive Server OmniConnect™ 功能的远程服务器上运行。它所基于的过程和方法都必须驻留在远程服务器上。

language java
指定外部例程用 Java 编写。这是 SQLJ 函数所必需的子句。

parameter style java
指定在运行期传递给外部例程的参数是 Java 参数。这是 SQLJ 函数所必需的子句。

external
表示 create function 为使用 SQL 之外的编程语言编写的外部例程定义一个 SQL 名称。

name
指定外部例程（Java 方法）的名称。指定的名称 ('*java_method_name* [*java_datatype*[{, *java_datatype*} ...]]') 是一个字符串文字，必须用单引号引起来。

java_method_name
指定外部 Java 方法的名称。

java_datatype
指定可映射的或其结果集可映射的 Java 数据类型。这是 Java 方法签名。

示例

创建一个调用 java.lang.Math.sqrt() 方法的函数 square_root：

```
create function square_root
    (input_number double precision) returns
        double precision
    language java parameter style java
    external name 'java.lang.Math.sqrt'
```

用法

- 不能创建与 Adaptive Server 内置函数同名的 SQLJ 函数。
- 可以用相同的类和方法名称创建用户定义的函数 （基于 Java 静态方法）和 SQLJ 函数。

  注释　注意：Adaptive Server 的搜索顺序可确保始终最先找到 SQLJ 函数。

- 在一条 create function 语句中最多可以包含 31 个参数。

- 创建函数时， Adaptive Server 会通过检查确定该函数是 SQL 用户定义的函数还是 SQLJ 用户定义的函数。如果是 SQLJ 函数，Adaptive Server 将检查 “sa” 权限。如果是 SQL 函数， Adaptive Server 将检查 create function 权限。

权限

对 create function (SQLJ) 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 create function 特权。您必须具有 create any function 特权才能为其他用户运行 create function。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者或是具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
97	install	create function	• *角色* – 当前活动角色 • *关键字或选项* – NULL • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

有关 create function 的详细信息，请参见 《Adaptive Server Enterprise 中的 Java》。

命令 create function (SQLJ), drop function (SQLJ).

系统过程 sp_depends, sp_help, sp_helpjava, sp_helprotect.

create index

说明

创建表中一个或多个计算列或非计算列的索引。创建分区索引。

允许将计算列像普通列一样用作索引键，并创建基于函数的索引。基于函数的索引具有一个或多个用作索引键的表达式。

现有的 create index 语法可创建计算列的索引，但基于函数的索引需要额外的语法。

语法

```
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]
```

创建索引分区：

```
index_partition_clause::=
    [local index [partition_name [on segment_name]
    [, partition_name [on segment_name]...]]]
```

对于基于函数的索引：

```
create [unique | nonclustered] index index_name
    on [[database.]owner.]table_name
     (column_expression [asc | desc]
    [, column_expression [asc | desc]]...
```

参数

unique

禁止重复的索引值 （也称为 “键值”）。系统在创建索引 （如果数据已经存在）以及每次用 insert 或 update 添加数据时检查重复的键值。如果有重复的键值或有多行包含空值，则此命令失败， Adaptive Server 输出一条错误消息并给出重复条目。

警告！ 如果表包含任何非空 text、unitext 或 image 列，则 Adaptive Server 不检测重复行。

如果使用 allow_dup_row 选项创建索引，则生成重复键值的 update 和 insert 命令能够成功。

组合索引 （键值由多个列组成的索引）也可以唯一。

缺省值为非唯一。若要对包含重复行的表创建非唯一聚簇索引，请指定 allow_dup_row 或 ignore_dup_row。请参见第 143 页的 “重复行”。

在域、列表和散列分区表上创建唯一本地索引时，索引键列表是分区键列表的超集。

clustered

意味着当前数据库设备上的行的物理顺序与这些行的索引顺序相同。聚簇索引的底或**叶级**包含实际的数据页。聚簇索引检索数据几乎总是比非聚簇索引快。每个表只允许有一个聚簇索引。请参见第 142 页的 “创建聚簇索引”。

如果未指定 clustered，则假定为 nonclustered。

nonclustered

意味着行的物理顺序与它们的索引顺序不同。非聚簇索引的叶级包含指向数据页上的行的指针。每个表可以有多达 249 个非聚簇索引。

index_name

是索引的名称。索引名称在表内必须唯一，但在数据库内不需要唯一。

table_name

是索引的一个或多个列所在表的名称。如果该表位于另一数据库中，请指定数据库名，如果数据库中有多个具有该名称的表，请指定所有者的名称。 *owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

column_expression

是有效的 Transact-SQL 表达式，该表达式至少引用一个基列，并且不包含来自其它表、局部和全局变量、集合函数或子查询的列。

注释　*column_expressions* 替换了 Adaptive Server 15.0 以前的版本中使用的 *column_name* 变量。

asc | desc

指定以指定列的升序还是降序创建索引。缺省设置是按升序排列。

fillfactor

指定 Adaptive Server 对现有数据创建新索引时使每一页达到的填满程度。 fillfactor 百分比仅在创建了索引后才有意义。随着数据的变化，页不会保持在任何特定的填充程度。

指定的值不保存在 sysindexes 中。使用 sp_chgattribute 创建存储的 fillfactor 值。

fillfactor 的缺省值是 0；当您未在 create index 语句中包含 with fillfactor 时使用此值 （除非已使用 sp_configure 更改了该值）。指定 fillfactor 时，请使用介于 1 和 100 之间的值。

fillfactor 值为 0 可创建具有完全填充页的聚簇索引及具有完全填充叶页的非聚簇索引。它在聚簇索引和非聚簇索引的索引 B 树内都保留适量的空间。通常不需要更改 fillfactor。

如果将 fillfactor 设置为 100，Adaptive Server 在创建聚簇索引和非聚簇索引时，会使每页的占满度达到 100%。将 fillfactor 的值设置为 100 仅对只读表 （永不向其中添加数据的表）才有意义。

如果 fillfactor 的值小于 100 （0 除外，值为 0 是一种特殊情况），Adaptive Server 创建新索引时就不会占满每一页。创建表的索引时，如果表最终将包含大量数据，那么将 fillfactor 的值设置为 10 可能是一种明智的选择，但较小的 fillfactor 值会导致每个索引 （或索引和数据）占用较多的存储空间。

警告！ 使用 fillfactor 创建聚簇索引会影响数据占用的存储空间，因为创建聚簇索引时 Adaptive Server 会重新分配数据。

max_rows_per_page

限制数据页和索引的叶级页上的行数。与 fillfactor 不同，max_rows_per_page 的值将保持不变，直到使用 sp_chgattribute 更改它。

如果不指定 max_rows_per_page 的值， Adaptive Server 将在创建表时使用值 0。表和聚簇索引的值范围在 2K 页上为 0 到 183K，在 16K 页上为 0 到 1486K。

非聚簇索引的每页最大行数取决于索引键的大小。如果指定的值过高， Adaptive Server 会返回错误消息。

将 max_rows_per_page 设置为 0 将创建具有充满的页的聚簇索引和具有充满的叶页的非聚簇索引。它在聚簇索引和非聚簇索引的索引 B 树内都保留适量的空间。

如果将 max_rows_per_page 设置为 1，Adaptive Server 会创建叶级上为每页一行的聚簇和非聚簇索引。使用低值可以减少经常访问的数据的锁争用。然而， max_rows_per_page 值较低会导致 Adaptive Server 创建具有并非完全填充的页的新索引，从而使用存储空间，并会导致更多的页面拆分。

如果启用了 CIS，则不能对远程服务器使用 max_rows_per_page。

警告！ 使用 max_rows_per_page 创建聚簇索引会影响数据占用的存储空间，因为创建聚簇索引时 Adaptive Server 会重新分配数据。

---

with reservepagegap = *num_pages*

指定填充页与空白页的比率，扩充 I/O 分配操作过程中要保留该比率的空白页。对于每个指定的 *num_pages*，都会预留一个空白页供索引将来扩展之用。有效值为 0 到 255。缺省值为 0。

with consumers

指定应该为索引创建执行排序操作的消耗程序进程数。用于对索引排序的实际消耗程序进程数可能与指定值不同，取决于可用的工作进程数和数据分区数。

ignore_dup_key

取消试图将重复的键条目插入具有唯一索引 （聚簇或非聚簇）的表中的操作。Adaptive Server 取消重复键的试图的 insert 或 update，并给出信息性消息。取消之后，包含重复键的事务继续进行直到完成。

无论 ignore_dup_key 设置与否，都不能在已包含重复值或多个空值的列上创建唯一索引。如果试图这样做， Adaptive Server 会显示含有第一个重复值的错误消息。 Adaptive Server 在列上创建唯一索引之前，必须消除重复值。

ignore_dup_row
允许在包含重复行的表上创建一个新的非唯一聚簇索引。ignore_dup_row 从表中删除重复行，并取消任何会创建重复行的 insert 或 update，但不回退整个事务。请参见第 143 页的 “重复行”。

allow_dup_row
允许在包含重复行的表上创建一个非唯一聚簇索引，并允许用 update 和 insert 语句复制行。请参见第 143 页的 “重复行”。

sorted_data
如果表中的数据已经是排序顺序，则将加速聚簇索引或唯一非聚簇索引的创建（例如，当使用 bcp 命令将已排序的数据复制到一个空表中时）。请参见第 145 页的 “使用 sorted_data 选项加速排序”。

with statistics using *num_steps* values
指定为用于优化查询的直方图生成的梯级数。如果忽略该子句：

- 如果当前没有存储前导索引列的直方图，则缺省值为 20。
- 如果索引列的前导列的直方图已存在，则使用当前的梯级数。

如果指定 *num_steps* 为 0，则重新创建索引，但不会覆盖系统表中该索引的统计信息。

实际梯级数可能与您指定的梯级数不同；如果用 *num_steps* 指定的直方图梯级数为 *M*，histogram_tuning_factor 参数为 *N*，则实际梯级数介于 *M* 和 *M*N* 之间，具体取决于分布中存在的频率单元数。

on *segment_name*
在已命名段上创建索引。在使用 on *segment_name* 选项之前，请用 disk init 初始化设备，然后用 sp_addsegment 将段添加到数据库。要生成数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。可在两个位置使用 on *segment_name*：

- 紧邻 *index_partition_clause* 之前 – 定义全局缺省值，该值将用于未在 *index_partition_clause* 中显式定义段的所有分区。
- 内置于该子句本身 – 允许您为每个单个分区指定段

有关在上述两个位置使用 on *segment_name* 的示例，请参见示例部分。

本地索引
对于语义分区表，指定一个始终与其基表均分的索引，即表和索引享有相同的分区键和分区标准。对于循环分区表，本地索引表示表的每个索引分区中的索引键引用一个且仅引用一个表分区中的数据行。

对于语义分区表和循环分区表，每个表分区都只有一个对应的索引分区。

partition_name

指定要在其中存储索引的新分区的名称。分区名称在表或索引上的分区组内必须唯一。如果使用 set quoted_identifier on，则分区名称可以是分隔标识符。否则，分区名称必须是有效的标识符。

如果省略 *partition_name*， Adaptive Server 将创建一个格式为 *table_name_partition_id* 的名称。 Adaptive Server 将截断超过允许的最大长度的分区名称。

示例

示例 1 对 authors 表的 au_id 列创建名为 au_id_ind 的索引：

```
create index au_id_ind on authors (au_id)
```

示例 2 对 authors 表的 au_id 列创建名为 au_id_ind 的唯一聚簇索引：

```
create unique clustered index au_id_ind
on authors (au_id)
```

示例 3 对 titleauthor 表的 au_id 和 title_id 列创建名为 ind1 的索引：

```
create index ind1 on titleauthor (au_id, title_id)
```

示例 4 对 authors 表的 zip 列创建名为 zip_ind 的非聚簇索引，将每个索引页填满四分之一并将排序可以使用的消耗程序进程数限制为 4：

```
create nonclustered index zip_ind
on authors (postalcode)
with fillfactor = 25, consumers = 4
```

示例 5 创建 pub_id 以升序排列而 pubdate 以降序排列的索引：

```
create index pub_dates_ix
on titles (pub_id asc, pubdate desc)
```

示例 6 创建 title_id 的索引，为优化程序统计信息使用 50 个直方图梯级，并在索引中为每 40 页留 1 个空白页：

```
create index title_id_ix
on titles(title_id)
with reservepagegap = 40,
statistics using 50 values
```

示例 7 在已分区的 salesdetail 表上创建本地聚簇索引。clust_idx 索引继承 salesdetail 的分区策略、分区键和分区界限。

```
create clustered index clust_idx
on salesdetail (ord_num) local index
```

示例 8 在已通过 date 列按范围分区的 sales 表上，创建非分区的、非聚簇全局索引。

```
create nonclustered index global_idx
```

```
on sales (order_num)
```

示例 9 首先，创建一个具有三个数据分区的表 pback_sales：

```
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))
```

然后，在分区 p1 上创建一个基于函数的本地索引：

```
create index fc_idx on pback_sales (c1*c2) local index
    p1
```

示例 10 创建基于函数的索引：

```
create index sum_sales on mytitles (price * total_sales)
```

示例 11 在分区名称之前和之后均指定 on *segment_name* 子句：

```
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
```

输出以下内容：

```
 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            NULL                           0
b           varchar     30 NULL  NULL    0 NULL         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

(2 rows affected)
index_ptn_name   index_ptn_seg
---------------- -------------
t1_i1_952063116  default
t1_i1_968063173  default
ip1              seg4
t1_i2_1000063287 seg3

(4 rows affected)
No defined keys for this object.
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
The attribute 'exp_row_size' is not applicable to tables with allpages lock scheme.
The attribute 'concurrency_opt_threshold' is not applicable to tables with
```

```
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)
```

用法

- 如果向更改索引中键的分布的表中添加数据，请定期运行 update statistics。查询优化程序使用由 update statistics 创建的信息选择在表上运行查询的最佳计划。
- 如果创建非聚簇索引时表包含数据，则 Adaptive Server runs 在新索引上运行 update statistics。如果创建聚簇索引时表包含数据，则 Adaptive Server 对所有表的索引运行 update statistics。
- 索引在连接中经常使用的所有列。
- 如果启用了 CIS，则重新构建 create index 命令，并直接将其传递给与表相关联的 Adaptive Server。
- 您不能在包括虚拟散列表的段上使用 create index （聚簇或非聚簇），因为虚拟散列表必须仅采用一个排它段，而排它段是不能被其它表或数据库共享的。
- writetext 可与 online 参数同时运行。

创建索引和存储过程

Adaptive Server 在执行完 create index 语句后自动重新编译存储过程。虽然在执行 create index 之前开始的即席查询继续起作用，但它们不利用新索引。

在 Adaptive Server 12.5 版本和更早版本中，高速缓存的存储过程忽略 create index。

有效地创建索引

- 索引加速数据检索，但会减慢数据更新。当段在独立的物理设备上时，为了获得更好的性能，可以在一个段上创建表，并在另一个段上创建其非聚簇索引。
- 如果表已分区且服务器配置为并行，则 Adaptive Server 可以并行创建索引。还可以使用排序缓冲区以减少排序期间需要的 I/O 量。请参见《性能和调优指南：优化程序和抽象计划》中的 “并行排序”。

- 应在创建任何非聚簇索引之前创建聚簇索引，因为在聚簇索引创建之后非聚簇索引会自动重建。
- 对仅数据锁定表使用并行排序时，工作进程数必须等于或大于分区数 （即使对空表也是如此）。还必须启用数据库选项 select into/bulkcopy/pllsort。

创建聚簇索引

- 表 “跟随”其聚簇索引。创建表时，如果用 on *segment_name* 扩展 create clustered index，表将迁移到创建索引的段上。

  如果在特定段上创建表，然后在不指定段的情况下创建聚簇索引，Adaptive Server 创建聚簇索引时会将表转到缺省的段上。

  因为 text、 unitext 和 image 数据存储在单独页链中，所以用 on *segment_name* 创建聚簇索引不会移动文本和图像列。
- 为创建聚簇索引， Adaptive Server 复制现有数据，索引创建完成后服务器删除原始数据。在创建聚簇索引之前，请使用 sp_spaceused 确保数据库至少有 120% 表大小的可用空间。
- 聚簇索引常根据表的主键 （唯一标识行的一列或多列）来创建。可以用 sp_primarykey 将主键记录在数据库中 （以供前端程序和 sp_depends 使用）。
- 若要允许在聚簇索引中有重复行，请指定 allow_dup_row。

创建压缩表上的索引

- 如果表需要排序， Adaptive Server 会在数据复制操作期间压缩所有可供压缩的行。
- 如果您使用 sorted_data 创建聚簇索引， Adaptive Server 不会执行数据复制，而且在构建聚簇索引时不压缩任何数据行。
- Adaptive Server 不压缩索引键值：它仅压缩数据行中的值。
- 您可以选择索引键列并创建唯一索引，即使键列被压缩也是如此。要执行非聚簇唯一性检查，请检查索引页中的非压缩索引键。
- Adaptive Server 使用非压缩索引键和行格式来验证是否支持 ignore_dup_key、 ignore_dup_row 和 allow_dup_row。
- Adaptive Server 仅将 fillfactor 参数应用于行级压缩。
- 将 fillfactor 参数应用于所有页锁定聚簇索引的数据页时， Adaptive Server 会考虑：
  - 最终压缩行格式
  - 压缩所需的空间

Adaptive Server 可能会额外压缩后续页压缩操作所使用的页空间，从而导致 fillfactor 值降低。

- Adaptive Server 在页级应用 respagegap 参数，以使其不受压缩影响。
- max_rows_per_page 仅包括页的数据行，不包括隐藏的页字典、索引和字符编码条目。

创建加密列的索引

如果指定加密密钥而未使用任何初始化矢量或随机填充，则可以创建加密列的索引。在对数据进行等于和不等于匹配时，加密列的索引将非常有用，但它不能用于匹配不区分大小写的数据或任何数据的范围搜索。

若要改进等于和不等于搜索以及连接的性能，可以创建加密列的索引。

如果您执行以下操作，

create index 将报告错误：

- 使用引用加密列的表达式创建函数索引
- 在用初始化矢量或随机填充加密的列中创建索引

注释　不能在函数索引表达式中使用加密列。

指定索引是按升序还是按降序排列

在索引列名称之后用 asc 和 desc 关键字指定索引键的排列顺序。通过创建索引使列按查询的 order by 子句中指定的相同顺序排列，在查询处理过程中便不需要执行排序步骤。请参见 《性能和调优指南：锁定》 中的 “为改善性能建立索引”。

索引的空间要求

- 分配给表和索引的空间增量为一次 1 个扩充，或 8 页。每当一个扩充填满后，就会分配另一个扩充。使用 sp_spaceused 显示索引分配的和使用的空间量。
- 在某些情况下，使用 sorted_data 选项允许 Adaptive Server 跳过复制数据行，如 第 145 页的表 1-6 中所述。在这种情况下，只需要索引结构自身的足够的额外空间。根据键大小的不同而不同，这通常占表大小的 20%。

重复行

- 创建非唯一的非聚簇索引时， ignore_dup_row 和 allow_dup_row 选项是不相关的。 Adaptive Server 在每个非聚簇索引内部附加了一个唯一的行标识号，所以即使相同的数据值也不会出现重复行。
- ignore_dup_row 和 allow_dup_row 相互排斥。

- 在所有页锁定表中，非唯一聚簇索引允许采用重复键，但不允许采用重复行，除非指定 allow_dup_row。该行为对于仅数据锁定表是不同的，表 1-4 对此进行了详细描述。
- allow_dup_row 允许在包含重复行的表上创建非唯一的聚簇索引。如果不使用 allow_dup_row 选项创建了表的非唯一的聚簇索引，则不能使用 insert 或 update 命令创建新的重复行。

  如果表中存在具有唯一性的索引，则唯一性的要求优先于 allow_dup_row 选项。如果表中的任何列上存在唯一索引，则不能使用 allow_dup_row 创建索引。
- ignore_dup_row 选项还可与非唯一的聚簇索引一起使用。ignore_dup_row 选项可消除批处理数据中的重复项。 ignore_dup_row 会取消可能创建重复行的任何 insert 或 update，但不回退整个事务。
- 表 1-4 说明 allow_dup_row 和 ignore_dup_row 如何影响创建表 （包含重复行）的非唯一聚簇索引以及向表中输入重复行的尝试。

表 1-4: 非唯一聚簇索引的重复行选项

设置选项	在有重复行的表上创建索引	向有索引的表中插入重复行
未设置选项	create index 失败。	insert 失败。
设置 allow_dup_row	create index 完成。	insert 完成。
设置 ignore_dup_row	创建索引，但删除重复行；错误消息。	插入重复行之外的所有行；错误消息。

表 1-5 说明了不同类型的索引可使用哪些索引选项：

表 1-5: 索引选项

索引类型	选项
聚簇	ignore_dup_row | allow_dup_row
唯一，聚簇	ignore_dup_key
非聚簇索引	否
唯一，非聚簇	ignore_dup_key

使用唯一约束代替索引

- 作为 create index 的一种替代方法，可以通过用 create table 或 alter table 语句指定唯一约束隐式地创建唯一索引。唯一约束在表的列上创建聚簇或非聚簇索引。这些*隐式*索引在约束之后命名，且它们遵循与用 create index 创建的索引相同的规则。
- 不能用 drop index 语句删除支持唯一约束的索引。它们在使用 alter table 语句删除约束时或删除表时被删除。有关唯一约束的详细信息，请参见 create table。

使用 *sorted_data* 选项加速排序

- 在某些情况下，通过跳过排序步骤和消除将数据行复制到新页的需要， sorted_data 选项可以减少创建索引所需的时间。对大型表操作时速度的提高尤其明显，对 1GB 以上的表操作时，速度将成倍提高。

  如果指定 sorted_data，但数据未按排序顺序排列， Adaptive Server 将显示错误消息，该命令失败。

  在创建非唯一的非聚簇索引时，除非没有任何一行有重复键才会成功。如果有若干行有重复键， Adaptive Server 会显示出错信息，命令失败。

- sorted_data 对创建聚簇索引的影响取决于表是否已分区以及是否在 create index 命令中使用了某些其它选项。若使用了某些选项，则对未分区表需要数据复制；对已分区表需要排序加数据复制；而其它选项仅在下列情况下需要数据复制：
  - 使用 ignore_dup_row 选项
  - 使用 fillfactor 选项
  - 使用 on *segmentname* 子句指定与表数据所在段不同的段
  - 使用 max_rows_per_page 子句指定一个与表相关联的值不同的值
- 表 1-6 显示何时需要排序，何时要为分区表和未分区表复制表。

表 1-6: 使用 sorted_data 选项创建聚簇索引

选项	分区表	未分区表
未指定选项	只有在循环分区表上创建聚簇索引时才需要并行排序；复制数据，在分区上均匀分配数据；创建索引树。	并行或非并行排序；复制数据，创建索引树。
仅指定 with sorted_data 或 with sorted_data on *same_segment*	只创建索引树。不执行排序或复制数据。不并行运行。	只创建索引树。不执行排序或复制数据。不并行运行。
with sorted_data 和 ignore_dup_row 或 fillfactor 或 on *other_segment* 或 max_rows_per_page	并行排序；复制数据，在分区上均匀分配；创建索引树。	复制数据并创建索引树。不执行排序。不并行运行。

指定直方图梯级数

- 使用 with statistics 子句为索引的前导列指定直方图梯级数。直方图在查询优化程序为一个列确定与搜索参数相匹配的行数期间使用。
- 要重新创建索引而不更新列在 sysstatistics 中的值，请让梯级数为 0。这样可以避免覆盖已经用 optdiag 更改了的统计信息。

- 如果指定了具有某个值的 histogram_tuning_factor 参数，则 create index 使用 20 和 M*20 之间的任何梯级数，具体取决于已隔离的频率单元数。缺省值为 20，但是可以使用 using step values 选项指定其它值。

空间管理属性

- fillfactor、 max_rows_per_page 和 reservepagegap 以不同的方式帮助管理索引页上的空间：
  - fillfactor 适用于所有锁定方案的索引。对于所有页锁定表上的聚簇索引，它影响表的数据页。在其它索引上，它影响索引的叶级。
  - max_rows_per_page 只适用于所有页锁定表的索引页。
  - reservepagegap 适用于所有锁定方案的表和索引。
- reservepagegap 在以下情况下影响索引中的空间使用：
  - 创建索引时。
  - 在索引上执行 reorg 命令时。
  - 在创建聚簇索引之后重建非聚簇索引时。
- 如果在 create clustered index 命令中指定了 reservepagegap 值，则它适用于：
  - 所有页锁定表的数据和索引页
  - 仅 DOL 锁定表的索引页
- *num_pages* 值指定索引的叶级上的填充页与空白页的比率，因此当需要新空间时，索引可以分配接近于现有页的空间。例如，值为 10 的 reservepagegap 为每 9 个使用的页保留 1 个空白页。
- 在所有页锁定表上与 create clustered index 一起指定的 reservepagegap 将覆盖以前用 create table 或 alter table 指定的任何值。
- 可以用 sp_chgattribute 更改索引的空间管理属性。用 sp_chgattribute 更改属性不会立即影响表上索引的存储。将来进行大量分配时（如 reorg rebuild），请使用 sp_chgattribute 值。
- 由 sp_chgattribute 设置的 fillfactor 值存储在 sysindexes 中的 fill_factor 列中。当 alter table...lock 命令或 reorg rebuild 命令导致重新创建索引时，将应用 fillfactor。

索引选项和锁定方式

表 1-7 显示了所有页锁定表和仅数据锁定表支持的索引选项。在仅数据锁定表上，在 create index 期间，将强制执行 ignore_dup_row 和 allow_dup_row 选项，但在 insert 和 update 操作期间，则不会强制执行。仅数据锁定表始终允许插入重复行。

表 1-7: 创建锁定方案支持的索引选项

索引类型	所有页锁定表	DOL 锁定表	
		在索引创建期间	在插入期间
聚簇	allow_dup_row、ignore_dup_row	allow_dup_row、 ignore_dup_row	allow_dup_row
唯一聚簇索引	ignore_dup_key	ignore_dup_key	ignore_dup_key
非聚簇索引	无	无	无
唯一非聚簇索引	ignore_dup_key	ignore_dup_key	ignore_dup_key

表 1-8 显示尝试向带有聚簇索引的表中插入重复行的命令行为，以及何时删除和重新创建聚簇索引。

表 1-8: 重复行选项的强制实施和错误

选项	所有页锁定表	DOL 锁定表
未指定选项	插入失败，错误消息是 2615。重新创建索引成功。	插入成功。重新创建索引失败，错误消息是 1508。
allow_dup_row	插入并重新创建索引成功。	插入并重新创建索引成功。
ignore_dup_row	插入失败，错误消息是 “重复行被忽略” (Duplicate row was ignored)。重新创建索引成功。	插入成功。重新创建索引删除重复行。

对仅数据锁定表使用 *sorted_data* 选项

- 只能在批量复制到空表操作后立即使用 sorted_data 选项来 create index。对表的数据修改导致了额外的页分配时，不能使用 sorted_data 选项。
- 为空间管理属性指定不同的值可能会替换 sorted_data 的禁止排序功能。

获取有关表和索引的信息

- 每个索引 – 包括组合索引 – 在 sysindexes 中用一行表示。
- 有关通过索引检索数据的顺序和 Adaptive Server 安装的排序顺序影响的信息，请参见 order by 子句。
- 有关表的索引的信息，请执行 sp_helpindex。有关索引分区的信息，还可以执行 sp_helpartitions。
- 每个索引分区和数据分区在 syspartitions 中用一行表示。

创建计算列的索引

- 可以使用实现计算列作为索引键，就像它们是常规列一样。
- 若要将虚拟列转换为实现列并建立该列的索引，请在执行 create index 前使用 alter table modify 及 materialized 选项。

- 计算列无需是确定性的即可用作索引键；然而，必须小心非确定性列对引用它的查询的影响。

创建分区索引

- 本地索引从基表继承分区策略、分区列和分区界限 （对于域分区和列表分区）。
- Adaptive Server 维护本地索引，并在用不同分区键对基表重新分区后重建本地索引。
- Adaptive Server 支持：

索引类型	表类型
本地聚簇和非聚簇分区索引	分区表
全局、聚簇、未分区索引	循环分区表
全局、非聚簇、未分区索引	所有分区表

- 对于域、散列和列表分区表，聚簇索引始终是本地索引。无论语法中是否包含“local index”， Adaptive Server 都将创建本地聚簇索引。

创建基于函数的索引

- 可以直接对表达式创建索引。
- 表达式必须是确定性的。
- 由于 Adaptive Server 不检验表达式索引键的确定性属性，所以用户必须手动维护该属性。更改此属性可能导致意外结果。
- 由于基于函数的索引键必须具有确定性，所以其结果已预先求值，可重复使用而无需重新求值。 Adaptive Server 假设所有基于函数的索引键都是确定性的，并在查询中引用时使用对它们预先求值得到的值；仅当其基列值发生更改时才对它们重新求值。
- 一个索引可以有多个基于函数的索引键或一个基于函数的索引键与常规列的组合。
- 用作索引键的表达式必须是确定性的。表达式键不同于计算列索引键，后者只需求值一次，并且不需要确定性属性。而表达式在指定查询中每次出现时都必须重新求值，并且必须始终返回相同的结果。
- 如果基于函数的索引所引用的用户定义函数被删除或无效，调用该函数的任何操作都会失败。
- Adaptive Server 不支持基于函数的聚簇索引。
- 不能使用 sorted_data 选项创建基于函数的索引。
- 创建表达式的索引键后，仅当该表达式与用于创建索引键的表达式完全相同时，后续查询才会将该表达式视为索引键。

- 对基列执行任何 insert、 delete 和 update 操作都会导致 Adaptive Server 自动更新基于函数的索引键值。

标准　　符合 ANSI SQL 的级别Transact-SQL 扩展。

权限　　对 create index 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或具有 create any index 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是表所有者或具有 sa_role 的用户。 create index 权限缺省情况下授予表所有者，并且不能移交。

审计　　sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
104	create	create index	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - 索引名称 • *代理信息* - set proxy 有效时的初始登录名

另请参见　　**命令** alter table, create table, drop index, insert, order by 子句, set, update.

系统过程 sp_addsegment, sp_chgattribute, sp_helpcomputedcolumn, sp_helpindex, sp_helpsegment, sp_spaceused.

实用程序 optdiag.

create login

说明

创建登录帐户；指定口令、帐户的登录配置文件，以及为帐户分配的用户提供的参数。

语法

create login *login_name* with [encrypted]
password *password*
[*attribute_value_pair_list*]

参数

login_name

指定要创建的登录帐户名；它必须以字母字符开头且长度不能超过 30 个字符。

with encrypted

指定新登录帐户的加密口令。

password *passwordValue*

指定新登录帐户的口令。

attribute_value_pair_list

要添加到登录帐户的属性和对应值的列表。 *attribute_value_ pair_list* 是属性名和值。指定以下各项中的一个或多个：

参数	参数值	说明
login profile	有效值： • *login_profile_name* • ignore	• *login_profile_name* 将指定的登录配置文件绑定到指定的登录帐户。 • ignore 消除所有登录配置文件绑定。缺省的登录配置文件将不适用，而是会按照 15.7 之前的版本那样应用属性。 如果未指定登录配置文件，则应用缺省登录配置文件。请参见《安全性管理指南》中的“应用登录配置文件和口令策略属性”。
suid	有效值：介于 [-32768, 2147483647] 之间的唯一值，不包括 [-2, -1, 0, 1, 2]。	缺省情况下，会生成 suid 并在创建时自动将其分配给登录帐户。
fullname	*name_value*	拥有登录帐户的用户的全名。 缺省值为 NULL。
login script	*login_script_name*	指定有效的存储过程。登录脚本仅限于 120 个字符。
password expiration	有效范围：0 到 32767 天。	口令有效期。 缺省值为 0，意思是口令永不过期。
min password length	有效范围：0 到 30。	口令所需的最小长度。 缺省值为 6。

参数	参数值	说明
max failed attempts	有效范围：-1 到 32767。	允许的登录尝试次数，此后登录帐户将被锁定。 -1 指示将会记录失败次数，但不锁定。 缺省值为 0，表示不会记录失败次数，而且不会由于登录尝试失败而锁定帐户。
default database	*default_database_name*	指定用作缺省数据库的数据库。 缺省值为 Master。
default language	*default_language*	指定用作缺省语言的语言。 缺省值为 us_english
authenticate with	有效值：ASE、 LDAP、 PAM、 KERBEROS、 ANY	指定用于鉴定登录帐户的机制。 当使用 ANY 时，Adaptive Server 会检查有无定义的外部鉴定机制。如果定义了外部鉴定机制， Adaptive Server 会使用该机制，否则使用 ASE 机制。 如果未指定 authenticate with *鉴定机制*，则 ANY 会用于登录帐户。
exempt inactive lock	有效值：TRUE 或 FALSE	指定是否不让登录帐户因非活动而被锁定。 缺省值为 FALSE，表示不让帐户锁定。

示例

示例 1 创建一个口令为 itsA8ecret 的登录帐户，应用登录配置文件 emp_lp、服务器用户 ID 7，并指定该帐户不会因非活动而锁定。

```
create login ravi with password itsA8ecret login profile
emp_lp suid 7 exempt inactive lock true
```

用法

- 优先规则决定在从不同登录配置文件中获得属性时或使用 sp_passwordpolicy 指定值时如何应用登录帐户属性。
- 为便于管理，极力建议所有用户的 Adaptive Server 登录名必须与他们的操作系统登录名相同。这使得在操作系统和 Adaptive Server 间关联审计数据更加容易。否则，请记录操作系统和服务器登录名之间的对应关系。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 create login 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage any login 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sso_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
103	login_admin	create login	关键字包含：WITH *attribute_value_pair_list*

另请参见

命令 create login profile、 alter login、 alter login profile、 drop login、 drop login profile.

文档 有关创建登录帐户的详细信息，请参见 《安全性管理指南》。有关优先规则，请参见 《安全性管理指南》中的 “应用登录配置文件和口令策略属性”。

函数 lprofile_id、 lprofile_name.

系统过程 sp_passwordpolicy、sp_displaylogin、sp_displayroles、sp_locklogin.

create login profile

说明

用指定的属性创建登录配置文件。

语法

create login profile *login_profile_name* [as default]
[with { attributes from *login_name* | *attribute_value_pair_list* }]

参数

login_profile_name
指定要创建的登录配置文件的名称。

as default
将创建的登录配置文件设置为所有登录帐户 （sa 和 probe 除外）的缺省值。

with attributes from *login_name* | *attribute_value_pair_list*
如果指定了 *login_name*，则会创建一个具有从指定登录帐户获得的属性值的登录配置文件。*attribute_value_ pair_list* 指定属性名和对应的值。指定以下属性和值中的一个或多个：

- default database *default_database_name* – 指定缺省数据库。缺省值为 Master。
- default language *default_language* – 指定缺省语言。缺省值为 us_english。
- login script *login_script_name* – 指定有效的存储过程。登录脚本仅限于 120 个字符。
- authenticate with – 指定用于鉴定登录帐户的机制。有效值：ASE、 LDAP、 PAM、 KERBEROS、 ANY。

  当使用 ANY 时， Adaptive Server 会检查有无定义的外部鉴定机制。如果定义了外部鉴定机制， Adaptive Server 会使用该机制，否则使用 ASE 机制。

  如果未指定 authenticate with *鉴定机制*，则 ANY 会用于登录帐户。
- track lastlogin – 启用上次登录更新。有效值：TRUE、 FALSE。缺省值为 TRUE，表示更新。
- stale period – 指示允许登录帐户在因非活动而被锁定前保持非活动状态的持续时间。有效值为 1 到 32767 天。持续时间：D （天）、 W （周）、 M （月）、 Y （年）。缺省值为 D （天）。
- profile id – 与登录帐户的服务器用户 ID 共享 ID 空间。缺省情况下，会生成登录配置文件 ID 并在创建时自动将其分配给登录配置文件。

  有效值在登录帐户和登录配置文件之间是唯一的。范围：[-32768, 2147483647]，不包括：-2, -1, 0, 1, 2。

示例

示例 1 创建登录配置文件。未设置的属性值将遵循优先规则：

```
create login profile eng_lp
```

有关信息，请参见 《安全性管理指南》中的 “应用登录配置文件和口令策略属性”。

示例 2 创建一个登录配置文件并将登录属性值从登录帐户 ravi 传送到新登录配置文件 ravi_lp。未设置的属性值将遵循优先规则。

```
create login profile ravi_lp with attributes from ravi
```

示例 3 创建鉴定方法为 ASE 的登录配置文件 sa_login_profile。

```
create login profile sa_login_profile with authenticate
with ASE
```

用法

优先规则决定在从不同登录配置文件中获得属性时或使用 sp_passwordpolicy 指定值时如何应用登录帐户属性。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 create login profile 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage any login profile 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sso_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
137	security_profile	create login profile	关键字包含 DEFAULT 如果将登录配置文件设置为缺省值：{attributes from *login_name* | *attribute_value_pair_list*}

另请参见

命令 create login、alter login、alter login profile、drop login、drop login profile.

文档 有关创建登录配置文件、登录时调用登录脚本以及优先规则的信息，请参见 《安全性管理指南》。

函数 lprofile_id、 lprofile_name.

系统过程 sp_passwordpolicy、sp_displaylogin、sp_displayroles、sp_locklogin.

create plan

说明　创建抽象计划。

语法

create plan *query plan*
[into *group_name*]
[and set @*new_id*]

参数

query
是一个字符串文字、参数或包含查询的 SQL 文本的局部变量。

plan
是一个字符串文字、参数或包含抽象计划表达式的局部变量。

into *group_name*
指定抽象计划组的名称。

and set @*new_id*
在变量中返回抽象计划的 ID 号。

示例　**示例 1** 为指定查询创建抽象计划：

```
create plan "select * from titles where price > $20" " (t_scan titles)"
```

示例 2 为 dev_plans 组中的查询创建抽象计划，并在变量 @*id* 中返回计划 ID：

```
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
```

用法

- create plan 将抽象计划保存在用 into 指定的组中。如果未指定组名，则将计划保存在当前活动的计划组中。
- 查询和用 create plan 创建的抽象计划尚未经过有效的 SQL 语法检查，计划也尚未经过有效的抽象计划语法检查。此外，不检查计划是否与 SQL 文本兼容。应该通过运行在 create plan 语句中指定的查询，立即检查使用 create plan 创建的所有计划的正确性。
- 如果组中的另一个查询计划包含相同的 SQL 文本，则必须用 set plan replace on 启用 replace 模式。否则， create plan 命令失败。
- 在 and set 子句中使用 @*new_id* 之前，必须声明该参数。
- 使用 into 指定的抽象计划组必须已经存在。

- 如果使用 sp_configure "enable literal autoparam", 1 系统过程启用服务器范围的文字参数化， Adaptive Server 将忽略您在创建抽象计划时指定的所有文字参数化。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

无需任何权限即可使用 create plan。

另请参见

命令 set 计划.

文档 《性能和调优指南：优化程序和抽象计划》。 .

系统过程 sp_add_qpgroup、 sp_configure "enable literal autoparam"、 sp_find_qplan、 sp_help_qplan、 sp_set_qplan.

create precomputed result set

说明

创建预先计算结果集及其维护所需的策略。

语法

```
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
```

参数

precomputed result set | materialized view

创建物化视图或预先计算的结果集。

prs_name

预先计算的结果集的名称。完全限定的 *prs_name* 不能包括服务器或数据库名称。

alternative_column_name

表示预先计算的结果集列的名称。

constraint *constraint_name*

引入完整性约束的名称，该名称必须符合标识符规则，且在数据库中必须唯一。

如果不为唯一或主键约束指定名称，则 Adaptive Server 将以 *tabname_colname_tabindid* 格式生成名称，其中 *tabindid* 是表 ID 和索引 ID 的字符串并置。

unique (*column_name*,...)

约束所指定列中的值，使得任两行都不能具有相同的值。

{immediate | manual} refresh

确定刷新策略：

- immediate - （缺省）在更新基表的事务进行期间更新预先计算的结果集。
- manual - 显式更新预先计算的结果集。如果使用 manual 参数，直到显式发出 refresh 命令，基表更新才会反映在预先计算的结果集中。由于 manual 预先计算结果集并不进行维护， Adaptive Server 会认为它们是旧的 （即使在发出 refresh 参数后）。因此，只有在查询接受旧数据时，查询处理器才为查询重写选择此数据。

populate | nonpopulate

指定是在创建预先计算结果集后对其进行填充，还是只用元数据信息创建预先计算的结果集，然后再进行填充：

- populate - （缺省）结果集作为 create 命令的一部分进行填充。
- nonpopulate - 结果集不作为 create 命令的一部分进行填充。如果指定 nonpopulate，则无法对预先计算结果集运行 enable 参数。下次发出 refresh 命令时， Adaptive Server 会启用预先计算结果集。

enable | disable

指定预先计算的结果集是否可用于操作。此选项会覆盖所有其它预先计算结果集选项。

- enable - （缺省）可用于操作。只有配置为 enable 的预先计算结果集才按刷新策略进行维护。
- disable - 不可用于操作。维护或查询重写不考虑禁用的预先计算结果集。如果预先计算的结果集配置为 disable，则不：
  - 在优化期间用于查询重写，无论是否指定 use in optimization。
  - 填充，无论是否指定 with populate。

{enable | disable} use in optimization

指定是否包括预先计算结果集以在优化期间用于查询重写。缺省情况下会启用 use in optimization。查询重写是否考虑预先计算结果集取决于如何设置 refresh 参数：

- immediate - 所有查询都考虑使用。
- manual - 只在查询接受旧数据时才考虑使用。

lock {datarows | datapages | allpages}

表示预先计算的结果集所使用的锁定级别。

示例

创建 prs_1 预先计算结果集：

```
create precomputed result set prs_1
as select col1, col2 from test_table
```

用法

在创建或更改预先计算结果集之前，必须设置以下 set 参数：

- set ansinull on
- set arithabort on
- set arithignore off
- set string_rtruncation on

标准	create precomputed result set 命令是 Transact-SQL 扩展，不包括在 SQL 标准范围内。
权限	必须具有 create table 和 create view 特权才能创建预先计算结果集。
审计	不对预先计算结果集创建进行审计。

create procedure

说明

创建存储过程或可以带一个或多个用户提供的参数的扩展存储过程 (ESP)。

注释 注意：第 130 页的 “create function (SQLJ)”有关用于创建过程的 SQLJ 命令的语法和用法信息，请参见。

语法

```
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}
```

参数

procedure_name

是过程的名称。它必须符合标识符规则，并且不能是变量。指定所有者的名称，以创建由当前数据库中的其他用户拥有的另一个同名过程。 *owner* 的缺省值是当前用户。

;number

是一个可选的整数，用于将同名的过程分成一组，以便能用单个 drop procedure 语句一并删除它们。同一应用程序中使用的过程经常用此方法分组。例如，如果将与名为 orders 的应用程序一起使用的过程命名为 orderproc;1、 orderproc;2 等等，则以下语句将删除整个组：

```
drop proc orderproc
```

一旦将过程分组，就不能单独删除组中的过程。例如，以下语句是不允许的：

```
drop procedure orderproc;2
```

如果在**已评估的配置**中运行 Adaptive Server，则不能对过程分组。已评估的配置要求不允许过程分组，以便每个存储过程都有唯一的对象标识符，并可单独删除。如果不允许过程分组，系统安全员必须用 sp_configure 重新设置 allow procedure grouping。有关已评估配置的详细信息，请参见 《系统管理指南》。

parameter_name

是该过程的参数名。过程执行时，提供每个输入参数的值。在 create procedure 语句中，参数名是可选的 过程不需要带任何参数。

参数名前面必须带有 @ 符号，且必须符合标识符的规则。参数名（包括 @ 符号）最长为 30 个字符，标识符则更长。参数局限于过程：同一参数名可用于其它过程。

如果参数的值包含非字母数字字符，则必须用引号将其引起来。这包括数据库名或所有者名限定的对象名，因为它们包含一个句点。如果字符参数的值以数字字符开头，则还必须用引号将其引起来。

datatype[(*length*) | (*precision* [, *scale*])]

是参数的数据类型。请参见 《参考手册：构件块》第 1 章 “系统数据类型和用户定义的数据类型”中第 42 页的 “用户定义的数据类型”。存储过程参数的数据类型不能为 text、 unitext 或 image，也不能是基础类型为 text、 unitext 或 image 的用户定义数据类型。

char、 varchar、 unichar、 univarchar、 nchar、 nvarchar、 binary 和 varbinary 数据类型应该在括号中包括 *length*。如果省略了 length，Adaptive Server 会将参数值截断为 1 个字符。

float 数据类型需要一个括在括号中的二进制 *precision*。如果省略了 precision， Adaptive Server 将为平台使用缺省精度。

numeric 和 decimal 数据类型需要 *precision* 和 *scale*，用括号括起来并用逗号分隔。如果省略了 precision 和 scale，Adaptive Server 将使用缺省的精度 18 和标度 0。

default

定义过程参数的缺省值。如果定义了缺省值，则用户无需提供参数值就可以执行过程。缺省值必须是一个常量。如果过程使用带有关键字 like 的参数名 （请参见示例 2），则缺省值可包括通配符 %、 _、 [ ] 和 [^]。

缺省值可以为 NULL。过程定义可以指定如果参数值为 NULL 时应采取的操作 （请参见示例 3）。

output

表示该参数是返回参数。它的值可以返回给调用该过程的 execute 命令。使用返回参数向调用过程返回信息。

若要从多级嵌套过程返回参数值，则每个过程必须包含带参数名的 output 选项，同时包含调用最高级别过程的 execute。

output 关键字可以缩写为 out。

with recompile
表示每次执行过程时， Adaptive Server 都会创建新计划。当预计过程的执行不规则时，即需要新计划时，使用该可选子句。 with recompile 子句对扩展存储过程的执行没有任何影响。

with execute as
指定是以所有者还是调用方身份执行过程。以所有者身份执行时，对照过程所有者的特权检查过程内的所有操作。以调用方身份执行时，对照过程调用方的特权检查过程内的所有操作。

owner
以过程所有者身份检查运行时权限、执行 DDL 以及解析对象名称。也支持将 execute as definer 作为替代语法。

caller
以过程调用方身份检查运行时权限、执行 DDL 以及解析对象名称。也支持将 execute as invoker 作为替代语法。

SQL_statements
指定过程要进行的操作。可以包含任何数量和任何种类的 SQL 语句，create view、 create default、 create rule、 create procedure、 create trigger 和 use 除外。

create procedure SQL 语句通常包含控制流语言，其中包括以下一项或多项：declare、 if...else、 while、 break、 continue、 begin...end、 goto label、 return、 waitfor、 /* *comment* */。它们还可引用为过程定义的参数。

SQL 语句可以引用另一个数据库中的对象，只要是正确限定的即可。

external name
创建扩展存储过程。不能将 *number* 参数与 as external name 一起使用。

dll_name
指定动态链接库 (DLL) 或共享库 （包含实现扩展存储过程的函数）的名称。指定 *dll_name* 时可以不使用扩展名，也可使用平台特定的扩展名，例如在 Windows NT 上使用 *.dll*，在 Sun Solaris 上使用 *.so*。如果指定了扩展名，请用引号将整个 *dll_name* 引起来。

示例

示例 1 给定一个表名，过程 showind 会显示该表的名称和该表中任何列上的任何索引的名称和标识号：

```
create procedure showind @tabname varchar (30)
as
  select sysobjects.name, sysindexes.name, indid
  from sysindexes, sysobjects
  where sysobjects.name = @tabname
  and sysobjects.id = sysindexes.id
```

下面是执行 showind 的可接受的语法形式：

```
execute showind titles
execute showind @tabname = "titles"
```

或者，如果这是文件或批处理中的第一条语句：

```
showind titles
```

示例 2 如果用户不提供参数，则此过程将显示有关系统表的信息：

```
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
```

示例 3 此过程指定当参数为 NULL （即用户未提供参数）时要采取的操作：

```
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
```

示例 4 此过程将两个整数参数相乘，然后在 output 参数 *@result* 中返回它们的乘积：

```
create procedure mathtutor @mult1 int, @mult2 int,
  @result int output
as
select @result = @mult1 * @mult2
```

如果给过程传递三个整数，则执行过程时 select 语句将计算乘积并赋值，但不会显示返回参数：

```
mathtutor 5, 6, 32
 (return status 0)
```

示例 5 过程和 execute 语句都包含带一个参数名的 output，因此过程可以为调用方返回一个值：

```
declare @guess int
select @guess = 32
exec mathtutor 5, 6, @result = @guess output

 (1 row affected)
 (return status = 0)

Return parameters:

@result
-----------
         30
```

execute 语句中的输出参数 *@result* 和任何后续参数都必须以如下形式传递：

```
@parameter = value
```

- 无论返回参数值更改与否，都始终报告该值。
- *@result* 不必在调用批处理中声明，因为它是要传递到 `mathtutor` 的参数的名称。
- 虽然 *@result* 的更改值在 execute 语句中指派的变量 （此例为 *@guess*）中返回给调用方，但它将在其本身的标题 (*@result*) 下显示。

示例 6 可以在批处理或调用过程中的其它 SQL 语句中使用返回参数。本例说明如何在 execute 语句后的条件子句中使用 *@guess* 的值，方法是在过程调用时把它用另一个变量名 *@store* 存储。如果返回参数用于属于 SQL 批处理的 execute 语句，则在执行批处理中的后续语句前，将输出带有标题的返回值。

```
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!
```

示例 7 创建名为 xp_echo 的扩展存储过程，过程的输入参数是 *@in*，回应的输出参数是 *@out*。过程代码位于名为 xp_echo 的函数中，该函数编译并链接到一个名为 *sqlsrvdll.dll* 的 DLL 中：

```
create procedure xp_echo @in varchar (255),
        @out varchar (255) output
as external name "sqlsrvdll.dll"
```

示例 8 使用 execute as owner 子句创建过程。Jane 创建了过程，Bill 只需 execute 权限便可运行该过程。 Jane 创建并拥有 emp_interim 表。如果 Jane 没有 create table 权限，过程将失败：

```
create procedure p_emp
    with execute as owner as
    select * into emp_interim
    from jane.employee
grant execute on p_emp to bill
```

示例 9 使用 execute as caller 子句创建过程。 Jane 创建了过程， Bill 需要 execute 权限来运行该过程。Jane 同时拥有 p_emp 和 jane.employee。Bill 需要 jane.employee 的 select 权限。 Bill 创建并拥有 emp_interim 表。 Bill 必须具有 create table 权限：

```
create procedure p_emp
    with execute as caller as
    select * into emp_interim
    from jane.employee
grant execute on p_emp to bill
```

示例 10 使用引用名称未限定对象的 execute as owner 子句创建过程。Jane 创建了过程，Bill 执行过程。Adaptive Server 将搜索 Jane 拥有的名为 t1 的表。如果 jane.t1 不存在，Adaptive Server 将查找 dbo.t1。如果 Adaptive Server 将 t1 解析成 dbo.t1，则必须为 Jane 授予向 t1 插入内容的权限：

```
create procedure insert p
    with execute as owner as
    insert t1 (c1) values (100)
grant execute on insert p to bill
```

示例 11 使用引用名称未限定对象的 execute as caller 子句创建过程。Jane 创建了过程， Bill 执行过程。 Adaptive Server 将搜索 Bill 拥有的名为 t1 的表。如果 bill.t1 不存在， Adaptive Server 将查找 dbo.t1。如果 Adaptive Server 将 t1 解析成 dbo.t1，则必须为 Bill 授予向 t1 插入内容的权限：

```
create procedure insert p
    with execute as caller as
    insert t1 (c1) values (100)
grant execute on insert p to bill
```

示例 12 使用在其它具有完全限定名的数据库里调用嵌套过程的 execute as owner 子句创建过程。 Jane 创建了过程， Bill 执行过程。与 Jane 关联的登录名解析成 otherdb 中的用户 Jane。 Adaptive Server 检查 otherdb 中的用户 Jane 对 jim.p_child 是否具有 execute 权限。如果 jim.p_child 使用了 execute as owner 来创建，则以 Jim 的身份执行 p_child。如果 jim.p_child 使用了 execute as caller 或没用 execute as 子句来创建，则以 Jane 的身份执行 p_child：

```
create procedure p master
    with execute as owner
    as exec otherdb.jim.p_child
grant execute p master to bill
```

示例 13 使用在其它具有完全限定名的数据库里调用嵌套过程的 execute as caller 子句创建过程。Jane 创建了过程，Bill 执行过程。与 Bill 关联的登录名解析成 otherdb 中的用户 Bill。Adaptive Server 检查 otherdb 中的用户 Bill 对 jim.p_child 是否具有 execute 权限。如果 jim.p_child 使用了 execute as owner 来创建，则以 Jim 的身份执行 p_child。如果 jim.p_child 使用了 execute as caller 或没用 execute as 子句来创建，则以 Bill 的身份执行 p_child：

```
create procedure p master
    with execute as caller
    as exec otherdb.jim.p_child
grant execute on p master to bill
```

用法

- 为了避免看到因设置更改而引起的意外结果，请先将 set rowcount 0 作为初始语句运行，然后再执行 create procedure。set 的范围仅限于 create procedure 命令，一旦过程退出便会重置为以前的设置。
- 创建过程后，可以通过发出 execute 命令和过程名及任何参数来运行过程。如果过程是批处理中的第一条语句，则可以提供过程名而不使用关键字 execute。
- 可以用 sp_hidetext 隐藏过程的源文本，源文本存储在 syscomments 中。
- 如果存储过程批处理成功执行，则 Adaptive Server 将全局变量 *@@error* 设为 0。

限制

- 一个存储过程最多可以有 2048 个参数。
- 过程中局部变量与全局变量的最大数仅受可用内存的限制。
- 存储过程中的最大文本数是 16MB。
- 不能将 create procedure 语句与其它语句组合在同一批处理中。
- 虽然过程可以引用其它数据库中的对象，但只能在当前数据库中创建存储过程。过程中引用的大多数对象在创建过程时必须存在。但可以包括 drop table、create index 或 truncate table 等语句。即使创建过程时基础对象不存在， create procedure 语句中也可以包括这些语句。

  可以在过程中创建对象，然后引用它，只要对象在被引用之前创建即可。

  不能在过程中用 alter table 添加一列然后在过程中引用该列。
- 如果在 create procedure 语句中使用 select*，则过程不会选取任何已添加到表的新列（即使用 with recompile 选项来 execute）。必须 drop 该过程并重新进行创建。否则，当在两个表中都添加了新列时，过程中的 insert into table1 select * from table2 的 insert...select 语句就会导致错误结果。
- 在存储过程内，不能在创建某个对象 （包括临时表）并将其删除之后又以相同的名字创建新的对象。 Adaptive Server 是在执行而非编译存储过程时创建在该过程中定义的对象的。

警告！对数据库所做的某些更改 （例如删除和重新创建索引）可能导致对象 ID 发生更改。对象 ID 发生更改时，存储过程会自动重新编译，且大小会稍有增加。请为这种增加留出一些空间。

扩展存储过程

- 如果使用 as *external name* 语法，则 create procedure 将注册一个扩展存储过程 (ESP)。扩展存储过程执行过程语言函数而不是 Transact-SQL 命令。
- (Windows) ESP 函数不应调用 C 运行时信号例程。否则会导致 XP Server 失败，因为 Open Server™ 不支持 Windows NT 上的信号处理。
- 若要支持多线程， ESP 函数应该使用 Open Server srv_yield 函数，该函数挂起并重新安排 XP Server 线程，以允许执行相同优先级或较高优先级的其它线程。
- DLL 搜索机制与平台有关。在 Windows NT 上， DLL 文件名搜索的顺序：

a 装载应用程序的目录

b 当前目录

c 系统目录 (SYSTEM32)

d PATH 环境变量中所列目录

如果 DLL 不在前三个目录中，请设置 PATH 使其包含该 DLL 所在的目录。

在 UNIX 平台上，该搜索方法随特定平台的不同而不同。如果查找 DLL 或共享库失败，则它将搜索 *$SYBASE/lib*。

不支持绝对路径名。

execute as 存储过程

- set session authorization 语句不能在 execute as owner 存储过程内使用，即使该语句嵌在使用或没用 execute as 子句定义的嵌套过程内也如此。
- SQLJ 过程不支持 execute as 子句。
- 使用 execute as caller 创建的同一过程的过程高速缓存中的计划不在用户之间共享，因为过程中的对象必须解析为执行该过程的用户。因此，如果多位用户执行该过程，可能会增加过程高速缓存率。当用户再次执行该过程时，将重用特定用户的计划。

有关 execute as 存储过程的信息，请参见 《安全性管理指南》中的 *管理用户权限*。

系统过程

- 统管理员可以在 sybsystemprocs 数据库中创建新的系统过程。系统过程名必须以 “sp_” 开头。可以通过指定过程名称从任何数据库执行这些过程，而不必用 sybsystemprocs 数据库名称限定过程。有关创建系统过程的详细信息，请参见 《系统管理指南》。
- 系统过程结果随其执行的上下文的不同而不同。例如，执行 db_name() 系统函数的 sp_foo 会返回从中执行过程的数据库的名称。通过 pubs2 数据库执行时，将返回值 “pubs2”：

```
use pubs2
sp_foo
------------------------------
pubs2
```

通过 sybsystemprocs 执行时，将返回值 “sybsystemprocs”：

```
use sybsystemprocs
sp_foo
------------------------------
sybsystemprocs
```

嵌套过程

- 当一个存储过程调用另一个存储过程时，发生过程嵌套。
- 如果执行一个调用其它过程的过程，被调用过程可以访问由调用过程创建的对象。
- 嵌套级别在被调用过程开始执行时递增，并在被调用过程执行完成时递减。超过 16 层的最大嵌套值将导致事务失败。
- 可用过程名或用变量名替换真正的过程名来调用其它过程。
- 当前嵌套级别存储在 @@*nestlevel* 全局变量中。
- execute as 过程可与使用或没用 execute as 子句创建的嵌套过程嵌套

过程返回状态

- 存储过程可以返回一个称为**返回状态** 的整数值。返回状态要么说明过程成功执行，要么指定发生的错误的类型。
- 执行存储过程时，会自动返回相应的状态码。 Adaptive Server 当前返回以下状态码：

代码	含义
0	过程执行时没有发生错误
-1	缺失对象
-2	数据类型错误
-3	进程被选作死锁牺牲品
-4	权限错误
-5	语法错误
-6	杂类用户错误
-7	资源错误，如空间不足
-8	非致命内部问题
-9	达到系统限制
-10	致命内部不一致性
-11	致命内部不一致性
-12	表或索引损坏
-13	数据库损坏
-14	硬件错误

  代码 -15 到 -99 留做将来使用。

- 用户可以用 return 语句生成一个用户定义的返回状态。此状态可以是任何整数 （0 到 -99 的整数除外）。以下示例在书具有有效合同时返回 1，在所有其它情况下返回 2。

```
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)
```

- 如果在执行时发生了多个错误，将返回绝对值最高的代码。用户定义的返回值比系统定义的值优先级高。

对象标识符

- 若要更改存储过程的名称，请使用 sp_rename。
- 要更改扩展存储过程的名称，请删除该过程，重命名并重新编译支持函数，然后重新创建过程。
- 如果过程引用并非有效标识符的表名、列名或视图名，则必须在 create procedure 命令之前 set quoted_identifier on 并将每个这样的名称都用双引号引起来。执行过程时并不需要打开 quoted_identifier 选项。
- 如果重命名了过程所引用的任何对象，则必须删除并重新创建该过程。
- 在存储过程中，如果其他用户要使用该存储过程，则同 create table 和 dbcc 一起使用的对象名必须用其所有者名限定。例如，如果用户“mary”（拥有表 marytab）希望其他用户可以执行她的过程，则当该表与这些命令一起使用时，她应当在存储过程内限定该表的名字。这是因为对象名在过程运行时被解析。当另一个用户试图执行该过程时， Adaptive Server 查找用户 “mary”拥有的名为 marytab 的表，而不是查找执行该存储过程的用户拥有的名为 marytab 的表。

  这样，如果没有限定 marytab，且用户“john”尝试执行该过程，则 Adaptive Server 将查找过程所有者 （此例中为 “mary”）所拥有的名为 marytab 的表；如果该用户表不存在，则查找数据库所有者所拥有的表。例如，如果删除了表 mary.marytab，则过程将引用 dbo.marytab。

  在存储过程中同其它语句 （如 select 或 insert）一起使用的对象名无需限定，因为在过程编译时该名称就已被解析。

临时表和过程

- 如果临时表在当前会话中创建，则可以创建引用临时表的过程。过程退出时，在过程中创建的临时表将消失。请参见 《Transact-SQL 用户指南》。
- 系统过程 （如 sp_help）可使用临时表，但条件是必须从 tempdb 使用它们。

在过程中设置选项

可以在存储过程中使用 set 命令。大部分 set 选项在过程执行期间保持有效，然后返回到它们先前的设置。

然而，如果使用要求用户为对象所有者的 set 选项 （如 identity_insert），则不是对象所有者的用户将无法执行该存储过程。

获取有关过程的信息

- 若要获取被过程引用的对象的报告，请使用 sp_depends。
- 若要显示 create procedure 语句的文本 （存储在 syscomments 中），请以过程名作参数使用 sp_helptext。使用 sp_helptext 时，必须正在使用过程所驻留的数据库。若要显示系统过程的文本，请从 sybsystemprocs 数据库执行 sp_helptext。
- 若要查看系统扩展存储过程和它们支持的 DLL 的列表，请从 sybsystemprocs 使用 sp_helpextendedproc。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

创建过程时，不对过程引用的对象 （如表和视图）进行权限检查。因此，即使不能访问过程的对象，也可以成功创建过程。所有权限检查在用户执行该过程时进行。

执行过程时，对象上的权限检查取决于过程和所有被引用的对象是否由同一用户拥有。

- 如果过程的对象由不同的用户拥有，那么调用者必须得到可以直接访问这些对象的授权。例如，如果过程执行对用户不能访问的表的选择，过程的执行就会失败。
- 如果过程及其对象为同一用户所有，则应用特殊的规则。调用者自动拥有访问过程的对象的 “隐式权限”，即使调用者不能直接访问它们。不必授予用户直接访问表和视图的权限，可以给予他们有限制地访问存储过程的权限。这样，存储过程就可以是一种安全性机制。例如，过程的调用者可能只能访问表的某些行和列。请参见 《安全性管理指南》中的 “使用存储过程作为安全机制”。

以下介绍了取决于细化权限设置的 create procedure （以及在创建扩展过程时）的权限检查。

细化权限已启用	在启用细化权限的情况下，您必须具有 create procedure 特权。您必须具有 create any procedure 特权才能为其他用户运行 create procedure。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者或具有 create procedure 特权。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
11	create	create procedure	• *角色* - 当前活动角色 • *关键字或选项* - 对于 execute as owner，显示过程所有者名称和关键字 execute as owner。对于 execute as caller，显示过程调用者名称和关键字 execute as caller。 • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 begin...end, break, continue, declare, drop procedure, execute, goto label, grant, if...else, return, select, waitfor, while.

系统过程 sp_addextendedproc、 sp_helpextendedproc、 sp_helptext、 sp_hidetext、 sp_rename、 sp_help.

create procedure (SQLJ)

说明

通过将 SQL 包装添加到 Java 静态方法中来创建 SQLJ 存储过程。可以接受用户提供的参数并返回结果集和输出参数。

注释　注意：有关用于创建过程的 Transact- SQL 命令的语法和用法信息，请参见第 160 页的 “create procedure”。

语法

```
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
            ...]])]'
```

参数

sql_procedure_name

是过程的 Transact-SQL 名称，该名称必须符合标识符规则，并且不能是变量。指定所有者的名称，以创建由当前数据库中的其他用户拥有的另一个同名过程。 *owner* 的缺省值是当前用户。

in | out | inout

指定列出的参数的模式。 in 表示输入参数， out 表示输出参数， inout 表示参数既是输入参数又是输出参数。缺省模式是 in。

sql_parameter_name

是该过程的参数名。过程执行时，提供每个输入参数的值。参数是可选的， SQLJ 存储过程不需要带参数。

参数名必须符合标识符的规则。如果参数的值包含非字母数字字符，则必须用引号将其引起来。这包括数据库名或所有者名限定的对象名，因为它们包含一个句点。如果参数值以数字字符开头，则还必须用引号将其引起来。

sql_datatype [(*length*) | (*precision* [, *scale*])]
是参数的 Transact-SQL 数据类型。

sql_datatype 是 SQL 过程签名。

default
定义过程参数的缺省值。定义了缺省值后，在没有参数值的情况下也可以执行过程。缺省值必须是一个常量。如果过程使用带有关键字 like 的参数名，则缺省值可包括通配符 （%、 _、 [ ] 和 ^）。

缺省值可以为 NULL。过程定义可以指定参数值为 NULL 时采取某个操作。

modifies sql data
表示 Java 方法调用 SQL 操作，读取并修改数据库中的 SQL 数据。这是缺省的、同时也是唯一的实现方法。将它包含进来是为了在语法上与 ANSI 标准兼容。

dynamic result sets *integer*
指定 Java 方法可以返回 SQL 结果集。 *integer* 指定方法可以返回的最大结果集数。该值是实现定义的。

deterministic | not deterministic
支持该语法是为了与其他 SQLJ 兼容的供应商兼容。

language java
指定外部例程用 Java 编写。这是 SQLJ 存储过程所必需的子句。

parameter style java
指定在运行期传递给外部例程的参数是 Java 参数。这是 SQLJ 存储过程所必需的子句。

external
表示 create procedure 为使用 SQL 之外的编程语言编写的外部例程定义一个 SQL 名称。

name
指定外部例程 （Java 方法）的名称。指定的名称是一个字符串文字且必须用单引号引起来：

```
'java_method_name [ java_datatype
                  [{, java_datatype} ...]]'
```

java_method_name
指定外部 Java 方法的名称。

java_datatype
指定可映射的或其结果集可映射的 Java 数据类型。这是 Java 方法签名。

示例

示例 1 创建 SQLJ 过程 java_multiply，该过程将两个整数相乘并返回一个整数。

```
create procedure java_multiply (param1 integer,
      param2 integer, out result integer)
    language java parameter style java
    external name 'MathProc.multiply'
```

示例 2 返回始终大于 10 的值：

```
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)
```

另请参见 Transact-SQL create procedure 的示例。

用法

- 为了与 SQLJ ANSI 标准兼容， SQLJ create procedure 语法与 Transact-SQL create procedure 语法不同。Adaptive Server 以相同的方式执行每种存储过程。
- 为了避免看到因设置更改而引起的意外结果，请先将 set rowcount 0 作为初始语句运行，然后再执行 create procedure。set 的范围仅限于 create procedure 命令，一旦过程退出便会重置为以前的设置。
- 在一条 create procedure 语句中最多可以包含 31 个 in、inout 和 out 参数。
- 为了与 ANSI 标准相符合，不要在参数名前加 @ 符号。不过，从 isql 或其它非 Java 客户端执行 SQLJ 存储过程时，必须在参数名前加 @ 符号，这样可以保留命名顺序。

权限

创建过程时，不对过程引用的对象 （如表和视图）进行权限检查。因此，即使不能访问过程的对象，也可以成功创建过程。所有权限检查在用户执行该过程时进行。

执行过程时，对象上的权限检查取决于过程和所有被引用的对象是否由同一用户拥有。

- 如果过程的对象由不同的用户拥有，那么调用者必须得到可以直接访问这些对象的授权。例如，如果过程执行对用户不能访问的表的选择，过程的执行就会失败。

- 如果过程及其对象为同一用户所有，则应用特殊的规则。调用者自动拥有访问过程的对象的 “隐式权限”，即使调用者不能直接访问它们。不必授予用户直接访问表和视图的权限，可以给予他们有限制地访问存储过程的权限。这样，存储过程就可以是一种安全性机制。例如，过程的调用者可能只能访问表的某些行和列。

以下介绍了取决于细化权限设置的 create procedure （以及在创建扩展过程时）的权限检查。

细化权限已启用	在启用细化权限的情况下，您必须具有 create procedure 特权。您必须具有 create any procedure 特权才能为其他用户运行 create procedure。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者或具有 create procedure 特权。

另请参见

命令 create function (SQLJ), drop procedure.

系统过程 sp_depends, sp_help, sp_helpjava, sp_helprotect.

create proxy_table

说明

（仅限组件集成服务）创建代理表而不指定列列表。 CIS 通过从远程表获得的元数据派生列列表。

语法

```
create proxy_table table_name
        [external [table | directory | file]]
        at pathname
        [column delimiter "<string>"]
```

参数

table_name

指定后续语句要使用的本地代理表名称。 *table_name* 采用的形式为 *dbname.owner.object*，其中 *dbname* 和 *owner* 是可选的，分别表示本地数据库及所有者名。如果未指定 *dbname*，则表在本地数据库中创建；如果未指定 *owner*，则表由当前用户拥有。如果指定了 *dbname* 或 *owner* 两者之一，则必须将整个 *table_name* 放在引号中。如果只有 *dbname*，则需要为 *owner* 使用占位符。

external table

指定对象是一个远程表或视图。 external table 为缺省值，因此该子句是可选的。

external directory

指定对象是一个带路径的目录，格式如下：“*/tmp/directory_name* [;R]”，其中 “R” 表示 “递归”。

external file

指定对象是一个带路径的文件，格式如下：“*/tmp/filename*”。

at *pathname*

指定远程对象的位置。 *pathname* 采用的形式为 *server_name.dbname.owner.object*，其中：

- *server_name* - 包含远程对象的服务器的名称。
- *dbname* - （可选）包含此对象的远程服务器所管理的数据库的名称。
- *owner* - （可选）拥有远程对象的远程服务器用户的名称。
- *object* - 远程表或视图的名称。

string

列分隔符字符串可以是任何字符序列，但如果字符串的长度超出了 16 个字节，则只使用前 16 个字节。对映射到除文件之外的任何对象的代理表使用列分隔符会导致语法错误。

示例

此例创建了一个名为 t1 的代理表，它映射到远程表 t1。 CIS 从远程表派生列列表：

```
create proxy_table t1
at "SERVER_A.db1.joe.t1"
```

用法

- create proxy_table 是 create existing table 命令的变体。使用 create proxy_table 来创建代理表，但不要指定列的列表 （与 create existing table 不同）。 CIS 通过从远程表获得的元数据派生列列表。
- at 关键字所提供的位置信息与 sp_addobjectdef 所提供的信息相同。该信息存储在 sysattributes 表中。
- 如果远程服务器对象不存在，则拒绝该命令，并给出错误消息。
- 如果对象存在，则本地系统表被更新。每一列都被使用。获得表或视图的列及其属性。
- CIS 自动将列的数据类型转换成 Adaptive Server 数据类型。如果不能进行转换，则 create proxy_table 命令不允许定义表。
- 将提取远程服务器表的索引信息，并用于创建系统表 sysindexes 的行。这将用 Adaptive Server 的术语定义索引和键，并启用查询优化程序来考虑表中任何可能存在的索引。
- 定义了代理表之后，为表发出 update statistics 命令。这将允许查询优化程序在考虑连接顺序时做出正确选择。
- 如果由 *pathname* 标识的服务器不区分大小写 （例如 DB2 和 Oracle），则执行 create proxy_table *table_name* at *pathname* 时，将假定表名和列名的大小写与 *table_name* 的大小写相同。

  如果 *table_name* 是小写，则由不区分大小写的服务器返回的列 （通常为大写）以小写形式存储在 Adaptive Server 中。如果 *table_name* 是大写，则列名也以大写形式存储。如果 *table_name* 是大小写混合，则所有列名以其从远程节点收到时的形式存储。
- 临时表不支持 create proxy_table。
- 不能在同一批处理中结合使用 create proxy_table 语句和其它语句。
- 代理表仅存储元数据。因此，唯一使用的空间是在系统目录中创建条目的结果。假设每个表平均有两个索引，则一百个代理表估计要消耗 1MB 空间。
- create proxy table、create table at remote server 或 alter table 当前不支持 SQL 用户定义的函数。

  注释 注意：执行 SQL 函数需要使用语法 username.functionname()。
- 如果远程 Adaptive Server 表有一个或多个加密列， CIS 将更新 syscolumns 中的代理表元数据，以反映列的加密属性及其密钥 ID。

标准　　符合 ANSI SQL 的级别Transact-SQL 扩展。

权限　　create proxy_table 权限缺省情况下授予表所有者，并且不能移交。

审计　　sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
11	create	create procedure	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见　　**命令** create existing table, create table.

create role

说明

创建用户定义的角色；指定在创建特定角色时所允许的口令有效期、最小口令长度和最大失败登录次数。也可以在创建角色时将一个口令与该角色关联。

语法

create role *role_name* [with passwd "*password*"
[, {passwd expiration | min passwd length |
max failed_logins} *option_value*]]

参数

role_name

新角色的名称，必须对服务器唯一，而且符合标识符规则。*role_name* 不能为变量。

with passwd

添加一个用户激活角色所必须输入的口令。

password

是添加给角色的口令。口令的长度不得少于 6 个字符，并且必须符合标识符的规则。口令不能使用变量。

passwd expiration

password expiration interval 指定口令的有效期 （以天为单位）。可以是介于 0 和 32767 之间的任何值 （包括 0 和 32767）。例如，如果您在 2007 年 8 月 1 日上午 10:30 创建一个新登录名，其口令有效期为 30 天，那么该口令将在 2007 年 8 月 31 日上午 10:30 过期。

min passwd length

指定特定角色所需的最短口令长度。

max failed_logins

指定允许所指定登录名登录失败的尝试次数。

option_value

指定 passwd expiration、 min passwd length 或 max failed_logins 的值。

示例

示例 1 创建名为 doctor_role 的角色：

```
create role doctor_role
```

示例 2 创建名为 doctor_role 且口令为 “physician”的角色：

```
create role doctor_role with passwd "physician"
```

示例 3 将 passwd expiration 设置为 7 天。角色的口令将在经过指定期限（在此示例中为 7 天）后在上次更改口令的时间到期：

```
create role intern_role with passwd "temp244",
passwd expiration 7
```

示例 4 设置 intern_role 允许的最大失败登录次数：

```
create role intern_role with passwd "temp244"
max failed_logins 20
```

示例 5 设置 intern_role 的最短口令长度：

```
create role intern_role with passwd "temp244",
min passwd length 0
```

用法

- 从 master 数据库使用 create role。
- 如果为角色添加了口令，被授予担任此角色的用户必须指定口令才能激活角色。

  有关在创建后为角色添加口令的信息，请参见 alter role 命令。

  注释　注意：在 12.x 之前的版本中创建并附加到用户定义角色的口令不会过期。

- 角色名对于服务器必须是唯一的。
- 角色名不能与用户名相同。可以创建与用户同名的角色，但授予特权时， Adaptive Server 会将特权授予用户而不是角色，以便消除命名冲突。

  有关命名冲突的详细信息，请参见 grant role 命令。

限制

- 每个服务器会话可创建的最大角色数是 1024。不过，有 32 个角色被保留给 Sybase 系统角色，如 sa_role 和 sso_role。此外，用户定义的特殊角色 sa_serverprivs_role 由 Adaptive Server 创建。因此，每个服务器会话可创建的最大用户定义角色数是 991。
- 如果创建带有口令的角色，用户登录时缺省情况下不能激活该角色。如果被授予角色的用户需要在登录时缺省激活该角色，则不要创建带口令的角色。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 create role 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage roles 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sso_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
85	roles	create role、 drop role、 alter role、 grant role 或 revoke role	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter role, drop role, grant, revoke, set.

系统过程 sp_activeroles, sp_displaylogin, sp_displayroles, sp_helprotect, sp_modifylogin.

create rule

说明

为特定列或属于用户定义数据类型的任意列指定可接受值的域并创建访问规则。

语法

```
create [[and | or] access]] rule
        [owner.]rule_name
        as condition_expression
```

参数

access

指定将要创建访问规则。请参见 《系统管理指南》中的 “管理用户权限”。

rule_name

新规则的名称，该名称必须符合标识符规则，并且不能是变量。指定所有者的名称，以便在当前数据库中创建由其他用户拥有的另一个同名规则。 *owner* 的缺省值是当前用户。

condition_expression

指定定义规则的条件。它可以是任何在 where 子句中被视为有效的表达式，并可以包括算术运算符、关系运算符、 in、 like、 between 等。不过， *condition_expression* 不能引用任何列或其它数据库对象。可以包括不引用数据库对象的内置函数。

condition_expression 采用一个参数，该参数必须以 @ 符号为前缀并引用通过 update 或 insert 命令输入的值。编写规则时可以使用任何名称或符号来表示此值。将字符和日期常量用引号引起来，并在二进制常量前加上 “0x”。

示例

示例 1 创建名为 limit 的规则，该规则将 advance 的值限制为小于 $1000：

```
create rule limit
as @advance < $1000
```

示例 2 创建名为 pubid_rule 的规则，该规则将 pub_id 的值限制为 1389、0736 或 0877：

```
create rule pubid_rule
as @pub_id in ('1389', '0736', '0877')
```

示例 3 创建名为 picture 的规则，该规则将 value 的值限制为始终以指定的字符开始：

```
create rule picture
as @value like '_-%[0-9]'
```

用法

- 若要隐藏规则的文本，请使用 sp_hidetext。
- 若要重命名规则，请使用 sp_rename。

限制

- 只能在当前数据库中创建规则。
- 规则不适用于创建规则时数据库中已有的数据。
- 在单个批处理中， create rule 语句不能与其它语句一起使用。
- 不能将规则绑定到 Adaptive-Server-提供的数据类型或类型为 text、unitext、 image 或 timestamp 的列上。
- 在创建同名的新规则前必须删除旧规则，且在删除规则前必须解除绑定。请选择下列方法之一：

```
sp_unbindrule objname [, futureonly]
```

绑定规则

- 使用 sp_bindrule 将规则绑定到列或用户定义数据类型：

```
sp_bindrule rulename, objname [, futureonly]
```

- 绑定到用户定义数据类型的规则在您向此类型的列中插入值或更新该列时激活。规则并不测试插入此类变量中的值。
- 规则必须与列的数据类型兼容。例如，不能使用以下内容作为精确或近似数值列的规则：

```
@value like A%
```

  如果规则与所绑定到的列不兼容， Adaptive Server 会在试图插入值时 （而不是在您绑定它时）生成错误消息。
- 可以将规则绑定到列或数据类型，而不必解除现有规则的绑定。
- 绑定到列的规则总是优先于绑定到用户- 定义数据类型的规则，这与规则的绑定顺序无关。表 1-9 指出了将规则绑定到已存在规则的列和用户- 定义数据类型时的优先级。

表 1-9: 规则绑定优先级

新规则绑定到	旧规则绑定到用户定义数据类型	旧规则绑定到列
用户定义数据类型	新规则代替旧规则	无更改
列	新规则代替旧规则	新规则代替旧规则

- 规则并不替换列定义。如果将规则绑定到允许 NULL 值的列，即使规则文本中没有 NULL 值，也可以在列中显式或隐式插入 NULL 值。例如，如果创建一个指定 “@val in (1,2,3)” 或 “@amount > 10000” 的规则，并将此规则绑定到允许 NULL 值的表列，则仍然可以在该列中插入 NULL。列定义将覆盖该规则。

- 如果列既有缺省值又有关联的规则，缺省值必须在规则所定义的范围内。从不会插入与规则冲突的缺省值。每次尝试插入此类缺省值时， Adaptive Server 都会生成错误消息。
- 可以通过对 create table 语句使用 check 来定义规则，这样会创建完整性约束。但是，这些约束是专用于此表的；不能将它们绑定到其它表。有关完整性约束的信息，请参见 create table 和 alter table。
- 若要获取关于规则的报告，请使用 sp_help。
- 若要显示规则的文本 （存储在 syscomments 系统表中），请以规则名作为参数执行 sp_helptext。
- 将规则绑定到特定列或用户定义数据类型后，其 ID 就被存储到 syscolumns 或 systypes 系统表中。

标准

符合 ANSI SQL 的级别符合初级标准。

要使用符合 ANSI SQL 的语法创建规则，请使用 create table 语句的 check 子句。

权限

对 create rule 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 create rule 特权。您必须具有 create any rule 特权才能为其他用户运行 create rule。
细化权限已禁用	在禁用细化权限的情况下，您必须具有 create rule 特权，是数据库所有者或具有 sa_role 的用户。 您必须是具有 sa_role 的用户才能为其他用户使用 create rule。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
13	create	create rule	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter table, create default, create table, drop rule, drop table.

系统过程 sp_bindrule, sp_help, sp_helptext, sp_hidetext, sp_rename, sp_unbindrule.

create schema

说明

为数据库用户创建一个表、视图和权限的新集合。

语法

create schema authorization *authorization_name*
create_object_statement
[*create_object_statement* ...]
[*permission_statement* ...]

参数

authorization_name

是数据库中当前用户的名称。

create_object_statement

是 create table 或 create view 语句。

permission_statement

是 grant 或 revoke 命令。

示例

创建 newtitles、 newauthors、 newtitleauthors 表、 tit_auth_view 视图和相应的权限：

```
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
```

用法

- 只能在当前数据库中创建模式。

- *authorization_name* 也称为**模式授权标识符**，必须是当前用户的名称。
- 用户必须具有正确的命令权限（create table 和 create view）。如果用户根据另一数据库用户拥有的表创建视图，则会在某个用户试图通过此视图访问数据时而不是创建此视图时，检查对视图的权限。
- create schema 命令由以下对象终止：
  - 常规命令终结符（在 isql 中，缺省为“go”）。
  - 除 create table、create view、grant 或 revoke 之外的任何其它语句。
- 如果 create schema 语句中的任何语句失败，整个命令将作为一个单元回退，所有命令均不起作用。
- create schema 向系统表中添加关于表、视图和权限的信息。可使用相应的删除命令（drop table 或 drop view）删除用 create schema 创建的对象。不能用标准的 grant 和 revoke 命令在模式创建语句外更改在模式中授予或撤消的权限。
- 仅限集群 – 除了从同一本地临时数据库中的表之外，不能包括引用本地临时数据库中的列的参照完整性约束。当 create schema 尝试从其它数据库的表中创建对本地临时数据库中列的引用时会失败。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

create schema 可由任何数据库用户执行。用户必须具有创建模式中指定的对象的权限，即 create table 和 create view 权限。

另请参见

命令 create table, create view, grant, revoke.

实用程序 isql.

create service

说明

使用指定的名称和参数包装存储过程中提供的 SQL 语句。

语法

```
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])
```

参数

service-name

用户定义的 Web 服务的名称。该名称可以是任何对存储过程有效的名称。当使用该服务名称调用 drop service 命令时，将删除相应的存储过程。如果指定了一个现有的服务名称，则会引发例外。

security_option_item

- clear – 表示使用 HTTP 访问此 Web 服务。
- ssl – 表示使用 HTTPS 访问此 Web 服务。

path

字符串文字，用于指定要附加到访问 Web 服务的 URL 的用户定义路径。缺省情况下， *path* 为空值。

alias-name

字符串文字，用于指定用户定义的 Web 服务别名。

parameter_name

用户定义的 Web 服务的参数名称。 Web 服务执行时，需提供该参数的值。参数名前面必须带有 @ 符号，且必须符合标识符的规则。这些情况与 create procedure 命令的 *parameter_name* 参数相同。

SQL_statements

用户定义的 Web 服务要执行的操作。可以包含任何数量和任何类型的 SQL 语句，但 create view、 create default、 create rule、 create procedure、 create trigger 和 use 除外。

type

- soap – 意味着使用 HTTP POST 请求，同时还意味着必须符合所有的 SOAP 规则。数据以 SQL/XML 格式返回。
- raw – 表示在不需要变更或重新格式化的情况下发送输出。这意味着使用 HTTP GET 请求。调用的存储过程可以指定精确的输出。
- xml – 表示结果集输出以 SQL/XML 格式返回。这意味着使用 HTTP GET 请求。

注释　有关 Adaptive Server 存储过程与 SOAP 用户定义 Web 服务之间的数据类型映射，请参见 《Web 服务用户指南》。

示例

示例 1 此示例创建了一个类型为 raw 的、用户定义的 Web 服务 rawservice，以返回当前数据库版本。从 pubs2 数据库的 isql 命令行输入了 create service 命令：

```
1> use pubs2
2> go
1> create service rawservice type raw as select
'<html><h1>' + @@version + '</h1></html>'
2> go
```

然后部署新创建的用户定义的 Web 服务：

```
1> sp_webservices 'deploy', 'all'
2> go
```

新创建的用户定义 Web 服务的 Web 服务定义语言位于 `http://myhost:8181/services/pubs2?wsdl` 中。

新创建的用户定义的 Web 服务通过以下 URL 提供，其中 bob 和 bob123 是用户定义 Web 服务创建者的用户 ID 和口令：

```
http://myhost:8181/services/pubs2?method=rawservice&username=bob&password=bob123
```

输出 （Adaptive Server Enterprise 版本字符串）将显示在浏览器窗口中的 HTML `<h1>` 标记中。

示例 2 此示例创建了一个类型为 xml 的、用户定义的 Web 服务 xmlservice，以返回当前数据库版本。从 pubs2 数据库的 isql 命令行输入了 create service 命令：

```
1> use pubs2
2> go
1> create service xmlservice userpath "testing" type xml
as select @@version
2> go
```

然后部署新创建的用户定义的 Web 服务：

```
1> sp_webservices 'deploy', 'xmlservice'
2> go
```

注释 有关 deploy 选项的详细信息，请参见 《参考手册：过程》中的 sp_webservices。

用户定义的 Web 服务的 WSDL 位于：

```
http://myhost:8181/services/pubs2/testing?wsdl
```

可通过浏览器从以下 URL 中调用用户定义的 Web 服务，其中 bob 和 bob123 是用户定义 Web 服务创建者的用户 ID 和口令：

```
http://myhost:8181/services/pubs2/testing?method=xmlervice&
username=bob&password=bob123
```

输出以 XML 格式显示在浏览器窗口中。

示例 3 此示例向 SOAP 客户端提供了用户定义的 Web 服务以执行存储过程 sp_who。提供了一个参数，且指定了可选的 userpath 标识：

```
create service sp_who_service userpath
'myservices/args' type soap @loginname varchar(30) as
exec sp_who @loginname
```

这里创建了 Web 服务作为 pubs2 数据库中的 sp_who_service。在配置之后，可以从以下 URL 处访问它：

```
http://localhost:8181/pubs2/myservices/args/sp_who_service
```

服务的 WSDL 可在以下网址找到：

```
http://localhost:8181/pubs2/myservices/args?wsdl
```

WSDL 文件中描述的 Web 方法的签名为：

```
DataReturn[] sp_who_service (xsd:string username,
    xsd:string password, xsd:string loginname)
```

新服务是由具有参数 loginname （其类型为 varchar(30)）的 SOAP 客户端调用的。

用法

除以下不同外，结果存储过程的行为与使用 create procedure 命令创建的存储过程相同 （遵循现有存储过程的执行、复制、 sp_helptext 和重新编译规则，且可使用 isql 来执行）：

- 结果存储过程仅可用 drop service 命令删除，而不能用 drop procedure 命令删除。

- 将使用重新创建 create service 命令所需的 DDL 来填充 syscomments 表。
- 指定的服务名称不能创建存储过程组。

注释　要通过 Adaptive Server Web 服务引擎获得用户定义的 Web 服务，必须使用 sp_webservices 的 deploy 选项。然而，即使没有部署，也可以从 isql 访问用户定义的 Web 服务的存储过程。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

要使用 Web 服务，必须已被显式授予 execute 权限。

创建服务时对象的权限

创建 Web 服务时， Adaptive Server 不对服务引用的对象（如表和视图）进行权限检查。因此，即使不能访问服务的对象，也可以成功创建 Web 服务。所有权限检查都在用户执行该 Web 服务时进行。

Web 服务执行时的对象权限

执行 Web 服务时，对象的权限检查取决于该 Web 服务以及所有被引用的对象是否由同一用户拥有。

- 如果 Web 服务的对象由不同的用户拥有，那么调用者必须得到可以直接访问这些对象的授权。例如，如果 Web 服务选择用户不能访问的表中的内容，则 Web 服务的执行就会失败。
- 但是，如果 Web 服务及其对象为同一用户所有，则应用特殊的规则。调用者自动拥有访问 Web 服务的对象的 “隐式权限”，即使调用者不能直接访问它们。不必授予用户直接访问表和视图的权限，可以给予他们有限制地访问存储过程的权限。这样，存储过程就可以是一种安全性机制。例如， Web 服务的调用者可能只能访问表的某些行和列。

  《系统管理指南》中详细说明了隐式权限的规则。

以下介绍了取决于细化权限设置的 create service 权限检查。

细化权限已启用	在启用细化权限的情况下，您必须具有 create procedure 特权。您必须具有 create any procedure 特权才能为其他用户运行 create service。
细化权限已禁用	在禁用细化权限的情况下，您必须具有 create procedure 特权，是数据库所有者或具有 sa_role 的用户。 您必须是具有 sa_role 的用户才能为其他用户使用 create rule。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
11	create	create services	• *角色* – 当前活动角色 • *关键字或选项* – NULL • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* - set proxy 有效时的初始登录名

create table

说明

- 创建新表和 （可选）完整性约束。
- 定义计算列。
- 定义表、行和分区的压缩级别。
- 定义加密列及其解密缺省值。
- 定义表的分区属性。用于创建表分区的语法在下面单独列出。请参见分区的语法。

语法

```
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}}]
                lob_compression = off | compression_level
        [on segment_name]
        [partition_clause]
        [[external table] at pathname]
        [for load]
        compression_clause::=
            with compression = {none | page | row}
```

将此语法用于分区：

```
    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]...)]}
```

将此语法用于计算列

```
    create table [[database.[owner].] table_name
        (column_name {compute | as}
```

computed_column_expression
[[materialized] [not compressed]] | [not materialized]}

使用此语法创建虚拟散列表

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

参数

table_name

是新表的显式名称。如果该表位于另一数据库中，请指定数据库名；如果数据库中有多个具有该名称的表，请指定所有者的名称。 *owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

不能对表名使用变量。表名在数据库内必须唯一，且对所有者也必须唯一。如果设置了 set quoted_identifier on，就可以对表名使用分隔标识符。另外，它必须符合标识符的规则。有关有效表名的详细信息，请参见 《参考手册：构件块》第 4 章 “表达式、标识符和通配符”中第 349 页的 “标识符”。

通过在表名前加井号 (#) 或 “tempdb..”可以创建临时表。请参见 《参考手册：构件块》第 4 章 “表达式、标识符和通配符”中第 352 页的 “以 # 开头的表 （临时表）”。

可以在另一数据库中创建表，只要您被列在 sysusers 表中并具有对该数据库的 create table 权限。例如，可以使用以下两种方法之一在数据库 otherdb 中创建一个名为 newtable 的表：

```
create table otherdb..newtable
```

```
create table otherdb.yourname.newtable
```

column_name

是表中列的名称。它在表中必须是唯一的。如果设置了 set quoted_identifier on，就可以为列使用分隔标识符。另外，它必须符合标识符的规则。有关有效列名的详细信息，请参见 《参考手册：构件块》第 4 章 “表达式、标识符和通配符”。

datatype

是列的数据类型。可接受系统数据类型或用户定义的数据类型。某些数据类型需要在括号中使用长度 *n*：

```
datatype (n)
```

其它类型要求使用精度 *p* 和标度 *s*：

datatype (*p*,*s*)

有关详细信息，请参见 《参考手册：构件块》第 1 章 “系统数据类型和用户定义的数据类型”。

如果数据库中启用了 Java， *datatype* 可以是安装在数据库中的 Java 类的名称，这个类可以是系统类或用户定义的类。有关详细信息，请参见 《Adaptive Server Enterprise 中的 Java》。

default

指定列的缺省值。如果您指定了缺省值，而用户在插入数据时没有为列提供值， Adaptive Server 就会插入缺省值。缺省值可以是常量表达式、内置值 （插入执行插入操作的用户的名称）或者是 null （插入空值）。Adaptive Server 以 *tabname_colname_objid* 为格式生成缺省名称，其中 *tabname* 是表名的前 10 个字符， *colname* 是列名的前 5 个字符，而 *objid* 是缺省对象 ID 号。使用 IDENTITY 属性为列声明的缺省值不会影响列值。

可以在不引用数据库对象的 create table 语句的 default 部分引用全局变量。但不能在 create table 的 check 部分使用全局变量。

constant_expression

是用作列的缺省值的常量表达式。它不能包含全局变量、任何列的名称或其它数据库对象，但可以包括不引用数据库对象的内置函数。缺省值必须与该列的数据类型兼容，否则在试图插入缺省值时， Adaptive Server 会生成数据类型转换错误。

user | null

指定如果用户不提供值， Adaptive Server 应插入用户名或 NULL 值作为缺省值。对于 user，列的数据类型必须是 char (30) 或 varchar (30)。对于 null，列必须允许空值。

identity

表示该列具有 IDENTITY 属性。数据库中的每个表都可以具有一个以下数据类型的 IDENTITY 列：

- 精确 numeric 类型，标度为 0，或
- 任意整数数据类型，包括有符号或无符号 bigint、 int、 smallint 或 tinyint。

IDENTITY 列不能更新，也不允许有空值。

IDENTITY 列用于存储由 Adaptive Server 自动生成的序列号，如发票编号或职员编号。 IDENTITY 列的值唯一地标识表的每一行。

null | not null
指定在无缺省值的情况下，Adaptive Server 在数据插入过程中的行为。

null 指定如果用户没有提供值，则 Adaptive Server 分配一个空值。

not null 指定如果没有缺省值，则用户必须提供一个非空值。

Bit 类型列的属性必须始终为 not null。

如果不指定 null 或 not null，缺省情况下 Adaptive Server 将使用 not null。不过，为了使此缺省值与 SQL 标准兼容，可以使用 sp_dboption 对它进行切换。

in row
指示 Adaptive Server 凡是在数据页中有足够空间的情况下都将 LOB 列中的数据存储为行内。 LOB 列的数据要么完全存储为行内，要么完全存储为行外。

length
（可选）指定 LOB 列数据可以存储为行内的最大大小。任何大于此值的数据都存储为行外，而等于或小于 *length* 的数据都存储为行内（只要页上有足够的空间）。

如果您不指定 *length*， Adaptive Server 会将数据库范围设置用于行内长度。

off row
（可选）提供在行外存储 LOB 列的缺省行为。 Adaptive Server 会对您的新表采用此行为，除非您指定了 in row。如果您不指定 off row 子句，而且设置了数据库范围的行内长度，create table 就会将 LOB 列创建为行内 LOB 列。

off row | in row
指定 Java-SQL 列是和行分开存储 (off row) 还是存储在行中直接分配的存储区 (in row)。

缺省值是 off row。请参见 《Adaptive Server Enterprise 中的 Java》。

size_in_bytes
指定行内列的最大大小。根据数据库服务器的页大小和其它变量，存储在行内的对象最多可占据约 16K 字节的空间。缺省值为 255 个字节。

constraint *constraint_name*

引入完整性约束的名称。

constraint_name 是约束的名称。它必须符合标识符的规则，并且在数据库中是唯一的。如果不指定引用或检查约束的名称， Adaptive Server 会生成 *tabname_colname_objectid* 格式的名称，其中：

- *tabname* – 是表名的前 10 个字符
- *colname* – 是列名的前 5 个字符
- *objectid* – 是约束的对象 ID 号

如果没有为唯一约束或主键约束指定名称， Adaptive Server 会生成格式为 *tabname_colname_tabindid* 的名称，其中 *tabindid* 是表 ID 和索引 ID 的字符串并置。

unique

约束所指定列中的值，使得任两行都不能具有相同的值。此约束将创建一个唯一索引，该索引仅在使用 alter table 删除了约束后才能被删除。

primary key

约束所指定列中的值，使得任两行都不能具有相同的值，因此值不能为 NULL。此约束将创建一个唯一索引，该索引仅在使用 alter table 删除了约束后才能被删除。

clustered | nonclustered

指定由 unique 或 primary key 约束创建的索引是聚簇索引还是非聚簇索引。 clustered 是主键约束的缺省值；nonclustered 是唯一约束的缺省值。每个表只能有一个聚簇索引。有关详细信息，请参见 create index。

asc | desc

指定为约束创建的索引以每列的升序还是降序创建。缺省设置是按升序排列。

fillfactor

指定 Adaptive Server 对现有数据创建新索引时使每一页达到的填满程度。 fillfactor 百分比仅在创建索引时使用。随着数据的变化，页不会维持在某个特定的填充程度。

fillfactor 的缺省值是 0，这个值在您未在 create index 语句中包括 with fillfactor 时使用 （除非使用 sp_configure 更改了该值）。指定 fillfactor 时，请使用介于 1 和 100 之间的值。

fillfactor 值为 0 可创建具有完全填充页的聚簇索引及具有完全填充叶页的非聚簇索引。它在聚簇索引和非聚簇索引的索引 B 树内都保留适量的空间。通常不需要更改 fillfactor。

如果将 fillfactor 设置为 100，Adaptive Server 在创建聚簇索引和非聚簇索引时，会使每页的占满度达到 100%。将 fillfactor 的值设置为 100 仅对只读表 （永不向其中添加数据的表）才有意义。

如果 fillfactor 的值小于 100（0 除外，0 是特殊情况），Adaptive Server 在创建新索引时不会填满每页。创建表的索引时，如果表最终将包含大量数据，那么将 fillfactor 的值设置为 10 可能是一种明智的选择，但较小的 fillfactor 值会导致每个索引 （或索引和数据）占用较多的存储空间。

如果启用了 CIS，则不能对远程服务器使用 fillfactor。

警告！ 使用 fillfactor 创建聚簇索引会影响数据占用的存储空间，因为创建聚簇索引时 Adaptive Server 会重新分配数据。

---

decrypt_default 能让 sso 指定向没有加密列解密权限的用户返回的值。解密缺省值将被替换为通过 select 语句检索到的 text、image 或 unitext 列。

max_rows_per_page

限制数据页和索引的叶级页上的行数。与 fillfactor 不同，max_rows_per_page 的值在插入或删除数据时保持不变。

如果不指定 max_rows_per_page 的值， Adaptive Server 将在创建表时使用值 0。表和聚簇索引的值在 0 到 256 之间。非聚簇索引的每页最大行数取决于索引键的大小；如果指定的值过高， Adaptive Server 会返回错误消息。

max_rows_per_page 值为 0 可创建具有完全填充页的聚簇索引及具有完全填充叶页的非聚簇索引。它在聚簇索引和非聚簇索引的索引 B 树中都保留适量空间。

对 max_rows_per_page 使用低值可减少经常访问的数据的锁争用。然而，使用低值也会导致 Adaptive Server 创建具有不完全填充页的索引，使用更多存储空间，并可能导致更多的页面拆分。

如果启用了 CIS 并创建了代理表，则会忽略 max_rows_per_page。代理表不包含任何数据。如果用 max_rows_per_page 创建了一个表，然后创建了代理表引用此表，那么当您通过代理表进行 insert 或 delete 时，将应用 max_rows_per_page 限制。

reservepagegap = *num_pages*

指定填充页与空白页的比率，扩充 I/O 分配操作过程中要保留该比率的空白页。对于每个指定的 *num_pages*，都会预留一个空白页供表将来扩展之用。有效值为 0 到 255。缺省值为 0。

dml_logging = {full | minimal}

确定 insert、update 和 delete 操作以及某些形式的批量插入的日志记录量。以下值之一：

- full – Adaptive Server 记录所有事务
- minimal – Adaptive Sever 不记录行或页更改

deferred_allocation

将表或索引的创建延迟到需要表时。在第一次 insert 时创建延迟表。

immediate_allocation

启用 sp_dboption 'deferred table allocation' 后显式创建表。

on *segment_name*

当与 constraint 选项一起使用时，指定要在已命名的段上创建索引。在使用 on *segment_name* 选项之前，必须使用 disk init 初始化设备，并使用 sp_addsegment 将段添加到数据库。要获得数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。

如果指定 clustered 并使用 on *segment_name* 选项，则整个表都将迁移到您指定的段，因为索引的叶级包含实际的数据页。

references

为参照完整性约束指定列列表。只能为一个列约束指定一个列值。通过将此约束包括到引用其它表的表中，插入到*引用*表中的任何数据都必须已经存在于*被引用*表中。

若要使用此约束，必须对被引用表拥有 references 权限。被引用表中的指定列必须由唯一索引 （由 unique 约束或 create index 语句创建）约束。如果未指定任何列，则在被引用表的适当列中必须有一个 primary key 约束。而且，引用表列的数据类型必须和被引用表列的数据类型匹配。

ref_table

是包含被引用列的表的名称。您可以引用其它数据库中的表。约束可引用最多 192 个用户表和内部生成的工作表。

ref_column

是被引用表中的列的名称。

match full

指定如果引用行的引用列中的所有值均为：

- 空值 – 参照完整性条件为真。
- 非空值 – 如果被引用行中每个对应的列在被引用表中都相等，则参照完整性条件为真。

如果不属于上述任何一种情况，则参照完整性条件在以下条件成立时为假：

- 所有的值都是非空值且不相等，或者
- 引用行的引用列中的某些值是非空值，而其它值为空。

check (*search_condition*)

指定列值的 check 约束以及 Adaptive Server 对表中的所有行都实施的 *search_condition* 约束。可以指定 check 约束作为表或列约束；create table 允许一个列定义中有多个 check 约束。

虽然可以在 create table 语句的 default 部分引用全局变量，但是不能在 check 部分使用它们。

这些约束可以包括：

- 用 in 引入的一系列常量表达式
- 由 like 引入的一组条件，可以包含通配符

列和表检查约束可引用表中的任何列。

表达式可包括算术运算符和函数。 *search_condition* 不能包含子查询、集合函数、主变量或参数。

encrypt [with *key_name*]

创建加密列。如果密钥在另一数据库中，则指定数据库名称。如果 *key_name* 对于数据库不唯一，则指定所有者的名称。 *owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

表的创建者必须对密钥具有 select 权限。如果不提供 *key_name*，Adaptive Server 将在数据库中查找缺省密钥。

keyname 标识使用 create encryption key 创建的密钥。表创建者必须对 *keyname* 具有 select 权限。如果未提供 *keyname*，Adaptive Server 将查找通过在 create encryption key 或 alter encryption key 中使用 as default 子句创建的缺省密钥。

有关支持的数据类型的列表，请参见 《加密列用户指南》中的 “加密数据”。

decrypt_default *constant_expression*

指定此列为没有解密权限的用户返回缺省值， *constant_expression* 是 Adaptive Server 对 select 语句返回的常量值，而不是解密值。该值只能在可空列中为 NULL。如果解密缺省值无法转换为列的数据类型，则仅当 Adaptive Server 执行查询时，它才会捕捉转换错误。

compression = *compression_level* | not compressed

指示行中的数据是否压缩以及压缩级别。

foreign key

指定列出的列是该表中的外键，这些外键的目标键为随后的 references 子句中列出的列。foreign key 语法只允许用于表级约束，不允许用于列级约束。

next_column | *next_constraint*
表示可以使用与列定义或表约束定义相同的语法包括附加列定义或表约束 （以逗号分隔）。

lock datarows | datapages | allpages
指定要对表使用的锁定方案。缺省值是对配置参数 lock scheme 的全服务器范围的设置。

exp_row_size = *num_bytes*
指定所需行宽；仅适用于 datarows 和 datapages 锁定方案，且仅适用于具有可变长度行的表。有效值可以是 0、 1 以及介于表的最小和最大行长度之间的任何值。缺省值为 0，表示应用服务器范围的设置。

identity_gap *value*
指定表的标识间隔。该值只替换此表的系统标识间隔设置。

value 是标识间隔量。有关设置标识间隔的详细信息，请参见第 228 页的“IDENTITY 列”。

transfer table [on | off]
将表标记为进行增量传输。此参数的缺省值为 off。

dml_logging
确定 insert、update 和 delete 操作以及某些形式的批量插入的日志记录量。以下值之一：

- full – Adaptive Server 记录所有事务
- minimal – Adaptive Sever 不记录行或页更改

compression
指示在表级或分区级设置压缩的级别。指定分区的压缩级别会覆盖表的压缩级别。 Adaptive Server 仅在配置了压缩的分区中压缩各个列。

- none - 不压缩该表或分区中的数据。对于分区， none 指示该分区中的数据保持不压缩，即使将表压缩改为 row 或 page 压缩也是如此。
- row – 压缩单个行中的一个或多个数据项。只有在压缩形式比非压缩形式更节省空间时， Adaptive Server 才会用 row 压缩形式存储数据。在分区级或表级设置 row 压缩。
- page – 当页面已满时，则会使用页级压缩来压缩现有的采用行压缩形式的数据行，从而创建页级字典、索引和字符编码条目。在分区级或表级设置 page 压缩。

  Adaptive Server 只有已经在行级压缩数据后才会在页级压缩数据，因此，将压缩级别设置为 page 意味同时进行 page 和 row 压缩。

lob_compression = off | *compression_level*

决定表的压缩级别。如果您选择 off，表将不具有 LOB 压缩。

compression_level

表压缩级别。压缩级别有：

- 0 – 不压缩行。
- 1 到 9 - Adaptive Server 使用 ZLib 压缩。通常，压缩级别数字越大，Adaptive Server 压缩 LOB 数据的程度就更大，压缩和非压缩数据之间的比率就越大（也就是说，压缩数据与非压缩数据大小相比，节省的空间量就越大（以字节为单位））。

  但是，压缩量取决于 LOB 内容，压缩级别越高，进程的 CPU 占用率就越高。也就是说，级别 9 提供的压缩率最高，但 CPU 使用率也最高。
- 100 – Adaptive Server 使用 FastLZ 压缩。这是 CPU 使用率最低的压缩率；通常用于较少的数据量。
- 101 – Adaptive Server 使用 FastLZ 压缩。与值为 100 时相比，值为 101 时的 CPU 占用率略高，但压缩率也略高。

压缩算法忽略不使用 LOB 数据的行。

on *segment_name*

指定要在其上放置表的段的名称。使用 on *segment_name* 时，必须已经用 create database 或 alter database 将逻辑设备指派给了数据库，且必须已经用 sp_addsegment 在数据库中创建了段。要获得数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。

当用于分区时，指定将在其上放置分区的段。

external table

指定对象是远程表还是视图。缺省值为 external table，因此可指定也可不指定该选项。

for load

创建仅可用于 bcp in 和 alter table unpartition 操作的表。可对使用 for load 创建的表使用 row_count()。

partition by range

指定要根据一个或多个分区列中指定的范围值对记录分区。

column_name

在 *partition_clause* 中使用时，指定一个分区键列。

partition_name

指定要在其上存储表记录的新分区的名称。分区名称在表或索引上的分区组内必须唯一。如果启用 set quoted_identifier，则分区名称可以是分隔标识符。否则，分区名称必须是有效的标识符。

如果省略 *partition_name*， Adaptive Server 将创建一个格式为 *table_name_patition_id* 的名称。Adaptive Server 将截断超过允许的最大长度的分区名称。

on *segment_name*

在 *partition_clause* 中使用时，指定将在其上放置分区的段。在使用 on *segment_name* 选项之前，必须使用 disk init 初始化设备，并使用 sp_addsegment 系统过程将段添加到数据库。要获得数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。

values <= constant | MAX

指定命名分区的上限值 （包括边界值）。为最高分区上限指定常量值将在表上施加隐式完整性约束。关键字 MAX 指定给定数据类型的最大值。

partition by hash

指定要按系统提供的散列函数对记录分区。该函数计算分区键的散列值，这些分区键指定将记录分配到的分区。

partition by list

指定要按命名列中指定的实际值对记录分区。只有一个列能够对列表分区表分区。最多可以为每个分区指定 250 个不同列表值。

partition by round-robin

指定要按顺序方式对记录分区。循环分区表没有分区键。用户和优化程序都不知道特定记录的分区。

at *pathname*

指定远程对象的位置。使用 at *pathname* 子句会导致创建一个代理表。

pathname 的格式为 *server_name.dbname.owner.object;aux1.aux2*，其中：

- *server_name* （必需） - 是包含远程对象的服务器的名称。
- *dbname* （可选）是包含此对象的远程服务器所管理的数据库的名称。
- *owner* （可选）是拥有远程对象的远程服务器用户的名称。
- *object* （必需）是远程表或视图的名称。
- *aux1.aux2* （可选） - 是在执行 create table 或 create index 命令期间传递给远程服务器的字符串。该字符串仅在服务器是 db2 类时使用。 aux1 是在其中放置表的 DB2 数据库， aux2 是在其中放置表的 DB2 表空间。

{compute | as}

可交替使用的保留关键字，用于表明列是计算列。

computed_column_expression

是任意有效的 T-SQL 表达式，该表达式不包含来自其它表、局部变量、集合函数或子查询的列。它可以是由一个或多个运算符连接起来的列名、常量、函数、全局变量、 case 表达式或它们的组合。除虚拟计算列引用实现计算列以外，不能在计算列之间进行交叉应用。

materialized | not materialized

指定是否实现计算列并将其物理地存储在表中。如果没有指定关键字，则计算列缺省为 not materialized，因此不物理地存储在表中。

using clustered

指示您要创建虚拟散列表。列列表将被视为此表的键列。

column_name [asc | desc]

由于行基于其散列函数放置，因此不能将 [asc | desc] 用于散列区域。如果提供虚拟散列表的键列顺序，该顺序将只用于溢出聚簇区域。

hash_factor

用于虚拟散列表的散列函数所必需的参数。对于散列函数，每个键列都需要一个散列因子。这些因子与键值一起使用，以便为特定行生成散列值。

with max *num_hash_values* key

可以使用的最大散列值数。定义此散列函数输出的上限。

示例

示例 1 使用 *@@spid* 全局变量与缺省参数创建 foo 表：

```
create table foo (
      a      int
    , b      int      default @@spid
)
```

示例 2 创建 titles 表：

```
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
)
```

示例 3 使用 for load 创建表 mytable：

```
create table mytable (
      col1    int
    , col2    int
    , col3    (char 50)
)
partitioned by roundrobin 3 for load
```

在取消分区之前，新表不可用于任何用户活动。

1　使用 bcp in 将数据装载到 mytable。

2　取消对 mytable 的分区。

现在，该表可用于任何用户活动。

示例 4 创建 compute 表。因为表名和列名（max 和 min）是保留字，所以用双引号引起来。因为 total score 列名中包含嵌入的空格，所以用双引号引起来。创建此表之前，必须先执行 set quoted_identifier on。

```
create table "compute" (
      "max"            int
    , "min"            int
    , "total score"    int
)
```

示例 5 在一步中创建具有唯一约束的 sales 表和聚簇索引。（在 pubs2 数据库安装脚本中，有单独的 create table 和 create index 语句）：

```
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)
)
```

示例 6 创建具有两个参照完整性约束和一个缺省值的 salesdetail 表。有一个名为 salesdet_constr 的表级参照完整性约束和 title_id 列上的一个无指定名称的列级参照完整性约束。两个约束都指定被引用表 （titles 和 sales）中具有唯一索引的列。带有 qty 列的 default 子句指定 0 作为其缺省值。

```
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)
)
```

示例 7 创建在 pub_id 列上具有检查约束的 publishers 表。此列级约束可用于代替 pubs2 数据库中的 pub_idrule。

```
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
)
```

示例 8 指定 ord_num 列作为 sales_daily 表的 IDENTITY 列。首次向表中插入行时，Adaptive Server 为 IDENTITY 列指派值 1。以后每次插入时，此列的值都会增加 1。

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

```
    , ord_amt          money             null
)
```

示例 9 为 new_titles 表指定数据页锁定方案和所需行宽 200：

```
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
```

示例 10 指定 datarows 锁定方案并将 reservepagegap 值设置为 16，以使扩充 I/O 操作为每 15 个填充页留出 1 个空白页：

```
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
```

示例 11 创建名为 big_sales 并且记录日志最少的表：

```
create table big_sales (
      storid        char(4)        not null
    , ord_num       varchar(20)    not null
    , order_date    datetime       not null
)
with dml_logging = minimal
```

示例 12 创建一个名为 im_not_here_yet 的延迟表：

```
create table im_not_here_yet (
col_1 int,
col_2 varchar(20)
)
with deferred_allocation
```

示例 13 创建一个名为 mytable 的表，该表使用锁定方案数据行并允许增量传输：

```
create table mytable (
      f1 int
    , f2 bigint not null
    , f3 varchar (255) null
)
lock datarows
with transfer table on
```

示例 14 创建一个名为 genre、采用行级压缩的表：

```
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
```

示例 15 在段 seg1、 seg2 和 seg3 上创建一个名为 sales 的表，且 seg1 上具有压缩：

```
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)
```

示例 16 创建 email 表，该表使用 LOB 压缩级别 5：

```
create table email (
      user_name      char (10)
    , mailtxt        text
    , photo          image
    , reply_mails    text)
    with lob_compression = 5
```

示例 17 创建一个唯一聚簇索引所支持的约束；索引顺序对 stor_id 为升序，对 ord_num 为降序：

```
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)
)
```

示例 18 在远程服务器 SERVER_A 上创建一个名为 t1 的表，并创建一个映射到该远程表的名为 t1 的代理表：

```
create table t1 (
      a    int
    , b    char (10)
)
at "SERVER_A.db1.joe.t1"
```

示例 19 创建一个名为 employees 的表。 name 的类型为 varchar，home_addr 是类型为 Address 的 Java-SQL 列，而 mailing_addr 是类型为 Address2Line 的 Java-SQL 列。Address 和 Address2Line 都是安装在数据库中的 Java 类：

```
create table employees (
      name            varchar (30)
    , home_addr       Address
    , mailing_addr    Address2Line
)
```

示例 20 创建一个带 identity 列的名为 mytable 的表。标识间隔设置为 10，表示将在内存中以十个 ID 号的块为单位分配 ID 号。如果服务器出现故障或没有等待就关闭，那么分配给行的最后一个 ID 号与分配给行的下一个 ID 号之间的最大间隔为 10 个编号：

```
create table mytable (
      IdNum    numeric (12,0)    identity
)
with identity_gap = 10
```

示例 21 创建表 my_publishers，该表根据 state 列中的值按列表分区。有关创建表分区的详细信息，请参见 《Transact-SQL 用户指南》。

```
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
)
```

示例 22 创建表 fictionsales，该表根据 date 列中的值按范围分区：

```
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
)
```

示例 23 创建表 currentpublishers，该表按循环分区：

```
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)
```

示例 24 创建表 mysalesdetail，该表根据 ord_num 列中的值按散列分区：

```
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
)
```

示例 25 创建名为 mytitles 的表，该表具有一个实现计算列：

```
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
)
```

示例 26 创建一个员工表，该表具有一个可为空的加密列。 Adaptive Server 使用数据库缺省加密密钥来加密 ssn 数据：

```
create table employee_table (
      ssn             char(15)    null
      encrypt name    char(50)
    , deptid          int
)
```

示例 27 创建具有用于信用卡数据的加密列的客户表：

```
create table customer (
ccard char(16) unique
   encrypt with cc_key
   decrypt_default 'XXXXXXXXXXXXXXXX', name char(30)
)
```

ccard 列具有唯一约束并将 cc_key 用于加密。由于存在 decrypt_default 指示符，当没有解密权限的用户选择 ccard 列时，Adaptive Server 将返回值“XXXXXXXXXXXXXXXX”，而不是实际数据。

示例 28 创建一个表，该表将 description 指定为 300 字节长的行内 LOB 列，将 notes 指定为没有指定长度的行内 LOB 列 （继承行外存储的大小），将 reviews 列指定为无论条件如何都在行外存储：

```
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
)
```

示例 29 在 pubs2 数据库的 order_seg 段中创建名为 orders 的虚拟散列表：

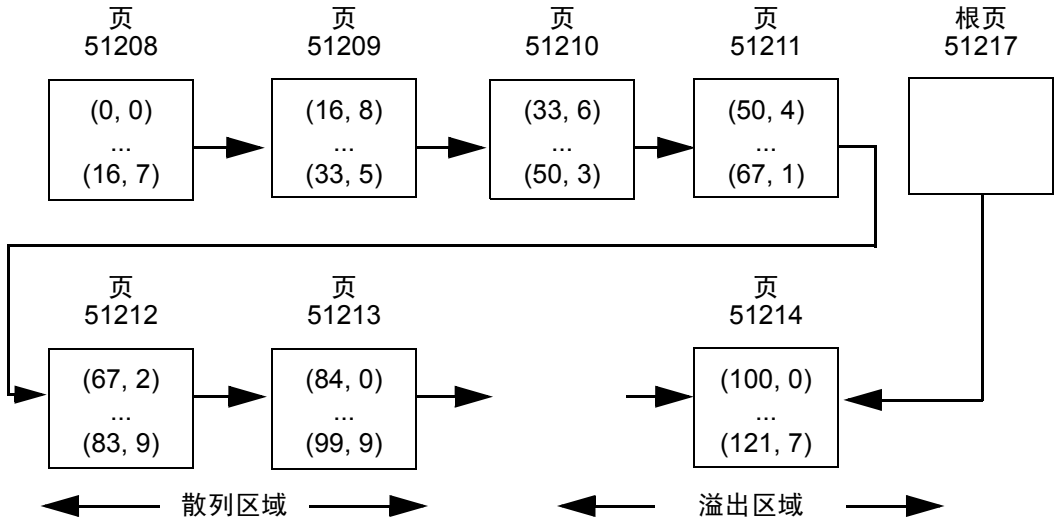
```
create table orders(
      id    int
```

```
    , age    int
    , primary key using clustered (id,age) = (10,1) with
max 1000 key
)
on order_seg
```

数据布局为：

- order_seg 段开始于 ID 为 51200 的页。
- 第一个数据对象分配映射 (OAM) 页的 ID 为 51201。
- 每页的最大行数为 168。
- 行宽为 10。
- 溢出聚簇区域的根索引页为 51217。

图 1-1：示例的数据布局



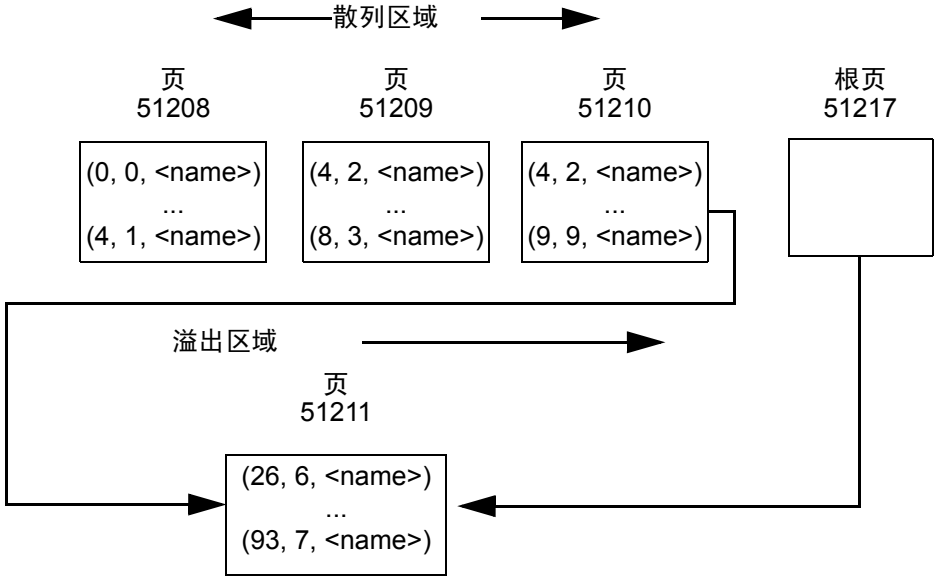
示例 30 在 pubs2 数据库的 order_seg 段中创建名为 orders 的虚拟散列表：

```
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
```

数据布局为：

- order_seg 段开始于 ID 为 51200 的页。
- 第一个数据 OAM 页的 ID 为 51201。
- 每页的最大行数为 42。
- 行宽为 45。
- 溢出聚簇区域的根索引页为 51217。

图 1-2：示例的数据布局



用法

- create table 创建表和可选完整性约束。除非在 create table 语句中指定了不同的数据库，否则将在当前打开的数据库中创建表。只要您被列在 sysusers 表中并在数据库中具有 create table 权限，您就可以在另一数据库中创建表或索引。
- 分配给表和索引的空间增量为一次 1 个扩充，或 8 页。每当一个扩充填满后，就会分配另一个扩充。若要查看表所分配和使用的空间量，请使用 sp_spaceused。
- 行内 Java 列的最大长度由根据表的方案、锁定风格和页大小决定的可变长度列的最大大小决定。
- create table 在创建表之前会针对检查约束进行错误检查。
- 当从带有一个定义为 char (*n*) NULL 的列的 CIS 中使用 create table 时， CIS 会在远程服务器上将该列创建为 varchar (*n*)。

- 在索引列名后使用 asc 和 desc 关键字指定索引的排序顺序。通过创建索引使各列按照在查询的 order by 子句中指定的顺序排列，可以避免在查询处理过程中进行排序。
- 如果应用程序在 DOL 锁定表中插入短行，并且稍后对其进行更新使得长度增加，则可使用 exp_row_size 减少将 DOL 锁定表中的行转移到新位置的次数。
- at 关键字所提供的位置信息与 sp_addobjectdef 所提供的信息相同。该信息存储在 sysattributes 表中。

限制

- 表中的最大列数取决于列的宽度和服务器的逻辑页大小：
  - 列大小的总和不能超过服务器的逻辑页大小。
  - 每个表的最大列数不能超过 1024。
  - 所有页锁定表的最大可变长度列数是 254。

    例如，如果服务器使用 2K 逻辑页大小，并且包括一个整数列组成的表，则表中的最大列数将远远小于 1024。（1024 * 4 字节超过了 2K 逻辑页大小。）

    只要最大列数不超过 1024，就可以在单个表中混合使用可变长度和固定长度的列。例如，如果服务器使用 8K 的逻辑页大小，则为 APL 配置的表可以具有 254 个可为空值的整数列（这些是可变长度列）和 770 个不可为空值的整数列，共计 1024 列。
- 每个数据库最多可以有 2,000,000,000 个表，且每个表最多可以有 1024 个用户定义的列。每个表的行数仅受到可用存储的限制。
- 虽然 Adaptive Server 并不在下面的环境中创建表，当您执行 DML 操作时，将会收到关于大小限制的错误：
  - 如果具有可变长度列的行的总行宽超出最大列大小
  - 如果单个可变长度列的长度超出最大列大小
  - 对于 DOL 锁定表，如果除初始列之外的任何可变长度列的偏移量超出了 8191 字节的限制
- 如果所有固定长度列的总大小加上行开销大于表的锁定方案和页大小允许的值，则 Adaptive Server 会报告错误。这些限制在表 1-10 中进行了说明。

表 1-10: 行和列的最大长度 – APL 和 DOL

锁定方案	页大小	最大行长度	最大列长度
APL 表	2K （2048 字节）	1962 字节	1960 字节
	4K （4096 字节）	4010 字节	4008 字节
	8K （8192 字节）	8106 字节	8104 字节
	16K （16384 字节）	16298 字节	16296 字节
DOL 表	2K （2048 字节）	1964 字节	1958 字节
	4K （4096 字节）	4012 字节	4006 字节
	8K （8192 字节）	8108 字节	8102 字节
	16K （16384 字节）	16300 字节	16294 字节 如果表不包含任何可变长度的列
	16K （16384 字节）	16300 （取决于 varlen 的最大起始偏移 = 8191）	8191-6-2 = 8183 字节 如果表至少包含一个可变长度列。 *
	*此大小包含六个字节的行开销和两个字节的行长度字段		

- 每行的可变长度数据的最大字节数根据表的锁定方案而定：

页大小	最大行长度	最大列长度
2K （2048 字节）	1962	1960
4K （4096 字节）	4010	4008
8K （8192 字节）	8096	8104
16K （16384 字节）	16298	16296

  DOL 表的最大列大小：

页大小	最大行长度	最大列长度
2K （2048 字节）	1964	1958
4K （4096 字节）	4012	4006
8K （8192 字节）	8108	8102
16K （16384 字节）	16300	16294

- 如果创建一个具有超过 8191 字节偏移量的可变长度列的 DOL 锁定表，则您不能向该列添加任何行。
- 如果创建具有 varchar、 nvarchar、 univarchar 或 varbinary 列的表，而列的总定义宽度大于所允许的最大行宽，则会出现一条警告消息，但仍会创建该表。如果试图在这样的行中插入超过最大数量的字节，或 update 某一行，使其总行宽大于最大长度，则 Adaptive Server 将产生一条错误消息，且命令失败。

- 当 create table 命令在 if...else 块或 while 循环中出现时， Adaptive Server 将在确定条件是否为真之前创建表模式。如果该表已经存在，则会导致错误。为避免这种情况，要么确保数据库中不存在具有相同名称的视图，要么按如下方式使用 execute 语句：

```
if not exists
     (select * from sysobjects where name="my
table")
begin
execute "create table mytable (x int)"
end
```

- 不能用声明缺省值或检查约束执行 create table，然后在同一批处理或过程的表中插入数据。可以将创建和插入语句分放在两个不同的批处理或过程中，或者使用 execute 分别执行操作。
- 不能在提供缺省值的 create table 语句中使用以下变量：

```
declare @p int
select @p = 2
create table t1 (c1 int default @p, c2 int)
```

  这样做将出现错误消息 154：“Variable is not allowed in default”。

- 虚拟散列表具有以下限制：

  create proxy table、create table at remote server 或 alter table 当前不支持 SQL 用户定义的函数。

  注释 注意：执行 SQL 函数需要使用语法 username.functionname()。

  - 虚拟散列表的行必须唯一。虚拟散列表不允许多个行具有相同的键列值，因为 Adaptive Server 不能将某行保留在散列区域中，而将具有相同键列值的另一行保留在溢出聚簇区域中。
  - 每个虚拟散列表都必须在排它段上创建。

创建压缩表

- 除非您另外声明，否则 Adaptive Server 会：
  - 在您创建表时将数据压缩设置为 NULL。
  - 在您修改数据时保留现有的压缩级别。
  - 将所有分区都设置为在 create table 子句中指定的压缩级别。
- 您可以创建具有表级压缩的表，但保留某些分区不被压缩，这样您便可以用活动分区的格式来维护非压缩数据，并定期根据情况压缩数据。

- Adaptive Server 支持对所有形式的分区 （循环分区除外）进行分区级压缩。
- 标记为 not compressed 的列不会被选择来进行行压缩或页压缩。不过，行内列 （包括已实现的计算列）符合压缩条件：
  - 所有短于 4 字节固定长度的数据都不符合行压缩条件。不过，Adaptive Server 可以在页索引压缩期间压缩这些数据类型。
  - 所有数据 （固定长度，或者等于或大于 4 字节的可变长度）都符合行压缩条件。
- 缺省情况下， Adaptive Server 创建不压缩的非实现计算列。
- Adaptive Server 先压缩符合行级压缩条件的列。如果压缩的行比不压缩的行长， Adaptive Server 会放弃压缩的行并将不压缩的行存储到磁盘上，以确保压缩不浪费空间。
- 数据页可以同时包含压缩的数据行和不压缩的数据行。
- 您可以压缩固定长度的列。
- 您可以使用 with exp_row_size 子句仅为固定长度的行创建压缩 DOL 表。不能对所有页锁定 (APL) 表使用 with exp_row_size 子句。
- 如果您指定预期的行宽，但不压缩的行长度小于预期行宽，Adaptive Server 将不压缩该行。
- 对某个表启用压缩后，对该表执行的所有 bcp 和 DML 操作都将压缩数据。
- 压缩可能允许在页上存储更多行，但它不会更改表的最大行宽。不过，它可以更改表的最小有效行宽。
- 对于可以进行行压缩或页索引压缩，但列的性质使得压缩不适用或无意义的列 （例如，使用 bit 数据类型、加密的列，或时间戳列），请使用 not compressed。
- 对表进行压缩不会压缩其索引。

有关压缩的限制

- 您不能压缩：
  - 系统表
  - 工作表
  - 行内 Java 列
  - 非实现计算列
  - IDENTITY 列

  - 为进行数据传输而添加的时间戳
  - 所有数据类型；有关不支持的数据类型的列表，请参见 《压缩用户指南》
  - 加密列
- 如果最小行宽超过了所配置的锁定方案和页大小组合的最大用户数据行宽，您就无法创建要进行压缩的表。例如，您无法创建一个仅数据锁定表，其页大小为 2K，且包括数据类型为 char(2007) 的列 c1，因为它超过了最大用户数据行宽。对于行压缩和页压缩，Adaptive Server 会像对待新表一样执行行宽检查。
- 您不能创建一个具有短的、小于 4 字节的固定长度列的表来进行 row 或 page 压缩。

对使用大对象 (LOB) 数据的表进行压缩

压缩 LOB 表中的数据包括以下限制。您不能：

- 压缩计算文本列
- 对常规列的 XML 数据发出 LOB 压缩子句 （例如， lob_compression =）
- 对系统表和工作表使用 LOB 压缩

列定义

- 基于用户定义的数据类型创建列时：
  - 不能更改长度、精度或标度。
  - 可以用 NULL 类型创建 NOT NULL 列，但不能创建 IDENTITY 列。
  - 可以用 NOT NULL 类型创建 NULL 列或 IDENTITY 列。
  - 可以用 IDENTITY 类型创建 NOT NULL 列，但该列会继承 IDENTITY 属性。不能用 IDENTITY 类型创建 NULL 列。
- 只有具有可变长度数据类型的列才能存储 NULL 值。当创建具有固定长度数据类型的 NULL 列时， Adaptive Server 会自动将其转换为相应的可变长度数据类型。 Adaptive Server 不会向用户告知类型的更改。

  表 1-11 列出了固定长度数据类型及其可转换为的可变长度数据类型。某些可变长度数据类型 （如 moneyn）是保留的数据类型，不能用来创建列、变量或参数：

表 1-11: 用于存储 NULL 值的可变长度数据类型

原始的固定长度的数据类型	转换为
char	varchar
nchar	nvarchar
binary	varbinary
datetime	datetimn
float	floatn
bigint、 int、 smallint、 tinyint	intn
unsigned bigint、 unsigned int、 unsigned smallint	uintn
decimal	decimaln
numeric	numericn
money 和 smallmoney	moneyn

- 您可以通过两种方式创建列缺省值：通过在 create table 或 alter table 语句中将缺省值声明为列约束，或通过使用 create default 语句创建缺省值并使用 sp_bindefault 将其绑定到列。
- 有关表及其列的报告，请执行系统过程 sp_help。

临时表

- 临时表存储在临时数据库 tempdb 中。
- 临时表名的前 13 个字符必须是每个会话唯一的。只有当前 Adaptive Server 会话才能访问这样的表。它们按名称后加上一个由系统提供的数字后缀的方式存储在 tempdb..objects 中，并在当前会话结束时或被显式删除时消失。
- 用 “tempdb..” 前缀创建的临时表在 Adaptive Server 用户会话之间是可共享的。仅当要在用户和会话间共享表时，才应使用 “tempdb..” 前缀从存储过程内创建临时表。若要避免无意中共享临时表，请在存储过程中创建和删除临时表时使用 “#” 前缀。
- 在一个 Adaptive Server session 会话期间，临时表可由多个用户使用。但是，通常不能确定特定的用户会话，因为临时表是用 “guest” 用户 ID 2 创建的。如果多个用户运行创建临时表的过程，每个用户都是一个 “guest” 用户，这样 uid 值是一样的。因此，无法得知临时表中的哪个用户会话是针对特定用户的。系统管理员可以用 create login 向临时表中添加用户，在这种情况下，临时表中的用户的会话可以使用各个 uid。
- 可将规则、缺省值和索引与临时表建立关联，但不能在临时表上创建视图，或将触发器与之建立关联。

- 创建临时表时，仅当 tempdb..systypes 中存在要使用的用户定义数据类型时才能使用该类型。要仅针对当前会话向 tempdb 添加用户定义的数据类型，请在使用 tempdb 时执行 sp_addtype。若要永久性地添加数据类型，应在使用 model 时执行 sp_addtype，然后重新启动 Adaptive Server，以将 model 复制到 tempdb 中。

使用索引

- 表 “跟随”其聚簇索引。如果在一个段上创建表，然后在另一个段上创建其聚簇索引，表将迁移到创建索引的段上。
- 如果段在单独的物理设备上，通过在一个段上创建表，并在另一个段上创建其非聚簇索引可以加快插入、更新和选择的速度。请参见《Transact-SQL 用户指南》中的 “使用聚簇或非聚簇索引”。

重命名表或表的列

- 使用 sp_rename 重命名表或列。
- 重命名表或其任一列后，使用 sp_depends 确定哪些过程、触发器和视图依赖于该表，并重新定义这些对象。

警告！ 如果不重新定义这些依赖的对象， Adaptive Server 重新编译后它们将无法使用。

定义完整性约束

- create table 语句通过一系列由 SQL 标准定义的完整性约束帮助控制数据库的完整性。这些完整性约束子句限制用户可插入表中的数据。您也可以用缺省值、规则、索引和触发器来强制实现数据库完整性。

  完整性约束的优点表现在：在创建表的进程中一步完成完整性控件的定义，并可简化创建这些完整性控件的过程。但是，与缺省值、规则、索引和触发器相比，完整性约束的范围较为有限，并且综合性也较差。

- 必须将对多列进行操作的约束声明为表级约束；将仅对一列进行操作的约束声明为列级约束。列级约束和表级约束之间存在差异，但用户很少会注意到这一差异。即，仅当修改列中的值时才会检查列级约束；而如果对行进行任何修改，都会检查表级约束，而不管是否更改了有关的列。

  列级约束应放在列名和数据类型之后，分隔逗号之前（请参见示例 5）。输入表级约束作为单独的逗号分隔子句（请参见示例 4）。Adaptive Server 对表级约束和列级约束的处理方式相同；二者的有效程度相同。

- 可在表级或列级创建以下类型的约束：

- unique 约束不允许一个表中有两行在指定的列有相同的值。此外， primary key 约束不允许列中有空值。
- **参照完整性** (references) 约束要求特定列中插入或更新的数据与指定表和列中的数据相匹配。
- check 约束限制插入到列中的数据值。

也可通过以下方法来强制实现数据完整性：即限制在列中使用空值（使用 null 或 not null 关键字）；为列提供缺省值 （使用 default 子句）。

- 可以使用 sp_primarykey、 sp_foreignkey 和 sp_commonkey 在系统表中保存信息，这样有助于阐明数据库中表之间的关系。这些系统过程并不强制实现键关系或复制 create table 语句中的 primary key 和 foreign key 关键字的功能。若要获取已经定义的键的报告，请使用 sp_helpkey。若要获取常用连接的报告，请执行 sp_helpjoins。
- Transact-SQL 为强制实现完整性提供了几种机制。除了可在 create table 过程中声明的约束，您还可以创建规则、缺省值、索引和触发器。表 1-12 总结了完整性约束并介绍了强制实现完整性的其它方法：

表 1-12: 强制实现完整性的方法

在 create table 中	其它方法
unique 约束	create unique index （在允许空值的列上）
primary key 约束	create unique index （在不允许空值的列上）
references 约束	create trigger
check 约束 （表级）	create trigger
check 约束 （列级）	create trigger 或 create rule 和 sp_bindrule
default 子句	create default 和 sp_bindefault

根据您的要求选择方法。例如，触发器对参照完整性 （如引用其它列或对象）的处理方式比在 create table 中声明的方法更为复杂。此外，由 create table 语句定义的约束是特定于该表的；与规则和缺省值不同，不能将它们绑定到其它表，并且只能用 alter table 来对它们进行删除或更改。即使在同一张表上，约束也不能包含子查询或集合函数。

- create table 可包含很多约束，具有以下限制：
  - unique 约束的数量受表可拥有的索引的数量的限制。
  - 一个表只能有一个 primary key 约束。
  - 表中每列只能包括一个 default 子句，但可以在同一列上定义不同的约束。

例如：

```
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)
```

新表 discount_titles 的列 title_id 是用各种完整性约束定义的。

- 可以创建错误消息并将它们绑定到参照完整性约束和 check 约束。用 sp_addmessage 创建消息并用 sp_bindmsg 将它们绑定到约束。请参见 sp_addmessage 和 sp_bindmsg。
- Adaptive Server 在强制实现参照约束前评估检查约束，并在强制实现所有完整性约束后评估触发器。如果有任何约束失败， Adaptive Server 将取消数据修改语句，不会执行任何相关的触发器。但是，违反约束*并不*回退当前事务。
- 在被引用表中，不能更新与引用表中的值相匹配的列值或删除与引用表中的值相匹配的行。首先在引用表中进行更新或删除，然后尝试在被引用表中进行更新或删除。
- 在删除被参照表之前，必须先删除参照表；否则将违反约束。
- 有关为表定义的约束的信息，请使用 sp_helpconstraint。

唯一约束和主键约束

- 可在列级或表级声明 unique 约束。 unique 约束要求指定列中所有的值都是唯一的。表中任意两行的指定列中不能有相同的值。
- primary key 约束是 unique 约束的限制性更强的形式。带有 primary key 约束的列不能包含空值。

注释 create table 语句的 unique 和 primary key 约束创建定义列的唯一或主键属性的索引。sp_primarykey、sp_foreignkey 和 sp_commonkey 定义列间的逻辑关系。必须使用索引和触发器强制实现这些关系。

- 表级 unique 或 primary key 约束在 create table 语句中作为单独的项出现，且必须包括所创建的表的一列或多列的名称。
- unique 或 primary key 约束在指定列上创建唯一索引。示例 3 中的 unique 约束创建了一个唯一的聚簇索引，与以下语句效果相同：

```
create unique clustered index salesind
    on sales (stor_id, ord_num)
```

只有索引名不同，通过命名约束可将其设置为 salesind。

- SQL 标准中的 unique 约束的定义规定，列定义不允许使用空值。缺省情况下，如果在列定义中省略 null 或 not null，Adaptive Server 会将列定义为不允许使用空值 （如尚未用 sp_dboption 进行更改）。在 Transact-SQL 中，可以将列定义为在 unique 约束下允许空值，因为用于强制约束的唯一索引允许插入空值。
- 缺省情况下， unique 约束创建唯一的非聚簇索引；primary key 约束创建唯一的聚簇索引。一个表只能有一个聚簇索引，所以只能指定一个 unique clustered 或 primary key clustered 约束。
- create table 的 unique 和 primary key 约束提供了相对于 create index 语句更简单的替代方法。然而：
  - 不能创建非唯一索引。
  - 不能使用所有由 create index 提供的选项。
  - 必须使用alter table drop constraint 删除这些索引。

参照完整性约束

- 参照完整性约束要求插入到定义约束的*引用*表中的数据必须在*被引用*表中有匹配的值。参照完整性约束满足以下条件之一：
  - 引用表的约束列中的数据包含空值。
  - 引用表的约束列中的数据与被引用表中相应列的数据值匹配。

  以 pubs2 数据库为例，插入到 salesdetail 表 （记录书的销售情况的表）中的行必须在 titles 表中有一个有效的 title_id。salesdetail 是引用表，而 titles 表是被引用表。现在， pubs2 使用触发器强制实现此参照完整性。但是， salesdetail 表能够包括此列定义和参照完整性约束来完成相同的任务：

  ```
  title_id tid
      references titles (title_id)
  ```

- 查询所允许的最大表引用数为 192。使用 sp_helpconstraint 可检查表的参照约束。
- 表可以包括对其自身的参照完整性约束。例如， pubs3 中的 store_employees 表（列出了雇员及其管理者）在 emp_id 和 mgr_id 列之间有以下自身引用：

  ```
  emp_id id primary key,
  mgr_id id null
          references store_employees (emp_id),
  ```

  此约束确保所有的管理者也是雇员，且为所有的雇员都指派了有效的管理者。

- 在删除引用表或删除参照完整性约束之前不能删除被引用表 （除非该表仅包括一个对其自身的参照完整性约束）。
- Adaptive Server 并不为临时表强制实现参照完整性约束。
- 若要创建一个表，使其引用另一用户的表，必须对被引用表拥有 references 权限。有关指派 references 权限的信息，请参见 grant 命令。
- 表级参照完整性约束在 create table 语句中作为单独的项出现。它们必须包括 foreign key 子句和一个或多个列名的列表。

  只有在通过 primary key 约束将被引用表中的列指定为主键时，references 子句中的列名才是可选的。

  被引用列必须由该被引用表中的唯一索引约束。可以用 unique 约束或 create index 语句创建该唯一索引。
- 引用表列的数据类型必须与被引用表列的数据类型匹配。例如，引用表 (test_type) 中 col1 的数据类型与被引用表 (publishers) 中 pub_id 的数据类型相匹配：

```
create table test_type
 (col1 char (4) not null
    references publishers (pub_id),
col2 varchar (20) not null)
```

- 定义参照完整性约束时，被引用表必须存在。对于互相交叉引用的表，使用 create schema 语句来同时定义两个表。作为替代方法，可以创建一个不带约束的表，然后再使用 alter table 添加。有关详细信息，请参见 create schema 或 alter table。
- create table 参照完整性约束提供了强制实现数据完整性的简单方法。与触发器不同，约束不能：
  - 对数据库中的相关表进行级联更改
  - 通过引用其它列或数据库对象强制实施复杂限制
  - 执行 “what-if” 分析

  当数据修改违反约束时，参照完整性约束并不回退事务。触发器允许依据处理参照完整性的方式选择回退或继续事务。

注释 Adaptive Server 在检查触发器之前将检查参照完整性约束，以使违反约束的数据修改语句也不会引发触发器。

使用跨数据库参照完整性约束

- 创建跨数据库约束时， Adaptive Server 会将以下信息存储在每个数据库的 sysreferences 系统表中：

表 1-13: 存储的有关参照完整性约束的信息

存储在 sysreferences 中的信息	包含有关被引用表的信息的列	包含有关引用表的信息的列
键列 ID	refkey1 到 refkey16	fokey1 到 fokey16
表 ID	reftabid	tableid
数据库 ID	pmrydbid	frgndbid
数据库名称	pmrydbname	frgndbname

- 可以删除引用表或其数据库。 Adaptive Server 自动从被引用数据库中删除外键信息。
- 由于引用表依赖于来自被引用表的信息， Adaptive Server 不允许您：
  - 删除被引用表，
  - 删除包含被引用表的外部数据库，或者
  - 使用 sp_renamedb 对其中任何一个数据库进行重命名。

  必须使用 alter table 删除跨数据库约束后才能执行其中任一操作。
- 每次添加或删除跨数据库约束或者删除包含跨数据库约束的表时，都请转储*上述两个*受影响的数据库。

  警告！ 装载包含跨数据库约束的数据库的早期转储可能会导致数据库损坏。

- sysreferences 系统表存储外部数据库的名称和 ID 号。如果您用 load database 更改数据库名或将其装载到其它服务器上，则 Adaptive Server 无法保证参照完整性。

  警告！ 在转储数据库以便用不同数据库名装载或转移到另一个 Adaptive Server 之前，请使用 alter table 删除所有的外部参照完整性约束。

check 约束

- check 约束可限制用户在表列中插入的值。 check 约束指定任何非空值在插入到表之前必须通过的 *search_condition*。 *search_condition* 可包括：
  - 用 in 引入的一系列常量表达式
  - 用 between 引入的一系列常量表达式
  - 由 like 引入的一组条件，可以包含通配符

表达式可包括算术运算符和 Transact-SQL 内置函数。*search_condition* 不能包含子查询、集合函数、主变量或参数。Adaptive Server 并不为临时表强制实现 check 约束。

- 列级 check 约束只能引用定义它时所在的列，而不能引用表中的其它列。表级 check 约束可引用表中的任何列。
- create table 允许在一个列定义中有多个 check 约束。
- check 完整性约束提供了使用规则和触发器的替代方法。它们特定于被创建时所在的表，且不能绑定到其它表中的列或用户定义数据类型。
- check 约束并不替换列定义。如果对允许空值的列声明 check 约束，即使 *search_condition* 中不包含 NULL 值，也可以在列中显式或隐式地插入 NULL 值。例如，如果创建了一个 check 约束在允许 NULL 值的表列上指定 “pub_id in (“1389”、“0736”、“0877”、“1622”、“1756”）” 或 “@amount > 10000”，您仍可将 NULL 值插入该列。列定义会替换 check 约束。

IDENTITY 列

- 首次向表中插入行时， Adaptive Server 为 IDENTITY 列指派值 1。每个新行的列值都比上一个值增加 1。此值优先于在 create table 语句中声明的或用 sp_bindefault 绑定到列的任何缺省值。

  可插入 IDENTITY 列的最大值为数值 $10^{precision} - 1$。对于整数标识，最大值为其类型的最大允许值 （例如对于 tinyint 为 255，对于 smallint 为 32767）。

  请参见 《参考手册：过程》中的 第 1 章 “系统数据类型和用户定义的数据类型”。

- 通过在 IDENTITY 列中插入值，可以指定该列的源值或恢复误删除的行。对基表使用 set identity_insert *table_name* on 后，表的所有者、数据库所有者或系统管理员可以显式地在 IDENTITY 列中插入值。除非创建了 IDENTITY 列的唯一索引，否则 Adaptive Server 不会检验值的唯一性。可以插入任何正整数。
- 如有必要，可用由表名限定的 syb_identity 关键字代替实际列名来引用 IDENTITY 列。
- 系统管理员可使用 auto identity 数据库选项自动将 10 位的 IDENTITY 列包含在新表中。若要在数据库中打开此功能，请使用：

  ```
  sp_dboption database_name, "auto identity", "true"
  ```

用户每次在数据库中创建表而不指定 primary 键、 unique 约束或 IDENTITY 列时，Adaptive Server 都会自动定义一个 IDENTITY 列。用 select * 语句检索列时，此 SYB_IDENTITY_COL 列是不可见的。必须在选择列表中显式地包含此列名。

- 服务器故障会使 IDENTITY 列值产生间隔。事务回退、删除行或手动在 IDENTITY 列中插入数据也都会形成间隔。间隔的最大大小取决于 identity burning set factor 和 identity grab size 配置参数的设置，或者在 create table 或 select into 语句中给定的 identity_gap 值。请参见《Transact-SQL 用户指南》的 “创建数据库和表”中的 “管理表中的标识间隔”。

指定锁定方案

若要指定表的锁定方案，请使用关键字 lock 和以下锁定方案之一：

- 所有页锁定，锁定数据页和受查询影响的索引
- 数据页锁定，仅锁定数据页
- 数据行锁定，仅锁定数据行

如果没有指定锁定方案，将使用服务器的缺省锁定方案。使用配置参数 lock scheme 可设置服务器范围的缺省值。

可以使用 alter table 命令更改表的锁定方案。

空间管理属性

- 空间管理属性 fillfactor、 max_rows_per_page、 exp_row_size 和 reservepagegap 以下列方式帮助管理表的空间使用情况：
  - fillfactor 在创建索引时在页上留出额外空间，但不会不断对 fillfactor 进行维护。
  - max_rows_per_page 限制数据或索引页的行数。它的主要用途是改善所有页锁定表的并发性，因为减少行数可减少锁争用。如果指定 max_rows_per_page 值和 datapages 或 datarows 锁定，会显示一条警告消息。表创建完毕，且值已存储在 sysindexes 中，但仅当稍后将锁定方案更改为 allpages 时应用该值。

- exp_row_size 指定数据行的所需行宽。它仅适用于数据行，而不适用于索引，且仅适用于具有可变长度列的仅数据锁定表。它用于减少仅数据锁定表中的转移行的数量。主要是那些首次插入时行具有空列或短列，但随后的更新又增加了这些列的大小的表需要它。exp_row_size 在数据页上保留空间以供要增长到指定大小的行使用。如果在创建所有页锁定表时指定了 exp_row_size，则会显示一条警告消息。表创建完毕，且值已存储在 sysindexes 中，但仅当稍后将锁定方案更改为 datapages 或 datarows 时才应用该属性。
- reservepagegap 指定空白页与整页的比率以供执行扩展分配的命令使用。它适用于所有锁定方案的数据和索引页。

- 表 1-14 显示了空间管理属性和锁定方案的有效组合。如果 create table 命令包括不兼容的组合，则会显示一条警告消息并创建表。值存储在系统表中但不会应用。如果表的锁定方案的更改使这些属性成为有效的，随后将使用它们。

表 1-14: 空间管理属性和锁定方案

属性	allpages	datapages	datarows
max_rows_per_page	可用	不可用	不可用
exp_row_size	不可用	可用	可用
reservepagegap	可用	可用	可用
fillfactor	可用	可用	可用

- 表 1-15 显示了空间管理属性的缺省值和使用缺省值的效果。

表 1-15: 空间管理属性的缺省值和效果

属性	缺省值	使用缺省值的效果
max_rows_per_page	0	使每页容纳尽可能多的行，最多可容纳 255 行
exp_row_size	0	使用服务器范围的缺省值，该值使用配置参数 default exp_row_size percent 设置
reservepagegap	0	在扩充分配期间不留下任何空白页
fillfactor	0	完全填满叶页，在索引页上留有空间

使用 *reservepagegap*

- 使用大量空间的命令通过分配扩展而不是分配单页来分配新空间。reservepagegap 关键字使这些命令留出空白页以使将来的页分配紧邻被拆分的页或移走行的页发生。表 1-16 显示了应用 reservepagegap 的时间。

表 1-16: 应用 reservepagegap 的情形

命令	应用于数据页	应用于索引页
快速 bcp	是	如果存在索引，则不使用快速 bcp
慢速 bcp	仅用于堆表，不用于具有聚簇索引的表	不执行扩充分配
select into	是	目标表上不存在索引
create index 或 alter table...constraint	是，对于聚簇索引	是
reorg rebuild	是	是
alter table...lock （对于从所有页锁定到 DOL 锁定的转换，或反向转换）	是	是

- 表的 reservepagegap 值存储在 sysindexes 中并在对表执行以上任一操作时使用。使用 sp_chgattribute 可更改存储的值。
- reservepagegap 不适用于工作表或对工作表的排序。

获取有关表的信息

- sp_help 显示关于表的信息，列出指派给指定表及其索引的任何属性（如高速缓存绑定），给出属性的类、名称、整数值、字符值和注释。
- sp_depends 显示关于数据库中依赖于表的视图、触发器和过程的信息。
- sp_helpindex 报告有关为表创建的索引的信息。
- sp_helpartition 报告有关表的分区属性的信息。

创建带分区的表

- 创建带分区的表之前，必须准备好将用于分区的磁盘设备和段。
- 域分区取决于排序顺序。如果更改了排序顺序，则必须按照新排序顺序对表重新分区。
- 域分区边界必须根据分区创建顺序升序排列。
- text、unitext、image 或 bit 类型、Java 数据类型的列或计算列不能是分区键的一部分，但分区表可以包含这些数据类型的列。组合分区键最多可以包含 31 列。
- 对于域分区和散列分区，分区键可以是最多 31 个列的组合键。但一般情况下，具有四个以上分区列的表会很难管理并且不很有用。
- 域分区和列表分区的边界值必须与对应分区键的数据类型兼容。如果指定了兼容但属于不同数据类型的边界值， Adaptive Server 会将该边界值转换为分区键的数据类型。 Adaptive Server 不支持：
  - 显式转换。

  - 导致数据丢失的隐式转换。
  - 将 NULL 作为域分区表的边界。
  - 从非二进制数据类型到 binary 或 varbinary 数据类型的转换。
- 可以在列表分区表的值列表中使用 NULL。
- 可以对包含 text 和 image 列的表分区，但分区不会影响 Adaptive Server 存储 text 和 image 列的方法，因为它们驻留在自己的分区上。
- 不能对远程表进行分区。
- Adaptive Server 将 NULL 视为比给定分区键列的任何其它分区键值小。

创建带计算列的表

- *computed_column_expression* 只能引用同一表中的列。
- *computed_column_expression* 的确定性属性对数据操作有显著影响。请参见 《Transact-SQL 用户指南》中的 “确定性属性”。
- 计算列不能有缺省值，并且不能是 identity 或 timestamp 列。
- 只能为实现计算列指定可为空性。如果不指定可为空性，则所有计算列缺省情况下都是可空的。虚拟计算列始终是可空的。
- 触发器和约束 （例如 check、 rule、 unique、 primary key 或 foreign key）仅支持实现计算列。不能将它们用于虚拟计算列。
- 如果计算列定义中用户定义的函数被删除或变为无效，则调用该函数的任何计算列操作都会失败。

创建带加密列的表

- 可以对以下数据类型加密：
  - 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
- 磁盘上的加密数据的基本数据类型为 varbinary。不会对空值加密。
- 如果您执行以下操作， create table 会显示错误：
  - 基于引用一个或多个加密列的表达式来指定计算列。
  - 将 encrypt 和 compute 参数用于相同的列。
  - 在 partition 子句中列出加密列
- 在执行 create table、alter table 和 select into 操作的过程中， Adaptive Server 计算加密列的最大内部长度。在做出有关模式安排和页大小的决策之前，数据库所有者必须知道加密列的最大长度。
- 如果指定加密密钥而未使用任何初始化矢量或随机填充，则可以创建加密列的索引。如果加密列使用了初始化矢量或随机填充，则对其执行 create index 命令时， Adpative Server 将显示错误。
- 在以下情况下，可以定义加密列的参照完整性约束：
  - 加密引用的列和被引用的列。
  - 用于对列执行加密的密钥指定 init_vector null，但您尚未指定 pad random。
- 不能加密计算列，并且加密列不能出现在定义计算列的表达式中。您不能在 create table 的 *partition_clause* 中指定加密列。

  请参见 《加密列用户指南》中的 “加密数据”。

创建虚拟散列表时的限制

- 您不能在包括虚拟散列表的段上使用 create table，因为虚拟散列表必须仅采用一个排它段，而排它段是不能被其它表或数据库共享的。
- 虚拟散列表的行必须唯一。虚拟散列表不允许多个行具有相同的键列值，因为 Adaptive Server 不能将某行保留在散列区域中，而将具有相同键列的另一行保留在溢出聚簇区域中。
- 不支持 truncate table。应改用 delete from *table_name*。
- SQL92 不允许一个关系中的两个唯一约束具有相同的键列。但是，虚拟散列表的主键子句不是标准的唯一约束，因此可以将具有相同键列的单独唯一约束声明为虚拟散列键。
- 由于在创建表之后无法创建虚拟散列聚簇索引，因此也无法删除虚拟散列聚簇索引。

- 必须在排它段上创建虚拟散列表。不能将指派给段用于创建虚拟散列表的磁盘设备与其它段共享。
- 不能在同一排它段上创建两个虚拟散列表。 Adaptive Server 支持每个数据库具有 32 个不同的段。缺省段、系统段和日志段这三个段为保留段，所以每个数据库的最大虚拟散列表数为 29 个。
- 不能将 alter table 或 drop clustered index 命令用于虚拟散列表。
- 虚拟散列表必须使用所有页锁定。
- 虚拟散列表的键列和散列因子必须使用 int 数据类型。
- 虚拟散列表中不能包括 text 或 image 列，也不能包括数据类型基于 text 或 image 数据类型的列。
- 不能创建分区的虚拟散列表。

为内存数据库和宽松持久性数据库创建表

- 由 create table 定义的表级日志记录设置也适用于通过 select into 创建的表。
- 虽然可在使用 full 持久性的数据库中创建具有最少日志记录的表，但数据库不会对这些表使用最少记录。 Adaptive Server 允许您将这些表设置为最少日志记录，以便于将这些数据库用作持久性设置为 no_recovery 的其它数据库的模板，其中最少记录在相关数据库中生效。

确定 hash_factor 的值

可将第一个键的散列因子保持为 1。其余所有键列的散列因子大于散列区域中允许的前一个键与其散列因子之乘积的最大值。

对于第一个键列的散列因子大于 1 的表， Adaptive Server 允许其页上具有较少的行。例如，如果表的第一个键列的散列因子为 5，则在页上的每一行之后，留给后四行的空间将保持为空。为了支持此功能，Adaptive Server 所需空间量为表空间的五倍。

如果键列的值大于或等于下一个键列的散列因子，会将当前行插入溢出聚簇区域，以避免散列区域中出现冲突。

例如， t 是具有键列 id 和 age 并且对应的散列因子为 (10,1) 的虚拟散列表。由于行 (5, 5) 和 (2, 35) 的散列值为 55，因此可能导致散列冲突。

但是，由于值 35 大于等于 10 （下一个键列 id 的散列因子），因此 Adaptive Server 会将第二行存储在溢出聚簇区域中，以避免散列区域中出现冲突。

在另一个示例中，如果 u 是主索引和散列因子为 (id1, id2, id3) = (125, 25, 5) 并且 *max hash_value* 为 200 的虚拟散列表：

- 行 (1,1,1) 的散列值为 155，因此会将该行存储在散列区域中。
- 行 (2,0,0) 的散列值为 250，因此会将该行存储在溢出聚簇区域中。
- 行 (0,0,6) 的散列因子为 6 x 5，该值大于等于 25，因此会将该行存储在溢出聚簇区域中。
- 行 (0,7,0) 的散列因子为 7 x 25，该值大于等于 125，因此会将该行存储在溢出聚簇区域中。

对共享磁盘集群的限制

- 除了从同一本地临时数据库中的表之外，不能包括引用本地临时数据库中的列的参照完整性约束。当 create table 尝试从其它数据库的表中创建对本地临时数据库中列的引用时会失败。
- 除非包含列的表驻留在同一本地临时数据库中，否则您不能用本地临时数据库中存储的加密密钥对该列加密。如果 alter table 尝试用本地临时数据库中的加密密钥对列加密，而表位于其它数据库中，该命令将失败。

Java-SQL 列

- 如果数据库中启用了 Java，则可创建具有 Java-SQL 列的表。有关详细信息，请参见 *Adaptive Server Enterprise 中的 Java*。
- Java-SQL 列的声明类 (*datatype*) 必须使用 Serializable 或 Externalizable 接口。
- 创建表时，不能将 Java-SQL 列：
  - 指定为外键
  - 在 references 子句中指定
  - 指定为具有 UNIQUE 属性
  - 指定为主键
- 如果指定了 in row，那么根据数据库服务器的页大小和其它变量，存储的值不能超过 16K 字节。
- 如果指定了 off row：
  - 不能在检查约束中引用列。
  - 不能在指定 distinct 的 select 中引用列。
  - 不能在比较运算符、判定或 group by 子句中指定列。

标准

符合 ANSI SQL 的级别符合初级标准。

Transact-SQL 扩展包括：

- 使用数据库名限定表或列名
- IDENTITY 列
- not null 缺省值
- asc 和 desc 选项
- The reservepagegap 选项
- lock 子句
- on *segment_name* 子句

有关数据类型兼容的信息，请参见 《参考手册：构件块》中的第 1 章 “系统数据类型和用户定义的数据类型”。

权限

任何用户均可创建禁用记录的临时表和新表。

下文说明了基于您的细化权限设置的 create table 的权限检查。

细化权限已启用	在启用细化权限的情况下，您必须具有 create table 特权。您必须具有 create any table 特权才能为其他用户运行 create table。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者、具有 sa_role 的用户或是具有 create table 特权的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息

10	create	create table	• *角色* - 当前活动角色
			• *关键字或选项* - NULL
			• *先前值* - NULL
			• *当前值* - NULL
			• *其它信息* - NULL
			• *代理信息* - set proxy 有效时的初始登录名
			• 如果 with transfer table [on | off] 的 with 选项为：
			  • on - Adaptive Server 会在审计记录的额外信息中输出 `WITH TRANSFER TABLE ON`。
			  • off - Adaptive Server 会输出 `WITH TRANSFER TABLE OFF`。

另请参见

命令 alter table, create existing table, create index, create rule, create schema, create view, drop index, drop rule, drop table.

系统过程 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

说明

创建用户定义的线程池。

有关进程模式的考虑事项

create thread pool 在进程模式下不受支持。

语法

create thread pool *pool_name* with thread count = *count*
[, pool description = *description*]
[idle timeout = *time_period*]

参数

pool_name
要创建的池的名称。

thread count = *count*
池中的线程数。必须大于或等于 1。

pool description = *description*
（可选）说明池的用途。必须少于 256 个字符。

idle timeout = *time_period*
线程在进入休眠状态之前查找工作的时间 （以毫秒为单位）。缺省值为 100 毫秒。值为 -1 表示线程永不进入休眠状态，即使没有工作可做，也继续耗用 CPU。值为 0 表示，如果找不到工作，线程会立即进入休眠状态。

示例

示例 1 创建一个名为 sales_pool、具有 10 个线程的线程池：

```
create thread pool sales_pool with thread count = 10
```

示例 2 创建一个名为 order_pool、包括说明的线程池：

```
create thread pool order_pool with thread count = 10,
pool description = 'used for handling order entry users'
```

示例 3 创建一个名为 order_pool、具有 2 个线程且 idle timeout 为 500 毫秒的线程池：

```
create thread pool order_pool with thread count = 2,
idle timeout = 500
```

用法

- 使用 sp_addexeclass 可将负载和用户创建的线程池关联。
- Adaptive Server 必须具有足够数量的可用引擎，才能使引擎池中的所有线程 （由 *count* 指定）联机。 max online engines 的值决定引擎总数。所有引擎池中的活动线程总数不能超过 max online engines 的值。
- *pool_name* 不能以 syb_ 开头，因为 syb_ 是为与 Sybase 有关的线程池保留的。
- 不能将 Transact-SQL 变量用作 create thread pool 的参数。

- idle timeout 值为 0 表示，如果找不到工作，线程会立即进入休眠状态。
- idle timeout 值为 -1 表示，线程永不进入休眠状态，即使没有工作可做，也继续耗用 CPU。
- 可以连同 create thread pool 一起发出 execute immediate。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展

权限

对 create thread pool 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage any thread pool 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。

审计

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
42			池名称和线程计数

create trigger

说明

创建触发器，它是一种经常用来强制实现完整性约束的存储过程。当用户试图对指定的表使用指定的数据修改语句时，就会自动执行触发器。

语法

```
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]...]
```

参数

trigger_name

是触发器的名称，该名称必须符合标识符规则，并且在数据库中是唯一的。指定所有者的名称，以创建由当前数据库中的其他用户拥有的另一个同名触发器。 *owner* 的缺省值是当前用户。如果使用所有者名限定触发器，则必须以相同的方式显式限定表名。

不能对触发器名使用变量。

table_name

是要在其上创建触发器的表的名称。如果数据库中存在多个同名表，请指定所有者名称。 *owner* 的缺省值是当前用户。

for | instead of

for – 用在 insert、 delete 或 update 之前，以指示创建触发器的用途。

instead of – 创建并填充触发器中使用的已插入和已删除**伪表**，以检查本应由原来的 insert、 delete 或 update 查询修改的行。

insert, update, delete

可采用任意组合形式。 delete 不能与 if update 子句一起使用。

SQL_statements

指定触发器状态和触发器动作。触发器状态决定尝试的 insert、update 或 delete 是否会导致执行触发器动作。SQL 语句通常包括前面带关键字 if 的子查询。在下面的示例 2 中，跟随关键字 if 的子查询即为触发器状态。

触发器行为在尝试用户动作 （insert、 update 或 delete）时生效。如果指定了多个触发器动作，用 begin 和 end 将它们组合在一起。

有关触发器定义中不允许使用的语句列表，请参见第 245 页的 “触发器和事务”。有关可以包括在触发器定义中的 deleted 和 inserted 逻辑表的信息，请参见第 244 页的 “deleted 和 inserted 逻辑表”。

if update

测试指定列是否包括在 update 语句的 set 列表中或者是否受 insert 的影响。 if update 允许指定的触发器动作与指定列的更新建立关联 （请参见示例 3）。可以指定多列，且可以在一个 create trigger 语句中使用多个 if update 语句 （请参见示例 5）。

示例

示例 1 如果有人试图在 titles 表中添加或更改数据，则显示消息：

```
create trigger reminder
on titles
for insert, update as
print "Don't forget to print a report for accounting."
```

示例 2 如果 titles 表中没有相应的 title_id，则禁止向 titleauthor 插入新行：

```
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
```

示例 3 如果更改了 publishers 表的 pub_id 列，则在 titles 表中进行相应的更改：

```
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
```

示例 4 如果从 titleauthor 中删除了任何行，则从 titles 表中删除相应的标题。如果书有多个作者， titleauthor 中对它的其它引用也将删除：

```
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
```

示例 5 禁止在周末对主键进行更新。禁止更新标题的价格或预付款，除非该标题的总收入超过了其预付款金额：

```
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
```

示例 6 使用 instead of 触发器更新联合视图：

```
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
```

示例 7 使用 instead of 触发器可实现加密列支持，以加密形式在数据库中存储数据，而不更改应用程序（用户定义函数 my_encrypt 和 my_decrypt 对数据执行加密和解密操作）：

```
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
```

用法

- 为了避免看到因设置更改而引起的意外结果，请先将 set rowcount 0 作为初始语句来运行，然后再执行 create trigger。 set 的范围仅限于 create trigger 命令，一旦过程退出便会重置为以前的设置。
- 每个数据修改语句只将触发器引发一次。包含 while 循环的复杂查询可能重复 update 或 insert 多次，且每次都会引发触发器。

触发器和参照完整性

- 触发器通常用于强制实施**参照完整性**（有关表或视图的主键和外键之间关系的完整性规则），以提供级联删除或级联更新（请分别参见示例 2、 3 和 4）。
- 只有在数据修改语句完成操作，并且 Adaptive Server 完成对所有数据类型、规则或完整性约束冲突的检查之后，触发器才引发。触发器和引发它的语句将被当作单个事务，可从触发器中回退。如果检测到一个严重错误，整个事务将被回退。
- 您也可以用由 create table 语句定义的约束代替 create trigger 强制实施参照完整性。有关完整性约束的信息，请参见 create table 和 alter table。

deleted 和 *inserted* 逻辑表

- deleted 和 inserted 是逻辑 （概念）表。就结构来说，它们对为之定义了触发器的表 – 也就是用户试图进行动作的表 – 以及保存可能会由用户动作更改的行的旧值或新值的表来说是相同的。

  注释 inserted 和 deleted 表在事务日志中均显示为视图，但它们在 syslogs 中为虚设表。

- deleted 和 inserted 表可由触发器进行检查，以确定是否执行触发器动作以及如何执行，但表自身不能由触发器的动作改变。
- deleted 表是和 delete 和 update 一起使用的，而 inserted 表是和 insert 和 update 一起使用的。update 是一个 delete 后跟一个 insert：它首先影响 deleted 表，然后影响 inserted 表。

触发器限制

- 只可在当前数据库中创建触发器。如果使用所有者名限定触发器，则必须以相同的方式显式限定表名。触发器可引用当前数据库之外的对象。
- 触发器不可应用到多个表。不过，可在同一 create trigger 语句中为多个用户操作 （如 insert 和 update）定义相同的触发器动作。一个表最多可以有三个触发器，分别用于 insert、 update 和 delete。
- 表或列中进行相同操作的新的触发器 （insert、 update 或 delete）会覆盖前一个触发器。在覆盖前一个触发器之前不会显示任何警告消息。
- 不能在特定于会话的临时表上创建触发器。
- 不能在视图上创建触发器。
- 不能在系统表上创建触发器。
- 不能使用从已插入或已删除表的 text、unitext 或 image 列中选择的触发器。
- Sybase 建议使用不提供向用户返回结果的 select 语句的触发器，因为允许对触发器表进行修改的特殊处理必须写入到每个用于这些返回结果的应用程序中。
- 如果触发器引用并非有效标识符的表名、列名或视图名时，必须在 create trigger 命令之前 set quoted_identifier on，并将每个这样的名称都用双引号引起来。触发器引发时并不需要打开 quoted_identifier 选项。

触发器和性能

- 就性能而言，触发器的开销通常很小。运行触发器所需的时间大多用于引用内存或数据库设备中的其它表。

- 经常被触发器引用的 deleted 和 inserted 表始终在内存中而不是在数据库设备上，因为它们是逻辑表。触发器所引用的其它表的位置将决定操作所需的时间。

在触发器内设置选项

可以在触发器内使用 set 命令。您所调用的 set 选项在触发器执行期间保持有效，然后会恢复其原来的设置。特别要指出的是， self_recursion 选项可在触发器内部使用，这样触发器本身进行的数据修改可再次引发触发器。

删除触发器

- 如果重命名触发器所引用的任一对象，必须删除并重新创建此触发器。可以使用 sp_rename 来重命名触发器。
- 删除表时，所有与它关联的触发器也都会被删除。

不会导致引发触发器的动作

- truncate table 命令不会被 delete 触发器捕获。虽然 truncate table 语句实际上与不带 where 子句的 delete 相似 （删除所有行），但它不记录对数据行的更改，因此不能引发触发器。

  因为缺省情况下，表的所有者具有 truncate table 命令的权限，并且不能移交，因此只有表所有者需要关注用 truncate table 语句无意中回避了 delete 触发器的问题。
- 无论是有记录还是无记录， writetext 命令都不会引发触发器。

触发器和事务

- 定义触发器时，在它应用的表上指定的动作以及触发器自身始终隐式地是事务的一部分。检测到错误后，触发器常用于回退整个事务，或者它们也可用于回退特定数据修改的影响：
  - 当触发器包含 rollbacktransaction 命令时，回退会中止整个批处理，且不执行批处理中的任何后续语句。
  - 当触发器包含 rollback trigger 时，回退仅影响导致触发器引发的数据修改。rollback trigger 命令可包含一条 raiserror 语句。执行批处理中的后续语句。
- 由于触发器将作为事务的一部分被执行，所以在触发器中不允许使用以下语句和系统过程：
  - 所有 create 命令，包括 create database、 create default、 create index、create procedure、create rule、create table、create trigger 和 create view
  - 所有 drop 命令
  - alter database 和alter table

- truncate table
- grant 和revoke
- update statistics
- sp_configure
- load database 和load transaction
- disk init, disk refit, disk reinit, disk remirror, disk remirror, disk unmirror
- select into

- 如果期望的结果 （例如摘要值）取决于数据修改所影响的行数，可使用 @@*rowcount* 测试多行数据修改 （基于 select 语句的 insert、delete 或 update），并采取适当措施。任何不返回行的 Transact-SQL 语句 （如 if 语句）都会将 @@*rowcount* 设置为 0，因此应在开始使用触发器时进行 @@*rowcount* 测试。

插入和更新触发器

- 执行 insert 或 update 命令时， Adaptive Server 同时向触发器表和 inserted 表插入行。 inserted 表中的行始终是触发器表中一行或多行的重复值。
- update 或 insert 触发器可以用 if update 命令来确定 update 或 insert 是否更改了特定的列。只要在选择列表或在 values 子句中为列赋值，对于 insert 语句来说，if update (*column_name*) 即为 true。显式 NULL 或缺省值为列赋值，从而激活触发器。但隐式 NULL 则不能如此。

  例如，如果创建如下的表或触发器：

```
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"
```

  将值插入到一列或两列会同时引发 aaa 列和 bbb 列的触发器：

```
insert junk (aaa, bbb)
values (1, 2)
aaa updated
bbb updated
```

  向 aaa 列插入显式 NULL 也会引发触发器：

```
insert junk
values (NULL, 2)
aaa updated
bbb updated
```

如果 aaa 列具有缺省值，也会引发触发器。

不过，如果 aaa 列没有缺省值，且没有显式地插入值， Adaptive Server 会生成隐式 NULL，且不会引发触发器：

```
insert junk (bbb)
values (2)
bbb updated
```

delete 语句的 if update 值决不会为 true。

嵌套触发器和触发器递归

- 缺省情况下 Adaptive Server 允许嵌套触发器。若要阻止触发器嵌套，请使用 sp_configure 将 allow nested triggers 选项设置为 0 （关闭）：

  ```
  sp_configure "allow nested triggers", 0
  ```

- 触发器可嵌套 16 层。如果一个触发器更改还具有另一个触发器的表，则第二个触发器也会引发，并且可以接着调用第三个触发器，依此类推。如果链中的任何触发器引发了无限循环，就会超出此嵌套级别，触发器将中止，回退包含此触发器查询的事务。

  注释　因为触发器是放置在事务中的，所以如果在一组嵌套触发器的任一层出现故障，都会取消整个事务：回退所有数据修改。为触发器提供消息和其它错误处理和调试辅助程序，来决定故障出现在哪里。

- 全局变量 @@*nestlevel* 包含当前执行的嵌套级别。每当存储过程（或触发器）调用另一个存储过程 （或触发器）时，嵌套级别就会增加 1。创建高速缓存的语句时，嵌套级别也会增加一级。如果超过了最大值 16，事务中止。

- 如果触发器调用一个所执行的操作会使触发器再次引发的存储过程，则只有在启用嵌套触发器后，该触发器才会被再次激活。除非触发器内有限制递归次数的条件，否则将导致嵌套级别溢出。

  例如，如果更新触发器调用一个执行更新操作的存储过程，则在 allow nested triggers 关闭时，只执行一次触发器和存储过程。如果 allow nested triggers 是打开的，且触发器或过程中的条件没有限制更新的数目，则过程或触发器循环会一直继续，直到超过最大嵌套值 16 级。

- 缺省情况下，不管 allow nested triggers 配置参数的设置如何，触发器都不调用其本身来响应对触发器内同一个表的第二次数据修改。set 选项 (self_recursion) 使得触发器能够因触发器内的数据修改而再次引发。例如，如果表的一个列上的更新触发器会更新另一列，则当禁用 self_recursion 时，更新触发器仅引发一次，但如果 self_recursion 打开，则更新触发器可引发多达 16 次。为了能够进行自递归，还必须启用 allow nested triggers 配置参数。

instead of 和 for 触发器

- 您可以交错嵌套 instead of 和 for 触发器。例如，对具有 instead of update 触发器的视图执行 update 语句会导致该触发器执行。如果触发器包含的 SQL 语句可更新其中已定义 for 触发器的表，则会引发该触发器。 for 触发器可能包含可更新另一个视图的 SQL 语句，该视图具有随后执行的 instead of 触发器，依此类推。
- instead of 和 for 触发器具有不同的递归行为。 for 触发器支持递归，而 instead of 触发器不支持递归。如果 instead of 触发器引用引发了该触发器的同一视图，则不会递归调用该触发器。相反，触发语句会直接应用于该视图；换句话说，就是将该语句解析为对该视图底层的基表的修改。在这种情况下，视图定义必须满足可更新视图的所有限制。如果该视图不可更新，则会产生错误。

  例如，如果将触发器定义为某个视图的 instead of update 触发器，则在 instead of 触发器中对该视图执行 update 语句不会导致该触发器再次执行。该触发器会对视图执行 update 语句，就好像视图不具有 instead of 触发器一样。必须将 update 更改的列解析为单个基表。

对 instead of 的限制：

- 如果触发器引用的表名、列名或视图名不是有效的标识符，则必须在执行 create trigger 命令之前，将 quoted_identifier 设置为 on，并用双引号引起每个这样的名称。触发器引发时， quoted_identifier 选项无需处于 on 状态，带括号的标识符同样有效。
- 结合使用 set cursor rows 命令和客户端游标 （可通过 Open Client 调用或 Embedded SQL™ 声明这些游标），可以防止 positioned delete 和 update 引发 instead of 触发器。 positioned update 语句是一个包含 where current of <*cursorname*> 子句的 SQL update 语句，该语句只更新游标 <*cursorname*> 当前所定位的行。
- searched delete 和 update 语句中不允许出现连接，否则会引发 instead of 触发器。
- 在游标上使用连接定义的 positioned delete 和 update 不会引发 instead of 触发器。

positioned delete （或 positioned update）是一个包含 where current of <*cursorname*> 子句的 SQL delete （或 update）语句，该语句仅删除（更新）游标 <*cursorname*> 当前所定位的行。

- 对于可引发 instead of 触发器的 positioned delete 和 update 语句，在声明游标时， instead of 触发器必须存在。

获取有关触发器的信息

- 触发器的执行计划存储在 sysprocedures 中。
- 每个触发器都分配有一个标识号，此标识号是作为新行与表 （在 deltrig 列它所应用的表）的对象 ID 存储在 sysobjects 中的，标识号还作为触发器所应用的表的 sysobjects 行的 deltrig 列、 instrig 列和 updtrig 列的条目存储。
- 若要显示触发器的文本 （存储在 syscomments 中），请使用 sp_helptext。

  如果系统安全员已经用 sp_configure 重置了 allow select on syscomments.text column 参数（这是在已评估的配置中运行 Adaptive Server 所必需的操作），那么只有触发器的创建者或系统管理员才能通过 sp_helptext 查看触发器的文本。
- 若要获取触发器的报告，请使用 sp_help。
- 若要获取被触发器引用的表或视图的报告，请使用 sp_depends。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

触发器创建时对象的权限 – 创建触发器时， Adaptive Server 不对对象（如触发器引用的表或视图）进行权限检查。因此，即使不能访问触发器的对象，也可以成功创建触发器。引发触发器时会检查所有权限。

触发器执行时的对象权限 – 执行触发器时，其对象的权限检查取决于触发器及其对象是否由同一用户所有。

- 如果触发器及其对象不为同一用户所拥有，必须已经授予引起触发器引发的用户直接访问对象的权限。例如，如果触发器执行对用户不能访问的表的选择，触发器的执行就会失败。另外，引起触发器引发的数据修改也会回退。
- 如果触发器及其对象为同一用户所有，则应用特殊的规则。用户自动拥有访问触发器的对象的隐式权限，即使用户不能直接访问它们。请参见 《系统管理指南》中有关细隐式权限规则的详细说明。

instead of 触发器和 for 触发器的权限 – instead of 触发器具有与 for 触发器相同的权限要求：若要创建具有 instead of 触发器的视图，必须授予用户对视图 （而非基础表）执行 insert/update/delete 的权限。

下文说明了基于您的细化权限设置的 create trigger 的权限检查。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者，并且不能撤消 create trigger 特权。您必须具有 create any trigger 特权才能对另一个用户表运行 create trigger。
细化权限已禁用	在禁用细化权限的情况下： 只有系统安全员才能授予或撤消创建触发器的权限。数据库所有者具有在任何用户表中创建触发器的隐式权限。用户只能在其拥有的表中创建触发器。 系统安全员可以撤消用户创建触发器的权限。撤消创建触发器的权限只影响系统安全员从中发出 revoke 命令的数据库。当系统安全员向被撤消权限的用户显式授予 create trigger 权限时，该用户运行 create trigger 命令的权限即得到恢复。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
12	创建	create trigger	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter table, create procedure, drop trigger, rollback trigger, set.

系统过程 sp_commonkey, sp_configure, sp_depends, sp_foreignkey, sp_help, sp_helptext, sp_primarykey, sp_rename, sp_spaceused.

create view

说明

创建一个视图，视图是查看一个或多个表中的数据的一种替代方法。

语法

```
create view [owner.]view_name
        [(column_name[, column_name]...)]
        as
        select [distinct] select_statement
        [with check option]
```

参数

view_name

是视图的名称。名称中不能包括数据库名。如果设置了 set quoted_identifier on，就可以使用分隔标识符。否则，视图名不能为变量，且必须遵循标识符的规则。指定所有者的名称，以创建由当前数据库中的其他用户拥有的另一个同名视图。*owner* 的缺省值是当前用户。

column_name

指定用于视图中列的标题的名称。如果设置了 set quoted_identifier on，就可以使用分隔标识符。否则，列名必须符合标识符的规则。

始终可以提供列名，但列名仅在下列情况下需要：

- 列是从算术表达式、函数、字符串并置或常量中派生出来的
- 两列或多列具有相同的名称 （通常是由于连接）
- 要为视图中的列指定一个与派生该列的列不同的名称 （请参见示例 3）

也可以在 select 语句中分配列名 （请参见示例 4）。如果未指定列名，视图列的名称将获得与 select 语句中的列的名称相同的名称。

select

开始定义视图的 select 语句。

distinct

指定视图不能包含重复行。

select_statement

完成定义视图的 select 语句。 select 语句可使用多个表和其它视图。

with check option

表示所有数据修改语句都通过视图选择标准进行了验证。所有通过视图插入或更新的行都必须可以通过该视图来查看。

示例

示例 1 创建一个从基表 titles 的 title、 type、 price 和 pubdate 列派生的视图：

```
create view titles_view
as select title, type, price, pubdate
from titles
```

示例 2 Creates “new view” from “old view.” Both columns are renamed in the new view.所有包含嵌入空白的视图名和列名都用双引号引起来。创建视图之前，必须使用 set quoted_identifier on。

```
create view "new view" ("column 1", "column 2")
as select col1, col2 from "old view"
```

示例 3 为价格低于 $5.00 的书创建一个包含书名、预付款及应付款的视图：

```
create view accounts (title, advance, amt_due)
as select title, advance, price * total_sales
from titles
where price > $5
```

示例 4 创建一个从 authors 和 publishers 这两个基表派生的视图。此视图包含所居住城市内有一个出版社的作者的姓名和居住城市：

```
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
```

示例 5 创建和前一示例中的定义相同的视图，但在 select 语句中包含列标题：

```
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
```

示例 6 创建视图 author_codes，此视图从列出了唯一的作者标识代码的 titleauthor 派生：

```
create view author_codes
as select distinct au_id
from titleauthor
```

示例 7 创建视图 price_list，此视图从列出了唯一的书价的 title 派生：

```
create view price_list (price)
as select distinct price
from titles
```

示例 8 创建 stores 表的视图，表中不包括位于加利福尼亚以外的店铺的信息。 with check option 子句根据视图的选择标准验证插入或更新的每一行。所有 state 值不是 “CA” 的行都会被拒绝：

```
create view stores_cal
as select * from stores
where state = "CA"
with check option
```

示例 9 创建从 stores_cal 派生的视图 stores_cal30。新视图从 stores_cal 继承 check 选项。所有通过 stores_cal30 插入或更新的行的 state 值都必须为 “CA”。因为 stores_cal30 没有 with check option 子句，所以可以为 payterms 的值不为 “Net 30”的行通过 stores_cal30 插入或更新行：

```
create view stores_cal30
as select * from stores_cal
where payterms = "Net 30"
```

示例 10 创建从 stores_cal 派生的视图 stores_cal30_check。新视图从 stores_cal 继承 check 选项。它本身还具有 with check option 子句。每个通过 stores_cal30_check 插入或更新的行都根据 stores_cal 和 stores_cal30_check 的选择标准进行验证。 state 值不为 “CA”或 payterms 值不为 “Net 30”的行将被拒绝：

```
create view stores_cal30_check
as select * from stores_cal
where payterms = "Net 30"
with check option
```

示例 11 创建视图时使用 SQL 派生表：.

```
create view psych_titles as
   select *
      from (select * from titles
               where type = "psychology") dt_psych
```

用法

- 您可以通过对视图而不是对其基础表授予权限来将视图用作安全机制。
- 可以使用 sp_rename 来重命名视图。
- 通过视图查询时， Adaptive Server 会进行检查，以确保在语句中的任何位置引用的数据库对象全都存在，它们在语句的环境中均有效，而且数据更新命令不违反数据完整性规则。如果上述检查中的任何一个失败，您都会看到错误消息。如果检查成功， create view 会将视图 “转换为”对基础表的操作。
- 有关视图的详细信息，请参见 《Transact-SQL 用户指南》。

视图的限制

- 只能在当前数据库中创建视图。
- 视图所引用的列数不能超过 1024。

- 不能在临时表上创建视图。
- 不能在视图上创建触发器或建立索引。
- 不能对视图中的 text、 unitext 或 image 列使用 readtext 或 writetext。
- 不能在定义视图的 select 语句中包含 order by、 compute 子句或关键字 into。
- 不能用包含 union 运算符的 select 语句从视图进行更新、插入或删除。
- 如果使用局部或全局变量创建视图， Adaptive Server 将发出错误消息 7351：“Local or global variables not allowed in view definition.”
- 可以在单个批处理中组合 create view 语句与其它 SQL 语句。

警告！ 当在 if...else 块或 while 循环中出现 create view 命令时，Adaptive Server 将在决定条件是否为真之前创建该视图的模式。如果该视图已经存在，则会导致错误。为了避免这种情况，请检验数据库中不存在具有相同名称的视图，或者按如下方式使用 execute 语句：

```
if not exists
    (select * from sysobjects where name="mytable")
begin
execute ("create table mytable (x int)")
end
```

视图解析

- 如果通过添加或删除列来改变视图基础表的结构，新列将不会出现在用 select * 子句定义的视图中，除非删除并重新定义视图。首次创建视图时，星号速记符得到了解释和扩展。
- 如果视图所依赖的表或视图已被删除，当有人试图使用该视图时，Adaptive Server 将返回错误消息。如果创建了具有相同名称和模式的新表或视图来替代已被删除的表或视图，则此视图将再次变为可用。
- 可以重新定义视图而不重新定义依赖于它的其它视图，除非重新定义使得 Adaptive Server 不能转换任何相关视图。

通过视图修改数据

- delete 语句不得用于多表视图。
- 除非基础表或视图中的所有 not null 列都包含在用来插入新行的视图中，否则不允许使用 insert 语句。Adaptive Server 无法为基础表或视图中的 not null 列提供值。

- 不能通过视图直接对计算列执行插入。计算列的值只能由 Adaptive Server 在内部生成。
- 在用 distinct 或 with check option 创建的连接视图中不允许使用 insert 语句。
- 在用 with check option 创建的连接视图中，允许使用 update 语句。如果任何受影响的列出现在 where 子句中，或出现在包含来自多个表的列的表达式中，更新将会失败。
- 如果通过连接视图来插入或更新行，所有受影响的列都必须属于同一基表。
- 不能对用 distinct 子句定义的视图进行更新或插入。
- 数据更新语句不能更改作为计算的视图中的任何列，也不能更改包含集合的视图。

IDENTITY 列和视图

- 不能使用 *column_name* = identity (*precision*) 语法向视图添加新的 IDENTITY 列。
- 要向 IDENTITY 列插入显式值，表的所有者、数据库所有者或系统管理员必须将列的基表设置为 set identity_insert *table_name* on，而不是通过插入所通过的视图。

group by 子句和视图

当出于安全性考虑而创建视图时，应谨慎使用集合函数和 group by 子句。 Transact-SQL 扩展允许命名不出现在 group by 子句中的列。如果命名不出现在 group by 子句中的列，则 Adaptive Server 会返回该列的详细数据行。例如，此 Transact- SQL 扩展列查询会为每 18 行返回一行 比您想得到的数据要多：

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

而下面的 ANSI 兼容的查询为每个类型返回一行 （共 6 行）：

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

请参见第 434 页的 “group by 和 having 子句”。

distinct 子句和视图

- distinct 子句将视图定义为不含重复行的数据库对象。如果某行的所有列值都与另一行的相同列值相匹配，则该行就被定义为另一行的重复行。 NULL 值被看作是其它空值的重复值。

查询视图的列的子集会得到看起来像重复行的内容。如果选择列的子集，其中有些包含相同的值，结果看起来包含重复行。但视图的基础行仍然唯一。第一次访问视图时 （进行投影和选择之前），Adaptive Server 对视图定义应用 distinct 要求，因此视图的所有行都互不相同。

作为集合函数或 group by 子句的一部分，您可以在视图定义的 select 语句中多次指定 distinct 来消除重复行。例如：

```
select distinct count (distinct title_id), price
from titles
```

- distinct 的范围仅适用于该视图，而不包括从 distinct 视图派生的任何新视图。

with check option 子句和视图

- 如果视图是用 with check option 创建的，则通过视图插入或创建的每一行都必须符合视图的选择标准。
- 如果视图是用 with check option 创建的，则所有从 “基” 视图派生的视图都必须满足其检查选项要求。通过派生视图插入或更新的每一行都必须能通过基视图查看。

获取有关视图的信息

- 若要创建有关视图依赖的表或视图的报告，以及依赖于视图的对象的报告，请执行 sp_depends。
- 若要显示视图的文本 （存储在 syscomments 中），请以视图名作为参数执行 sp_helptext。

从 SQL 派生表创建视图

- 要用 SQL 派生表创建视图，请在 create view 语句的 select 部分的 from 子句中添加派生表表达式 （请参见示例 11）。
- 如果派生表表达式可以更新，则使用 SQL 派生表创建的视图也可以更新。派生表表达式的更新规则遵循 create view 语句的 select 部分的更新规则。
- 如果用于派生表表达式的 insert 规则和权限设置遵守用于 create view 语句的 select 部分的 insert 规则和权限设置，则可通过包含 SQL 派生表的视图插入数据。
- 作为 create view 语句的一部分的派生表表达式中不允许使用临时表和局部变量。
- SQL 派生表不能具有未命名的列。
- 有关派生表表达式的详细信息，请参见 《Transact-SQL 用户指南》。

标准

符合 ANSI SQL 的级别符合初级标准。

在 select 列表中多次使用 distinct 关键字和使用 “*column_heading* = *column_name*” 属于 Transact-SQL 扩展。

权限

创建视图时， Adaptive Server 不对视图所引用的对象 （如表和视图）进行权限检查。因此，即使不能访问视图的对象，也可以成功创建视图。所有权限检查在用户调用该视图时进行。

调用视图时，对象上的权限检查取决于视图和所有被引用的对象是否由同一用户拥有。

- 如果视图及其对象的所有者不是同一用户，那么调用者必须得到可以直接访问这些对象的授权。例如，如果视图从调用者不能访问的表执行 select， select 语句就会失败。
- 如果视图及其对象为同一用户所有，则应用特殊的规则。调用者自动拥有访问视图的对象的隐式权限，即使调用者不能直接访问它们。不必授予用户直接访问表的权限，可以给予他们有限制地访问视图的权限。这样，视图就可以是一种安全机制。例如，视图的调用者可能只能访问表的某些行和列。《系统管理指南》中详细说明了隐式权限的规则。
- 如果表中的列已加密，您必须拥有解密权限才能从视图中执行选择操作。如果视图及其对象不归同一用户所有，则必须对表中的加密列拥有解密权限，才能从视图中执行选择操作。如果视图及其对象归同一用户所有，则在与表中的加密列对应的视图列中向必须从视图中执行选择操作的用户授予解密权限便足够了。

细化权限已启用	在启用细化权限的情况下，您必须有 create view 特权。您必须具有 create any view 特权才能为其他用户运行 create view。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者或者具有 create view 特权。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
16	创建	create view	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

文档 《参考手册：构件块》的第 4 章 “表达式、标识符和通配符”中的 “标识符” .

命令 create schema, drop view, select, update.

系统过程 sp_depends, sp_help, sp_helptext, sp_rename.

dbcc

说明

数据库一致性检查程序 (dbcc) 检查数据库的逻辑和物理一致性，并提供统计、计划和修复功能。

有些 dbcc 命令只适用于共享磁盘集群。请参见针对各集群分别列出的 dbcc 语法。

语法

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' ] ] )
```

仅适用于集群的 dbcc 语法：

```
dbcc nodetraceon(trace_flag_number)
dbcc nodetraceoff(trace_flag_number)

dbcc set_scope_in_cluster("cluster"|"instance"|"scope")

dbcc quorum
```

参数

addtempdb

将一个临时数据库添加到可用临时数据库的全局列表。如果数据库不存在或不是临时数据库，则会产生错误。如果数据库已经是列表的成员，则会显示信息性消息。

dbid

是数据库 ID。

database_name

是要检查的数据库名。如果不提供数据库名，则 dbcc 使用当前数据库。

checkalloc

检查指定数据库，来确定正确分配了所有页，且使用了所有分配的页。如果不提供数据库名，则 checkalloc 检查当前数据库。它通常使用 optimized 报告选项 （请参见 tablealloc）。

checkalloc 报告分配和使用的空间量。

fix | nofix

确定 dbcc 是否修正发现的分配错误。 checkalloc 的缺省模式是 nofix。必须将数据库置于单用户模式才能使用 fix 选项。有关 Adaptive Server 中页分配的详细信息，请参见 《系统管理指南》。

checkcatalog
在系统表中和系统表之间检查一致性。例如， checkcatalog 确认 syscolumns 中的每一个类型在 systypes 中都有一个匹配的条目，sysobjects 中的每个表和视图在 syscolumns 中都至少有一列，且 syslogs 中的最后一个检查点是有效的。请参见第 276 页的 “dbcc checkcatalog 执行的检查”。您可以在存档数据库中使用 checkcatalog，但不是 fix 版本的 checkcatalog。

checkcatalog 还报告已定义的任何段。如果不提供数据库名，则 checkcatalog 检查当前数据库。

fix
确定 dbcc 是否修复它所找到的 sysindexes 错误。 checkcatalog 的缺省模式是不修复错误。必须将数据库置于单用户模式才能使用 fix 选项。新的 sysindexes 检查可能会产生新的错误，它们不是由低于 12.5.2 版的 Adaptive Server 中的 dbcc checkcatalog 引起的。

checkdb
运行与 checktable 相同的检查，但是检查指定数据库中的每个表，包括 syslogs。如果不提供数据库名，则 checkdb 检查当前数据库。您可以在存档数据库中使用 checkdb。

skip_ncindex
导致 dbcc checktable 或 dbcc checkdb 跳过对用户表上非聚簇索引的检查。缺省情况下是检查所有索引。

checkindex
运行与 checktable 相同的检查，但是只检查指定的索引。您可以在存档数据库中使用 checkindex。

bottom_up
（仅针对仅数据锁定表）当在 checkindex 中指定此选项时，按照自下而上的顺序检查索引。 bottom_up 检查涉及检验每个数据行是否有对应的索引行。

partition_name | *partition_id*
为要检查的数据分区的名称或 ID。如果指定分区，则 dbcc 跳过全局索引。

checkstorage
检查指定数据库的分配、对象分配映射 (OAM) 页条目、页一致性、文本值列、文本值列的分配以及文本列链。每个 dbcc checkstorage 操作的结果都存储在 dbccdb 数据库中。有关使用 dbcc checkstorage 的详细信息，以及有关从 dbccdb 创建、维护和生成报告的详细信息，请参见 《系统管理指南》。

checktable
检查指定的表，来查看索引和数据页是否正确链接，索引的排序顺序是否正确，所有指针是否一致，每页上的数据信息是否合理，以及页偏移是否合理。您可以在存档数据库中使用 checktable。

对 dbcc checktable 的某些更改会引用虚拟散列表：

- 除了执行常规检查外， checktable 还会验证散列区域中数据和 OAM 页的布局是否正确。
  - 不在为每个布局的 OAM 页保留的扩充中分配数据页。
  - 只在分配单元的第一个扩充中分配 OAM 页。

table_name | *table_id*
是要检查的表的名称或对象 ID。

fix_spacebits
用于使用数据页或数据行锁定的表，并检查空白位的有效性，以及修正任何无效的空白位。空间位按页存储，并指示页中可用于新的插入的空间。

check spacebits
检查使用数据页或数据行锁定表的表中的空白位。如果指定 check spacebits，则 dbcc 不检查非聚簇索引。

checkverify
检验针对指定的数据库最后一次运行 dbcc checkstorage 的结果。有关使用 dbcc checkverify 的详细信息，请参见 《系统管理指南》。

ignore_exclusions
启用或禁用排除列表。值为缺省值 0 （启用排除列表）或 1 （禁用排除列表）。

complete_xact
通过提交或回退其工作尝试完成一个事务。 Adaptive Server 保留 master.dbo.systransactions 表中关于所有尝试完成的事务的信息，因此外部事务协调器能够知道事务是如何完成的。

警告！ 尝试完成处于就绪状态的事务时，可能会在整个分布式事务中造成不一致的结果。系统管理员尝试提交或回退事务的决定可能会与进行协调的 Adaptive Server 或协议的决定相矛盾。

---

xid
是 systransactions.xactname 列中的事务名称。也可以用 sp_transactions 确定有效的 xid 值。

1pc

通过正在协调此完成过程的外部事务管理器， 1pc 尝试完成一个进行单阶段提交协议 （不是常规的两阶段提交协议）优化的事务。此选项允许尝试提交未处于就绪状态的事务。

forget_xact

删除来自 master.dbo.systransactions 的尝试完成事务的完成状态。forget_xact 可以在系统管理员不希望协调服务知道正尝试完成的事务时，或者是当外部事务协调器不可用于清除 systransactions 中的提交状态时使用。

警告！ 不要在常规 DTP 环境中使用 dbcc forget_xact，因为应该允许外部事务协调器检测尝试完成的事务。 X/Open XA 兼容的事务管理器和 Adaptive Server 事务协调服务可自动清除 systransactions 中的提交状态。

---

dbrepair (*database_name*, dropdb)

删除损坏的数据库。不能对损坏的数据库执行 drop database。

发出此 dbcc 语句时，任何人都不能使用正被删除的数据库 （包括发出此语句的用户）。

engine

将 Adaptive Server 引擎脱机或联机。如果未指定 *enginenum*，则 dbcc engine (offline) 会使编号最高的引擎脱机。请参见 《系统管理指南》中的 “管理多处理器服务器”。

fix_text

在 Adaptive Server 字符集从任意字符集更改为新的多字节字符集之后升级 text 值。

更改为多字节字符集会使 text 数据的管理更为复杂。由于 text 值可以大到占据多页，所以 Adaptive Server 必须能够处理跨越页边界的字符。为此，服务器需要有关每个 text 页的其它信息。系统管理员或表所有者必须在每个包含 text 数据的表上运行 dbcc fix_text，才能计算所需的新值。请参见 《系统管理指南》。

indexalloc

检查指定索引，来确定正确分配了所有页，且使用了所有分配的页。这是 checkalloc 的较小版本，可对单个索引进行相同的完整性检查。您可以在存档数据库中使用 indexalloc。

indexalloc 生成与 tablealloc 相同的三种类型的报告：full、 optimized 和 fast。如果未指定任何类型或者使用 null， Adaptive Server 将使用 optimized。fix | nofix 选项也可用于 indexalloc，就像用于 tablealloc 一样。

注释 注意：只有包括报告的类型值 （full、 optimized、 fast 或 null）时，才可以指定 fix或 nofix。

---

table_name | *table_id*

是表名或表的对象 ID。

indid

是执行 dbcc indexalloc 期间所检查的索引的 ID。

fix_spacebits

用于 datapages 或 datarows lockscheme 类型的表，并检查空间位的有效性，以及修正任何无效的空间位。空间位按页存储，并指示页中可用于新的插入的空间。

check spacebits

检查数据页或数据行锁定表的空白位。如果指定 check spacebits，则 dbcc 不检查非聚簇索引。

full

报告所有类型的分配错误。

optimized

根据索引的对象分配映射 (OAM) 中列出的分配页生成报告。它不报告也无法修正 OAM 页中未列出的分配页中的未引用扩充。 optimized 选项是缺省选项。

fast

不生成分配报告，但对已引用而未在扩充中分配的页生成例外报告（2521 级别错误）。

fix | nofix

确定 indexalloc 是否修正表中发现的错误。对除系统表中的索引外的所有索引来说缺省值是 fix，系统表中的索引的缺省值是 nofix。若要将 fix 选项用于系统表，必须先将数据库设置为单用户模式。

只有包括报告的类型值 （full、 optimized、 fast 或 null）时才可以指定 fix 或 nofix。

partition_name | *partition_id*
如果指定了分区 ID，则分配检查将在由 (indid, partition id) 所标识的分区上执行。

pravailabletempdbs
显示可用临时数据库的全局列表。

rebuild_text
为 text、unitext 或 image 数据重建或创建内部 Adaptive Server 12.0 或更高版本数据结构。此数据结构可使 Adaptive Server 在数据查询期间执行随机访问和异步预取。您可以对数据库中的所有表、单个表或数据分区运行 rebuild_text。

table_name | *table_id* | "all"
是表名或表的对象 ID，或者数据库中的所有对象。

column
是文本列的 ID 或列名。 dbcc rebuild_text 会重建此列的每个文本值的内部数据结构。

text_page
是第一个文本页的逻辑页码。 dbcc rebuild_text 会重建此文本页的内部数据结构。

data_partition_name | *data_partition_id*
为数据分区的名称或 ID。如果指定了 *text_page*，则忽略 *data_partition_name* （或 *data_partition_id*）。

monitor increment, *group name*
increment 命令将指定组中的监控计数器的使用量计数增加 1，decrement 命令将指定组中的监控计数器的使用量计数减少 1。reset 命令将指定组中的监控计数器的使用量计数设置为 0。这将关闭此组的监控数据的收集。

group name 可以是下列值之一：

- ‘all’ – 通过选择 @@*monitors_active* 全局变量确定 all 组的使用量计数，该组包含大部分监控计数器。
- spinlock_s – 由 dbcc resource 命令报告的 spinlock_s 的使用量计数。
- appl – 由 dbcc resource 命令报告的 appl 的使用情况计数。

reindex

通过运行 dbcc checktable 的快速版本，检查用户表索引的完整性。它可以和表名或表的对象 ID （来自 sysobjects 的 id 列）一起使用。reindex 在发现第一个与索引相关的错误时会显示消息，然后删除并重新创建可疑索引。当 Adaptive Server 的排序顺序已经更改，且索引由 Adaptive Server 标记为“可疑”后，系统管理员或表所有者必须运行 dbcc reindex。

当 dbcc 发现损坏的索引时，它会删除并重新创建相应的索引。如果表的索引已经是正确的，或者表没有索引，则 dbcc reindex 不重建索引，而是输出信息性消息。

如果怀疑表包含损坏的数据，则 dbcc reindex 将中止。发生这种情况时，错误消息会指示用户运行 dbcc checktable。dbcc reindex 不允许重建系统表索引。如果有必要，在排序顺序更改后重新启动 Adaptive Server，然后作为恢复操作的一个自动执行的部分检查并重建系统索引。

stackused

报告自从服务器首次启动以来所使用的最大堆栈内存量。

serverlimits

显示 Adaptive Server 对多种实体施加的限制，包括标识符的长度以及不同对象的最大数量 （如表中的列数、表上的索引数、页大小、行开销等）。使用这些信息可确定 Adaptive Server 进程的各种大小特性。

tablealloc

检查指定的表或数据分区，以确定正确分配了所有页，且所有分配的页都已在使用。这是 checkalloc 的较小版本，可对单个表进行相同的完整性检查。它可以和表名或表的对象 ID （来自 sysobjects 的 id 列）一起使用。您可以在存档数据库中使用 tablealloc。有关 tablealloc 输出的示例，请参见 《系统管理指南》。

可使用 tablealloc 生成的三种类型的报告：full、 optimized 和 fast。如果未指定任何类型或者使用 null， Adaptive Server 将使用 optimized。

textalloc

检查数据库中的 text 或 image 页的分配完整性。可将 dbcc textalloc 用于存档数据库。

full

等同于表级 checkalloc，它报告所有类型的分配错误。

optimized

根据表的对象分配映射 (OAM) 页中列出的分配页生成报告。它不报告也无法修正 OAM 页中未列出的分配页中的未引用扩充。 optimized 选项是缺省选项。

fast
不生成分配报告，但对已引用而未在扩充中分配的页生成例外报告（2521 级别错误）。

fix | nofix
确定 tablealloc 是否修正了表中发现的分配错误。除系统表外的所有表的缺省值是 fix，系统表的缺省值是 nofix。若要将 fix 选项用于系统表，必须先将数据库设置为单用户模式。

只有包括报告的类型值 （full、 optimized、 fast 或 null）时才可以指定 fix 或 nofix。

data_partition_name | *data_partition_id*
为要检查的数据分区的名称或 ID。如果指定了分区，则 dbcc tablealloc 将跳过全局索引。

traceon | traceoff
在查询优化期间切换诊断显示。值 3604 和 3605 分别切换为向用户会话以及向错误日志发送跟踪输出。

tune
启用或禁用在特殊性能情况下使用的调优标志。每次重新启动 Adaptive Server 时，必须重新发出 dbcc tune。有关各选项的详细信息，请参见 《性能和调优指南：基础知识》。

upgrade_object
从存储在 syscomments 表中的文本升级编译对象。upgrade_object 参数有：

参数	说明
dbid	指定数据库 ID。如果不指定 *dbid*，将升级当前数据库中的所有编译对象。
dbname	指定数据库名。如果不指定 *dbname*，将升级当前数据库中的所有编译对象。
compiled_object_name	要升级的具体编译对象的名称。如果使用完全限定名，则 *dbname* 和 *database* 必须匹配，且必须用引号将完全限定名括起来。如果该数据库包含两个以上同名的编译对象，则应使用完全限定名称。否则，将分析所有同名对象，如果未发现错误则进行升级。
check	升级所有检查约束和规则。参照约束不是编译对象，不需要升级。
default	升级所有声明缺省值和用 create default 命令创建的缺省值。
procedure	升级所有存储过程。
rule	升级所有规则和检查约束。
trigger	升级所有触发器。
view	升级所有视图。

参数	说明
force	指定要升级指定的对象，不管其中是否包含 select * 子句。除非能确认 select * 语句不会返回意外结果，否则不要使用 force 选项。force 选项不会升级以下对象：包含保留字的对象、包含截断的源文本或丢失的源文本的对象、引用不存在的临时表的对象或者与带引号标识符设置不匹配的对象。必须先修复这些对象，然后再升级。

关键字 check、default、procedure、rule、trigger 和 view 指定要升级的编译对象类。当指定了一个类后，指定数据库中的所有该类对象都将被升级，前提条件是 dbcc upgrade_object 没有发现错误或潜在问题区域。

check
检查指定数据库的 syscomments 中指定的编译对象的语法。不在出现 select 时引发错误。

对于 upgrade_object，将升级所有检查约束和规则。参照约束不是编译对象，不需要升级。

force
强制 syscomments 中的对象进行升级，即使没有必要升级也是如此。

object_name
为编译对象的名称。

object_type
为 Adaptive Server 编译的下列对象类型之一：procedure、 function、 view、 trigger、 default、 rule 以及 condition。

compiled_object_name
是要升级的具体编译对象的名称。如果使用完全限定名称，则 *database_name* 和 *database* 必须匹配，且必须用引号括住完全限定名称。如果该数据库包含两个以上同名的编译对象，则应使用完全限定名称。否则，将分析所有同名对象，如果未发现错误则进行升级。

default
升级所有声明缺省值和用 create default 命令创建的缺省值。

procedure
升级所有存储过程。

rule
升级所有规则和检查约束。

trigger
升级所有触发器。

view
升级所有视图。

关键字 check、default、procedure、rule、trigger 和 view 指定要升级的编译对象类。当指定了一个类后，指定数据库中的所有该类对象都将被升级，前提条件是 dbcc upgrade_object 没有发现错误或潜在问题区域。

force
指定要升级指定的对象，不管其中是否包含 select * 子句。除非能确认 select * 语句不会返回意外结果，否则不要使用 force 选项。force 选项不会升级以下对象：包含保留字的对象、包含截断的源文本或丢失的源文本的对象、引用不存在的临时表的对象或者与带引号标识符设置不匹配的对象。必须先修复这些对象，然后再升级。

trace_flag_number
是要启用或禁用的跟踪标志的编号。

cluster
将 dbcc 命令范围设置为集群。后续的 dbcc 命令将在整个集群范围内产生影响。

instance
将 dbcc 命令范围设置为当前实例。后续的 dbcc 命令只影响本地实例。

scope
显示 dbcc 命令的当前范围，即 cluster 或 instance。

示例

示例 1 检查 pubs2 以查找页分配错误：

```
dbcc checkalloc (pubs2)
```

示例 2 检查 pubs2 的数据库一致性并将信息放入 dbccdb 数据库：

```
dbcc checkstorage (pubs2)
```

示例 3 检查 salesdetail 表：

```
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.
```

示例 4 尝试中止事务 “distributedxact1”：

```
dbcc complete_xact (distributedxact1, "rollback")
```

示例 5 更改字符集后升级 blurbs 的文本值：

```
dbcc fix_text (blurbs)
```

示例 6 在禁用排除列表的情况下在数据库 my_db 中对表 tab 运行 checkverify：

```
dbcc checkverify(my_db, tab)
```

示例 7 在启用排除列表的情况下在数据库 my_db 中对表 tab 运行 dbcc checkverify：

```
dbcc checkverify (my_db, tab, 0)
```

示例 8 在禁用排除列表的情况下在数据库 my_db 中对表 tab 运行 dbcc checkverify，请输入：

```
dbcc checkverify (my_db, tab, 1)
```

示例 9 从 master.dbo.systransactions 中删除事务 “distributedxact1”的信息：

```
dbcc forget_xact (distributedxact1)
```

示例 10 返回有关索引 （其 titleauthor 表上的 indid 值为 2）的分配的完整报告，并修复所有分配错误：

```
dbcc indexalloc ("pubs..titleauthor", 2, full)
```

示例 11 显示可用临时数据库的全局列表：

```
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.
```

示例 12 为 blurbs 表中的所有 text 和 image 列重建或创建内部 Adaptive Server 数据结构：

```
dbcc rebuild_text (blurbs)
```

示例 13 检查位于 smallsales 分区 （包含小于 5000 的所有图书销售额）的 titles 表部分。

```
dbcc checktable (titles, NULL, "smallsales")
```

示例 14 dbcc reindex 在 titles 表中发现了一个或多个损坏的索引：

```
dbcc reindex (titles)

One or more indexes are corrupt.They will be rebuilt.
```

示例 15 检查自从 Adaptive Server 启动以来所使用的最大堆栈内存量：

```
dbcc stackused
```

示例 16 升级 listdb 数据库中的所有存储过程：

```
dbcc upgrade_object(listdb, 'procedure')
```

示例 17 升级 listdb 数据库中的所有规则和检查约束。应使用双引号括住 rule，因为 set quoted identifiers 为 off。

```
dbcc upgrade_object(listdb, list_proc)
```

示例 18 显示简化的输出，其中显示 Adaptive Server 中的各种限制形式：

```
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
...
```

注释　注意：若要显示服务器中的完整限制列表，请执行 dbcc traceon (3604)获取客户端会话的输出。

示例 19 为此表返回经过优化的分配报告，但不修复任何分配错误：

```
dbcc tablealloc (publishers, null, nofix)
```

示例 20 对 smallsales 分区执行分配检查。检查 smallsales 上的所有本地索引，而将全局索引排除在外：

```
dbcc tablealloc (titles, null, null, smallsales)
```

示例 21 使用 sp_transactions 来确定因为未处于 “就绪”状态而未尝试提交的单阶段提交事务的名称。接下来，本示例解释如何使用 1pc 参数来成功提交事务：

```
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_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM000FU_00003M00
        00Y_:SYBBEV0A_LRM
(1 row affected)

 (return status = 0)
```

如果您尝试提交此事务，则 Adaptive Server 将发出错误消息：

```
dbcc complete_xact
("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM000FU_00003M000
0Y_:SYBBEV0A_LRM", "commit"))
```

Adaptive Server 发出的错误消息为：

```
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.
```

因为此事务处于“完成”状态，检验此事务已提交后，可以使用单阶段提交协议优化来尝试完成此事务。可以使用 dbcc complete_xact（“1pc”）参数提交此事务：

```
dbcc complete_xact
("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM000FU_00003M000
0Y_:SYBBEV0A_LRM", "commit", "1pc")
DBCC execution completed.If DBCC printed error messages, contact a user with
system administrator (SA) role.
```

可以用 dbcc forget_xact 命令从 systransactions 中删除此事务：

```
dbcc forget_xact
("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM0
00FU_00003M0000Y_:SYBBEV0A_LRM")
DBCC execution completed.If DBCC printed error messages, contact a user with
system administrator (SA) role.
```

如果再次运行 sp_transactions，则上一事务不会出现：

```
sp_transactions
xactkey  type  coordinator  starttime  state  connection  dbid  spid
      loid   failover  srvnname  namelen  xactname
-------- ----- ------------ ---------- ------ ----------- ----- -----
      ------ --------- --------- -------- --------
 (0 row affected)
```

示例 22 启用跟踪标志 3604：

```
dbcc nodetraceoff(3604)
DBCC execution completed.If DBCC printed error
messages, contact a user with system administrator (SA)
role.
```

示例 23 将 dbcc 范围设置为集群：

```
dbcc set_scope_in_cluster('cluster')
```

示例 24 将 dbcc 范围设置为实例：

```
dbcc set_scope_in_cluster('instance')
```

示例 25 显示 dbcc 命令的当前范围：

```
dbcc set_scope_in_cluster('scope')
```

用法

- dbcc checkstorage 如果用于非散列表的除第一个数据页之外的任何数据页为空，则报告软故障。然而，dbcc checkstorage 不为虚拟散列表的散列区域报告此软故障。虚拟散列表的散列区域中的任何数据页均可为空。

- 可在数据库处于活动状态时运行 dbcc，但是不能使用 dbrepair (*database_name*, dropdb) 选项以及带 fix 选项的 dbcc checkalloc。
- dbcc 在检查数据库对象时会锁定它们。有关在使用 dbcc 时尽可能减少性能问题的信息，请参见 《系统管理指南》中的 dbcc 讨论。
- 执行 dbcc 命令时，用户不能访问存档数据库。如果您在 dbcc 命令的执行过程中试图访问存档数据库，则会收到一条消息，说明数据库正处于单用户模式。
- 大多数 dbcc 命令都可用于内存数据库和宽松持久性数据库。
- 对于处于联机或脱机状态下的存档数据库，可以使用 dbcc 命令的变体。然而，您只能对联机存档数据库使用带有 fix 选项的 dbcc。
- 若要用用户名或数据库名限定表或索引的名称，请将限定的名称用单引号或双引号引起来。例如：

  ```
  dbcc tablealloc ("pubs2.pogo.testtable")
  ```

- 不能在用户定义的事务中运行 dbcc reindex。
- dbcc fix_text 可以生成大量的日志记录，这些记录可能会填满事务日志。 dbcc fix_text 设计为在一系列小事务中进行更新：万一出现日志空间故障，则只有少量工作丢失。如果日志空间已用完，则清除日志，并使用在初始 dbcc fix_text 失败时升级的同一个表重新启动 dbcc fix_text。
- 如果您使用的是已复制的数据库，并且需要将转储从以前版本的 Adaptive Server 装载到当前版本中，请使用 dbcc dbrepair。例如：
  - 将转储从早期版本的 Adaptive Server 的生产系统装载到当前版本的 Adaptive Server 的测试系统，或者，
  - 在热备份应用程序中，使用早期版本的 Adaptive Server 的活动数据库中的数据库转储初始化当前版本的 Adaptive Server 的备用数据库。
- 如果试图在更改为多字节字符集后在 text 值上使用 select、 readtext 或 writetext，且没有运行 dbcc fix_text，则命令会失败，并会有一个错误消息指示您在表上运行 dbcc fix_text。但是，更改了字符集之后，即使不运行 dbcc fix_text，您也可以删除 text 行。
- dbcc 输出是作为消息或错误而不是作为结果行发送的。客户端程序和脚本应该检查相关的错误处理程序。
- 如果表是分区的， dbcc checktable 返回有关每个分区的信息。

- 已升级到 Adaptive Server 12.x 版或更高版本的 text 和 image 数据不会自动从原来的存储格式升级。若要提高查询性能，并启用对这些数据的预取，请对已升级的 text 和 image 列使用 rebuild_text 关键字。
- 过去使用的堆栈内存量仅能指出将来可能需要的堆栈内存量。Adaptive Server 所需的堆栈内存可能比过去所用的还要多。定期运行 dbcc stackused 可了解当前堆栈的内存使用情况。
- dbcc upgrade_object check 用于检测升级 Adaptive Server 前由 Adaptive Server 缺陷引起的 syscomments 文本损坏。这种 syscomments 文本损坏后果严重，因为它会导致升级过程失败。
- 如果 dbcc upgrade_object check 报告任何错误，则必须删除并重新创建 *compiled_object*。

dbcc complete_xact

dbcc complete_xact 使系统管理员能够在外部事务协调器无法提交或回退分布式事务的环境中提交或回退分布式事务。在低于 Adaptive Server 15.0 的版本中，除非事务处于 “就绪” 状态，否则不能尝试提交，事务协调器使用两阶段提交协议来提交事务。但是在某些情况下，事务协调器可能需要使用单阶段提交协议来优化事务的提交。

1pc 尝试完成一个进行单阶段提交协议 （不是常规的两阶段提交协议）优化的事务，它通过正在协调此完成过程的外部事务管理器来执行该操作。尝试提交此类事务要求事务处于 “完成” (done) 状态 （由 sp_transactions 报告）。

注释 尝试完成事务之前，系统管理员应该尽一切努力确定协调事务管理器是提交还是回退了分布式事务。

---

dbcc checkcatalog 执行的检查

dbcc checkcatalog 检查：

- 对于 sysindexes 中映射到域、散列或列表分区表的每一行，在 sysobjects 中都存在满足下列条件的一行：sysindexes.conditionid 等于 sysobjects.id。 dbcc checkcatalog 还对 sysindexes 中映射到带有分区条件的循环分区表的每一行执行此检查。
- 对于 sysindexes 中映射到域、散列或列表分区表的每一行，在 sysprocedures 中都存在满足下列条件的一行：sysindexes.conditionid 等于 sysprocedures.id。 dbcc checkcatalog 还对 sysindexes 中映射到带有分区条件的循环分区表的每一行执行此检查。

- 对于 sysindexes 中映射到域、散列或列表分区表的每一行，在 syspartitionkeys 中存在满足下列条件的一行：sysindexes.id 等于 syspartitionkeys.id 且 sysindexes.indid 等于 syspartitionkeys.indid。 dbcc checkcatalog 还对 sysindexes 中映射到带有分区条件的循环分区表的每一行执行此检查。
- 对于 sysindexes 中的每一行， syspartitions 中存在同时满足下面两个条件的一行或多行：sysindexes.id 等于 syspartitions.id 且 sysindexes.indid 等于 syspartitions.indid。
- 对于 sysobjects 中类型为 N 的每一行，在 sysindexes 中都存在 sysindexes.conditionid 等于 sysobjects.id 的一行。
- 对于 syspartitions 中的每一行，在 sysindexes 中存在满足下列条件的一行：syspartitions.id 等于 sysindexes.id 且 syspartitions.indid 等于 sysindexes.indid。
- 对于 syspartitionkeys 中的每一行，在 sysindexes 中存在满足下列条件的一行：syspartitionkeys.id 等于 sysindexes.id 且 syspartitionkeys.indid 等于 sysindexes.indid。
- 对于 syspartitions 中的每一行，在 syssegments 中存在满足以下条件的一行：syspartitions.segments 等于 syssegments.segment。
- 对于 systabstats 中的每一行，在 syspartitions 中都存在满足下列条件的一行：syspartitions.id 等于 systabstats.id、 syspartitions.indid 等于 systabstats.indid 且 syspartitions.partitionid 等于 systabstats.partitionid。

  文本索引 (indid=255) 在 systabstats 中没有条目。
- 对于 sysstatistics 中的每一行，在 sysobjects 中都存在满足以下条件的一行：sysstatistics.id 等于 sysobjects.id。
- 对于 sysobjects 中的每个加密密钥行， Adaptive Server 都会检查 sysencryptkeys 中是否有定义该密钥的行。
- 对于标记为加密的 syscolumns 中的每个列， Adaptive Server 都将在 sysobjects 和 sysencryptkeys 中检验密钥。
- dbcc checkcatalog 确保：
  - 对于 sysencryptkeys 中的每个密钥副本，sysencryptkeys 中均存在对应的基本密钥。如果不存在基本密钥，则 Adaptive Server 会发出错误消息。
  - 对于每个密钥副本， sysusers 中均存在对应的 uid。如果不存在 uid，则 Adaptive Server 会发出错误消息。

- 对于列中定义的每个解密缺省值，sysobjects 和 sysattributes 中均存在对应的解密缺省值。如果不存在对应的解密缺省值，则 Adaptive Server 会发出错误消息。

使用 dbcc checktable

如果日志段位于它自己的设备上，那么对 syslogs 表运行 dbcc checktable 将报告已用的日志和可用空间。例如：

```
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.
```

如果日志段不在自己的设备上，就会显示以下消息：

```
*** NOTICE:  Notification of log space used/free cannot be reported because the
log segment is not on its own device.
```

除了执行常规检查外， checktable 还会验证在创建表期间执行的预分配是否正确：

- 预分配的页数与为指定的 *max hash key* 值分配的数据页总数匹配。
- 在预分配方案指定只允许对象分配映射 (OAM) 页的扩充中，不对数据页进行预分配。
- 只在分配单元的第一个扩充中分配 OAM 页。

使用 dbcc nodetraceoff 和 dbcc nodetraceon （仅限集群）

dbcc traceon 和 dbcc traceoff 对整个集群应用跟踪标志，而 dbcc nodetraceoff 和 dbcc nodetraceon 在本地应用跟踪标志。

使用 dbcc quorum （仅限集群）

- dbcc quorum 输出传送到：
  - 启动 Adaptive Server 的终端 （缺省情况下）
  - 客户端会话 （如果打开了跟踪标志 3604 或 3605）
- dbcc quorum 接受整数参数作为要输出的视图记录数。例如，若要输出 20 条最新的视图记录，请使用：

  ```
  dbcc quorum(20)
  ```

- 如果不包括参数， dbcc quorum 将输出 10 条最新的视图记录。
- 发出 dbcc quorum (-1) 则可查看所有记录。

用于共享磁盘集群的 dbcc checkstorage 的限制：

- dbcc checkstorage 中不能包括仅限实例的命名高速缓存。如果您这样做， dbcc checkstorage 将发出以下错误消息：

  ```
  The cache %1! cannot be used because it is an
  instance only cache
  ```

- 若要针对本地临时数据库运行 dbcc checkstorage，必须从拥有本地临时数据库的同一实例中运行该命令。
- 由于性能原因， Cluster Edition 中的 dbcc checkstorage 可能不在集群中查询页的最新版本。这可能会导致 Cluster Edition 比其它版本报告更多软故障。

  对于由单个实例更新数据库的明确分区的应用程序， dbcc checkstorage 的行为与早期非 Adaptive Server Cluster Edition 版本相同。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 dbcc 命令的权限检查因您的细化权限设置而异。有关每个 dbcc 命令的权限要求，请参见表 1-17。

注释　如果符合下表中列出的 dbcc 命令的任何一个要求 （权限或所有权），便可运行该命令。

表 1-17: dbcc 命令的权限要求

DBCC 命令名	权限要求	
	细化权限已禁用	细化权限已启用
addtempdb	• 数据库所有者 • sa_role	• 数据库所有者 • own database
checkalloc	• dbcc checkalloc • 数据库所有者 • sa_role	• dbcc checkalloc
checkcatalog	• dbcc checkcatalog • 数据库所有者 • sa_role	• dbcc checkcatalog
checkdb	• dbcc checkdb • 数据库所有者 • sa_role	• dbcc checkdb
checkindex	• dbcc checkindex • 表所有者 • sa_role	• dbcc checkindex • 表所有者

DBCC 命令名	权限要求	
	细化权限已禁用	细化权限已启用
checkstorage	• dbcc checkstorage • 数据库所有者 • sa_role	• dbcc checkstorage
checktable	• dbcc checktable • 表所有者 • sa_role	• dbcc checktable • 表所有者
checkverify	• dbcc checkverify • 数据库所有者 • sa_role	• dbcc checkverify
complete_xact	• sa_role	• manage server
dbrepair dropdb	• 数据库所有者 • sa_role	• 数据库所有者 • own database
engine	• sa_role	• manage server
fix_text	• dbcc fix_text • 对象所有者 • sa_role	• dbcc fix_text • 对象所有者
forget_xact	• sa_role	• manage server
indexalloc	• dbcc indexalloc • 表所有者 • sa_role	• dbcc indexalloc • 表所有者
monitor	• sa_role	• manage server
rebuild_text	• 表所有者 • sa_role	• 表所有者 • manage database
reindex	• dbcc reindex • 表所有者 • sa_role	• dbcc reindex • 表所有者
stackused	• sa_role	• manager server
tablealloc	• dbcc tablealloc • 表所有者 • sa_role	• dbcc tablealloc • 表所有者
textalloc	• dbcc textalloc • 表所有者 • sa_role	• dbcc textalloc • 表所有者

DBCC 命令名	权限要求	
	细化权限已禁用	细化权限已启用
traceon | traceoff nodetraceon | nodetraceoff		
跟踪标志 – 3604、 3605	• sa_role	• set tracing • monitor qp performance • set switch
所有其它跟踪标志	• sa_role	• set switch
tune ascinserts	• dbcc tune • 表所有者 • sa_role	• dbcc tune • 表所有者
tune 所有其它参数	• dbcc tune • sa_role	• dbcc tune
upgrade_object	• 对象所有者 • 数据库所有者 • sa_role	• manage database • 对象所有者
set_scope_in_cluster	• sa_role	• manage cluster
quorum	• sa_role	• manage cluster

审计　　sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
81	dbcc	dbcc	• *角色* – 当前活动角色 • *关键字或选项* – 任一 dbcc 关键字 （例如 checkstorage）和该关键字的选项 • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见　　**命令** drop database.

系统过程 sp_configure, sp_helpdb.

deallocate cursor

说明 使游标不可访问并释放所有提交到该游标的内存资源。

语法 deallocate [cursor] *cursor_name*

参数 *cursor_name*
是要释放的游标的名称。

示例 **示例 1** 释放名为 “authors_crsr”的游标：

```
deallocate cursor authors_crsr
```

示例 2 释放名为 “authors_crsr”的游标，但从语法中省略 cursor：

```
deallocate authors_crsr
```

用法

- 可将 deallocate cursor 用于存档数据库。
- 如果游标不存在，则 Adaptive Server 返回错误消息。
- 在将游标的名称作为另一 declare cursor 语句的一部分使用之前必须释放游标。
- 在存储过程或触发器中指定时， deallocate cursor 对内存资源使用情况没有影响。
- 无论游标是打开的还是关闭的，都可以对它使用 deallocate 命令。

标准 符合 ANSI SQL 的级别Transact-SQL 扩展。

权限 使用 deallocate cursor 无需任何权限。

另请参见 **命令** close, declare cursor.

deallocate locator

说明　删除内存中存储的大对象 (LOB) 并使其 LOB 定位符无效。

语法　deallocate locator *locator_descriptor*

参数　*locator_descriptor*

LOB 定位符的有效表示：宿主变量、局部变量或定位符的实际二进制值。

示例　释放 *@v* 引用的 LOB：

```
deallocate locator @v
```

用法

- 在事务内使用 deallocate locator。 Adaptive Server 会在事务结束时自动释放每个定位符。
- deallocate locator 可以节约内存。在事务内创建许多 LOB 定位符时，请在各个 LOB 和定位符不再需要时使用 deallocate locator 来删除它们。

权限　任何用户都可以执行 deallocate locator。

另请参见　**命令**：truncate lob.

Transact-SQL 函数：locator_literal、 locator_valid、 return_lob、 create_locator.

declare

说明

为批处理或过程声明局部变量的名称和类型。

语法

变量声明：

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

变量赋值：

```
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]]
```

参数

@variable_name

必须以 @ 开头且必须遵循标识符规则。

datatype

可以是系统数据类型，也可以是用户定义数据类型。

示例

示例 1 声明两个变量并根据变量中的值输出字符串：

```
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
```

示例 2 如果 titles 表中的最高书价高于 $20.00，则输出 “Ouch!”：

```
declare @veryhigh money
select @veryhigh = max (price)
    from titles
if @veryhigh > $20
    print "Ouch!"
```

用法

- 使用 select 语句为局部变量赋值。
- 过程中的最大参数数目是 2048。局部变量或全局变量的数目仅受可用内存的限制。 @ 符号表示变量名。

- 局部变量往往被用作 while 循环或 if...else 块的计数器。在存储过程中，它们被声明为过程执行时由过程自动的、非交互式的使用。局部变量必须在声明它们的批处理或过程中使用。
- 给局部变量赋值的 select 语句通常返回单值。如果有多个值需要返回，会为变量赋予最后一个值。为变量赋值的 select 语句不能用于在同一语句中检索数据。
- print 和 raiserror 命令可接受局部变量作为参数。
- 用户不能在 select 语句中直接创建全局变量，也不能直接更新全局变量的值。

标准　符合 ANSI SQL 的级别Transact-SQL 扩展。

权限　使用 declare 无需任何权限。

另请参见　**命令** print, raiserror, select, while.

declare cursor

说明

通过将 select 语句与游标名关联来定义游标。可将 declare cursor 用于存档数据库。

语法

```
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]}]
```

参数

cursor_name

是定义的游标的名称。

select_statement

是定义游标结果集的查询。有关详细信息，请参见 select。

semi_sensitive

指定独立于游标进行的数据更改可以对游标结果集可见。相关数据的可见性取决于优化程序所选择的查询计划。如果计划中没有创建工作表，则数据更改对结果集可见。缺省值为 semi_sensitive。

insensitive

指定独立于游标进行的数据更改对游标结果集不可见。如果不指定此参数，则缺省值为 semi_sensitive。不能更新非敏感游标。

scroll | no scroll

指定声明的游标是否可滚动。可滚动游标允许您非顺序读取游标结果集，从而可以向前和向后扫描游标。不能更新可滚动游标。

release_locks_on_close

能让您配置每个游标锁释放行为，以便在关闭游标时能释放共享锁，即使事务处于活动状态也是如此。此选项适用于所有类型的游标。

for read only

指定游标结果集不能更新。

for update

指定游标结果集可以更新。

of *column_name_list*

是来自定义为可更新的游标结果集 （用 *select_statement* 指定）的列的列表。 Adaptive Server 还允许包括游标的 *select_statement* 列列表中没有指定的列 （且不包括在结果集中），但这些列是该 *select_statement* 中指定的表的一部分。

示例

示例 1 为 authors_crsr 游标定义结果集，其中包括 authors 表中居住在加利福尼亚以外的所有作者：

```
declare authors_crsr cursor
```

```
for select au_id, au_lname, au_fname
from authors
where state != 'CA'
```

示例 2 定义 titles_crsr 游标的只读结果集，其中包括 titles 表的商业类型书籍：

```
declare titles_crsr cursor
for select title, title_id from titles
where title_id like "BU%"
for read only
```

示例 3 定义 pubs_crsr 游标的可更新结果集，其中包括 publishers 表的所有行。它定义了要更新的每个出版社的地址（city 和 state 列）：

```
declare pubs_crsr cursor
for select pub_name, city, state
from publishers
for update of city, state
```

示例 4 定义 stores_scrollcrsr 的非敏感可滚动结果集，其中包含位于加利福尼亚的书店：

```
declare stores_scrollcrsr insensitive scroll cursor
for select stor_id, stor_name
from stores where state = 'CA'
```

示例 5 定义 stores_scrollcrsr 的非敏感不可滚动结果集，其中包含位于加利福尼亚的书店：

```
declare stores_scrollcrsr insensitive no scroll cursor
for select stor_id, stor_name
from stores where state = 'CA'
```

用法

游标限制

- declare cursor 语句必须出现在该游标的任一 open 语句之前。
- 不能将其它语句和 declare cursor 置于同一 Transact-SQL 批处理中。
- 在客户端的 declare cursor 语句的一个 update 子句中，可以包括多达 1024 列。
- *cursor_name* 必须是有效的 Adaptive Server 标识符，长度不超过 30 个字符。
- declare cursor 语句的 for update 子句中不能包含加密列。
- 不能更新可滚动游标。
- 不能更新非敏感游标。

游标 *select* 语句

- *select_statement* 可使用 Transact-SQL select 语句的完整语法与语义，同时具有下列限制：
  - 必须包含一个 from 子句
  - 不能包含 compute、 for browse 或 into 子句
  - 可以包含 holdlock 关键字
- *select_statement* 可以包含对 Transact-SQL 参数名或 Transact-SQL 局部变量 （除语言之外的所有游标类型）的引用。名称必须引用在过程、触发器或包含 declare cursor 语句的语句批处理中定义的 Transact-SQL 参数和局部变量。

  除非游标打开，否则在 declare cursor 语句中引用的参数和局部变量不必包含有效值。
- *select_statement* 可以包含对触发器中使用的 inserted 和 deleted 临时表的引用。

游标范围

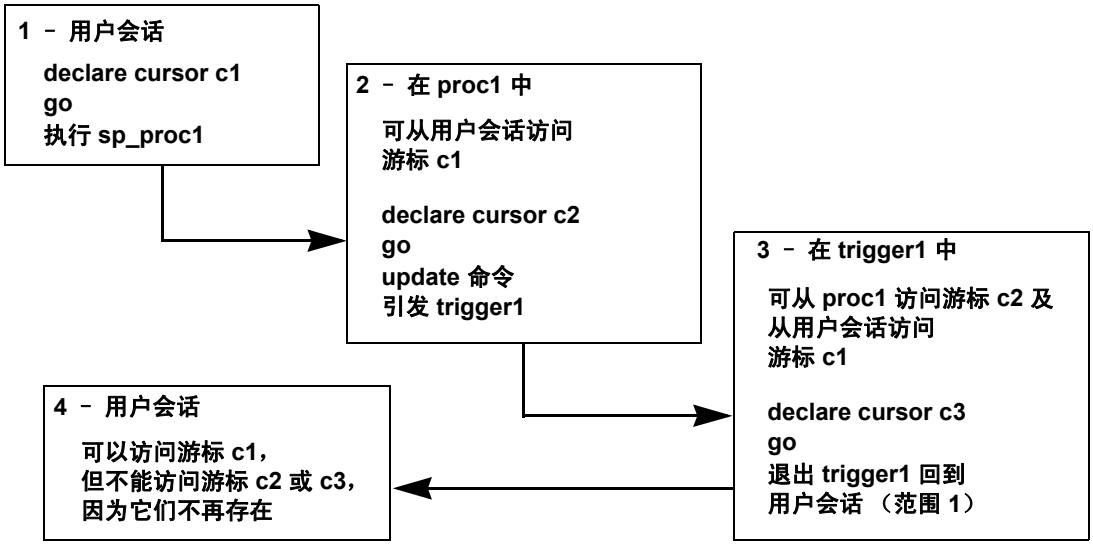
- 游标的存在取决于其范围。范围是指使用游标的环境，即在用户会话内、存储过程内还是触发器内。

  在用户会话内，游标只在用户结束会话之前存在。对其他用户启动的任何其它会话而言，游标并不存在。用户注销后， Adaptive Server 将释放该会话中创建的游标。

  如果 declare cursor 语句是存储过程或触发器的一部分，则在其中创建的游标将应用于存储过程或触发器范围以及启动该存储过程或触发器的范围。所有嵌套的存储过程或触发器将不能访问触发器内的 inserted 或 deleted 表上声明的游标。但是，在触发器范围内可以访问触发器中在 inserted 或 deleted 表上声明的游标。存储过程或触发器执行完成后， Adaptive Server 将释放在其中创建的游标。

  图 1-3 演示了游标在范围间的运行方式。

图 1-3：游标在范围内的运行方式



- 给定范围内的游标名必须唯一。 Adaptive Server 只在运行时才检测特定范围内的名称冲突。如果只执行一个游标，存储过程或触发器就可以定义两个同名的游标。例如，以下存储过程有效，因为在其范围内只定义了一个 names_crsr 游标：

```
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
```

结果集

- 游标结果集的行可能无法反映实际的基表行的值。例如，用 order by 子句声明的游标通常需要创建内部表才能对游标结果集的行进行排序。 Adaptive Server 不锁定与内部表中的行相对应的基表中的行，这样一来其它客户端就可更新这些基表行。在这种情况下，从游标结果集返回到客户端的行将不与基表的行同步。

- 该游标执行 fetch 操作返回行的同时将生成游标结果集。这意味着处理游标 select 查询的方式与处理常规 select 查询的方式相同。此进程（称为**游标扫描**）可缩短周转时间，并使应用程序不需要的行无需再被读取。

  游标扫描有一个限制就是只能使用表的唯一索引。但是，如果由游标结果集引用的基表都没有由与游标位于同一锁定空间的另一进程更新，则此限制不是必要的。 Adaptive Server 允许在无唯一索引的表上声明游标，但任何试图更新同一锁定空间上的表的操作都会关闭表上的所有游标。

可更新游标

- 使用 declare cursor 定义游标后， Adaptive Server 将确定游标是可更新的还是只读的。如果：
  - 游标可更新 – 可通过该游标更新或删除行；即，使用 *cursor_name* 进行位置 update 或 delete。
  - 游标是只读的 – 不能使用 *cursor_name* 进行定位 update 或 delete。
- 使用 for update 或 for read only 子句可显式定义游标为可更新的或只读的。如果游标的 *select_statement* 中包含以下结构，则不能定义它为可更新游标：
  - distinct 选项
  - group by 子句
  - 集合函数
  - 子查询
  - union 运算符
  - at isolation read uncommitted 子句

  如果未指定 for update 或 read only 子句，则 Adaptive Server 将进行检查以了解游标是否可更新。

  如果将包括 order by 子句的语言或服务器类型的游标声明为其 *select_statement* 的一部分，Adaptive Server 也会将游标定义为只读。Adaptive Server 句柄以不同的方式更新客户端类型和执行类型的游标，从而消除了此限制。

在游标关闭时释放锁

如果游标扫描在隔离级别 1 发生，则 release_locks_on_close 不起作用。

如果在游标关闭前事务已提交或回退，则在隔离级别 2 和 3 的缺省行为是， Adaptive Server 释放游标在此之前获取的共享锁（最后一个获取的行上的锁除外）。如果您使用 release_on_locks_close，则游标获取的共享锁一直存在到游标关闭为止。

可使用 sp_cursorinfo 确定是否通过 release_on_locks_close 参数声明了游标：

```
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.
游标 ID 为 917505。
The cursor has been successfully opened 0 times.
The cursor will remain open when a transaction is
committed or rolled back.
```

可更新游标和所有页锁定

- 如果不用 for update 子句指定 *column_name_list*，则查询中的所有指定列都可更新。Adaptive Serve 在扫描基表时尝试对可更新游标使用唯一索引。对于游标， Adaptive Server 会将包含 IDENTITY 列的索引视为唯一索引，即使该游标并未如此声明。
- 如果没有指定 for update 子句， Adaptive Server 将选择任何唯一索引，尽管在指定表列没有唯一索引的情况下，它也可以使用其它索引或表扫描。但如果指定 for update 子句， Adaptive Server 必须使用为一个或多个列定义的唯一索引扫描基表。如果不存在，则返回一个错误。
- 多数情况下，只在 for update 子句的 *column_name_list* 中包括要更新的列。如果表只有一个唯一索引，则无需在 for update *column_name_list* 中包括该表的列；游标扫描期间， Adaptive Server 会找到该列。如果表有多个唯一索引，则 for update *column_name_list* 中不能包含其中任何一个。

  这可使 Adaptive Server 对其游标扫描使用该唯一索引，这样有助于防止一种称为 **Halloween 问题**的更新异常。防止出现 Halloween 问题的另一种方法是用 unique auto_identity index 数据库选项创建表。请参见 《系统管理指南》。

  当客户端更新游标结果集行的一列，而该列又定义了行从基表返回的顺序时，就会出现 Halloween 问题。例如，如果 Adaptive Server 使用索引访问一个基表，且该索引键由客户端更新，则更新的索引行可以在索引内移动，并且可供游标再次读取。这就是只逻辑创建游标结果集的可更新游标的结果。此游标结果集实际上是派生此游标的基表。

如果指定 read only 选项，则不能通过使用游标名执行 update 或 delete 来更新游标结果集。

使用可滚动游标

- 如果在执行 declare cursor 时指定了 insensitive 或 semi_sensitive，则缺省敏感性为隐式的，游标敏感性将取决于优化程序所选择的查询计划。如果查询计划有任何创建的工作表，则游标变为非敏感。
- 如果将游标敏感性指定为半敏感 (semisensitive)，则敏感性也取决于查询计划。
- 如果指定 insensitive，则游标为 *read_only*。不能在游标声明中使用 for update 子句。
- 如果不指定游标的可滚动性，则表示值为 no scroll。
- 所有可滚动游标都为只读游标。在游标声明中不能使用 for update 子句。

标准

符合 ANSI SQL 的级别符合初级标准。

权限

使用 declare cursor 无需任何权限。

另请参见

命令 open.

delete

说明

删除表中的行。

语法

```
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
```

参数

from (after delete)

是用于与其它版本的 SQL 兼容的可选关键字。

view_name | *table_name*

是从其中删除行的视图或表的名称。如果该视图或表位于另一数据库中，请指定该数据库名称；如果在数据库中有多个具有该名称的视图或表，请指定所有者的名称。*owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

where

是标准 where 子句。有关详细信息，请参见where 子句。

from （*table_name* 或 *view_name* 后）

允许在指定删除哪些行时指定要用于 where 子句的多个表或视图。此 from 子句允许您根据存储在其它表中的数据删除一个表中的行，从而给予您嵌入式 select 语句的大部分能力。

top *unsigned_integer*

用于将行数限制为 *unsigned_integer* 指定的行数。

readpast

指定 delete 命令跳过持有不兼容锁的所有页或行，而不必等待锁或超时。对于数据页锁定表， *readpast* 会跳过持有不兼容锁的页上的所有行；对于数据行锁定表，它会跳过持有不兼容锁的所有行。

index *index_name*

指定访问 *table_name* 要使用的索引。当从视图中进行删除时不能使用此选项。

prefetch *size*

为绑定到配置了大 I/O 的高速缓存的表指定 I/O 大小 （单位为千字节）。从视图中进行删除时不能使用 prefetch。 sp_helpcache 显示对象绑定到的高速缓存或缺省缓存的有效大小。

在使用 prefetch 并指定预取大小 (*size*) 时，最小值是 2K 或逻辑页大小与 2 的任意次方的乘积 （最大为 16K）。 prefetch 大小选项如下 （以千字节为单位）：

逻辑页大小	预取大小选项
2	2, 4, 8 16
4	4, 8, 16, 32
8	8, 16, 32, 64
16	16, 32, 64, 128

在查询中指定的 prefetch 大小仅仅是一个建议。要使用指定的大小，应将数据高速缓存按该大小进行配置。如果未将数据高速缓存配置为特定的大小，则使用缺省的 prefetch 大小。

若要配置数据高速缓存大小，请使用 sp_cacheconfigure。

注释 如果启用 CIS，则不能对远程服务器使用 prefetch。

lru | mru

指定对表使用的缓冲区替换策略。使用 lru 强制优化程序将表读入 MRU/LRU （最近使用最多的/最近使用最少的）链上的高速缓存。使用 mru 可放弃高速缓存中的缓冲区，并将其替换为该表的下一个缓冲区。当从视图中进行删除时不能使用此选项。

plan "*abstract plan*"

指定用来优化查询的抽象计划。它可以是用抽象计划语言指定的完整或部分计划。请参见 《性能和调优指南：优化程序和抽象计划》中的 “创建和使用抽象计划”。

where current of *cursor_name*

导致 Adaptive Server 删除 *cursor_name* 的当前游标位置指示的表或视图中的行。

示例

示例 1 删除 authors 表中的所有行：

```
delete authors
```

示例 2 删除 authors 表中的一行或多行：

```
delete from authors where au_lname = "McBadden"
```

示例 3 删除 titles 表中作者为 Bennet 的书的行。

```
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 数据库包含一个阻止删除记录在 sales 表中的标题的触发器 (deltitle)，若要使此示例能够工作，请删除此触发器。

示例 4 删除当前由游标 title_crsr 指示的 titles 表中的一行：

```
delete titles where current of title_crsr
```

示例 5 确定 IDENTITY 列的值为 4 的行并将其从 authors 表中删除。注意用 syb_identity 关键字代替了 IDENTITY 列的实际名称：

```
delete authors where syb_identity = 4
```

示例 6 删除 authors 中的行，跳过所有锁定的行：

```
delete from authors from authors readpast
where state = "CA"
```

示例 7 删除 stores 中的行，跳过所有锁定的行。如果 authors 中有锁定的行，则查询会阻塞在这些行上，等待锁被释放：

```
delete stores from stores readpast, authors
where stores.city = authors.city
```

用法

- 可在一条 delete 语句中引用多达 15 个表。
- 在 Adaptive Server 12.5.2 版之前的版本中，有时不用工作表就能解析对视图执行的带有 union all 子句的 update 和 delete 查询，这偶尔会导致不正确的结果。在 Adaptive Server 12.5.2 及更高版本中，始终使用 tempdb 中的工作表解析对带有 union all 子句的视图执行 update 和 delete 的查询。

限制

- 不能在多表视图 （from 子句指定多个表的视图）上使用 delete，即使可以在同一视图上使用 update 或 insert。通过多表视图删除一行会更改多个表，而这是不允许的。只影响视图的一个基表的 insert 和 update 语句是允许的。
- Adaptive Server 将一个 delete 中同一表的两种不同名称作为两个表对待。例如，下面在 pubs2 中发出的 delete 命令将 discounts 指定为两个表 （discounts 和 pubs2..discounts)：

  ```
  delete discounts
  from pubs2..discounts, pubs2..stores
  where pubs2..discounts.stor_id =
      pubs2..stores.stor_id
  ```

  在这个例子中，连接中不包括 discounts，因此 where 条件对每一行都为真；Adaptive Server 删除 discounts 中的所有行 （这并非期望的结果）。若要避免此问题，请在整个语句中使用相同的表的名称。
- 如果要删除通过参照约束从其它表引用的表中的一行，允许删除前 Adaptive Server 会检查所有的引用表。如果试图删除的行包含一个被引用表用作外键的主键，则不允许删除此行。

删除表中的所有行

- 如果不使用 where 子句，在 delete [from] 后命名的表中的所有行都将被删除。表中虽然没有数据，但会一直存在，直到您发出 drop table 命令。
- truncate table 和不指定行的 delete 就功能来说是相同的，但 truncate table 要更快。delete 一次删除一行并记录这些事务。truncate table 删除整个数据页，且并不记录这些行。

  delete 和 truncate table 都回收由数据及其相关索引所占用的空间。
- 不能在分区表上使用 truncate table 命令。若要删除分区表的所有行，请使用不带 where 子句的 delete 命令，或在发出 truncate table 之前取消表的分区。

delete 和事务

在链式事务模式中，如果没有当前处于活动状态的事务，则每个 delete 语句会隐式地开始一个新的事务。请使用 commit 完成删除，或使用 rollback 撤消更改。例如：

```
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
```

此批处理会开始一个事务 （使用链式事务模式）并删除 sales 表中日期早于 2006 年 1 月 1 日的行。如果它删除了与一个店铺相关的所有 sales 条目，则回退所有对 sales 的更改并结束事务。否则它提交删除并结束事务。有关链式模式的详细信息，请参见 《Transact-SQL 用户指南》。

delete 触发器

可以定义一个触发器，从而在对指定表发出 delete 命令时执行指定的操作。

使用 *delete where current of*

- 用游标使用子句 where current of。用子句 where current of 删除行之前，先用 declare cursor 定义游标并用 open 语句将其打开。用一个或多个 fetch 语句将游标定位到要删除的行上。游标名不能是 Transact-SQL 参数或局部变量。游标必须是可更新游标，否则 Adaptive Server 会返回错误。对游标结果集进行的任何删除操作也会影响派生该游标行的基表行。使用游标时，每次只能删除一行。
- 如果游标的 select 语句中包含连接子句，那么即使游标是可更新的，也不能删除游标结果集中的行。用 delete...where current of 指定的 *table_name* 或 *view_name* 必须是在定义该游标的 select 语句的第一个 from 子句中指定的表或视图。
- 删除游标结果集中的一行后，游标定位在游标结果集中下一行的前面。要访问下一行，必须发出 fetch 命令。如果删除的行是游标结果集中的最后一行，则游标将被放置到结果集的最后一行之后。下面描述了受 delete 影响的打开的游标的位置和行为：
  - 如果客户端删除一行 （使用另一个游标或常规的 delete），而此行代表此客户端所拥有的另一打开的游标的当前游标位置，则隐式地设置每个受影响的游标的位置为下一可用行之前。但是，客户端不能删除代表另一客户端游标的当前游标位置的行。
  - 如果一个客户端删除一行，而此行代表由连接操作定义并由同一客户端所拥有的另一游标的当前游标位置，则 Adaptive Server 接受此 delete 语句。但它会隐式地关闭由连接定义的游标。

使用 *readpast*

- readpast 使得仅数据锁定表上的 delete 命令能够执行，而不会被其它任务持有的不兼容锁阻塞。
  - 在数据行锁定表中，readpast 会跳过共享、更新或排它锁由另一任务所持有的所有行。

- 在数据页锁定表中，readpast 会跳过共享、更新或排它锁由另一任务所持有的所有页。
- 有排它锁时指定 readpast 阻塞的命令。
- 如果为所有页锁定表指定了 readpast 选项，该选项将被忽略。一旦发现不兼容锁，命令就会阻塞。
- 如果会话范围的隔离级别为 3，将自动忽略 readpast 选项。此命令在级别 3 执行。此命令在任何带有不兼容锁的行或页上阻塞。
- 如果会话的事务隔离级别是 0，使用 readpast 的 delete 命令将不会发出警告消息。对于数据页锁定表，带有 readpast 的 delete 将修改所有页上没有被不兼容锁锁定的所有行。对于数据行锁定表，它影响所有没有被不兼容锁锁定的行。
- 如果 delete 命令应用于带有两个或多个文本列，并且每个文本列都有一个不兼容的锁的行， readpast 锁定会跳过此行。

使用 *index*、 *prefetch* 或 *lru* | *mru*

index、prefetch 和 lru | mru 选项可覆盖 Adaptive Server 优化程序作出的选择。请慎用这些选项，并始终用 set statistics io on 检查性能影响。请参见《性能和调优指南：监控和分析》中的 “使用 set statistics 命令”。

标准

符合 ANSI SQL 的级别符合初级标准。在 from 子句中使用多个表和用数据库名限定表名属于 Transact-SQL 扩展。

readpast 是一个 Transact-SQL 扩展。

权限

如果 set ansi_permissions 为 on，则除具有 delete 语句所需的常规权限外，还必须对 where 子句中出现的所有列具有 select 权限。缺省情况下，ansi_permissions 为 off。

下文说明了基于您的细化权限设置的 delete 的权限检查。

细化权限已启用	在启用细化权限的情况下，您必须是表或视图所有者，或者是拥有 delete 权限的用户或具有 delete any table 权限的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须具有 delete 权限，或者是表的所有者或是具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
18	delete	从表中删除数据的 delete 命令	• *角色* - 当前活动角色 • *关键字或选项* - delete • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名
19	delete	从视图中删除数据的 delete 命令	• *角色* - 当前活动角色 • *关键字或选项* - delete • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见　**命令** create trigger, drop table, drop trigger, truncate table, where 子句.

delete statistics

说明

从 sysstatistics 系统表中删除统计数据。

语法

delete [shared] statistics *table_name*
[partition *data_partition_name*]
[(*column_name*[, *column_name*] ...)]

参数

shared
删除 master 数据库中的 sysstatistics 的模拟统计信息。

table_name
删除表中所有列的统计信息。

data_partition_name
删除该数据分区的所有统计信息。全局统计信息不会被删除。

column_name
删除指定列的统计信息。

示例

示例 1 删除 titles 表中所有列的密度、选择性和直方图：

```
delete statistics titles
```

示例 2 删除 titles 表中 pub_id 列的密度、选择性和直方图：

```
delete statistics titles (pub_id)
```

示例 3 删除 titles 表的 smallsales 分区的密度、选择性和直方图：

```
delete statistics titles partition smallsales
```

示例 4 删除 pub_id 和 pubdate 的密度、选择性和直方图，而不影响单列 pub_id 或单列 pubdate 的统计信息：

```
delete statistics titles (pub_id, pubdate)
```

示例 5 删除 pub_id 列以及数据分区 smallsales 的密度、选择性和直方图：

```
delete statistics titles partition smallsales (pub_id)
```

用法

- delete statistics 不影响 systabstats 表中的统计信息。
- delete statistics on a data partition 不会删除全局统计信息。

- 发出 drop table 命令时会删除 sysstatistics 中相应的行。使用 drop index 时不会删除 sysstatistics 中的行。这就允许查询优化程序继续使用索引统计信息而不会产生维护表上索引的开销。

  警告！密度、选择性和直方图对好的查询优化来说是必要的。delete statistics 是作为工具提供的，用来删除优化程序不用的统计信息。如果无意中删除了查询优化所需的统计信息，请在表、索引或列上运行 update statistics。

- 使用 optdiag 实用程序命令装载模拟统计信息会在 master..sysstatistics 表中添加少量的行。如果不再使用模拟统计信息，则可以使用 delete shared statistic 命令删除 master..sysstatistics 中的信息。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 delete statistics 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或是具有 delete statistics 权限的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是表所有者、具有 sso_role 的用户或者是拥有 delete statistics 权限的用户。 delete statistics 权限可由表所有者或系统管理员授予或转给任何人。

另请参见

命令 create index, grant, revoke, update.

实用程序 optdiag.

disk init

说明

使物理设备或文件可供 Adaptive Server 使用。

语法

```
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"]
```

参数

device_name

是数据库设备名或文件名。此名称必须遵循标识符规则并用单引号或双引号引起来。此名称用于 create database 和 alter database 命令。

physical_name

是数据库设备或高速缓存的全名。此名称必须用单引号或双引号引起来。当物理设备路径为相对路径时， disk init 返回警告。

cache_name

在其中创建磁盘的高速缓存的名称。

inmemory

指示您正在创建内存设备。

skip_alloc

disk init 命令的布尔参数。在非 Windows 文件系统和 Windows 原始系统上创建的设备支持该参数。当 skip_alloc 设置为 true 时，用户可以避免用零初始化页。 skip_alloc 的缺省值为 false。

vdevno

是虚拟设备号，它在与 Adaptive Server 关联的数据库设备中必须是唯一的。设备号 0 是为主设备保留的。否则，有效设备号必须在 1 和 2,147,483,647 之间。

若要确定虚拟设备号，请查看 sp_helpdevice 报告的 device_number 列，并使用下一未用整数。

size

为分配给新设备的空间量。以下是示例单位指示符，交替使用了大写、小写以及单引号和双引号：“k”或“K”（千字节）、“m”或“M”（兆字节）、“g”或“G”（千兆字节）以及“t”或“T”（兆兆字节）。 Sybase 建议始终包括单位指示符。如果不包括单位指示符，则引号是可选的。不过，如果包括单位指示符，则必须使用引号。可接受的值有：

- `5120` = 10MB
- `"5120"` = 10MB
- `"10M"` = 10MB

vstart

是供 Adaptive Server 开始使用数据库设备的起始虚拟地址，即偏移量。以下是示例单位指示符，交替使用了大写、小写以及单引号和双引号：“k”或“K”（千字节）、“m”或“M”（兆字节）、“g”或“G”（千兆字节）以及“t”或“T”（兆兆字节）。 Sybase 建议始终包括单位指示符。如果不包括单位指示符，则引号是可选的。不过，如果包括单位指示符，则必须使用引号。

偏移的大小取决于输入 vstart 值的方式。

- 如果未指定单位大小，则 vstart 将使用 2K 页作为其起始地址。例如，如果指定 `vstart = 13`，则 Adaptive Server 使用 13 * 2K 页作为起始地址的偏移量。
- 如果指定了单位值，vstart 就将该单位值用作起始地址。例如，如果指定 `vstart = "13M"`，则 Adaptive Server 将起始地址的偏移量设置为 13MB。

vstart 的缺省值 （通常为首选值）为 0。如果指定设备中的块数达不到可用的 vstart + size 的块数之和，则 disk init 命令将失败。如果在 AIX 操作系统上运行逻辑卷管理器，则 vstart 应为 2。除非有 Sybase 技术支持部门的指导，否则不要指定 vstart。

cntrltype

指定磁盘控制器。其缺省值为 0。除非有 Sybase 技术支持部门的指导，否则不要重新设置 cntrltype。

dsync

指定对数据库设备的写入操作是刷新到存储介质中，还是仅在使用操作系统文件时放入缓冲区。本选项只在初始化操作系统文件时才有意义；初始化原始分区上的设备时不起作用。缺省情况下，所有操作系统文件都通过将 dsync 设置为 true 进行初始化。

directio

允许您将 Adaptive Server 配置为跳过操作系统缓冲区高速缓存，直接将数据传输到磁盘。 directio 为静态参数，需要重新启动 Adaptive Server 才能生效。

缺省情况下，directio 对于非集群 Adaptive Server 设置为 false，对于集群 Adaptive Server 设置为 true。

对于原始设备，将忽略 directio 参数。

instance = "*instance_name*"

（仅限于集群）将设备指定为私有，将其拥有实例设置为 *instance_name*。

示例

示例 1 不用零初始化页：

```
disk init name="d2",
physname="/usr/sybase/devices/d3.dat",
skip_alloc="true",
size="10G"
```

如果 skip_alloc 设置为 true， Adaptive Server 在磁盘初始化期间将不分配空间。

示例 2 在 UNIX 系统上初始化 10MB 的磁盘：

```
disk init
name = "user_disk",
physname = "/dev/rxy1a",
vdevno = 2, size = 5120
```

示例 3 在 UNIX 操作系统文件上初始化 10MB 的磁盘。Adaptive Server 使用 dsync 设置打开设备文件，并确保对文件的写入操作直接在存储介质上进行：

```
disk init
name = "user_file",
physname = "/usr/u/sybase/data/userfile1.dat",
vdevno = 2, size = 5120, dsync = true
```

示例 4 创建名为 “user_disk”的设备，并且该设备使用 directio 直接将数据写入磁盘：

```
disk init
name = "user_disk",
physname = "/usr/u/sybase/data/userfile1.dat",
size = 5120,
directio= true
```

示例 5 创建名为 inmemory_dev 的设备：

```
disk init name = inmemory_dev,
physname = 'imdb_cache',
size = '3G',
type = 'inmemory'
```

用法

- Sybase 建议您只在 HP-UX、Windows 和 Linux 平台上将块设备用作数据库设备。如果尝试在 Sybase 建议不要使用块设备的平台上创建块设备， disk init 和 disk reinit 将显示警告消息。
- 使用 skip_alloc 可加速非 NT 文件系统和 NT 原始系统的崩溃恢复。而且，将 skip_alloc 与 directio 功能结合使用还可以更快速地创建设备，并可改进更新的持久性。无论空间可用性如何， skip_alloc 总是输出一条警告消息，指出要确保 Adaptive Server 具有所需的空间以供以后使用。
- 主设备是由安装程序初始化的；您不需要用 disk init 初始化此设备。
- 使用 disk init 创建的设备有权限限制。
- 若要成功地完成磁盘初始化，“sybase”用户必须对将要被初始化的设备拥有相应的操作系统权限。
- 可将 size 指定为浮点型，但会被下舍至最接近的 2K 的倍数。
- 如果不对 size 使用单位指示符，则 disk init 将使用 2K 的虚拟页大小。
- 您可用 disk init 进行初始化的磁盘区段的最小大小是以下两者中的较大者：
  - 一兆字节
  - 服务器的逻辑页大小的一个分配单元
- directio 和 dsync 互斥。如果某台设备的 dsync 已设置为 true，则不能将此设备的 directio 设置为 true。若要为设备启用 directio，您必须先将 dsync 重置为 false。
- directio 并非在所有平台上都可用。如果在不支持该参数的平台上发出带有 directio 参数的 disk init，Adaptive Server 将发出消息 `No such parameter:'directio'`。
- 对每个新的数据库设备使用 disk init。每次发出 disk init 时，都会向 master..sysdevices 添加一行。新的数据库设备不会自动成为缺省数据库存储池的一部分。使用 sp_diskdefault 为数据库设备指派缺省状态。
- 每次使用 disk init 之后请用 dump database 或 dump transaction 命令备份 master 数据库。这使得在 master 损坏的情况下能够更容易和安全地进行恢复。如果用 disk init 添加了设备且备份 master 失败，则您可以用 disk reinit 恢复更改，然后停止并重新启动 Adaptive Server。

- 使用 create database 或 alter database 命令的 on 子句为数据库设备分配用户数据库。
- 将数据库的事务日志 （系统表 syslogs）放置在与存储其它数据库的设备不同的设备上的首选方法是对 create database 使用 log on 扩展。或者，您可以在创建数据库时指定至少两个设备，然后执行 sp_logdevice。您还可以用 alter database 将数据库扩展到另外的设备上，然后运行 sp_logdevice。 log on 扩展会立即将整个日志移动到单独的设备上。 sp_logdevice 方法在原来的设备上保留一部分系统日志，直到事务活动使得迁移完成。
- 若要获取系统上的所有 Adaptive Server 设备 （数据库及转储设备）的报告，请执行 sp_helpdevice。
- 使用 sp_dropdevice 删除数据库设备。必须先删除设备上的所有现有数据库。

使用 *dsync*

注释 对于存储关键数据的任何设备，不要将 dsync 设置为 false。唯一的例外是 tempdb，它可以安全地存储在 dsync 设置为 false 的设备上。

- （仅限 UNIX）在原始设备上，不能进行以下操作：
  - 将 directio 或 dsync 设置为 true
  - 通过 sp_deviceattr 存储过程将 directio 或 dsync 设置为 true。

注释 对于 HPUX，只有 dsync 选项适用。

这样做将会返回类似以下的消息：

```
You cannot set option dsync for raw device 'dev/raw/raw235'
```

或

```
You cannot set attribute dsync for raw device 'myrawdisk1'
```

- 当 dsync 打开时，可确保在物理存储介质上进行对数据库设备的写入操作，并且 Adaptive Server 可以在出现系统故障时恢复设备上的数据。
- 当 dsync 关闭时，UNIX 文件系统可能会对数据库设备的写入进行缓冲。即使尚未修改物理介质， UNIX 文件系统仍可能将更新标记为已经完成。出现系统故障时，不能保证物理介质上已进行数据更新，并且 Adaptive Server 可能无法恢复数据库。
- 对于主设备文件， dsync 始终处于打开状态。

- 仅在出现系统故障后无需恢复设备上的数据库时，关闭 dsync 值。例如，对于只存储 tempdb 数据库的设备，可以考虑将 dsync 关闭。
- Adaptive Server 会忽略存储在原始分区上的设备的 dsync 设置 确保在物理存储介质上进行对这些设备的写入操作，无论怎样设置 dsync。
- disk reinit 确保在 master 数据库已损坏或者上次转储 master 之后又添加了设备时， master..sysdevices 仍然是正确的。

为内存数据库和宽松持久性数据库创建设备

- 内存设备的逻辑名不能与创建设备所在的高速缓存的名称相同。
- 内存设备将保留用于您指派的第一个内存数据库。
- 必须使用物理设备名作为内存设备的逻辑名。
- 内存设备的逻辑名必须在所有设备中唯一。
- 您不能：
  - 在创建内存设备时使用以下参数：vstart、 cntrltype、 dsync、 directio 和 skip_alloc。
  - 创建的内存设备比所在的高速缓存大。
  - 从常规命名高速缓存 （包括缺省数据高速缓存）中创建内存设备。
  - 创建内存设备后增大其大小。但是，可以使用 sp_cacheconfig 增大内存高速缓存的大小，以及使用 disk init 创建新内存设备。
  - 对内存设备使用以下命令：disk resize、disk mirror、disk remirror、disk unmirror、 disk refit 和 disk reinit。
  - 将 sp_deviceattr 和 sp_diskdefault 系统过程用于磁盘设备。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 disk init 的权限检查因您的细化权限设置而异。必须使用 master 数据库才能使用 disk init。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage disk 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。 disk init 权限不能移交。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
20	disk	disk init	• *角色* – 当前活动角色 • *关键字或选项* – disk init • *先前值* – NULL • *当前值* – NULL • *其它信息* – 磁盘名称 • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 alter database, create database, disk refit, disk reinit, dump database, dump transaction, load database, load transaction.

系统过程 sp_diskdefault, sp_dropdevice, sp_helpdevice, sp_logdevice.

disk mirror

说明

创建在主设备发生故障时立即接替运行的软件镜像。

语法

```
disk mirror
        name = "device_name",
        mirror = "physicalname"
        [, writes = {serial | noserial}]
        [clear = {TRUE | FALSE}]
```

参数

name

为要镜像的数据库设备的名称。它存储在 sysdevices 表的 name 列中。名称必须用单引号或双引号引起来。

镜像

是将成为辅助设备的数据库镜像设备的全路径名。它必须用单引号或双引号引起来。如果辅助设备是一个文件，则 *physicalname* 应该是清楚地标识此文件的路径名称，该名称由 Adaptive Server 创建。*physicalname* 的值不能是现有文件。

writes

允许您选择是否对设备强制执行串行写入。缺省情况下 (serial)，应保证开始对辅助设备写入前完成对主数据库设备的写入。如果主设备和辅助设备位于不同的物理设备上，则串行写入可确保在出现电源故障时至少有一个磁盘不受影响。 serial 写入通常比 noserial 写入要慢。

clear

用零初始化镜像设备，以确保基础文件系统为镜像设备保留了空间。缺省值 FALSE 不清除镜像，并且向设备执行的写入操作可能会由于文件系统缺少空间而失败。如果指定 TRUE，则清除镜像，并强制文件系统为设备保留空间。

示例

示例 1 tranlog 是原始设备的逻辑设备名。 tranlog 设备通过 disk init 初始化并且作为事务日志设备使用 （如 create database...log on *tranlog*）。以下命令可镜像事务日志设备：

```
disk mirror
  name = "tranlog",
  mirror = "/dev/rxy1e"
```

示例 2 在 *mirror.dat* 文件上为数据库设备 user_disk 创建软件镜像：

```
disk mirror
    name = "user_disk",
    mirror = "/server/data/mirror.dat"
```

用法

- 磁盘镜像可为用户数据库设备、主数据库设备或用于用户数据库事务日志的数据库设备创建软件镜像。如果数据库设备发生故障，其镜像会立即接替它运行。

磁盘镜像不会干涉数据库中正在进行的活动。不关闭 Adaptive Server 即可镜像数据库设备或取消数据库设备的镜像。

- 每次使用 disk mirror 之后请使用 dump database 备份 master 数据库。这使得在 master 损坏的情况下能够更容易和安全地进行恢复。
- 如果在已镜像设备中读写时失败， Adaptive Server 会对损坏的设备取消镜像并显示错误消息。 Adaptive Server 将继续运行，但不进行镜像。系统管理员必须使用 disk remirror 命令重新启动镜像。
- 此命令中的 clear 选项在用于 NT 平台时不起作用。
- 可以镜像主设备、存储数据的设备及存储事务日志的设备。但不能镜像转储设备。
- 镜像设备；不镜像数据库。
- 设备及其镜像组成一个逻辑设备。 Adaptive Server 将镜像设备的物理名存储在 sysdevices 表的 mirrorname 列。它不需要 sysdevices 中单独的条目，并且不应使用 disk init 初始化它。
- 若要保持使用异步 I/O，应始终将能够执行异步 I/O 的设备镜像到能够执行异步 I/O 的其它设备。在多数情况下，这意味着将原始设备镜像到原始设备，将操作系统文件镜像到操作系统文件。

  如果操作系统不能在文件上执行异步 I/O，则将原始设备镜像到某个常规文件时会产生错误消息。将常规文件镜像到原始设备可行，但不使用异步 I/O。
- 在支持异步 I/O 的系统中，writes 选项允许您指定向第一个设备的写入操作是否必须在开始向第二个设备写入前完成 (serial)，或者指定两个 I/O 请求是否立即排队，且分别位于镜像的两侧 (noserial)。在任何一种情况下，如果写入不能完成，则 I/O 错误将造成设备损坏而无法镜像。
- 镜像所有缺省数据库设备，以便在 create database 或 alter database 命令影响缺省列表中的数据库设备时得到保护。
- 若要获得更好的保护，请镜像用于事务日志的数据库设备。
- 始终将用户数据库事务日志放在单独的数据库设备上。要将数据库事务日志 （即系统表 syslogs）放到存储该数据库其余部分的设备之外的设备上，可在创建数据库时指定数据库设备和日志设备。或者，还可以用 alter database 将数据库扩展到另外的设备，然后运行 sp_logdevice。

- 如果要为 master 数据库镜像数据库设备，可以在用 dataserver 实用程序重新启动 Adaptive Server 时使用 -r 选项和 UNIX 的镜像名称。将它添加到该服务器的 *RUN_servername* 文件，以便通知 startserver 实用程序这件事。例如，若要启动名为 master.dat 的主设备及其镜像 mirror.dat，请输入：

  ```
  dataserver -dmaster.dat -rmirror.dat
  ```

  请参见 《实用程序指南》中的 dataserver 和 startserver。

- 如果镜像一个具有未分配空间 （供附加 create database 和 alter database 语句用来分配设备的某些部分的空间）的数据库设备，disk mirror 在进行分配的时候即对这些分配进行镜像，而不是在发出 disk mirror 命令时进行镜像。

- 要获取系统中所有 Adaptive Server 设备 （用户数据库设备及其镜像以及转储设备）的报告，请执行 sp_helpdevice。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 disk mirror 的权限检查因您的细化权限设置而异。执行 disk mirror 时，必须使用 master数据库。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage disk 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。 disk mirror 权限不能移交。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
23	disk	disk mirror	• *角色* – 当前活动角色 • *关键字或选项* – disk mirror • *先前值* – NULL • *当前值* – NULL • *其它信息* – *磁盘名称* • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 alter database, create database, disk init, disk refit, disk reinit, disk remirror, disk unmirror, dump database, dump transaction, load database, load transaction.

系统过程 sp_diskdefault, sp_helpdevice, sp_logdevice.

实用程序 dataserver、 startserver.

disk refit

说明

通过 sysdevices 中的信息重建 master 数据库的 sysusages 和 sysdatabases 系统表。

语法

disk refit

用法

- 在 disk refit 重建系统表之后， Adaptive Server 会自动关闭。
- 在 disk reinit 之后将 disk refit 作为该过程的一部分使用，来恢复 master 数据库。

注释 您必须首先用跟踪标志 3608 启动 Adaptive Server，然后运行 disk refit。但是，在用任何跟踪标志启动 Adaptive Server 之前，一定要阅读《故障排除和错误消息指南》中的信息。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 disk refit 的权限检查因您的细化权限设置而异。执行 disk refit 时，必须使用 master 数据库。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage disk 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。 disk refit 权限不能移交。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
21	disk	disk refit	• *角色* - 当前活动角色 • *关键字或选项* - disk refit • *先前值* - NULL • *当前值* - NULL • *其它信息* - 磁盘名称 • *代理信息* - set proxy 有效时的初始登录名

另请参见

文档 《系统管理指南》。 .

命令 disk init, disk reinit.

系统过程 sp_addumpdevice, sp_helpdevice .

disk reinit

说明

重建 master 数据库的 sysdevices 系统表。将 disk reinit 作为过程的一部分使用，以恢复 master 数据库。

语法

```
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"]
```

参数

name

数据库设备的名称，必须遵循标识符规则并用单引号或双引号引起来。此名称用于 create database 和 alter database 命令。

physname

是数据库设备名。物理名称必须用单引号或双引号引起来。

vdevno

是虚拟设备号，它在与 Adaptive Server 关联的数据库设备中必须是唯一的。设备号 0 是为主设备保留的。否则，有效设备号介于 1 和 2,147,483,647 之间。

若要确定虚拟设备号，请查看 sp_helpdevice 报告的 device_number 列，并使用下一未用整数。

size

为正在重新初始化的设备的当前大小。以下是示例单位指示符，交替使用了大写、小写以及单引号和双引号：“k”或“K”（千字节）、“m”或“M”（兆字节）、“g”或“G”（千兆字节）以及“t”或“T”（兆兆字节）。Sybase 建议始终包括单位指示符。如果不包括单位指示符，则引号是可选的。不过，如果包括单位指示符，则必须使用引号。

vstart

是供 Adaptive Server 开始使用数据库设备的起始虚拟地址，即偏移量。以下是示例单位指示符，交替使用了大写、小写以及单引号和双引号：“k”或“K”（千字节）、“m”或“M”（兆字节）、“g”或“G”（千兆字节）以及“t”或“T”（兆兆字节）。Sybase 建议始终包括单位指示符。如果不包括单位指示符，则引号是可选的。不过，如果包括单位指示符，则必须使用引号。如果不提供单位指示符，则假定提供的值的单位为兆字节。偏移的大小取决于输入 vstart 值的方式。

- 如果未指定单位大小，则 vstart 将使用 2K 页作为其起始地址。例如，如果指定 `vstart = 13`，则 Adaptive Server 使用 13 * 2K 页作为起始地址的偏移量。
- 如果指定了单位值，vstart 就将它用作起始地址。例如，如果指定 `vstart = "13M"`，则 Adaptive Server 将起始地址的偏移量设置为 13MB。

vstart 的缺省值（通常为首选值）为 0。如果指定设备中的可用块数达不到 vstart + size 的块数之和，则 disk reinit 将失败。

注释 注意：如果在 AIX 操作系统上运行逻辑卷管理器，则 vstart 应为 2。

除非有 Sybase 技术支持部门的指导，否则不要指定 vstart。

cntrltype

指定磁盘控制器。其缺省值为 0。除非有 Sybase 技术支持部门的指导，否则不要重设此值。

dsync

指定对数据库设备的写入操作是刷新到存储介质中，还是仅在使用操作系统文件时放入缓冲区。本选项只在初始化操作系统文件时才有意义；初始化原始分区上的设备时不起作用。缺省情况下，所有操作系统文件初始化时都将 dsync 设置为 true。

directio

允许您将 Adaptive Server 配置为跳过操作系统缓冲区高速缓存，直接将数据传输到磁盘。directio 为静态参数，需要重新启动 Adaptive Server 才能生效。

缺省情况下，directio 对于非集群 Adaptive Server 设置为 false，对于集群 Adaptive Server 设置为 true。

对于原始设备，将忽略 directio 参数。

instance = "*instance_name*"

（仅限于集群）将设备指定为私有，将其拥有实例设置为 *instance_name*。

示例

示例 1 向 sysdevices 表中添加一个新行。此新行包含当前正在重新初始化的现有设备的特性：

```
disk reinit
name = "user_file",
physname = "/usr/u/sybase/data/userfile1.dat",
vdevno = 2, size = 5120, dsync = true
```

示例 2 向 sysdevices 表中添加一个新行，将数据直接传输到磁盘上。此新行包含当前正在重新初始化的现有设备的特性：

```
disk reinit
name = "user_disk",
physname = "/usr/u/sybase/data/userfile1.dat",
size = 5120, directio= true
```

用法

- Sybase 建议您只在 HP-UX、Windows 和 Linux 平台上将块设备用作数据库设备。如果尝试在 Sybase 建议不要使用块设备的平台上创建块设备， disk init 和 disk reinit 将显示警告消息。
- disk reinit 确保在 master 数据库已损坏，或者上次转储 master 之后又添加了设备时， master..sysdevices 仍然是正确的。
- disk reinit 与 disk init 类似，但前者不初始化数据库设备。
- 可将 *size* 指定为 float，但大小会向下舍入为最接近的 2K 倍数。
- 如果不对 *size* 使用单位指示符，则 disk reinit 将使用 2K 的虚拟页大小。
- 缺省情况下， directio 选项对于所有平台都设置为 false （关闭）。
- 有关恢复 master 数据库的完整信息，请参见 《系统管理指南》。

使用 *dsync*

注释　对于存储关键数据的任何设备，不要将 dsync 设置为 false。唯一的例外是 tempdb，它可以安全地存储在 dsync 设置为 false 的设备上。

- 启用 dsync 时，可确保在物理存储介质上进行对数据库设备的写入操作，并且 Adaptive Server 可以在出现系统故障时恢复设备上的数据。
- directio 和 dsync 互斥。如果某台设备的 dsync 已设置为 “true”，则不能为此设备将 directio 设置为 “true”。若要为设备启用 directio，则必须先将 dsync 重置为 “false”。

- 当 dsync 关闭时，UNIX 文件系统可能会对数据库设备的写入进行缓冲。即使尚未修改物理介质， UNIX 文件系统仍可能将更新标记为已经完成。出现系统故障时，不能保证物理介质上已进行数据更新，并且 Adaptive Server 可能无法恢复数据库。
- 对于主设备文件， dsync 始终处于打开状态。
- 仅在出现系统故障后无需恢复设备上的数据库时，关闭 dsync 值。例如，对于只存储 tempdb 数据库的设备，可以考虑将 dsync 关闭。
- Adaptive Server 会忽略存储在原始分区上的设备的 dsync 设置 确保在物理存储介质上进行对这些设备的写入操作，无论怎样设置 dsync。
- dsync 设置不用于 Windows NT 平台。
- disk reinit 确保在 master 数据库损坏或者上次转储 master 之后又添加了设备时， master..sysdevices 仍然是正确的。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 disk reinit 的权限检查因您的细化权限设置而异。执行 disk reinit 时，必须使用 master 数据库。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage disk 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。 disk reinit 权限不能移交。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
22	disk	disk reinit	• *角色* - 当前活动角色 • *关键字或选项* - disk reinit • *先前值* - NULL • *当前值* - NULL • *其它信息* - 磁盘名称 • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter database, create database, dbcc, disk init, disk refit.

系统过程 sp_addumpdevice, sp_helpdevice.

disk remirror

说明

在磁盘镜像因镜像设备发生故障而停止或者被 disk unmirror 命令临时禁用后重新启动它。

语法

```
disk remirror
        name = "device_name"
```

参数

name

要重新镜像的数据库设备的名称，它记录在 sysdevices 表的 name 列中，而且必须用单引号或双引号引起来。

示例

重新开始数据库设备 user_disk 的软件镜像：

```
disk remirror
  name = "user_disk"
```

用法

- 磁盘镜像可为用户数据库设备、主数据库设备或用于用户数据库事务日志的数据库设备创建软件镜像。如果数据库设备发生故障，其镜像会立即接替它运行。

  在镜像设备故障使得磁盘镜像临时停止，或通过 disk unmirror 命令的 mode = retain 选项临时禁用磁盘镜像之后，使用 disk remirror 命令重新建立镜像。 disk remirror 命令将保留的磁盘上的数据复制到镜像。

- 每次使用 disk remirror 之后请使用 dump database 命令备份 master 数据库。这使得在 master 损坏的情况下能够更容易和安全地进行恢复。

- 如果用 mode = remove 选项永久禁用了镜像，在使用 disk remirror 之前必须删除包含镜像的操作系统文件。

- 镜像数据库设备而不是数据库。

- 不关闭 Adaptive Server 即可镜像或重新镜像数据库设备或者取消数据库设备的镜像。磁盘镜像不会干涉数据库中正在进行的活动。

- 如果对镜像设备的读取或写入不成功， Adaptive Server 会对损坏的设备取消镜像并显示错误消息。 Adaptive Server 将继续运行，但不进行镜像。系统管理员必须使用 disk remirror 重新启动镜像。

- 除了镜像用户数据库设备外，还应该始终将用户数据库事务日志放在单独的数据库设备上。也可以镜像用于事务日志的数据库设备，从而得到更好的保护。若要将数据库事务日志 （即系统表 syslogs）放置到与存储该数据库其余部分的设备不同的设备上，可在创建数据库时指定数据库设备和日志设备。或者，使用 alter database 指向另一个设备，然后运行 sp_logdevice。

- 如果要为 master 数据库镜像数据库设备，可以在用 dataserver 实用程序重新启动 Adaptive Server 时使用 -r 选项和 UNIX 的镜像名称。将此选项添加到该服务器的 *RUN_servername* 文件，以使 startserver 实用程序获知此情况。例如，以下命令可启动名为 master.dat 的主设备及其镜像 mirror.dat：

  ```
  dataserver -dmaster.dat -rmirror.dat
  ```

  请参见 《实用程序指南》中的 dataserver 和 startserver。

- 要获取系统中所有 Adaptive Server 设备 （用户数据库设备及其镜像以及转储设备）的报告，请执行 sp_helpdevice。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 disk remirror 的权限检查因您的细化权限设置而异。执行 disk remirror 时，必须使用 master 数据库。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage disk 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。 disk remirror 权限不能移交。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
25	disk	disk remirror	• *角色* - 当前活动角色 • *关键字或选项* - disk remirror • *先前值* - NULL • *当前值* - NULL • *其它信息* - *磁盘名称* • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter database, create database, disk init, disk mirror, disk refit, disk reinit, disk unmirror, dump database, dump transaction, load database, load transaction.

系统过程 sp_diskdefault, sp_helpdevice, sp_logdevice.

实用程序 dataserver、 startserver.

disk resize

说明

动态增加 Adaptive Server 使用的设备的大小。

语法

```
disk resize
        name = "device_name",
        size = additional_space
```

参数

name

是要增加其大小的设备的名称。

additional_space

要为设备增加的额外空间量。

示例

若要将 testdev 的大小增加 4MB，请输入：

```
disk resize
name = "test_dev",
size = "4M"
```

用法

- disk resize 命令用于动态增加磁盘的大小。
- 设备重新调整后，应转储主设备，这样可以维护 sysdevices 表中的设备大小。如果尝试从旧的主设备转储中恢复，存储在 sysdevices 中的信息不是当前信息。
- 当增加设备大小后，将继续设置原来在设备上设置的所有属性。
- 磁盘物理初始化期间，如果由于磁盘空间不足而发生错误，那么 disk resize 将使数据库设备扩展到错误发生前的点。

  例如，在一台使用 4K 逻辑页的服务器上，如果试图为设备增加 40MB 的大小，但是只有 39.5MB 可用，那么设备只能扩展 39.5MB。扩展的大小 (39.5MB) 中， Adaptive Server 只使用了 39MB。最后的 0.5MB 虽已分配但未被使用，这是因为 4K 服务器为设备配置的最小值是 1MB。

  要利用最后的 0.5MB，请确保至少还有 1.5MB 可供设备使用，然后重新运行 disk resize，同时指定 1.5MB 为增量。
- 不能使用 disk resize 减少设备大小。
- *device_name* 必须有一个有效标识符。设备是使用 disk init 命令初始化的，它必须是一个有效的 Adaptive Server 设备，而不能是转储设备或装载设备。

- 以下是示例单位指示符，交替使用了大写、小写以及单引号和双引号：“k”或“K”（千字节）、“m”或“M”（兆字节）、“g”或“G”（千兆字节）以及“t”或“T”（兆兆字节）。 Sybase 建议始终包括单位指示符。尽管单位分类符是可选的，Sybase 建议始终在 disk resize 命令中包括单位分类符，以避免和分配的实际页数混淆。

  必须用单引号或双引号将单位指示符引起来。如果不使用单位指示符，缺省大小为磁盘页数。

- 调整操作进程中应始终禁用镜像。调整操作完毕后才可以重新建立镜像。

标准

符合 ANSI SQL 的级别：Transact-SQL 扩展。

权限

对 disk resize 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage disk 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
100	disk	disk resize	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - 索引名 • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 create database、 disk init、 drop database、 load database.

系统过程 sp_addsegment、 sp_dropsegment、 sp_helpdb、 sp_helpsegment、 sp_logdevice、 sp_renamedb、 sp_spaceused.

disk unmirror

说明

挂起用 disk mirror 命令启动的磁盘镜像，以允许硬件维护或硬件设备的更改。

语法

```
disk unmirror
        name = "device_name"
        [, side = {"primary" | secondary}]
        [, mode = {retain | remove}]
```

参数

name

要取消镜像的数据库设备的名称。名称必须用单引号或双引号引起来。

side

指定取消 primary 设备的镜像还是禁用 secondary 设备 （镜像）。缺省情况下是取消辅助设备的镜像。

mode

确定取消镜像是临时的 (retain) 还是永久的 (remove)。缺省情况下取消镜像是临时的。

如果计划以后在相同配置中重新镜像数据库设备，则指定 retain。此选项模仿主设备失败时的情况：

- I/O 仅指向*不*取消镜像的设备。
- sysdevices 的 status 列指示镜像已被停用。remove 删除对镜像设备的所有 sysdevices 引用。
- status 列指示镜像功能将被忽略。
- 如果主设备处于失效状态，则 phyname 列由 mirrorname 列中辅助设备的名称替换。
- mirrorname 列设置为 NULL。

示例

示例 1 挂起数据库设备 user_disk 的软件镜像：

```
disk unmirror
name = "user_disk"
```

示例 2 挂起辅助端上数据库设备 user_disk 的软件镜像：

```
disk unmirror name = "user_disk", side = secondary
```

示例 3 挂起数据库设备 user_disk 的软件镜像并删除对镜像设备的所有设备引用：

```
disk unmirror name = "user_disk", mode = remove
```

用法

- 磁盘镜像可为用户数据库设备、主数据库设备或用于用户数据库事务日志的数据库设备创建软件镜像。如果数据库设备发生故障，其镜像会立即接替它运行。

  disk unmirror 可永久或临时性地禁用原数据库设备或其镜像，以使得 Adaptive Server 不再能够读取或写入该设备。它不删除操作系统中的相关文件。

- 取消磁盘镜像可改变 master 数据库中的 sysdevices 表。每次使用 disk unmirror 之后请使用 dump database 命令备份 master 数据库。这使得在 master 损坏的情况下能够更容易和安全地进行恢复。
- 可以在使用数据库设备时取消其镜像。
- 不能在 dump database、load database 或 load transaction 正在进行时取消数据库设备的镜像。 Adaptive Server 显示一条消息，询问是中止转储或装载，还是将 disk unmirror 推迟到转储或装载完成后进行。
- 不能在 dump transaction 正在进行时取消数据库日志设备的镜像。Adaptive Server 显示一个消息，询问是中止转储或装载，还是将 disk unmirror 推迟到转储或装载完成后进行。

  注释 dump transaction 注意：取消日志设备的镜像时， with truncate_only 和dump transaction with no_log 不受影响。

- 应对所有缺省数据库设备进行镜像，以便在 create 或 alter database 命令影响缺省列表中的数据库设备时得到保护。
- 当向已镜像设备的读取或写入不成功时， Adaptive Server 会自动取消镜像坏设备并显示错误消息。 Adaptive Server 将继续运行，但不进行镜像。系统管理员必须用 disk remirror 命令重新启动镜像。
- 要获得系统中所有 Adaptive Server 设备的报告 （用户数据库设备及其镜像以及转储设备），请执行 sp_helpdevice。
- 在使用 disk unmirror 命令的 mode = retain 选项使得镜像临时停止之后使用 disk remirror 重新建立镜像。如果用 mode = remove 选项永久禁用了镜像，在使用 disk remirror 之前必须删除包含镜像的操作系统文件。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 disk unmirror 的权限检查因您的细化权限设置而异。执行 disk unmirror 时，必须使用 master数据库。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage disk 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。 disk unmirror 权限不能移交。

审计　　sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
24	disk	disk unmirror	• *角色* - 当前活动角色 • *关键字或选项* - disk unmirror • *先前值* - NULL • *当前值* - NULL • *其它信息* - 磁盘名称 • *代理信息* - set proxy 有效时的初始登录名

另请参见　　**命令** alter database、 create database、 disk init、 disk mirror、 disk refit、 disk reinit、disk remirror、dump database、dump transaction、load database、load transaction.

系统过程 sp_diskdefault, sp_helpdevice, sp_logdevice.

实用程序 dataserver、 startserver.

drop database

说明

从 Adaptive Server 中删除一个或多个数据库，包括存档数据库。

语法

drop database *database_name* [, *database_name*] ...

参数

database_name

是要删除的数据库的名称。使用 sp_helpdb 可获得数据库的列表。

示例

示例 1 删除 publishing 数据库及其所有内容：

```
drop database publishing
```

示例 2 key_db 是加密密钥所驻留的数据库， col_db 是包含加密列的数据库。 Adaptive Server 将引发错误，并使删除 key_db 的操作失败。但将成功删除 col_db。若要删除这两个数据库，请先删除 col_db：

```
drop database col_db, key_db
```

用法

- 当删除存档数据库时，将从 scratch 数据库的 sysaltusages 表中删除该数据库所有的行。这需要 scratch 数据库中的日志空间。
- 删除一个数据库就会删除此数据库及其所有对象，释放其存储分配，并清除其在 master 数据库的 sysdatabases 和 sysusages 系统表中的所有条目。
- drop database 清除 master..sysattributes 中附属于被删除的数据库的可疑页条目。

加密列和 *drop database*

为了防止密钥意外丢失，如果数据库包含的密钥当前用于加密其它数据库中的列， drop database 将会失败。若要删除数据库，请执行下列操作：

- 使用 alter table 来解密列，或者使用其它密钥来修改要加密的列。
- 删除包含加密列的表或数据库。

限制

- 删除数据库时，必须使用 master 数据库。
- 不能删除使用中的数据库 （由其他用户打开进行读写操作的数据库）。
- 不能使用 drop database 删除由其它数据库中的表所引用的数据库。确定哪些表和外部数据库在当前数据库的主键表上有外键约束，执行：

```
select object_name (tableid), frgndbname
    from sysreferences
    where frgndbname is not null
```

使用 alter table 删除跨数据库约束，然后重新发出 drop database 命令。

- 可以使用 drop database 删除损坏的数据库。如果 drop database 由于数据库损坏而无法运行，请使用 dbcc dbrepair 修复数据库：

  ```
  dbcc dbrepair (database_name, dropdb)
  ```

- 如果启用审计，则不能删除 sybsecurity 数据库。禁用审计时，只有系统安全员才能删除 sybsecurity。

标准　符合 ANSI SQL 的级别Transact-SQL 扩展。

权限　对 drop database 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是数据库的所有者，或拥有对数据库的 own database 特权。要删除 sybsecurity，您必须是数据库的所有者或拥有 manage auditing 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库的所有者、具有 sa_role 的用户，或者拥有 sso_role 的用户 （对于 sybsecurity）。

审计　sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
26	drop	drop database	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见　**命令** alter database, create database, dbcc, use.

过程 sp_changedbowner, sp_helpdb, sp_renamedb, sp_spaceused.

drop default

说明

删除用户定义的缺省值。

语法

drop default [*owner*.]*default_name*
[, [*owner*.]*default_name*] ...

参数

default_name

是现有缺省值的名称。执行 sp_help 可显示现有缺省值的列表。指定所有者的名称，以删除由当前数据库中的其他用户拥有的同名缺省值。 *owner* 的缺省值是当前用户。

示例

删除数据库中用户定义的缺省值 datedefault：

```
drop default datedefault
```

用法

- 不能删除当前绑定到列或用户定义数据类型的缺省值。在删除缺省值前应使用 sp_unbindefault 解除对它的绑定。
- 如果不先解除当前缺省值的绑定，则不能将新的缺省值绑定到列或用户定义数据类型。新的缺省值会覆盖旧的缺省值。
- 删除 NULL 列的缺省值时，NULL 就成为该列的缺省值。删除 NOT NULL 列的缺省值时，如果插入数据时不显式地为该列输入一个值，就会出现错误消息。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop default 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是缺省值的所有者或拥有 drop any default 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是缺省值的所有者或拥有 sa_role 的用户。

drop default 权限缺省情况下授予缺省值的所有者，并且不能移交。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
31	drop	drop default	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 create default.

系统过程 sp_help、 sp_helptext、 sp_unbindefault.

drop encryption key

说明

允许密钥所有者删除命名的加密密钥。

语法

drop encryption key [[*database*.][*owner*].]*keyname*

用于显式删除外部登录口令服务密钥的语法是：

```
drop encryption key syb_extpasswdkey
    with password encryption downgrade
```

用于显式删除隐藏文本服务密钥的语法是：

```
drop encryption key syb_syscommkey_dddddd
```

或：

```
drop encryption key syb_syscommkey with text encryption downgrade
```

参数

database
是数据库的名称。

owner
是所有者。

keyname
是密钥的名称。

syb_extpasswdkey
服务任务的名称。

当指定 with password encryption downgrade 时， Adaptive Server 会用 15.7 之前版本中使用的算法重置外部登录口令，而 Replication Agent 口令、 CIS 和 RTMS 外部登录口令会被重置为无效的值。

删除密钥后，管理员必须手动重新输入口令，才能继续使用相应的服务。

syb_syscommkey_ddddddd
是要删除的各个 syscomments 服务密钥的显式名称。

syb_syscommkey with text encryption downgrade
Adaptive Server 会用 15.7 之前的版本中使用的算法来重新加密 syscomments 中的所有隐藏文本。

示例

删除加密密钥 `cc_key`：

```
drop encryption key cust.dbo.cc_key
```

用法

- 如果密钥有密钥副本，则将副本与基本密钥一起删除。
- 如果任何数据库中有任意列使用该密钥加密，则此命令将失败。

- drop encryption key 不能检查处于已存档、可疑、脱机、未恢复状态或当前处于装载状态的数据库是否存在由密钥加密的列。此命令发出一条警告消息，列出不可用的数据库，但不会失败。在数据库联机后，任何包含使用删除的密钥加密的列的表将无法使用。要恢复密钥，系统管理员必须装载删除此密钥之前创建的数据库转储。

权限

对 drop encryption key 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是密钥的所有者或拥有 manage any encryption key 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是密钥的所有者或拥有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
109		drop encryption key	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

create encryption key、 alter encryption key、 sp_encryption、 sp_help

drop function

说明　从当前数据库中删除一个或多个用户定义的函数。

语法　drop function{ [*owner_name* .]*function_name* } [,...n]

参数

owner_name

是拥有该用户定义函数的用户 ID 的名称。必须是现有用户 ID。

function_name

是要删除的用户定义的函数的名称。可以选择是否指定所有者名称，但不能指定服务器名和数据库名。

示例　删除 bonus 函数：

```
drop function bonus
```

用法　drop function 从当前数据库中删除用户定义的标量 SQL 函数。

权限　对 drop function 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是函数的所有者或拥有 drop any function 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是函数的所有者或拥有 sa_role 的用户。 权限是不可转让的。

drop function (SQLJ)

说明

删除 SQLJ 函数。

语法

drop func[tion] [*owner*.]*function_name*
[, [*owner*.]*function_name*] ...

参数

[*owner*.]*function_name*
是 SQLJ 函数的 SQL 名称。

示例

删除 SQLJ 函数 square_root：

```
drop function square_root
```

用法

drop function 仅删除当前数据库中用户创建的函数。它不删除系统函数。

权限

对 drop function (SQLJ) 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是函数的所有者或拥有 drop any function 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是函数的所有者或拥有 sa_role 的用户。 权限是不可转让的。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
98	drop	drop function	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

文档 有关 SQLJ 函数的详细信息，请参见 《Adaptive Server Enterprise 中的 Java》。

命令 create function (SQLJ).

drop index

说明

从当前数据库的表中删除索引。

语法

drop index *table_name.index_name*
[, *table_name.index_name*] ...

参数

table_name
是索引列所在的表。该表必须在当前数据库中。

index_name
是要删除的索引。在 Transact-SQL 中，数据库中的索引名不必唯一，但表中的索引名必须唯一。

示例

从 authors 表中删除 au_id_ind：

```
drop index authors.au_id_ind
```

用法

- 发出 drop index 命令后，您就重新获得所有原来由索引占用的所有空间。这个空间可用于任何数据库对象。
- 不能在系统表上使用 drop index。
- drop index 不能删除支持唯一约束的索引。若要删除此类索引，请通过 alter table 删除约束，或者删除表。有关唯一约束索引的详细信息，请参见 create table。
- 不能删除任何打开的游标正在使用的索引。有关哪些游标是打开的以及它们使用哪些索引的信息，请使用 sp_cursorinfo。
- 若要获取表上存在哪些索引的信息，请使用下面的命令，其中 objname 是表名：

```
sp_helpindex objname
```

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop index 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表的所有者。
细化权限已禁用	在禁用细化权限的情况下，您必须是表的所有者。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
105		drop index	• *角色* – 当前活动角色 • *关键字或选项* – NULL • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 create index、 setuser.

系统过程 sp_cursorinfo、 sp_helpindex、 sp_spaceused.

drop login

说明

删除登录帐户或帐户列表。

语法

drop login *login_name* [, *login_name_list*] [with override]

参数

login_name
　指定要删除的登录帐户的名称。

login_name_list
　指定要删除的登录帐户列表。

with override
　删除登录名，即使有无法检查其有无登录引用的不可用数据库也是如此。

示例

删除登录帐户 ravi 和 vinod。

```
drop login ravi, vinod
```

用法

- 执行 drop login 将从 Adaptive Server 中删除一个用户登录名，同时从 master.dbo.syslogins 中删除该用户的条目。
- Adaptive Server 会重新使用已删除的登录名的服务器用户 ID，这会使责任受到影响。您应该避免彻底删除帐户，而尽量使用 sp_locklogin 锁定所有不再使用的帐户。
- 如果需要删除登录名，一定要审计这些事件 （使用 sp_audit）以便您能够拥有对这些事件的记录。
- drop login 删除所有与已删除的登录名关联的资源限制。
- 如果要删除的登录名是服务器上任意数据库中的用户，则 drop login 失败。使用 sp_dropuser 删除某个数据库的用户。如果用户在数据库中拥有任何对象，则不能从数据库删除此用户。
- 如果要删除的登录名是系统安全员，则 drop login 将验证是否至少存在一个其它已解锁的系统安全员的帐户。如果不存在，则 drop login 会失败。同样， drop login 还确保始终至少存在一个已解锁的系统管理员帐户。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop login 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是拥有 manage any login 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是拥有 sso_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
139	login_admin	drop login	关键字包含： *loginname1[, ...[, loginnameN*]]

另请参见

命令 create login profile、create login、drop login profil、alter login、alter login profile， .

文档 有关删除登录帐户的详细信息，请参见 《安全性管理指南》。

函数 lprofile_id、 lprofile_name.

系统过程 sp_passwordpolicy、sp_displaylogin、sp_displayroles、sp_locklogin.

drop login profile

说明

删除登录配置文件或登录配置文件列表。

语法

drop login profile *login_profile_name* [, *login_profile_name_list*]
[with override]

参数

login_profile_name
指定要删除的登录配置文件的名称。

login_profile_name_list
指定要删除的登录配置文件的列表。

with override
强制删除绑定到登录帐户的登录配置文件。与删除的登录配置文件相关的登录帐户将与缺省登录配置文件相关联。

示例

示例 1 如果登录配置文件 group1_login_profile 不绑定到一个或多个登录帐户，则删除它。

```
drop login profile group1_login_profile
```

示例 2 由于 sa_login_profile 绑定到一个或多个登录帐户，生成了错误。

```
drop login profile sa_login_profile

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.
```

示例 3 sa_login_profile 将被删除，即使它绑定到一个或多个登录帐户也是如此。

```
drop login profile sa_login_profile with override
```

用法

- 命令 drop login profile 将删除不绑定到登录帐户的登录配置文件。
- 使用 drop login profile with override 可强制删除绑定到登录帐户的登录配置文件。如果登录配置文件没有与登录帐户绑定，则登录帐户将绑定到缺省登录帐户 （如果存在的话）。如果缺省登录配置文件也不存在，则会使用早于 Adaptive Server 15.7 版的优先规则来检查有无登录帐户。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop login profile 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是拥有 manage any login profile 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是拥有 sso_role 的用户才能删除登录配置文件。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
141	security_profile	drop login profile	关键字包含： *login_profile_name* [WITH OVERRIDE]

另请参见

命令 create login profile、 create login、 alter login profile、 alter login、 drop login .

文档 有关删除登录配置文件的详细信息，请参见 《安全性管理指南》。

函数 lprofile_id、 lprofile_name.

系统过程 sp_passwordpolicy、sp_displaylogin、sp_displayroles、sp_locklogin.

drop precomputed result set

说明

删除预计算的结果集。

语法

drop {precomputed result set | materialized view}
[*owner_name*.]*prs_name*

参数

materialized view | precomputed result set
删除物化视图或预先计算结果集。

prs_name
预先计算结果集的名称。完全限定的 *prs_name* 不得含有服务器或数据库名称。

示例

删除 authors_prs 预先计算的结果集：

```
drop precomputed result set authors_prs
```

用法

标准

drop precomputed result set 命令是 Transact-SQL 扩展，不在 SQL 标准范围内。

权限

您必须是预先计算结果集的所有者。

审计

未审计 “删除预先计算结果集”。

drop procedure

说明

删除过程。

语法

drop proc[edure] [*owner*.]*procedure_name*
[, [*owner*.]*procedure_name*] ...

参数

procedure_name

是要删除的 Transact-SQL 或 SQLJ 过程的名称。指定所有者的名称，以删除由当前数据库中的其他用户拥有的另一个同名过程。 *owner* 的缺省值是当前用户。

示例

示例 1 删除存储过程 showind：

```
drop procedure showind
```

示例 2 取消扩展存储过程 xp_echo 的注册：

```
drop procedure xp_echo
```

用法

- drop procedure 删除用户定义的存储过程、系统过程以及扩展存储过程 (ESP)。
- 每次用户或程序执行过程时 Adaptive Server 都会检查其是否存在。
- 过程组 （即拥有相同名称，但却具有不同 number 后缀的多个过程）可以用单个 drop procedure 语句删除。例如，如果将与名为 orders 的应用程序一起使用的过程命名为 orderproc;1、orderproc;2 等等，下面的语句将删除整个组：

  ```
  drop proc orderproc
  ```

  一旦将过程分组，就不能单独删除组中的过程。例如，以下语句是不允许的：

  ```
  drop procedure orderproc;2
  ```

  不能删除作为过程组的扩展存储过程。
- sp_helptext 显示过程的文本，后者存储在 syscomments 中。
- sp_helpextendedproc 显示 ESP 及其相应的 DLL。
- 删除 ESP 会将过程从系统表中删除，因此会取消过程的注册。它对基础 DLL 没有影响。
- drop procedure 仅删除当前数据库中用户创建的存储过程。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop procedure 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是过程的所有者或拥有 drop any procedure 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是过程的所有者或拥有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
28	drop	drop procedure	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 create procedure、 create procedure (SQLJ).

系统过程 sp_depends、 sp_dropextendedproc、 sp_helpextendedproc、 sp_helptext、 sp_rename.

drop role

说明

删除用户定义的角色。

语法

drop role *role_name* [with override]

参数

role_name

是要删除的角色的名称。

with override

覆盖删除角色的限制。使用 with override 选项时，不必检查是否删除了每个数据库中的角色权限就可删除任何角色。

示例

示例 1 仅当撤消了所有数据库中的所有权限时才能删除指定角色。删除角色前，系统管理员或对象所有者必须撤消在每个数据库中所授予的权限，否则命令将失败：

```
drop role doctor_role
```

示例 2 从所有数据库中删除指定角色，并删除权限信息以及对此角色的任何其它引用：

```
drop role doctor_role with override
```

用法

- 删除角色前不必先删除成员资格。删除一个角色将自动删除此角色中的所有用户成员资格，无论是否使用了 with override 选项。
- 从 master 数据库中使用 drop role。

限制

- 不能使用 drop role 删除系统角色。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop role 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是拥有 manage roles 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是拥有 sso_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
85	roles	create role、 drop role、 alter role、 grant role 或 revoke role	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter role、 create role、 grant、 revoke、 set.

系统过程 sp_activeroles、sp_displaylogin、sp_displayroles、sp_helprotect、.

drop rule

说明

删除用户定义的规则。

语法

drop rule [*owner*.]*rule_name*[, [*owner*.]*rule_name*] ...

参数

rule_name

是要删除的规则的名称。指定所有者的名称，以删除由当前数据库中的其他用户拥有的另一个同名规则。 *owner* 的缺省值是当前用户。

示例

从当前数据库中删除 pubid_rule 规则：

```
drop rule pubid_rule
```

用法

- 删除规则前，使用 sp_unbindrule 解除绑定。如果规则未解除绑定，就会出现错误消息，且 drop rule 命令失败。
- 不必解除当前规则的绑定就可将新的规则绑定到列或用户定义数据类型。新的规则会覆盖旧的规则。
- 删除规则后， Adaptive Server 将新数据输入先前由该规则控制的列时，将没有这些约束。不管怎样，现有的数据都不会受到影响。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop rule 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是规则的所有者或拥有 drop any rule 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是规则的所有者或拥有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
30	drop	drop rule	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 create rule.

系统过程 sp_bindrule、 sp_help、 sp_helptext、 sp_unbindrule.

drop service

说明

drop service 命令用于从当前数据库中删除用户定义的 Web 服务。元数据和对应的存储过程都将被删除。

语法

drop service *service-name*

参数

service-name

是用户定义的 Web 服务的名称。该名称可以是任何对存储过程有效的名称。如果指定了一个不存在的服务名称，则会产生异常。此外，您也不能删除其它会话当前正在使用的服务。

示例

此示例删除名为 sp_who_service 的用户定义的 Web 服务：

```
drop service sp_who_service
```

用法

您必须先取消配置用户定义的 Web 服务，然后才能将其删除。

权限

对 drop role 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是服务的所有者或拥有 drop any procedure 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是拥有 sa_role 的用户。

另请参见

命令 create service.

存储过程 sp_webservices.

文档 《Web 服务用户指南》

drop table

说明

从数据库中删除表定义及其所有的数据、索引、分区属性、触发器、加密属性和权限。

语法

drop table [[*database*.]*owner*.]*table_name*
[, [[*database*.]*owner*.]*table_name*] ...

参数

table_name

是要删除的表的名称。如果该表位于另一数据库中，请指定数据库名称；如果数据库中有多个具有同样名称的表，请指定所有者的名称。*owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

示例

从当前数据库中删除表 roysched 及其数据和索引：

```
drop table roysched
```

用法

- 使用 drop table 时，表上的所有规则或缺省值都将丢失其绑定，且与之有关的触发器都将自动删除。如果重新创建表，就必须重新绑定相应的规则和缺省值，并重新创建所有触发器。
- 删除表时，任何与该表关联的分区条件也将被删除。
- 删除表将会删除与该表的列关联的所有解密缺省值，还将删除这些列的加密属性。
- 删除表时受到影响的系统表为 sysobjects、 syscolumns、 sysindexes、 sysprotects、 syscomments、 syspartitions、 syspartitionkeys 和 sysprocedures。
- 如果启用了 CIS，且删除的表是用 create existing table 创建的，则该表不会从远程服务器上删除。相反， Adaptive Server 删除系统表上对该表的引用。

限制

- 不能在系统表上使用 drop table 命令。
- 可以删除任何数据库中的表，只要您是表的所有者即可。例如，可以使用以下两种方法之一删除 otherdb 数据库中名为 newtable 的表：

  ```
  drop table otherdb..newtable
  drop table otherdb.yourname.newtable
  ```

- 如果 delete 表中的所有行，或对它使用 truncate table 命令，则该表仍然会存在，直到对它使用 drop。

删除具有跨数据库参照完整性约束的表

- 创建跨数据库约束时， Adaptive Server 会将以下信息存储在每个数据库的 sysreferences 系统表中：

表 1-18: 存储的有关参照完整性约束的信息

存储在 sysreferences 中的信息	包含有关被引用表的信息的列	包含有关引用表的信息的列
键列 ID	refkey1 到 refkey16	fokey1 到 fokey16
表 ID	reftabid	tableid
数据库名称	pmrydbname	frgndbname

- 由于引用表依赖于来自被引用表的信息， Adaptive Server 不允许您：
  - 删除被引用表，
  - 删除包含它的外部数据库，或
  - 使用 sp_renamedb 对其中任何一个数据库进行重命名。

  使用 sp_helpconstraint 确定哪些表引用您要删除的表。在重新发出 drop table 之前使用 alter table 删除约束。
- 可以删除进行引用的表或其数据库。 Adaptive Server 自动从被引用数据库中删除外键信息。
- 每次添加或删除跨数据库约束或者删除包含跨数据库约束的表时，都请转储上述两个受影响的数据库。

警告！ 装载这些数据库的早期转储可能会导致数据库损坏。有关装载具有跨数据库参照完整性约束的数据库的详细信息，请参见 《系统管理指南》。

标准　符合 ANSI SQL 的级别Transact-SQL 扩展。

权限　对 drop table 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表的所有者或拥有 drop any table 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是表的所有者或拥有 sa_role 的用户。

审计　sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
27	drop	drop table	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见	**命令** alter table、 create table、 delete、 truncate table.
	系统过程 sp_depends、 sp_help、 sp_spaceused.

drop thread pool

说明

删除用户定义的池。

有关进程模式的考虑事项

drop thread pool 在进程模式中不受支持。

语法

drop thread pool *pool_name*

参数

pool_name
要删除的池的名称。

示例

删除名为 sales_pool 的池：

```
drop thread pool sales_pool
```

用法

- Adaptive Server 会将与删除的线程池关联的任务重新分配给 syb_default_pool。
- 不能删除当前正在使用执行类定义的线程池。可使用 sp_dropexeclass 删除执行类。
- 不能删除系统创建的线程池 （以 syb_ 开头）。
- 线程池必须等待当前运行的任务交出控制权，才能被删除，这可能会导致 Adaptive Server 在删除池时略有延迟。
- 在要删除的线程池中运行的任务将迁移到 syb_default_pool。
- 不能将 Transact-SQL 变量用作 drop thread pool 的参数。
- 可以连同 drop thread pool 一起发出 execute immediate。

标准

符合 ANSI SQL 的级别：Transact-SQL 扩展。

权限

对 drop thread pool 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是拥有 manage any thread pool 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是拥有 sa_role 的用户。

审计

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
144			池名称

drop trigger

说明

删除触发器。

语法

drop trigger [*owner*.]*trigger_name*
[, [*owner*.]*trigger_name*] ...

参数

trigger_name
是要删除的触发器的名称。指定所有者的名称，以删除由当前数据库中的其他用户拥有的另一个同名触发器。 *owner* 的缺省值是当前用户。

示例

从当前数据库中删除 trigger1：

```
drop trigger trigger1
```

用法

- drop trigger 删除当前数据库中的触发器。
- 若要为同一操作 （insert、 update 或 delete）创建新的触发器，不需要显式从表中删除触发器。在表或列中，同一操作的每个新的触发器会覆盖前一个触发器。
- 删除表时， Adaptive Server 会自动删除与之关联的所有触发器。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop trigger 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是触发器的所有者或拥有 drop any trigger 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是触发器的所有者或拥有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
29	drop	drop trigger	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 create trigger.

系统过程 sp_depends、 sp_help、 sp_helptext.

drop view

说明

从当前数据库中删除一个或多个视图。

语法

drop view [*owner*.]*view_name* [, [*owner*.]*view_name*] ...

参数

view_name

是要删除的视图的名称。指定所有者的名称，以删除由当前数据库中的其他用户拥有的另一个同名视图。 *owner* 的缺省值是当前用户。

示例

从当前数据库中删除视图 new_price：

```
drop view new_price
```

用法

- 使用 drop view 命令时，会从系统表 sysobjects、 syscolumns、 syscomments、 sysdepends、 sysprocedures 和 sysprotects 中删除视图的定义及其它有关信息 （包括特权）。
- 每次视图被引用时 （例如被其它视图或存储过程引用）都会检查视图是否存在。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 drop view 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是视图的所有者或拥有 drop any view 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是视图的所有者或拥有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
33	drop	drop view	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL

另请参见

命令 create view.

系统过程 sp_depends、 sp_help、 sp_helptext.

dump configuration

说明

创建 Adaptive Server 配置文件的备份并保存到指定的转储目录。备份由 Adaptive Server 而不是 Backup Server 创建。

语法

```
dump config[uration]
    [with {
        file = (file_option, file_option...)}
    } ]
```

参数

with file = (*file_option*, *file_option*...)}

允许基于指定的选项创建一个或多个文件的备份。

有效的 *file_option* 值包括：

- server_config[uration] – 服务器配置文件。
- dump_history – 转储历史记录文件。
- cluster_config[uration] – 集群配置文件。

  在 Adaptive Server Cluster Edition 的集群配置中，每个实例都可以有自己的服务器配置文件。 dump configuration 命令转储执行命令的实例的服务器配置文件。集群配置文件从仲裁设备中生成，命名为 *cluster.cfg*，附加有当前时间戳。
- 'all' – （缺省值）所有列出的 （已知的）配置文件。

如果省略 *file_option*，则备份所有现有配置文件。

示例

示例 1 创建服务器配置文件和转储历史记录文件的备份：

```
dump configuration
    with file = (server_config, dump_history)
```

示例 2 创建所有列出的 （已知的）配置文件的备份：

```
dump configuration
```

用法

- *ist_option* 的值 'all' 必须用单引号或双引号引起来。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 dump configuration 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是拥有 manage dump configuration 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是拥有 oper_role 或 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
147	dump_config	dump configuration	• *角色* – 当前活动角色 • *关键字或选项* – NULL • *先前值* – NULL • *当前值* – NULL • *其它信息* - 用户提供的配置文件选项 • *代理信息* – set proxy 有效时的初始登录名

另请参见

文档 《系统管理指南》中的 “备份和恢复用户数据库”。 .

命令 dump database、 load database、 load transaction、 online database.

系统过程 sp_addumpdevice、 sp_dboption、 sp_dropdevice、 sp_helpdevice、 sp_hidetext、 sp_logdevice、 sp_volchanges.

dump database

说明

以一种可用 load database 命令读取的格式制作整个数据库 （包括事务日志）的备份副本。转储和装载通过 Backup Server 执行。

如果您不转储压缩数据，则 load database 操作的目标平台不必和发生 dump database 操作的源平台相同。压缩数据的转储和装载必须发生在同一平台上。不过，可从大型平台向小型平台执行 dump database 和 load database，或者从小型平台向大型平台执行。

有关您的站点许可使用 Tivoli Storage Manager 时的 dump database 语法，请参见 《将 Backup Server 与 IBM Tivoli Storage Manager 配合使用》。

语法

```
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) 使用此语法可在 Tivoli Storage Manager 提供备份服务时复制数据库。

```
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]
        } ]
```

参数

config[uration] = *config_name*

读取指定的转储配置并使用指定的值执行转储操作。

如果使用转储配置，则不能将分条目录指定为命令的参数。 Adaptive Server 在转储配置指定的分条目录中创建转储文件。使用以下约定命名转储文件：

database_name.dump_type.date-timestamp.stripeID

明确指定的命令参数覆盖了转储配置所指定的参数值。

database_name

是您从中复制数据的数据库的名称。可将数据库名称指定为文字、局部变量或存储过程参数。

compress::*compression_level*

已不再使用，提供它的目的仅仅是为了与旧版本应用程序兼容。改为将 "compression = *compress_level*" 用于压缩。有关 compress 选项的详细信息，请参见 《系统管理指南，卷 2》中的 “备份和恢复用户数据库”。

注释　Sybase 建议首选使用本机 "compression = *compress_level*" 选项，旧 "compress::*compression_level*" 选项其次。该本机选项允许压缩本地和远程转储，并且它创建的转储将在装载期间描述其自身的压缩级别。保留旧选项是为了与以前的应用程序兼容。

---

to *stripe_device*

是要向其复制数据的设备。有关指定转储设备时使用何种形式的信息，请参见第 362 页的 “指定转储设备”。

at *backup_server_name*

是 Backup Server 的名称。转储到缺省 Backup Server 时不要指定此参数。仅当通过网络转储到远程 Backup Server 时才指定此参数。使用此选项可指定多达 32 个远程 Backup Server。通过网络进行转储时，请指定在转储设备附加到的计算机上运行的远程 Backup Server 的*网络名*。对于使用接口文件的平台， *backup_server_name* 必须出现在接口文件中。

density = *density_value*

替换磁带设备的缺省密度。有效密度为 800、 1600、 6250、 6666、 10000 和 38000。并不是所有的值对于每个磁带驱动器都有效；请使用适合您的磁带驱动器的正确密度。

blocksize = *number_bytes*

替换转储设备的缺省块大小。块大小必须至少为一个数据库页 （对于大多数系统为 2048 字节），且必须为数据库页大小的整数倍。若要获得最佳性能，请将 blocksize 指定为 2 的乘方，例如 65536、131072 或 262144。

capacity = *number_kilobytes*

是设备可写入单个磁带卷的最大数据量。容量至少应为 5 个数据库页，但应小于设备的推荐容量。

计算容量的一般规则是使用设备制造商给出的设备最大容量的 70%，留出 30% 的容量用于记录间隙和磁带标志之类的开销。最大容量是驱动器上设备的容量，而不是驱动器本身的容量。此规则在多数情况下适用，但可能由于各供应商和设备的开销存在差异而不能全部适用。

在不能可靠检测到磁带结束标志的 UNIX 平台上，请指明可转储到磁带的千字节数。对于作为物理路径名指定的转储设备，必须提供 capacity。如果将转储设备作为逻辑设备名指定，则除非指定容量，否则 Backup Server 使用存储在 sysdevices 系统表中的 *size* 参数。

compression = *compress_level*

是 0 到 9 之间的数字、100 或 101。对于一位数的压缩级别，0 表示不进行压缩，而 9 表示最高级别的压缩。压缩级别 100 和 101 提供更加快捷、更加有效的压缩模式，其中 100 提供更加快捷的压缩，而 101 提供程度更高的压缩。如果不指定 *compression_level*，Adaptive Server 将不会压缩转储。

注释　Sybase 建议首选使用本机 "compression = *compress_level*" 选项，为了与以前的应用程序兼容而保留的旧选项 "compress::*compression_level*" 其次。

---

dumpvolume = *volume_name*

建立指派给卷的名称。 *volume_name* 的最大长度为 6 个字符。覆盖现有转储、转储到新磁带或者转储到内容不可识别的磁带时， Backup Server 会在 ANSI 磁带标签中写入 *volume_name*。load database 命令会检查标签，如果装载了错误的卷，则会生成错误消息。

警告！请在创建时为每个磁带卷贴上标签，以便操作员装载正确的磁带。

---

with shrink_log

在使用 alter database log off 命令缩减日志中的空间时，如果在数据库中创建了空洞，则会使用它。此命令会在数据库不在转储序列中时自动删除数据库结尾的空洞。同样， dump database 将会在数据库不在转储序列中时（也就是说，当您由于 dump transaction 不被允许而被迫运行 dump database 时，例如，当执行了任何最少日志记录命令时）自动删除数据库结尾的任何空洞。dump database 的 with shrink_log 选项会删除数据库结尾的空洞，无论数据库是否在转储序列中都是如此。

with verify[= header | full]

允许 Backup Server 在数据页复制到存档中时对数据页执行最低限度的标头或结构行检查。此时不会对 gam、 oam、 allocation pages、 indexes、 text 或 log 页进行结构检查。唯一的其它检查在页码与页头匹配的页上执行。

stripe on *stripe_device*

是附加的转储设备。可以使用多达 32 个设备，其中包括在 to *stripe_device* 子句中命名的设备。 Backup Server 将数据库分成几个大致相等的部分，并将每个部分发送到不同的设备。转储是在每个设备上同时进行的，减少了进行转储所需的时间，并且转储过程中需要的卷更改更少。请参见第 362 页的 “指定转储设备”。

dismount | nodismount

在支持逻辑卸下的平台上，确定磁带是否保持装入状态。缺省情况下，转储完成时将卸下用于转储的全部磁带。使用 nodismount 命令可使磁带供其它转储或装载使用。

nounload | unload

确定转储完成后是否回绕磁带。缺省情况下磁带不会回绕，从而使您可以向同一磁带卷进行其它转储。请为要添加到多转储卷的最后一个转储文件指定 unload。这样，在转储完成后就会回绕并卸载磁带。

passwd = *password*

是您提供的口令，用来防止转储文件被未经授权的用户使用。口令长度必须介于 6 到 30 个字符之间。口令不能使用变量。请参见 《系统管理指南，卷 1》中的 “管理 Adaptive Server 登录、数据库用户和客户端连接”。

retaindays = *number_days*

（UNIX 系统）转储到磁盘时，请指定天数， Backup Server 保护转储在此天数内不被覆盖。如果试图在过期前覆盖转储，则在覆盖未过期卷前， Backup Server 会要求进行确认。

注释 此选项仅适用于转储到磁盘时；它不适用于磁带转储。

---

number_days 必须是正整数，或者，对于可立即覆盖的转储则为 0。如果不指定 retaindays 值，则 Backup Server 将使用由 sp_configure 设置的 tape retention in days 值。

noinit | init

确定是将转储附加到现有的转储文件还是重新初始化 （覆盖）磁带卷。缺省情况下， Adaptive Server 将转储附加在最后一个磁带结束标记之后，从而使您可以向同一个卷转储其它数据库。新的转储只能附加到多卷转储的最后一个卷上。对转储到磁带的第一个数据库使用 init 以覆盖其内容。

在需要 Backup Server 存储或更新磁带配置文件中的磁带设备特性时，可使用 init 命令。请参见 《系统管理指南》。

file = *file_name*

是转储文件名。该名称不得超过 17 个字符，且必须符合操作系统对文件名的约定。请参见第 363 页的 “转储文件”。

notify = {client | operator_console}
替换缺省的消息显示目标。

在提供操作员终端功能的操作系统上，始终会将卷更改消息发送到运行 Backup Server 的计算机的操作员终端上。使用 client 可将其它 Backup Server 消息发送到启动 dump database 的终端会话。

在不提供操作员终端功能的操作系统 （如 UNIX）上，消息将发送到启动 dump database 的客户端。使用 operator_console 将消息发送到运行 Backup Server 的终端。

syb_tsm::*obj_name*
是调用 *libsyb_tsm.so* 模块的关键字，该模块用于实现 Backup Server 和 Tivoli Storage Manager 之间的通信。

object_name
是 TSM 服务器上备份对象的名称。

示例

示例 1 使用 *dmp_cfg2* 转储配置转储数据库：

```
dump database testdb using config=dmp_cfg2
```

示例 2 使用 *dmp_cfg2* 转储配置转储数据库。在转储操作中创建的存档文件带口令保护：

```
dump database testdb using config=dmp_cfg2
    with passwd='mypass01'
```

注释　口令必须用单引号或双引号引起来。

示例 3 使用 *dmp_cfg2* 转储配置执行数据库转储，明确指定压缩级别 6，从而覆盖在 *dmp_cfg2* 中配置的压缩级别：

```
dump database testdb using config=dmp_cfg2
    with compression=6
```

示例 4 将数据库 pubs2 转储到磁带设备。如果磁带具有 ANSI 磁带标签，则此命令将该转储附加到磁带上已有的文件之后，因为未指定 init 选项：

```
dump database pubs2
    to "/dev/nrmt0"
```

示例 5（仅限 UNIX）使用 REMOTE_BKP_SERVER Backup Server 转储 pubs2 数据库。此命令指定了三个转储设备，因此 Backup Server 会将约数据库的三分之一转储到每个设备。此命令可将转储附加到磁带上现有的文件。 retaindays 选项指定 14 天不得覆盖磁带：

```
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
```

示例 6 init 选项初始化磁带卷，覆盖所有现有文件：

```
dump database pubs2
    to "/dev/nrmt0"
    with init
```

示例 7 在转储完成时回绕转储卷：

```
dump database pubs2
    to "/dev/nrmt0"
    with unload
```

示例 8 （仅限 UNIX）notify 子句将请求卷更改的 Backup Server 消息发送到发起转储请求的客户端，而不是将其发送到缺省位置，即 Backup Server 计算机的主控台：

```
dump database pubs2
    to "/dev/nrmt0"
    with notify = client
```

示例 9 使用压缩级别 4 创建 pubs2 数据库的压缩转储，并保存为名为 *dmp090100.dmp* 的本地文件：

```
dump database pubs2 to
    "compress::4::/opt/bin/Sybase/dumps/dmp090100.dmp"
```

此外，可以使用 compression = *compression_level* 语法以压缩级别 100 创建 pubs2 数据库的压缩转储，并保存为名为 *dmp090100.dmp* 的本地文件：

```
dump database pubs2 to "/opt/bin/Sybase/dumps/dmp090100.dmp"
    with compression = 100
```

示例 10 将 pubs2 数据库转储到名为 “remotemachine” 的远程计算机，并使用压缩级别 4：

```
dump database pubs2 to "/Syb_backup/mydb.db" at remotemachine
    with compression ="4"
```

示例 11 将 pubs2 数据库转储到 TSM 备份对象 “obj1.1”：

```
dump database pubs2 to "syb_tsm::obj1.1"
```

示例 12 使用多个分条将 pubs2 数据库转储到 TSM 备份对象 “obj1.2”：

```
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"
```

示例 13 删除 sales_db1 中的最后一个片段，它是数据库结尾的数据库空洞。

select * 指示数据库结尾有空洞：

```
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).
```

运行 select * 确认是否成功删除了该片段：

```
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
```

用法

- 如果在使用 sp_hidetext 之后执行跨平台的 dump 和 load，则必须手动删除并重新创建所有隐藏对象。
- dump database 在三个阶段中执行。每个阶段完成后，都会有一条进度消息通知您。转储操作完成后，该消息将反映执行期间所做的所有更改，阶段 3 中进行的更改除外。
- 表 1-19 描述了用于备份数据库的命令和系统过程：

表 1-19: 用于备份数据库和日志的命令

目的	使用
对整个数据库进行例行转储，包括事务日志。	dump database
对事务日志进行例行转储，然后截断不活动的部分。如果 dump transaction 与 dump database 同时运行，则不会截断事务日志的不活动部分。	dump transaction
在数据库设备发生故障后转储事务日志。	dump transaction with no_truncate
截断日志而不进行备份，然后复制整个数据库。	dump transaction with truncate_only dump database
在常用方法因日志空间不足而失败后截断日志，然后复制整个数据库。	dump transaction with no_log dump database
响应 Backup Server 卷更改消息。	sp_volchanged

限制

- 物理设备的最大文件路径/文件名大小为 127 个字符。
- 如果数据库有代理表，则这些代理表为数据库保存集的一部分。代理表的内容数据不会保存；仅保存和恢复指针。
- 不能使用 with shrink_log 选项删除数据库结尾的空洞。
- 不能在同一磁带中混合 Sybase 转储和非 Sybase 数据 （如 UNIX 存档）。
- 如果数据库具有跨数据库的参照完整性约束，则 sysreferences 系统表存储外部数据库的名称 （而不是 ID 号）。如果您使用 load database 命令更改数据库名称或将其装载到其它服务器上，则 Adaptive Server 无法保证参照完整性。

  警告！ 在转储数据库以用不同的名称装载它或将其转移到另一个 Adaptive Server 上之前，请使用 alter table 删除所有外部参照完整性约束。

- 不能在用户定义的事务中使用 dump database。
- 如果在 dump transaction 正在进行的数据库上发出 dump database 命令，则 dump database 会休眠，直到事务转储完成。
- 使用 1/4 英寸盒式磁带时，每个磁带只能转储一个数据库或事务日志。
- 不能转储具有脱机页的数据库。若要强制脱机页联机，请使用 sp_forceonline_db 或 sp_forceonline_page。

- 在针对跨平台转储和装载运行 dump database 之前，请将数据库转至事务抑制状态：
  a　执行 dbcc checkdb 和 dbcc checkalloc 命令，检验数据库是否在顺利运行。
  b　为了防止在 dump database 过程中其它进程的打开事务进行并发更新，请使用 sp_dboption 命令将数据库模式更改为单用户模式。
  c　使用 sp_flushstats 将统计信息刷新到 systabstats。
  d　等待 10 到 30 秒钟，该时间的长短取决于数据库的大小和活动。
  e　针对数据库运行 checkpoint 以刷新已更新的数据库。
  f　运行 dump database。
- dump transaction 和 load transaction 不允许跨平台使用。
- 对远程 backupserver 执行 dump database 和 load database操作不支持跨平台进行。
- 不能跨平台装载有口令保护的转储文件。
- 如果对已分析的 XML 对象执行 dump database 和 load database，则在完成 load database 之后，必须重新分析该文本。
- Adaptive Server 不能转换存储为 binary、 varbinary 或 image 列的嵌入数据结构。
- 不允许对 master database 执行跨平台的 load database 操作。
- 执行 load database 之后，存储过程和其它编译对象在初次执行时需要从 syscomments 中的 SQL 文本重新编译。

  如果您无权从文本重新进行编译，则拥有该权限的人必须使用 dbcc upgrade_object 从文本重新进行编译才能升级对象。

注释　如果将 master 数据库中的 syslogins 系统表内的登录记录从 Solaris 迁移到 Linux，则可以执行 bcp -c 字符格式批量复制，Solaris 中的登录口令就将在 Linux 上兼容。对于所有其它组合和平台，由于口令不兼容，因此必须重新创建登录记录。

安排转储

- Adaptive Server 数据库转储是动态的 – 可以在数据库活动时发生。不过，它们可能会轻微地减慢系统，所以最好在数据库更新不频繁时使用 dump database。

- 定期且经常备份 master 数据库。除了定期备份，还要在每次发出 create database、alter database 和 disk init 命令之后转储 master 数据库。
- 每次对 model 数据库进行更改后都要进行备份。
- 创建数据库后都要立即使用 dump database 复制整个数据库。在运行 dump database 前，不能在新数据库上运行 dump transaction。
- 每次添加或删除跨数据库约束或者删除包含跨数据库约束的表时，都请转储上述两个受影响的数据库。

警告！装载这些数据库的早期转储可能会导致数据库损坏。

- 制定一份用于定期备份用户数据库及其事务日志的日程表。
- 使用阈值以自动进行备份过程。要利用 Adaptive Server 的最后机会阈值，请创建日志段与数据段不在同一设备上的用户数据库。有关阈值的详细信息，请参见 《系统管理指南》。

转储系统数据库

- master、 model 和 sybsystemprocs 数据库的事务日志没有单独的段。应使用 dump transaction with truncate_only 清除日志，然后使用 dump database 备份数据库。
- 一旦出现影响 master 数据库的故障，将需要 master 数据库的备份来执行恢复过程。有关备份和恢复 master 数据库的分步指导，请参见《系统管理指南》。
- 如果使用可移动介质进行备份，则整个 master 数据库必须容纳于单个卷中，除非还有另外的 Adaptive Server 可响应卷更改消息。

指定转储设备

- 可以将转储设备指定为文字、局部变量或存储过程参数。
- 不能转储到空设备 （UNIX 上为 */dev/null*）。
- 磁带和磁盘设备支持转储到多个分条。仅磁带设备支持在设备上放置多个转储。
- 可将本地转储设备指定为：
  - sysdevices 系统表中的逻辑设备名称
  - 绝对路径名
  - 相对路径名

  Backup Server 使用 Adaptive Server 中的当前工作目录解析相对路径名。

- 通过网络转储时，必须指定转储设备的绝对路径名。路径名在运行 Backup Server 的计算机上必须是有效的。如果名称包括任何非字母、数字或下划线 (_) 的字符，都必须用引号将它引起来。
- 转储设备上的所有权和权限问题可能会干扰 dump 命令的使用。sp_addumpdevice 将设备添加到系统表，但不保证可以向该设备转储或将文件创建为转储设备。
- 可以同时运行多个转储 （或装载），只要它们使用不同的转储设备即可。
- 如果设备文件已经存在，Backup Server 会覆盖它；但不会截断它。例如，假如将数据库转储到设备文件，且设备文件变为 10MB。如果从此数据库到此设备的下一个转储较小，则设备文件仍然是 10MB。

转储压缩数据

- 不能在一个平台上创建压缩表的转储并将该转储装载到另一个平台上。
- 压缩数据直接转储到某个存档位置。
- 在包含任何形式的压缩或解压缩行的压缩表上执行的 create index 命令在 load transaction 期间将会完全恢复。

确定磁带设备特性

- 如果发出了不带 init 限定符的 dump 命令，且 Backup Server 不能确定此设备类型，则 dump 命令将失败。请参见 《系统管理指南》。

Backup Server

- Backup Server 必须与 Adaptive Server 在同一台计算机上运行。Backup Server 必须在 master..sysservers 表中列出。此条目是在安装或升级期间创建的；请勿将其删除。
- 如果备份设备位于另一台计算机上以通过网络进行转储，那么在该远程计算机上还必须安装 Backup Server。

转储文件

- 使用 init 选项转储数据库会覆盖磁带或磁盘上的任何现有文件。
- 如果执行两次或多次对一个磁带设备的转储，并且对这些转储使用相同的文件名 （用 FILENAME 参数指定），则 Adaptive Server 将把第二个转储附加到存档设备。您将无法恢复第二个转储，因为 Adaptive Server 将查找具有指定文件名的转储映像的第一个实例，并恢复这个映像。 Adaptive Server 并不搜索后续的同名转储映像。

- Backup Server 将转储文件名发送到由 with notify 子句指定的位置。在存放备份磁带之前，操作员应利用数据库名、文件名、日期和其它相关信息为磁带制作标签。装载无标识标签的磁带时，可使用 with headeronly 和 with listonly 选项确定其内容。

文件名和存档名

- 转储文件名称标识转储的数据库及转储时间。不过，在语法中，根据转储到磁盘或 UNIX 磁带的不同， *file_name* 具有不同的意义：

  ```
  file = file_name
  ```

  对于到磁盘的转储，磁盘文件的路径名也是其文件名。

  对于到 UNIX 磁带的转储，路径名并非文件名。文件交换的 ANSI 标准格式在 HDR1 标签中包含有文件名字段。对于遵循 ANSI 规范的磁带，标签中的这个字段标识的是文件名。 ANSI 规范仅适用于磁带的标签，不适用于磁盘文件。

  这就造成了两个问题：

  - UNIX 不遵循磁带文件名的 ANSI 约定。 UNIX 认为磁带的数据是不带标签的。虽然数据可以分成多个文件，但这些文件没有名称。
  - 在 Backup Server 中，ANSI 磁带标签用于存储关于存档的信息，否定 ANSI 意义。因此，磁盘文件也有 ANSI 标签，因为那里存储了档案名。

  filename 的含义随所执行的转储的类型而变。例如，在以下语法中：

```
dump database database_name to 'filename' with file='filename'
```

  - 第一个 *filename* 指输入的显示此文件的路径名。
  - 第二个 *filename* 实际上是档案名，也就是存储在档案的 HDR1 标签中的名称，用户可以用 dump 或 load 命令的 file=*filename* 参数指定。

  指定了档案名时，服务器在装载数据库时使用此名称查找选择的档案。

  如果没有指定档案名，服务器会装载它所遇到的第一个档案。

  在上述两种情况下， file='*archivename*' 确定了存储在 HDR1 标签中的名称，后续 load 将使用此名称验证其正在查找的数据是正确的。

  如果不指定档案名，dump 会创建一个名称，而 load 则使用它遇到的第一个名称。

to '*filename*' 子句中的 *filename* 的意义根据您执行磁盘转储还是磁带转储而不同：

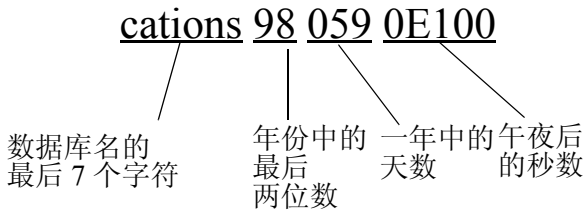
- 如果是转储到磁带，'*filename*' 是磁带设备的名称。
- 如果是转储到磁盘，它就是磁盘文件的名称。

如果进行的是磁盘转储，而 ‘*filename*’ 并非完整路径，则会在文件名前加上服务器的当前工作目录。

- 如果正转储到磁盘而没有指定文件名，则 Backup Server 会通过并置以下内容创建缺省的文件名：
  - 数据库名的最后七个字符
  - 两位数的年份数值
  - 三位数表示的一年中的一天 (1 – 366)
  - 创建转储文件时的十六进制编码时间

  例如，文件 *cations980590E100* 包含 1998 年第 59 天生成的 publications 数据库的一个副本：

图 1-4：对于转储到磁带的数据库的文件命名约定



卷名

- 转储卷根据 ANSI 磁带标签标准标注。标签信息应包含逻辑卷号和设备在分条集中的位置。

- 装载过程中， Backup Server 使用磁带标签来检验卷的安装顺序是否正确。这使您可以从比转储时所用设备数更少的设备进行装载。

注释 通过网络进行转储和装载时，必须为每个操作指定相同数量的分条设备。

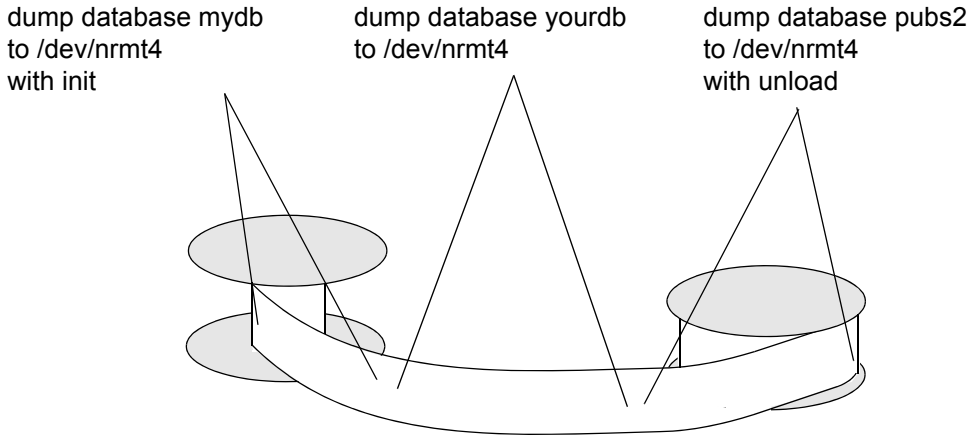
更改转储卷

- （UNIX 系统）在磁带容量满后请求更改卷。安装另一个卷后，操作员通过在可以与 Backup Server 通信的任何 Adaptive Server 上执行 sp_volchanged 通知 Backup Server。
- 如果 Backup Server 检测到当前装入的卷有问题，它会通过向客户机或其操作员主控台发送消息来请求更改卷。操作员可用 sp_volchanged 系统过程响应这些消息。

附加到或覆盖卷

- 缺省情况下 (noinit)， Backup Server 将在同一磁带卷上写入连续转储，这可以有效利用高容量的磁带介质。数据将被添加到最后一个磁带结束标记的后面。新的转储只能附加到多卷转储的最后一个卷上。在写入磁带前， Backup Server 将检验第一个文件是否还处于有效期内。如果磁带包含非 Sybase 数据，则 Backup Server 会予以拒绝，以避免破坏具有潜在价值的信息。
- 使用 init 选项重新初始化卷。如果指定 init，则 Backup Server 将覆盖现有的任何内容，即使磁带包含非 Sybase 数据、第一个文件还没有过期，或者磁带有 ANSI 访问限制。
- 图 1-5 演示如何使用下列方法将三个数据库转储到单个卷：
  - init 初始化磁带以存放第一个转储
  - noinit （缺省值）附加后续转储
  - unload 在最后一个转储之后回绕并卸载磁带

图 1-5：将多个数据库转储到同一卷上



从 32 位 OS 转储到 64 位 OS

从 32 位版本的 Adaptive Server 进行的数据库转储与同一平台的 64 位版本的 Adaptive Server 完全兼容，反之亦然。

转储其设备被镜像的数据库

- 在 dump database 开始时，Adaptive Server 会将所有数据库和日志设备的主设备名传递给 Backup Server。如果主设备已取消镜像， Adaptive Server 会改为传递辅助设备名。如果任何指定的设备在 Backup Server 完成数据传输前发生故障，则 Adaptive Server 将中止转储。
- 如果用户在 dump database 正在进行时试图取消任何指定数据库设备的镜像，则 Adaptive Server 会显示一条消息。执行 disk unmirror 命令的用户可中止转储或将 disk unmirror 推迟到转储完成后执行。

性能注释

由于在 dataserver 中设计索引的目的是提供最佳搜索路径，因此索引行的排序原则是快速访问表中的数据行。包含行标识符 (RID) 的索引行被视为二进制，可用于快速访问用户表。

在同一体系结构平台中，索引行的顺序保持有效，并且选择条件的搜索顺序采用正常途径。但是，当在不同的体系结构间转换索引行时，优化的执行顺序无效，这会导致跨平台转储和装载期间用户表上的索引无效。

装载来自不同体系结构的数据库转储 （例如从大型平台转储到小型平台）时，某些索引将被标记为可疑：

- APL 表上的非聚簇索引。
- DOL 表上的聚簇索引。

- DOL 表上的非聚簇索引。

要修复目标系统上的索引，在从不同的体系结构转储装载之后，您可以：

- 删除并重新创建所有索引，或者，
- 使用 sp_post_xpload。请参见 《参考手册：过程》中的 “系统过程”。

在大表上重新创建索引会比较耗时。利用 sp_post_xpload，只需一条命令即可验证索引、删除无效索引以及重新创建删除的索引。

使用 sp_post_xpload 可能比逐个删除并重新创建索引更为耗时。对于大小超过 10GB 的数据库， Sybase 建议您使用删除并重新创建索引的方法。

存档数据库的压缩转储

要对存档数据库使用压缩转储：

- 使用 dump database 或 dump tran 命令的 with compression = <*compression level*> 选项创建压缩转储。
- 创建内存池以便访问存档数据库。

注释 使用 “compress::”生成的转储无法装载到存档数据库中。因此，本章中提到的任何压缩都是指使用 with compression = <compression level> 选项生成的转储。

对传统数据库使用此压缩选项进行转储不存在兼容性问题。

压缩转储的兼容性问题

利用 with compression = *compression_level* 选项生成的压缩转储的格式已更改。 Backup Server 15.0 ESD #2 版和更高版本会生成不同于较早版本格式的压缩转储。因此：

- 使用 Backup Server 15.0 ESD #2 版及更高版本创建的压缩转储只能装载到使用 Backup Server 15.0 ESD #2 版或更高版本的 15.0 ESD #2 之前的安装中。
- 如果使用的是 15.0 ESD #2 之前的安装，且希望将您的转储用于存档数据库，请使用 Backup Server 15.0 ESD #2 版或更高版本来创建压缩数据库转储。

  注释 可以将 15.0 ESD #2 Backup Server 用于转储和装载。

加密列和 *dump database*

dump 和 load 用于处理加密列的密文，以确保加密列中的数据在磁盘上仍保持加密状态。

dump 和 load 与整个数据库有关。缺省密钥和在同一数据库中创建的密钥将同与它们相关的数据一起转储和装载。

如果密钥与它们加密的列位于单独的数据库中， Sybase 建议：

- 当转储包含加密列的数据库时，还应转储创建密钥的数据库。如果自上次转储以后添加了新的密钥，则必须执行此操作。
- 当转储包含加密密钥的数据库时，应转储包含用该密钥加密的列的所有数据库。这样可保持加密的数据与可用密钥同步。
- 在装载包含加密密钥的数据库和包含加密列的数据库后，同时将两个数据库联机。

由于加密列对密钥数据库具有元数据依赖性，因此如果要将密钥数据库装载为具有其它名称的数据库，请执行下列步骤 （如果您的数据与密钥存储在同一数据库中，则不需要执行下列步骤）：

1　在转储包含加密列的数据库之前，使用 alter table 解密数据。

2　转储包含密钥和加密列的数据库。

3　装载数据库之后，通过 alter table 用新命名数据库中的密钥对数据重新加密。

加密密钥和加密列之间的一致性问题类似于跨数据库参照完整性的一致性问题。请参见 《系统管理指南》中的 “跨数据库约束和装载数据库”。

dump database 和对 Tivoli Storage Manager 的支持

有关您的站点支持 TSM 时创建备份的详细信息，请参见 Tivoli Storage Manager 的文档。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 dump database 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是数据库的所有者，或对数据库拥有 dump database 特权或 own database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库的所有者或具有以下任一角色的用户： • sa_role 或 • replication_role 或 • oper_role

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
34	dump	dump database	• *角色* – 当前活动角色 • *关键字或选项* – NULL • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

文档 《系统管理指南》中的 “备份和恢复用户数据库”。 .

命令 dump transaction、 load database、 load transaction.

系统过程 sp_addthreshold、 sp_addumpdevice、 sp_dropdevice、 sp_dropthreshold、 sp_helpdb、 sp_helpdevice、 sp_helpthreshold、 sp_hidetext、 sp_logdevice、 sp_spaceused、 sp_volchanged.

dump transaction

说明

如果 dump transaction 命令并未与另一条 dump database 命令同时运行，则制作事务日志的副本并删除不活动的日志部分。

有关您的站点许可使用 Tivoli 时的 dump tranaction 语法，请参见 Tivoli Storage Manager (TSM) 语法。

语法

进行例行日志转储：

```
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}]
```

截断日志而不生成备份副本：

```
dump tran[saction] database_name
    with truncate_only
```

截断已达到容量上限的日志。*仅在不得以的情况下才应使用这种方法，因为您将失去日志的内容*：

```
dump tran[saction] database_name
    with no_log
```

在某个数据库设备出现故障后备份日志：

```
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 提供备份服务时复制事务日志：

```
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]
    } ]
```

基于配置文件中指定的设置来转储事务：

```
dump transaction database_name
    using config = configuration_name
    [with {
        verify[ = header | full]
        }]
```

参数

database_name

是您从中复制数据的数据库的名称。可以将数据库名以文字、局部变量或参数形式指定给一个存储过程。

configuration_name

是唯一的转储配置名称。转储配置中指定的参数值用于执行转储操作。

命令中明确指定的任何其它参数均可覆盖转储配置所指定的值。

如果使用转储配置，则不能将分条目录指定为命令的参数。 Adaptive Server 在转储配置指定的分条目录中创建转储文件。转储文件名使用此约定：

Database_Name.Dump_Type.Date-Timestamp.StripeID

compress::*compression_level*

是 0 到 9 之间的数字、100 或 101。对于一位数的压缩级别，0 表示不进行压缩，而 9 表示最高级别的压缩。压缩级别 100 和 101 提供更加快捷、更加有效的压缩模式，其中 100 提供更加快捷的压缩，而 101 提供程度更高的压缩。如果不指定 *compression_level*，Adaptive Server 将不会压缩转储。

有关 compress 选项的详细信息，请参见 《系统管理指南》中的 “备份和恢复用户数据库”。

注释　compression = *compress_level* 选项允许您在本地和远程计算机上压缩转储文件，这不同于 compress::*compression_level* 选项，后者只能在本地计算机上压缩转储文件。

从 Adaptive Server 15.0 版开始，Sybase 支持并建议使用本机 compression = *compression_level* 语法。

---

truncate_only

删除日志不活动的部分而不制作备份副本。该选项在日志段与数据段位于同一设备的数据库上使用。不要指定转储设备或 Backup Server 名称。

no_log

删除日志的不活动部分，而不制作备份副本，也不在事务日志中记录过程。仅在用完日志空间，无法运行常规 dump transaction 命令时才应使用 no_log。仅在不得已的情况下才应使用 no_log，且应仅在 dump transaction with truncate_only 失败时才使用。

to *stripe_device*

是数据转储到的设备。有关指定转储设备时使用何种形式的信息，请参见第 362 页的 “指定转储设备”。

at *backup_server_name*

是 Backup Server 的名称。如果是转储到缺省 Backup Server，请不要指定此参数。仅当通过网络转储到远程 Backup Server 时才指定此参数。使用此选项可指定多达 32 个不同的远程 Backup Server。通过网络进行转储时，请指定在转储设备附加到的计算机上运行的远程 Backup Server 的 *network name*。对于使用接口文件的平台，*backup_server_name* 必须出现在接口文件中。

density = *density_value*

替换磁带设备的缺省密度。有效密度为 800、 1600、 6250、 6666、 10000 和 38000。并不是所有的值对于每个磁带驱动器都有效；请使用适合您的磁带驱动器的正确密度。

blocksize = *number_bytes*

替换转储设备的缺省块大小。块大小必须至少为一个数据库页 （对于大多数系统为 2048 字节），且必须为数据库页大小的整数倍。

注释 注意：应尽可能使用缺省块大小，它最适合您的系统。

capacity = *number_kilobytes*

是设备可写入单个磁带卷的最大数据量。容量至少应为 5 个数据库页，但应略小于设备的推荐容量。

计算容量的一般规则是使用设备制造商给出的设备最大容量的 70%，并留出 30% 的容量用于记录间隙和磁带标志之类的开销。此规则在多数情况下适用，但可能由于各供应商和设备的开销存在差异而不能全部适用。

在不能可靠检测到磁带结束标志的 UNIX 平台上，必须指明可转储到磁带的千字节数。对于作为物理路径名指定的转储设备，必须提供 capacity。如果将转储设备作为逻辑设备名指定，那么，除非指定容量。否则 Backup Server 使用存储在 sysdevices 系统表中的 *size* 参数。

compression = *compress_level*

是 0 到 9 之间的数字、100 或 101。对于一位数的压缩级别，0 表示不进行压缩，而 9 表示最高级别的压缩。压缩级别 100 和 101 提供更加快捷、更加有效的压缩模式，其中 100 提供更加快捷的压缩，而 101 提供程度更高的压缩。如果不指定 *compression_level*，Adaptive Server 将不会压缩转储。

注释　Sybase 建议首选使用本机 "compression = *compress_level*" 选项，旧选项 "compress::*compression_level*" 其次。该本机选项允许压缩本地和远程转储，并且它创建的转储将在装载期间描述其自身的压缩级别。保留旧选项是为了与以前的应用程序兼容。

---

dumpvolume = *volume_name*

建立指派给卷的名称。 *volume_name* 的最大长度为 6 个字符。覆盖现有转储、转储到新磁带或者转储到内容不可识别的磁带时， Backup Server 会在 ANSI 磁带标签中写入 *volume_name*。 load transaction 命令会检查标签，如果装载了错误的卷，则会生成错误消息。

stripe on *stripe_device*

是附加的转储设备。最多可以使用 32 个设备，其中包括在 to *stripe_device* 子句中指定的设备。Backup Server 将日志分成几个大致相等的部分，并将每个部分发送到不同的设备。转储是在所有设备上同时进行的，从而减少了所需的时间和卷的更改数量。请参见第 362 页的 “指定转储设备”。

dismount | nodismount

（在支持逻辑卸下的平台上）确定磁带是否保持装入状态。缺省情况下，转储完成时将卸下用于转储的全部磁带。使用 nodismount 命令可使磁带供其它转储或装载使用。

nounload | unload

确定转储完成后是否回绕磁带。缺省情况下磁带不会回绕，从而使您可以向同一磁带卷进行其它转储。请为要添加到多转储卷的最后一个转储文件指定 unload。这样，在转储完成后就会回绕并卸载磁带。

retaindays = *number_days*

（在 UNIX 平台上）指定天数， Backup Server 保护转储在此天数内不被覆盖。如果试图在过期前覆盖转储，则在覆盖未过期卷前，Backup Server 会要求进行确认。

注释 此选项对于磁盘、 1/4 英寸盒式磁带和单文件介质是有意义的。在多文件介质上，此选项对除第一卷外的所有卷都是有意义的。

对于可以立即覆盖的转储，其 *number_days* 值必须是正整数或 0。如果不指定 retaindays 值，Backup Server 会使用由 sp_configure 设定的服务器范围的 tape retention in days 值。

noinit | init

确定是将转储附加到现有的转储文件还是重新初始化 （覆盖）磁带卷。缺省情况下， Adaptive Server 将转储附加在最后一个磁带结束标记之后，从而使您可以向同一个卷转储其它数据库。新的转储只能附加到多卷转储的最后一个卷上。对转储到磁带的第一个数据库使用 init，以覆盖其内容。

在需要 Backup Server 存储或更新磁带配置文件中的磁带设备特性时，可使用 init 命令。请参见 《系统管理指南》。

file = *file_name*

是转储文件名。该名称不得超过 17 个字符，且必须符合操作系统对文件名的约定。如果不指定文件名，则 Backup Server 创建缺省文件名。请参见第 363 页的 “转储文件”。

no_truncate

使用指向 master 数据库中事务日志的指针转储事务日志，即使无法访问包含数据库中数据段的磁盘。当事务日志驻留在未损坏的设备上，而 master 数据库和用户数据库驻留在不同的物理设备上时， with no_truncate 选项提供最新的日志恢复。

如果将 dump tran 和 no_truncate 一起使用，则前者后面应该接 dump database，而不是另一个 dump tran。如果装载使用 no_truncate 选项生成的转储，则 Adaptive Server 将阻止您装载任何后续转储。

notify = {client | operator_console}

替换缺省的消息显示目标。

- 在提供操作员终端功能的操作系统上，始终会将卷更改消息发送到运行 Backup Server 的计算机的操作员终端上。使用 client 可将其它 Backup Server 消息发送到启动 dump database 的终端会话。
- 在不提供操作员终端功能的操作系统（如 UNIX）上，消息将发送到启动 dump database 的客户端。使用 operator_console 将消息发送到运行 Backup Server 的终端。

with standby_access

指定仅转储完成的事务。转储继续到它可发现的事务刚刚完成、没有其它活动事务的最远的点。

syb_tsm

是调用 libsyb_tsm.so 模块的关键字，该模块用于实现 Backup Server 和 TSM 之间的通信。

object_name

是 TSM 服务器上备份对象的名称。

configuration_name

是唯一的转储配置名称。转储配置中指定的参数值用于执行转储操作。

如果命令中明确指定了其它参数，则其会覆盖转储配置所指定的参数值。

如果使用转储配置，则不能将分条目录指定为命令的参数。 Adaptive Server 在转储配置指定的分条目录中创建转储文件。使用以下约定命名转储文件：

Database Name.Dump Type.Date-Timestamp.StripeID

示例

示例 1 将事务日志转储到磁带，将其附加到磁带上的文件，因为没有指定 init 选项：

```
dump transaction pubs2
    to "/dev/nrmt0"
```

示例 2 使用 Backup Server REMOTE_BKP_SERVER 转储 mydb 数据库的事务日志。Backup Server 向两个设备中的每一个转储约一半日志。init 选项覆盖磁带上的所有现有文件。 retaindays 选项指定 14 天不得覆盖磁带：

```
dump transaction mydb
    to "/dev/nrmt4" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
with init, retaindays = 14
```

示例 3 将 inventory_db 事务日志文件中完成的事务转储到设备 dev1 中：

```
dump tran inventory_db to dev1 with standby_access
```

示例 4 采用压缩级别 100 将 pubs2 数据库的事务日志转储到 TSM 备份对象 “demo2.2”中。

```
dump transaction pubs2 to "syb_tsm::demo2.2"
    with compression = 100
```

示例 5 使用 “dmp_cfg2”配置转储数据库。在转储过程中创建的存档文件带口令保护：

```
dump transaction testdb using config = 'dmp_cfg2'
   with passwd = 'my_pass01'
go
```

示例 6 使用 “dmp_cfg2”配置通过压缩级别 6 （也就是覆盖 “*dmp_cfg2*”中指定的压缩级别）转储数据库：

```
dump transaction testdb using config = 'dmp_cfg2'
   with compression = 6
go
```

用法

- 如果在使用 sp_hidetext 之后执行跨平台的 dump 和 load，则必须手动删除并重新创建所有隐藏对象。
- 表 1-20 描述了用于备份数据库和日志的命令和系统过程。

表 1-20: 用于备份数据库和日志的命令

目的	使用
对整个数据库进行例行转储，包括事务日志。	dump database
对事务日志进行例行转储，然后截断不活动的部分。如果 dump transaction 与 dump database 同时运行，则不会截断日志的不活动部分。	dump transaction
在数据库设备发生故障后转储事务日志。	dump transaction with no_truncate
截断日志而不进行备份。 然后复制整个数据库。	dump transaction with truncate_only dump database
在常规方法因日志空间不足而失败时截断日志。 然后复制整个数据库。	dump transaction with no_log dump database
响应 Backup Server 卷更改消息。	sp_volchanged

限制

- 物理设备的最大文件路径/文件名大小为 127 个字符。
- 不能转储到空设备 （UNIX 上为 */dev/null*）。
- 不能在事务中使用 dump transaction 命令。
- 使用 1/4 英寸盒式磁带时，每个磁带只能转储一个数据库或事务日志。

- 必须先完全转储新创建的数据库，然后才能运行 dump transaction *database_name* to。
- 在数据库中执行未记录的操作后，将不能使用 dump transaction *database_name* to。
- 启用 trunc log on chkpt 数据库选项时，或者启用 select into/bulk copy/pllsort 并用 select into、快速批量复制操作、缺省未记录的 writetext 操作或并行排序进行最小记录更改后，不能发出 dump the transaction log，而应使用 dump database。

警告！请勿用 delete、update 或 insert 命令修改日志表 syslogs。

- 如果数据库不具有在独立于数据段的设备上的日志段，则不能使用 dump transaction 复制和截断日志。
- 如果用户或阈值过程在正在运行 dump database 或另一个 dump transaction 的数据库上发出一个 dump transaction 命令，则第二个命令将休眠，直至第一个命令完成。
- 要恢复数据库，请使用 load database 装载最近的数据库转储，然后使用 load transaction 按转储的顺序装载每个后续事务日志转储。
- 每次添加或删除跨数据库约束或者删除包含跨数据库约束的表时，都请转储上述两个受影响的数据库。

警告！装载这些数据库的早期转储可能会导致数据库损坏。

- 不能在同一磁带中混合 Sybase 转储和非 Sybase 数据 （如 UNIX 存档）。
- 在包含脱机页的数据库上不能使用 with no_log 或 with truncate_only 转储事务。

使用 *with no_truncate* 选项的限制

正常情况下，如果执行以下操作， Adaptive Server 会返回错误消息：

- 完全转储新创建的数据库之前就运行 dump transaction *database_name* to：

```
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.
```

- 在数据库中执行最少日志记录的操作后，使用 dump transaction *database_name* to：

  ```
  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.
  ```

  有关详细信息，请参见第 385 页的 “完全可恢复的 DDL 和转储事务”。

- 在执行 dump transaction with truncate_only 后，使用dump transaction *database_name* to：

  ```
  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.
  ```

然而，如果在 dump transaction *database_name* to *dump_file* 命令中使用 with no_truncate 选项，则 Adaptive Server 不会对数据库执行检查，因此不会返回上述任何错误消息。 Adaptive Server 假定您的数据库丢失了一些数据 （例如，从发生故障的磁盘），因此不可访问。

然而，尝试装载事务时会收到错误消息。 load transaction 过程可能会失败，并显示以下错误消息：

```
Specified file 'dump device' is out of sequence.Current
timestamp is <X> while dump was from <Y>.
```

设备出现故障后复制日志

- 设备发生故障后，请使用 dump transaction with no_truncate 复制日志，而不截断它。仅当事务日志在一个单独的段上，且 master 数据库可访问的情况下，才可使用此选项。
- 由 dump transaction with no_truncate 创建的备份是日志的最新转储。恢复数据库时，最后装载该转储。

转储不带有独立日志段的数据库

- 如果数据库没有日志段在独立于数据段的设备上，请使用 dump transaction with truncate_only 从日志中删除已提交的事务，而不备份其副本。

警告！ dump transaction with truncate_only 没有提供恢复数据库的方式。应及早运行 dump database 以确保可恢复性。

- 可在 master、 model 和 sybsystemprocs 数据库上使用 with truncate_only，这些数据库的日志段和数据段在同一个设备上。
- 也可以在将事务日志和数据存储在同一个设备中的极小数据库上使用 with truncate_only。
- 任务关键用户数据库应该有日志段在独立于数据段的设备上。请使用 create database 的 log on 子句创建带有单独日志段的数据库，或使用 alter database 和 sp_logdevice 将日志传送到单独的设备上。

只转储完成的事务

- 使用 with standby_access 选项转储事务日志以将其载入用作数据库热备份服务器的服务器中。
- 使用 with standby_access 转储事务日志时，转储继续到日志中所有以前事务都已完成且没有属于打开事务的记录的最远点。
- 在您依次装载两个或更多事务日志，并希望在两次装载之间数据库保持联机状态的任何情况下，都必须使用 dump tran[saction]...with standby_access。
- 在装载使用 with standby_access 选项生成的转储之后，请使用带有 for standby_access 选项的 online database 命令使数据库可访问。

警告！ 如果某一事务日志包含打开的事务，并且您未使用 with standby_access 选项转储该事务，则 Adaptive Server 不会允许您装载该日志、使数据库联机，然后装载后续事务转储。如果要装载一系列事务转储，则只有在装载最初用 with standby_access 进行转储的事务或装载整个事务系列之后才能使数据库联机。

不带日志转储

警告！ 在转储事务日志的常规方法（dump transaction 或 dump transaction with truncate_only）因为日志空间不足而失败，不得已时再使用 dump transaction with no_log。dump transaction with no_log 没有提供恢复数据库的方法。应及早运行 dump database 以确保可恢复性。

- dump transaction...with no_log 截断日志，而不记录转储事务事件。因为未复制数据，所以它只需要数据库的名称。
- 每次使用 dump transaction...with no_log 都被视为一个错误并记录在 Adaptive Server 错误日志中。

- 如果您创建了日志段与数据段不在同一个设备上的数据库，编写了足以频繁转储事务日志的最后机会阈值过程，并且为日志和数据库分配了足够的空间，则不必使用 with no_log。如果必须使用 with no_log，需要增加转储频率和日志空间量。

安排转储

- 事务日志转储是动态的 – 可以在数据库活动时发生。它们可能会轻微地减慢系统，所以请在数据库更新不频繁时进行转储。
- 制定一份用于定期备份用户数据库及其事务日志的日程表。
- dump transaction 比 dump database 使用的存储空间更小且花费的时间也更短。通常，事务日志转储比数据库转储更频繁。

使用阈值以自动执行 *dump transaction*

- 使用阈值以自动进行备份过程。要利用 Adaptive Server 的最后机会阈值，请创建日志段与数据段不在同一设备上的用户数据库。
- 当日志段上的空间低于最后机会阈值时， Adaptive Server 执行最后机会阈值过程。在最后机会阈值过程中包括 dump transaction 命令有助于避免用尽日志空间。请参见 sp_thresholdaction。
- 可以使用 sp_addthreshold 添加第二个阈值以监控日志空间。有关阈值的详细信息，请参见 《系统管理指南》。

指定转储设备

- 可以将转储设备指定为文字、局部变量或存储过程参数。
- 可将本地转储设备指定为：
  - sysdevices 系统表中的逻辑设备名称
  - 绝对路径名
  - 相对路径名

  Backup Server 使用 Adaptive Server 中的当前工作目录解析相对路径名。
- 磁带和磁盘设备支持转储到多个分条。仅磁带设备支持在设备上放置多个转储。
- 当通过网络转储时，请指定转储设备的绝对路径名。路径名在运行 Backup Server 的计算机上必须是有效的。如果名称包括任何非字母、数字或下划线 (_) 的字符，都必须用引号将它引起来。
- 转储设备上的所有权和权限问题可能会干扰 dump 命令的使用。sp_addumpdevice 将设备添加到系统表，但不保证可以向该设备转储或将文件创建为转储设备。

- 可以同时运行多个转储 （或装载），只要它们使用不同转储设备即可。

确定磁带设备特性

如果发出了不带 init 限定符的 dump transaction 命令，且 Backup Server 不能确定此设备类型，则 dump transaction 命令将失败。请参见 《系统管理指南》。

Backup Server

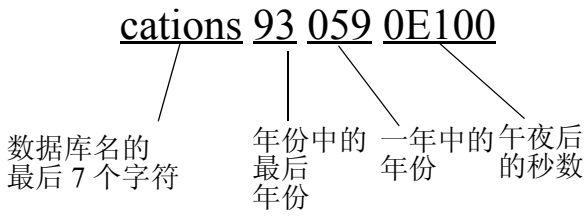
- Backup Server 必须与 Adaptive Server 在同一台计算机上运行。Backup Server 必须在 master..sysservers 表中列出。这个条目是在安装或升级过程中创建的，不得删除。
- 如果备份设备位于另一台计算机上以通过网络进行转储，那么在该远程计算机上还必须安装 Backup Server。

转储文件

- 用 init 选项转储日志将覆盖磁带或磁盘上的任何现有文件。
- 转储文件名标识转储哪个数据库及何时转储。如果不指定文件名，则 Backup Server 通过并置以下部分创建缺省的文件名：
  - 数据库名的最后七个字符
  - 两位数的年份数值
  - 三位数表示的一年中的一天 (1 366)
  - 创建转储文件时的十六进制编码时间

  例如，文件 *cations930590E100* 包含 1993 年第 59 天生成的 publications 数据库的一个副本：

图 1-6：事务日志转储的文件命名约定



- Backup Server 将转储文件名发送到由 with notify 子句指定的位置。在存放备份磁带之前，操作员应利用数据库名、文件名、日期和其它相关信息为磁带制作标签。装载无标识标签的磁带时，可使用 with headeronly 和 with listonly 选项确定其内容。

卷名

- 转储卷根据 ANSI 磁带标签标准标注。标签信息应包含逻辑卷号和设备在分条集中的位置。
- 装载过程中， Backup Server 使用磁带标签来检验卷的安装顺序是否正确。这使您可以从比转储时所用设备数更少的设备进行装载。

注释 通过网络进行转储和装载时，必须为每个操作指定相同数量的分条设备。

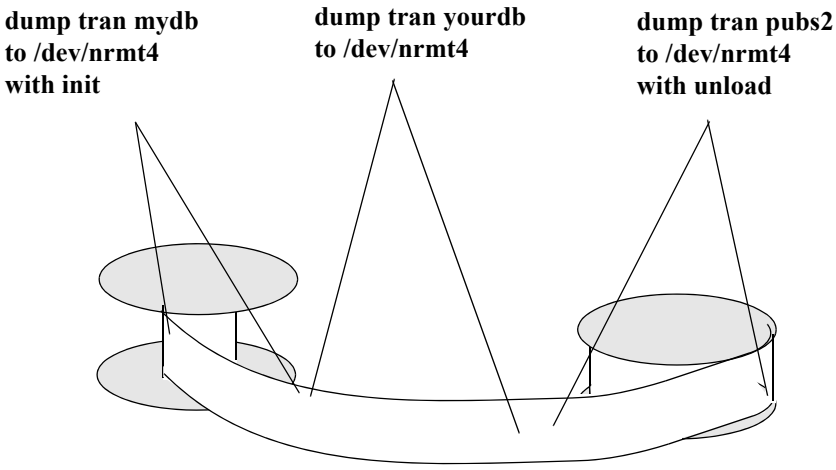
更改转储卷

- （在 UNIX 系统中） Backup Server 在磁带容量满后请求更改卷。装入另一卷后，操作员通过在可以与 Backup Server 通信的任何 Adaptive Server 上执行 sp_volchanged 系统过程来通知 Backup Server。
- 如果 Backup Server 检测到当前装入的卷有问题 （例如，如果装入了错误的卷），它通过向客户机或其操作员主控台发送消息来请求更改卷。操作员可用 sp_volchanged 系统过程响应这些消息。

附加到或覆盖卷

- 缺省情况下 (noinit)， Backup Server 将在同一磁带卷上写入连续转储，这可以有效利用高容量的磁带介质。数据将被添加到最后一个磁带结束标记的后面。新的转储只能附加到多卷转储的最后一个卷上。在写入磁带前， Backup Server 将检验第一个文件是否还处于有效期内。如果磁带包含非 Sybase 数据，则 Backup Server 会予以拒绝，以避免破坏具有潜在价值的信息。
- 使用 init 选项重新初始化卷。如果指定 init，则 Backup Server 将覆盖现有的任何内容，即使磁带包含非 Sybase 数据、第一个文件还没有过期，或者磁带有 ANSI 访问限制。
- 图 1-7 说明了如何将三个事务日志转储到单个卷。请选择下列方法之一：
  - init 初始化磁带以存放第一个转储
  - noinit （缺省值）附加后续转储
  - unload 在最后一个转储之后回绕并卸载磁带

图 1-7：将三个事务日志转储到单个卷



转储存储在镜像设备上的日志

- 在 dump transaction 开始时， Adaptive Server 将每个逻辑日志设备的主设备名传递给 Backup Server。如果主设备已取消镜像， Adaptive Server 会改为传递辅助设备名。如果指定的设备在 Backup Server 完成数据传输之前发生故障，则 Adaptive Server 将中止转储。
- 如果在 dump transaction 正在进行时试图取消指定日志设备的镜像，则 Adaptive Server 会显示一条消息。执行 disk unmirror 命令的用户可以中止转储或将 disk unmirror 推迟到转储完成后执行。
- dump transaction with truncate_only 和 dump transaction with no_log 不使用 Backup Server。当日志设备取消镜像后 （设备故障或 disk unmirror 命令导致），这些命令不受影响。
- dump transaction 只复制日志段。当取消只读数据设备的镜像时 （设备故障或 disk unmirror 命令导致），该命令不受影响。

完全可恢复的 DDL 和转储事务

在低于 15.7 版的 Adaptive Server 中，某些操作是最少日志记录的。由于在最少日志记录的操作后不允许 dump transaction，因此，此限制影响：

- 超大型数据库 (VLDB) 安装的可恢复性和操作可扩展性，其中 dump database 可能十分耗时。

- 数据库的最新可恢复性。即使最少日志记录的操作可完全从服务器故障中恢复，在最后一个成功事务转储后进行的更改也可能会在数据设备损坏或者数据库损坏时丢失。您不能在最少日志记录的操作后使用 dump tran with no_truncate 来转储日志，然后使用转储的事务日志来恢复数据库。
- 您不能使用 dump transaction 将数据库恢复到特定时刻，然后执行 load tran with until_time。

从 Adaptive Server 15.7 开始，您可以使用 dump transaction 来完全恢复在早期版本的 Adaptive Server 中最少日志记录的以下操作：

- select into，包括选择到代理表中
- 需要移动数据的 alter table 命令
- reorg rebuild

在 master 数据库中使用 sp_dboption 可完全记录缺省情况下以最少日志记录模式记录的命令。

dump transaction 和 Tivoli Storage Manager

有关您的站点支持 TSM 时创建备份的详细信息，请参见 Tivoli Storage Manager 的用户文档。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 dump transaction 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是数据库的所有者，或对数据库拥有 dump database 特权或 own database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库的所有者或具有以下任一角色的用户： • sa_role 或 • replication_role 或 • oper_role

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
35	dump	dump transaction	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

文档 《系统管理指南》中的 “备份和恢复用户数据库”。 .

命令 dump database、 load database、 load transaction、 online database.

系统过程 sp_addumpdevice、 sp_dboption、 sp_dropdevice、 sp_helpdevice、 sp_hidetextsp_logdevice、 sp_volchanged.

execute

说明

运行过程或动态执行 Transact-SQL 命令。

语法

```
[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]
```

或

```
exec[ute] ("string" | char_variable
    [+ "string" | char_variable]...)
```

参数

execute | exec

用于执行存储过程或扩展存储过程 (ESP)。如果批处理中有多条语句，则此关键字是必需的。

execute 还用于执行包含 Transact-SQL 的字符串。

@return_status

是一个保存存储过程返回状态的可选整数变量。在 *@return_status* 语句中使用 execute 之前，必须在批处理或存储过程中声明它。

server

是远程服务器的名称。可以在另一台 Adaptive Server 上执行过程，只要拥有使用该服务器的权限和在数据库上执行过程的权限即可。如果指定了服务器名但未指定数据库名，则 Adaptive Server 在缺省数据库中查找过程。

database

是数据库名。如果过程在另一数据库中，指定数据库名。 *database* 的缺省值是当前数据库。可以在另一个数据库中执行过程，只要是该数据库的所有者或具有在该数据库中执行过程的权限即可。

owner

是过程所有者的名称。如果数据库中存在多个同名过程，指定所有者的名称。 *owner* 的缺省值是当前用户。仅当数据库所有者拥有该过程或者是您自己拥有它时，所有者的名称才是可选的。

procedure_name

是用 create procedure 定义的过程的名称。

number

是一个可选的整数，用于将同名的过程组成一组，以便能用一条 drop procedure 语句一并删除它们。同一应用程序中使用的过程经常用此方法分组。例如，如果将与名为 orders 的应用程序一起使用的过程命名为 *orderproc;1*、 *orderproc;2* 等等，则下列语句将删除整个组：

```
drop proc orderproc
```

将过程分组后，就不能单独删除组中的过程。例如，不能执行以下语句：

```
drop procedure orderproc;2
```

parameter_name

是在 create procedure 中定义的该过程的参数名。参数名前面必须有 @ 符号。

如果使用 “*@parameter_name = value*” 形式，则参数名和常量不必以 create procedure 中定义的顺序提供。但是，如果任何参数使用了此形式，则其所有后续参数都必须使用此形式。

value

是该过程的参数值。如果不使用 “*@parameter_name = value*” 形式，则必须以 create procedure 中定义的顺序提供参数值。

@variable

是用来存储返回参数的变量的名称。

output

表示此存储过程将返回一个返回参数。存储过程中匹配的参数也必须已经用关键字 output 创建。

output 关键字可以缩写为 out。

with recompile

强制编译新计划。如果提供的参数不规则或数据已经发生了重大更改时使用该选项。更改后的计划用于后续执行。执行扩展系统过程时，Adaptive Server 忽略此选项。

注释　多次使用 execute procedure with recompile 可能对过程高速缓存性能有不利影响。由于每次使用 with recompile 都会生成新计划，因此如果没有足够的空间可用于新计划，则有用的性能计划可能会被推出高速缓存。

---

string

是包含要执行的 Transact-SQL 命令部分的文字字符串。对用文字字符串提供的字符数没有限制。

char_variable

是提供 Transact-SQL 命令的文本的变量名。

示例

示例 1 这三个语句都以参数值 titles 执行 showind：

```
execute showind titles
exec showind @tabname = titles
```

如果这是批处理或文件中的唯一语句：

```
showind titles
```

示例 2 在远程服务器 GATEWAY 上执行 checkcontract。在 *@retstat* 中存储指示成功还是失败的返回状态：

```
declare @retstat int
execute @retstat = GATEWAY.pubs.dbo.checkcontract
"409-56-4008"
```

示例 3 执行 roy_check，传递三个参数。第三个参数 *@pc* 为 output 参数。执行该过程后，变量 *@percent* 中有返回值：

```
declare @percent int
select @percent = 10
execute roy_check "BU1032", 1050, @pc = @percent output
select Percent = @percent
```

示例 4 如果不提供参数，则该过程显示有关系统表的信息：

```
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
```

示例 5 执行 xp_echo，并传入值 “Hello World!”该扩展存储过程的返回值存储在名为 *result* 的变量中：

```
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
```

示例 6 最后的 execute 命令将字符串值和字符变量并置以发出 Transact-SQL 命令：

```
select name from sysobjects where id=3

declare @tablename char (20)
declare @columname char (20)
```

```
select @tablename="sysobjects"
select @columname="name"
execute ('select ' + @columname + ' from ' + @tablename
+ ' where id=3')
```

示例 7 执行 sp_who：

```
declare @sproc varchar (255)
select @sproc = "sp_who"
execute @sproc
```

用法

- 只要存档数据库内允许有任何引用存档数据库的语句，就可以对存档数据库使用 execute。存储过程内部或外部的事务不允许执行 execute 命令。
- 过程结果随执行过程的数据库的不同而不同。例如，用户定义的系统过程 sp_foo （执行 db_name() 系统函数）返回执行过程的数据库的名称。当从 pubs2 数据库执行时，它返回值 “pubs2”：

  ```
  exec pubs2..sp_foo
  ------------------------------
  pubs2
  (1 row affected, return status = 0)
  ```

  当从 sybsystemprocs 执行时，它返回值 “sybsystemprocs”：

  ```
  exec sybsystemprocs..sp_foo
  ------------------------------
  sybsystemprocs
  (1 row affected, return status = 0)
  ```

- 有两种提供参数的方式 – 按位置或通过使用：

  ```
  @parameter_name = value
  ```

  若使用第二种形式，则不必以 create procedure 中定义的顺序提供参数。

  如果使用 output 关键字并打算在批处理或过程的其它语句中使用返回参数，则此参数值必须作为变量传递。例如：

  ```
  parameter_name = @variable_name
  ```

  执行扩展存储过程时，请按名称或值传递所有参数。在执行 ESP 的单个 execute 命令调用中，不能混合使用按值和按名称传递的参数。

- exec (*@parameter_name*) 的动态 SQL 语句同样有效；但它可能需要更多键击。例如，动态 SQL 命令 exec (@sproc ="7") 将整数值 7 传递给过程，但这也可以使用 exec @sproc 7 来实现。

- 不可将 text、unitext 和 image 列用作存储过程的参数或传递给参数的值。
- 执行为 create procedure 中未定义为返回参数的参数指定 output 的过程会导致错误。
- 不能使用 output 向存储过程传递常量；返回参数需要变量名。必须在执行过程之前声明变量的数据类型并为其赋值。返回参数不可为 text、 unitext 或 image 数据类型。
- 如果语句是批处理的第一条语句，则不必使用关键字 execute。批处理是以自成一行的词 “go”结束的输入文件的段。
- 由于过程的执行计划是在其首次运行时存储的，因此之后的运行时间要比同等独立语句的运行时间短很多。
- 当一个存储过程调用另一个存储过程时，就会发生嵌套。嵌套级别在被调用过程开始执行时递增，并在被调用过程执行完成时递减。创建高速缓存的语句时，嵌套级别也会增加一级。超过 16 层的最大嵌套值将导致事务失败。当前的嵌套级别存储在 *@@nestlevel* 全局变量中。
- 当前 Adaptive Server 使用返回值 0 和 -1 到 -14 之间的值表明存储过程的执行状态。- 15 到 -99 之间的值留做将来使用。有关值的列表，请参见 return。
- 参数不属于事务，所以如果在后来回退的事务中更改了参数，其值并不会还原到它的先前值。返回到调用者的值总是过程返回时的值。
- 如果在存储过程中使用 select *，则该过程不会选取您可能已使用 alter table 添加到表中的任何新列，即使使用了 with recompile 选项也如此。为此，您必须删除并重新创建该存储过程，否则基于 select * 的 insert 可能造成错误结果。即使新添加的列已绑定一个缺省值，insert 对于新添加的列的结果仍为 NULL。

  当删除并重新创建存储过程，或重装数据库时，如果目标表的列定义与 select * 结果不匹配，则将显示错误消息。
- 通过远程过程调用执行的命令不能回退。
- 当 Adaptive Server 执行扩展存储过程时，忽略 with recompile 选项。

动态执行的 Transact-SQL

- 当与 *string* 或 *char_variable* 选项一起使用时，execute 并置提供的字符串和变量，以执行结果 Transact-SQL 命令。这种形式的 execute 命令可用于 SQL 批处理、过程和触发器。

- 不能提供 *string* 和 *char_variable* 选项来执行下列命令：use、exec(*string*)（不是 execute 存储过程）、 connect、 begin transaction、rollback、 commit 和 dbcc。
- *string* 或 *char_variable* 选项的内容不能引用 SQL 批处理或过程中声明的局部变量。
- *string* 和 *char_variable* 选项可以并置以创建新表。在同一个 SQL 批处理或过程中，用 execute 创建的表仅对其它 execute 命令可见。完成 SQL 批处理或过程之后，动态创建的表是持久的，且对其它命令可见。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对于执行用 *string* 或 *char_variable* 选项定义的 Transact-SQL 命令的用户，系统将检查其执行该命令的权限，除非过程是使用执行模式 “dynamic ownership chain”设置的。请参见 proc_role。

对 execute 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是过程的所有者或对此过程拥有 execute 权限的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须对此过程拥有 execute 权限。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
38	exec_procedure	执行过程	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - 所有输入参数 • *代理信息* - set proxy 有效时的初始登录名
39	exec_trigger	执行触发器	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 create procedure、 drop procedure、 return.

系统过程 sp_addextendedproc、 sp_depends、 sp_dropextendedproc、sp_helptext、 sp_procxmode.

fetch

说明

从游标结果集中返回一行或一组行。

语法

```
fetch [next | prior | first | last | absolute
        fetch_offset | relative fetch_offset]
        [from] cursor_name
        [into fetch_target_list]
```

参数

next | prior | first | last | absolute | relative

为指定读取方向的关键字。对于不可滚动的游标，无需指定读取方向。如果指定了读取方向，则可以使用任何其它选项从可滚动游标访问行。使用 absolute 或 relative 时，必须指定 *fetch_offset*。

[from] *cursor_name*

是游标的名称。 from 是可选的。

fetch_offset

指定从特定位置开始的偏移值。当您指定 absolute 或 relative 时，*fetch_offset* 是必需的。 *fetch_offset* 可以是标度为零的有符号数字文字，或者是类型为整数或零标度数字的 Transact-SQL 变量。请参见第 397 页的 “可滚动游标定位规则”。

fetch_target_list

是一个放置游标结果的由逗号分隔的参数或局部变量的列表。这些参数和变量必须在 fetch 之前声明。

示例

示例 1 从由 authors_crsr 游标定义的游标结果集中返回一行信息：

```
fetch authors_crsr
```

示例 2 从由 pubs_crsr 游标定义的游标结果集中将一行信息返回到变量 *@name*、 *@city* 和 *@state* 中：

```
fetch pubs_crsr into @name, @city, @state
```

示例 3 对于可滚动游标，可结合使用数字文字偏移与方向关键字 absolute。本示例中，指定了第 25 行。输入：

```
fetch  absolute 25 from pubs_crsr
    into @name, @city, @state
```

示例 4 要使用表示第 25 行的 Transact-SQL 变量，请输入：

```
declare @offset int
select @offset = 25
fetch absolute @offset from c1
```

用法

限制

- 要使用 fetch，必须先声明游标并将其 open 。

- 可以对存档数据库使用 fetch。
- *cursor_name* 不能是 Transact-SQL 参数或局部变量。
- 对于非滚动游标，不可 fetch 已经读取的行。无法回溯结果集，但可以关闭游标并重新打开以再次创建游标结果集并从头开始。
- Adaptive Server 要求 *fetch_target_list* 中的变量和定义该游标的 select 语句所指定的目标列表表达式之间是一一对应关系。变量或参数的数据类型必须兼容游标结果集中列的数据类型。
- 设置链式事务模式时，如果当前没有活动的事务， Adaptive Server 将隐式地用 fetch 语句启动事务。但是，只有当设置了 close on endtran 选项，并且在最初打开游标的事务结束之后游标仍保持打开时，才会发生此种情况，因为 open 语句也自动启动事务。

游标位置

- 对于不可滚动游标，在读取完所有行之后，游标指向结果集的最后一行。如果再次进行读取， Adaptive Server 将通过 *@@sqlstatus* 和 *@@fetch_status* 全局变量返回警告，变量中的值表明已没有更多数据，并且游标位置移动到结果集之外。不可再从当前游标位置进行更新或删除。
- 如果使用 fetch into，当发生错误时 Adaptive Server 不会前移游标，因为 *fetch_target_list* 中的变量的数目与定义该游标的查询所指定的目标列表表达式的数目不相等。但是，即使发生变量的数据类型与游标结果集中的列的数据类型不兼容的错误，也会前移游标位置。

确定读取的行数

- 可以每次 fetch 一行或多行。使用 set 命令的 cursor rows 选项来指定要获取的行数。

获取有关存取的信息

- *@@sqlstatus* 全局变量保存执行 fetch 语句产生的状态信息 （警告除外）。其值反映读取的上一个游标：

值	说明
0	表示 fetch 语句成功完成。
1	表示 fetch 语句导致了错误。
2	表示结果集中不再有数据。如果当前的游标位置在结果集中的最后一行并且客户端对该游标提交了 fetch 语句，就会出现这一警告。

- *@@fetch_status* 全局变量提供有关 fetch 是否在可滚动游标中成功执行的信息：

值	说明
0	表示 fetch 语句成功完成。
-1	表示 fetch 操作失败，或者读取的行在结果集之外。
-2	留作将来使用。

- 只有 fetch 语句能设置 @@*sqlstatus* 和 @@*fetch_status*。其它语句对 @@*sqlstatus* 或 @@*fetch_status* 无效。
- 指定的游标是仅向前游标还是可滚动游标将影响 @@*rowcount* 的值。如果游标为缺省的不可滚动游标，则 @@*rowcount* 只能向前逐一递增，直至读取完结果集中的所有行。

  一旦从游标结果集读取了所有行， @@*rowcount* 就表示该游标结果集中的总行数。执行 fetch 之后的 @@*rowcount* 可获取针对该 fetch 操作中指定的游标所读取的行数。

  如果游标可滚动，则 @@*rowcount* 无最大值。有关 @@*rowcount* 的详细信息，请参见 《参考手册：构件块》。

使用可滚动游标

fetch_direction：

- 如果未指定，则缺省值为 next。
- 如果不是 next，则游标必须声明为可滚动。
- *fetch_offset* 必须是精确的、标度为零的有符号数字。
- 将游标定位在末行之后或首行之前，则不会有数据返回，也不会有错误产生。
- 为 absolute，则当 *fetch_offset* >0 时，偏移从结果集的首行之前的位置开始算起。如果 *fetch_offset* <0，则偏移从结果集的末行之后的位置开始算起。
- 为 relative，则当 *fetch_offset* n>0 时，游标定位在当前位置之后的 *n* 行；如果 *fetch_offset* n<0，则游标定位在当前位置之前的 abs (*n*) 行。

结果集中指定的行号从 1 开始计起；首行行号为 1。

每次 fetch 读取多行

缺省行为是，每次 fetch 将一行返回到客户端。每次 fetch 返回的行数可通过输入以下命令更改为另一个数：

```
set cursor rows number for cursor_name
```

number 指定游标可执行的每次 fetch 读取的行数。此数可以是无小数点的数字文字，或者是类型为 integer 的局部变量。如果 *cursor rows* 大于 1，则在 fetch 之后将有多行返回到客户端。某些情况下， fetch 返回的行数可能小于指定的行数，这取决于游标的位置。当前游标位置总是为一行。

可滚动游标定位规则中使用的术语

这些是在接下来的 “可滚动游标定位规则”中使用的术语。

- *curRowsetStart* – 游标的当前位置。
- *new_CurRowsetStart* – 游标新的当前位置。
- *total_rows* – 游标结果集中的总行数。
- *before_first* – 游标结果集第一行前面的行位置。此变量值为 0。
- *after_last* – 游标结果集最后一行后面的行位置。此变量的值为 *total_rows* + 1。
- *first_row* – 游标结果集第一行所在的位置。此变量值为 1。
- *last_row* – 游标结果集最后一行所在的位置。此变量的值与 *total_rows* 相同。
- *fetchSize* – 每次 fetch 操作所请求的行数。

可滚动游标定位规则

这些规则控制在使用 fetch_orientation 选项读取游标行时游标的位置，其中 *cursPos* 为游标位置。请参见 fetch_orientation 选项语法：

Fetch first

new_CurRowsetStart 始终移至 *first_row*，而不考虑 *CurRowsetStart* 的位置以及 *fetchSize* 的值。

Fetch last

- 如果 *total_rows* >= *fetchSize*，
  则 *new_CurRowsetStart* = *total_rows* – *fetchSize* + 1。
- 如果 *total_rows* < *fetchSize*，
  则 *new_CurRowsetStart* 位于 *first_row*。

Fetch next

- 如果 *CurRowsetStart* 为 *before_first*，
  则 *new_CurRowsetStart* 位于 *first_row*。
- 设 *curPos* = (*CurRowsetStart* + *fetchSize*)，
  - 若 *curPos* <=*total_rows*，则 *new_CurRowsetStart* = *curPos*

  - 若 *curPos* > *total_rows*，则 *new_CurRowsetStart* 为 *after_last*

- 如果 *CurRowsetStart* 为 *after_last* 行，
  则 *new_CurRowsetStart* 仍位于 *after_last*。

Fetch prior

- 当下列任一条件成立时，
- *new_CurRowsetStart* 为 *before_first*：
  - (*CurRowsetStart* >= 1) 并且 (*CurRowsetStart* - *fetchSize* <=0)
  - *CurRowsetStart* 为 *before_first*
- 设 *curPos* = *CurRowsetStart* – *fetchSize*；当且仅当 1 <=*curPos* <=*total_rows* 时，
  then *new_CurRowsetStart* = *curPos*.
- 如果 （*CurRowsetStart* 为 *after_last*），设 *curPos* = *total_rows* – *fetchSize* + 1
  如果 *curPos* > 0，则 *new_CurRowsetStart* = *curPos*
  如果 *curPos* <= 0，则 *new_CurRowsetStart* 为 *before_first*

Fetch relative

- 如果 （*CurRowsetStart* 为 *before_first*）并且 (*fetch_offset* > 0)，
  则 *new_CurRowsetStart* = *fetch_offset*。
- *new_CurRowsetStart* 为 *before_first*：
  - （*CurRowsetStart* 为 *before_first*）且 (*fetch_offset* < 0)
  - （*CurRowsetStart* 位于 *first_row*）且 (*fetch_offset* < 0)
  - （*CurRowsetStart* 为 *after_last*）
    且 ((*CurRowsetStart* + *fetch_offset* + 1) <= 0)
- 如果 (1 < *CurRowsetStart* <= *total_rows*)，
  设 *curPos* = *CurRowsetStart* + *fetch_offset*，则：
  - *new_CurRowsetStart* 位于 *first_row* （当且仅当
    (*curPos* < 1) 且 *abs* (*fetch_offset*) <= *fetchSize* 时）
  - *new_CurRowsetStart* 位于 *first_row* 之前 （当且仅当
    (*curPos* < 1) 并且 (*abs* (*fetch_offset*) *fetchSize*) 时）
  - 当且仅当 (0 < *curPos* <=*total_rows*) 时， *new_CurRowsetStart* = *curPos*
  - 当且仅当 *curPos* > *total_rows* 时， *new_CurRowsetStart* 为 *after_last*

- 如果 （*CurRowsetStart* 为 *after_last*），
  设 *curPos* = *CurRowsetStart* + *fetch_offset* +1，则：
  - 当且仅当 1 <= *curPos* <= *total_rows* 时， *new_CurRowsetStart* = *curPos*
  - 当且仅当 *curPos* <= 0 时， *new_CurRowsetStart* 为 *before_first*
  - 当且仅当 *curPos* > *total_rows* 时， *new_CurRowsetStart* 为 *after_last*

Fetch absolute

- 如果 *fetch_offset* = 0，则 *new_CurRowsetStart* 为 *before_first*
- 如果 *fetch_offset* > *total_rows*，则 *new_CurRowsetStart* 为 *after_last*
- 如果 0 < *fetch_offset* <= *total_rows*，则 *new_CurRowsetStart* = *fetch_offset*
- 如果 (*fetch_offset* < 0) 并且 (*abs* (*fetch_offset*) > *total_rows*)，
  设 *abs_offset* = *abs* (*fetch_offset*)
  当且仅当 *abs_offset* > *fetchSize* 时，*new_CurRowsetStart* 为 *before_first*
  当且仅当 *abs_offset* <= *fetchSize* 时，*new_CurRowsetStart* 位于 *first_row*
- 如果 (*fetch_offset* < 0) 并且 (*abs* (*fetch_offset*) <= *total_rows*)
  new_CurRowsetStart = *total_rows* + *fetch_offset* + 1

标准

符合 ANSI SQL 的级别符合初级标准。

读取多行是一种 Transact-SQL 扩展。

权限

fetch 权限缺省情况下授予所有用户。

另请参见

命令 declare cursor, open, set.

goto label

说明

分岔转到用户定义的标签。

语法

label:
goto *label*

示例

显示名为 restart 的标签的使用：

```
declare @count smallint
select @count = 1
restart:
  print "yes"
select @count = @count + 1
while @count <=4
  goto restart
```

用法

- 标签名必须符合标识符的规则，且在声明时必须后接冒号 (:)。当与 goto 一起使用时，它不后接冒号。
- 为了避免 goto 和标签之间的无穷循环，使 goto 依赖于 if 或 while 测试，或者其它一些条件。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 goto 无需任何权限。

另请参见

命令 if...else, while.

grant

说明

为单个用户、用户组和角色指派权限。

语法

授予访问数据库对象的权限：

```
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]
```

授予使用内置函数的权限：

```
grant select
    on [builtin] builtin
    to {name_list | role_list}
    [granted by grantor]
```

授予执行特定命令的系统特权：

```
grant {all [privileges] | privilege_list}
    to {public | name_list | role_list}
    [granted by grantor]
```

授予 dbcc 特权：

```
grant {dbcc_privilege [on database ]
        [, dbcc_privilege [on database ], ...]}
    to {user_list | role_list }
    [granted by grantor]
```

授予特定系统表的缺省权限：

```
grant default permissions on system tables
```

授予被授予者权限，允许其将服务器用户标识切换成任何其它服务器登录名，并根据目标登录名的角色限制其使用：

```
grant set proxy to name_list
    [restrict role role_list | all | system]
    [granted by grantor]
```

参数

all

当用于指派访问数据库对象的权限时，all 指定授予适用于指定对象的所有权限 （decrypt 权限除外）。所有对象所有者都可以使用含有对象名的 grant all 授予各自对象的权限。 decrypt 权限必须单独授予。

在未启用细化权限的情况下，系统管理员或数据库的所有者可以使用 grant all 指派用于创建数据库对象的特权 （请参见 “授予执行特定命令的系统特权”的语法）。由系统管理员使用时，grant all 指派所有的 create 特权 （create database、 create default、 create procedure、 create rule、 create table、 create function 和 create view）。数据库所有者使用 grant all 或在 master 数据库外执行 grant all 时， Adaptive Server 将授予除 create database 之外的所有 create 权限并显示信息性消息。

在启用细化权限的情况下，不支持使用 grant all 授予所有 create 特权。有关详细信息，请参见 《安全性管理指南》中的 “使用细化权限”。

不能将 all 用于包含 where 子句的 grant 语句。

permission_list

是授予的对象访问权限的列表。如果列出的权限不止一个，可用逗号将它们隔开。下表说明了可针对每种对象类型授予的访问权限：

对象	permission_list 可包括
列	select、 update、 references、 decrypt 列名可以在 *column_list* 中指定。
加密密钥	select
存储过程	execute
SQL 函数	execute
表	select、 insert、 delete、 update、 references、 update statistics、delete statistics、truncate table、decrypt、transfer table、 identity_insert *、 identity_update *
视图	select、insert、delete、update、decrypt、identity_insert *、identity_update *

注释 标有星号 (*) 的权限仅在启用细化权限时才能授予。

correlation_name

仅用于 grant ... where 命令，作为别名来引用 where 子句中 *table_name* 内的列。

table_name

是您在授予其使用权限的表的名称。该表必须在当前数据库中。对于每个 grant 语句只能列出一个对象。

column_list

是由逗号分隔的、该权限适用的一个或多个命名列。如果指定了列，则只能授予 select、 references、 decrypt 和 update 权限。

如果使用 where 子句对一个或多个命名列发出 grant 命令，则 Adaptive Server 将按照以下方式对用户的 select、update 或 delete 命令实施行级访问：

- 在目标列表或用户 select 语句的 where 子句中引用 grant select 语句上的一个或多个命名列
- 在用户 update 语句的目标列表中引用 grant update 语句上的一个或多个命名列
- 对于会话已将 ansi_permissions 设置为开的用户 update 或 delete 语句，在其 where 子句中引用 grant select 上的一个或多个列。

view_name

是您在授予其使用权限的视图的名称。该视图必须位于当前数据库中。

stored_procedure_name

是您在授予其权限的存储过程的名称。该存储过程必须位于当前数据库中。

key_name

是您在授予其访问权限的加密密钥的名称。 *key_name* 必须存在于当前数据库中。

SQL_function_name

是您在授予其权限的 SQL 函数的名称。此存储函数必须位于当前数据库中。对于每个 grant 语句只能列出一个函数。

where *search_conditions*

充当行过滤器，与 select、update 或 delete 语句中指定的任何 where 子句结合使用。仅当向表授予 select、update 和 delete 特权时，才可以使用 where 语法。 *search_conditions* 可以使用一般 where 子句中允许的所有语法。如果 where 子句访问没有授权的表，则必须使用子查询。关于使用 grant 语句上的 where 子句的信息，请参见 《安全性管理指南》中的 *“授予谓词特权”*。

as *pred_name*

是谓词的名称，在当前数据库中授予者所拥有的其它对象名称中必须是唯一的，并且符合标识符规则。如果您省略 *pred_name*， Adaptive Server 会向 grant 谓词指派一个唯一的名称，使用 sp_helprotect 可查看该名称。*pred_name* 不能用于没有 where 子句的 grant 语句。谓词可由 revoke 命令按名称引用。

public
指所有用户。对于对象访问权限， public 不包括对象所有者。对于对象创建权限或 set proxy 授权， public 不包括数据库所有者。

name_list
是由逗号分隔的用户名和组名的列表。

role_list
是您要授予其权限的角色 （系统定义角色或用户定义角色）的列表。

with grant option
允许 *name_list* 中指定的用户向其他用户授予对象访问权限。使用 with grant option 只能向单个用户授予权限，而不能向 “public”或者向组或角色授予权限。不能使用 with grant option 授予谓词特权

granted by *grantor*
将授予者指定为数据库中未执行该命令的用户。

grantor
当前数据库中的有效用户名，授予者的用户标识 （而非执行者的用户标识）将会作为授权者记录在系统目录 sysprotects 中。

builtin
是内置函数。在内置函数名允许使用相同的名称区分表和可授予内置函数之前，指定关键字 *builtin*。可授予 *builtin* 函数为 set_appcontext、get_appcontext、 list_appcontext、 authmech、 rm_appcontext 和 next_identity （需要对 IDENTITY 列的 select 权限）。

privilege_list
是列有可授权系统特权的列表。系统特权包括服务器范围和数据库范围特权。有关可授予的系统特权，请参见表 1-21、表 1-22。有关如何授予系统特权的详细信息，另请参见 “用法”部分。使用逗号分隔多个命令。

dbcc_privilege
是您在授予的 dbcc 特权的名称。它不能是变量。表 1-21 和 表 1-22 包括可授予的服务器范围 dbcc 和数据库范围 dbcc 特权。

注释 不能向 public 或组授予或撤消 dbcc 特权。

database

是您正在授予其权限的数据库的名称。它用于授予数据库范围 dbcc 特权。 on database 子句是可选的，数据库必须是当前数据库。被授予者必须是目标数据库中的有效用户。 *database* 应符合标识符的规则，而且不能是变量。

如果在同一命令中存在多个授予的操作，则 *database* 必须是唯一的。

set proxy

授予用户充当另一用户的权限。如果被授予者不具有已授予的 *role_list* 中的角色，而目标登录名具有已授予的 *role_list* 中的任何角色，对该目标登录名进行 set proxy 将失败。

system

被授予者不能与拥有其它系统角色的用户交换其标识。 *sysystem* 只能用于 set proxy 参数。

restrict role *role_list*

只有被授予者和目标登录名拥有 *role_list* 中包括的任何角色，才允许其交换标识。

all

被授予者可将其标识授予与其具有相同角色集的任何用户。也就是说，被授予者不能通过执行 set proxy 命令来继承任何新角色。

系统表的缺省权限

指定您授予对第 425 页的 “授予对系统表的缺省权限”中所列出的系统表的缺省权限。

示例

示例 1 授予 Mary 和 “sales” 组在 titles 表上使用 insert 和 delete 命令的权限：

```
grant insert, delete
on titles
to mary, sales
```

示例 2 授予 “public”（包括所有用户） get_appcontext 函数的 select 权限：

```
grant select on builtin get_appcontext to public
```

将其与以下命令比较，如果存在一个名为 get_appcontext 的表，以下命令将授予该表的 select 权限：

```
grant select on get_appcontext to public
```

明确在 grant 语句中包括 builtin 参数可确保您不会错误选择与某个函数同名的表 本例中，存在与 get_appcontext 函数同名的 get_appcontext 表。

示例 3 有两种方法可以将 titles 表的 price 和 advance 列的 update 权限授予 “public”（包括所有用户）：

```
grant update
```

```
on titles (price, advance)
to public
```

或：

```
grant update (price, advance)
on titles
to public
```

示例 4 向用户 Mary 授予对 titles 表的 transfer table 权限：

```
grant transfer table on titles to mary
```

示例 5 授予 Mary 和 John 使用 create database 和 create table 命令的权限。Mary 和 John 的 create table 权限仅适用于 master 数据库：

```
grant create database, create table
to mary, john
```

示例 6 授予所有用户对 titles 表的完整访问权限 （decrypt 权限除外）：

```
grant all on titles
to public
```

示例 7 给 Mary 在 authors 表上使用 update 命令的权限和将该权限授予其他用户的权限：

```
grant update on authors
to mary
with grant option
```

示例 8 给 Bob 在 titles 表的 price 列上使用 select 和 update 命令的权限和将该权限授予其他用户的权限：

```
grant select, update on titles (price)
to bob
with grant option
```

示例 9 授予所有系统安全员执行 new_sproc 存储过程的权限：

```
grant execute on new_sproc
to sso_role
```

示例 10 授予 James 在引用 titles 表的 price 列的另一个表上创建参照完整性约束的权限：

```
grant references on titles (price)
```

```
to james
```

注释　在创建包含参照完整性约束的表以引用另一用户的表之前，必须被授予对被引用表的 references 权限。该表在被引用的列上还必须具有一个唯一约束或唯一索引。有关参照完整性约束的详细信息，请参见 create table。

示例 11 密钥所有者执行时，授予数据库所有者使用 ssn_key 指定列加密的权限。数据库所有者需要对 ssn_key 具有 select 权限，以便在执行 create table、 alter table 或 select into 时进行引用：

```
grant select on ssn_key to dbo
```

示例 12 授予 Bob 创建加密密钥的权限：

```
grant create encyption key to Bob
```

示例 13 授予对 customer 表中所有加密列的 decrypt 权限：

```
grant decrypt on customer to accounts_role
```

示例 14 向 Joe 授予对 master 的 dump any database 特权，允许他转储任何数据库：

```
1> use master
2> go
1> grant dump any database to joe
2> go
```

示例 15 向 Joe 授予对数据库 pubs2 的 create any object 特权，允许其代表自己或者 pubs2 中的其他用户创建任何对象特权：

```
1> use pubs2
2> go
1> grant create any object to joe
2> go
```

示例 16 向 Alex 授予 manage roles。这会返回一条错误，因为服务器范围特权需要 master 为当前数据库：

```
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.
```

示例 17 通过使用角色，系统管理员允许 Carlos 对任何符合以下两种条件之一的数据库执行 dbcc checkalloc：Carlos 是该数据库的有效用户；该数据库支持 “guest”用户。

注释 注意：如果 master中已经存在 “guest”用户，则不必将 Carlos 作为真实用户添加到 master 数据库中。

```
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
```

示例 18 授予 Frank （master 数据库中的有效用户）对服务器中的所有数据库执行 dbcc checkdb 的能力：

```
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 frank
2> go

New user added.
 (return status = 0)

1> grant dbcc checkdb any database to frank
2> go
```

Frank 现在可对服务器中的、他是其中的有效用户的每个数据库执行 dbcc checkdb 命令：

```
% 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.
```

注释　不能向 public 或组授予或撤消 dbcc 特权。

示例 19 如果 Walter 需要成为 pubs2 的维护用户，但系统管理员不希望授予他对其它数据库的管理员级特权，则系统管理员可执行以下命令：

```
1> use pubs2
2> go
1> grant dbcc checkdb on pubs2 to walter
2> go
```

注释　系统管理员必须在目标数据库（在本例中为 pubs2）中，同时，Walter 必须是该目标数据库中的有效用户。 on pubs2 子句是可选的。

Walter 现在能够在 customers 数据库上执行 dbcc checkdb 命令，而不会遇到错误。

示例 20 错误地将 grant dbcc 和 revoke dbcc 应用于组或 public：

```
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)

11> 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.
```

示例 21 不能使用 grant 选项授予系统特权：

```
grant change password to alex with grant option

Msg 156, Level 15, State 1:
Line 1:
Incorrect syntax near the keyword 'with'.
```

示例 22 允许 Harry 在 authors 表上使用 truncate table 和 updates statistics：

```
grant truncate table on authors to harry
grant update statistics on authors to harry
```

示例 23 允许 Billy 在 authors 表上使用 delete statistics 命令：

```
grant delete statistics on authors to billy
```

示例 24 授予角色为 oper_role 的所有用户 truncate table、 update 和 delete statistics 特权 （如果 Billy 和 Harry 具有角色 oper_role，则他们现在可以对 authors 执行这些命令）：

```
grant truncate table on authors to oper_role
grant update statistics on authors to oper_role
grant delete statistics on authors to oper_role
```

示例 25 通过存储过程隐式授予 truncate table、 delete statistics 和 update statistics 权限。例如，假设 Billy 拥有 authors 表，他可以执行以下命令授予 Harry 在 authors 表上运行 truncate table 和 update statistics 的特权：

```
create procedure sproc1
as
truncate table authors
update statistics authors
go
grant execute on sproc1 to harry
go
```

也可以通过存储过程在列级隐式授予 update statistics 和 delete statistics 的权限。

示例 26 授予 Harry 和 Billy 执行 set proxy 或 set session authorization 的权限，使其可以在服务器上充当另一个用户：

```
grant set proxy to harry, billy
```

示例 27 授予具有 sso_role 权限的用户执行 set proxy 或 set session authorization 的权限，使其可以在服务器上充当另一个用户：

```
grant set session authorization to sso_role
```

示例 28 将 set proxy 授予 Joe，但限制他将标识切换为具有 sa、 sso 或 admin 角色的任何用户 （但是，如果他已经具有这些角色，则他可以对具有这些角色的任何用户执行 set proxy）：

```
grant set proxy to joe
restrict role sa_role, sso_role, admin_role
```

如果 Joe 尝试将其标识切换为某个具有 admin_role 的用户 （在本示例中，为 Our_admin_role）时，除非他已经具有 admin_role，否则此命令失败：

```
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.
```

Joe 被授予 admin_role 并重试此命令后，此命令成功：

```
grant role admin_role to joe
set proxy Our_admin_role
```

示例 29 限制 Joe 在切换标识时被授予任何新的角色：

```
grant set proxy to joe
restrict role all
```

Joe 只能将 set proxy 授予具有与他相同角色 （或更少特权的角色）的用户。

示例 30 限制 Joe 在使用 set proxy 时获得任何新的系统角色：

```
grant set proxy to joe
restrict role system
```

如果目标登录名具有 Joe 所没有的系统角色，则 set proxy 失败。

示例 31 仅允许学员查看其各自等级的相关信息：

```
grant select on grades
  where user_name(uid) = USER as predicate_grades
  to public
```

示例 32 允许注册学员查看所有课程的相关信息。第一个 grant 允许任何用户浏览所提供的课程和章节。第二个 grant 仅允许用户查看自己注册的课程。

```
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
```

注册学员输入以下查询时，会被限定为查看自己的课程 （因为已选中 with_honors 列）：

```
select course_id, quarter, with_honors
  from enrollment
```

同样，当注册学员尝试使用以下查询查看用户所参与的课程数时：

```
select course_id, count(uid) from enrollment
  group by course_id
```

Adaptive Server 将返回一个行，显示用户注册的课程数。

示例 33 用户 Smith 授予 John 对 mary.books 的 select 权限，表所有者 Mary 为授予者：

```
grant select on mary.books to john
granted by mary
```

示例 34 用户 Smith 授予用户 John create table 权限， dbo 为授予者：

```
grant create table to john
granted by dbo
```

示例 35 在禁用细化权限的情况下，授予系统特权 manage any login 将导致错误：

```
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.
```

示例 36 授予系统特权 own database 时，必须指定 on *database* 子句：

```
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 tdb1 to smith
2>go
```

用法

语法替代

可在 grant 语法中用关键字 from 替代 to。

使用 set fipsflagger

在 set fipsflagger 选项启用的情况下，当执行 grant dbcc 时，它将发出下面的警告：

```
SQL statement on line number 1 contains Non-ANSI
text.The error is caused due to the use of DBCC.
```

“特权” (Privileges)

服务器范围系统特权

表 1-21 列出所有可授予服务器范围系统特权。必须在 master 数据库中授予服务器范围特权。有关每个被授予特权可执行的操作，请参见《安全性管理指南》中 “使用细化权限”一章的表 8-15：服务器范围特权。

注释　在表 1-21 中，如果禁用细化权限，则只能授予标有星号 (*) 的特权。

dbcc 特权语法 dbcc dbcc_subcmd on all 是 dbcc dbcc_subcmd any database 的别名。这两种语法均受支持。

表 1-21: 可授予的服务器范围系统特权

类别	特权	
特权管理	• manage security permissions • manage server permissions	
审计管理	• manage auditing	
登录名和角色管理	• allow exceptional login • change password • manage any login	• manage any login profile • manage remote login • manage roles

类别	特权	
数据库管理	• checkpoint (on *database*) • checkpoint any database • create database* • dump any database • dump database (on *database*) • load any database • load database (on *database*) • manage any database	• mount any database • online any database • online database (on *database*) • own any database • own database (on *database*) • quiesce any database • unmount any database
服务器管理	• manage any thread pool • manage cluster • manage disk • manage security configuration	• manage server • manage server configuration • shutdown
DBCC 特权	• dbcc checkalloc any database* • dbcc checkcatalog any database* • dbcc checkdb any database* • dbcc checkindex any database* • dbcc checkstorage any database* • dbcc checktable any database* • dbcc checkverify any database*	• dbcc fix_text any database* • dbcc indexalloc any database* • dbcc reindex any database* • dbcc tablealloc any database* • dbcc textalloc any database* • dbcc tune*
应用程序管理	• manage any execution class • manage any ESP • manage data cache • manage dump configuration	• manage lock promotion threshold • monitor qp performance • manage resource limit
其它	• connect* • kill • kill any process • map external file • monitor server replication • set proxy	• set tracing* • set tracing any process • set switch • show switch • use any database • use database

数据库范围特权

表 1-22 列出所有可授予数据库范围系统特权。数据库范围特权必须在要执行特权的数据库中授予。有关每个被授予特权可执行的操作，请参见《安全性管理指南》中“使用细化权限”一章的表 8-16：数据库范围特权。

注释　在表 1-22 中，如果禁用细化权限，则只能授予标有星号 (*) 的特权。

表 1-22: 可授予的数据库范围特权

类别	特权	
特权管理	• manage any object permission • manage database permissions	
用户管理	• manage any user	
设置用户	• setuser*	
复制管理	• manage replication	
数据库管理	• manage database	
查询计划管理	• manage abstract plans	
DBCC 特权	• dbcc checkalloc * • dbcc checkcatalog * • dbcc checkdb * • dbcc checkindex * • dbcc checkstorage * • dbcc checktable * • dbcc checkverify *	• dbcc fix_text * • dbcc indexalloc * • dbcc reindex * • dbcc tablealloc * • dbcc textalloc * • manage checkstorage • report checkstorage
系统目录	• select any audit table • select any system catalog • truncate any audit table	
一般对象	• alter any object owner • create any object • drop any object	
加密密钥	• create encryption key * • manage any encryption key • manage column encryption key	• manage master key • manage service key
缺省值	• create default * • create any default • drop any default	

类别	特权	
函数	• create function *	• drop any function
	• create any function	• execute any function
索引	• create any index	
过程	• create procedure *	• execute any procedure
	• create any procedure	• drop any procedure
规则	• create rule *	
	• create any rule	
	• drop any rule	
表	• alter any table	• insert any table
	• create any table	• manage any statistics
	• create table *	• references any table
	• decrypt any table	• reorg any table
	• delete any table	• select any table
	• drop any table	• transfer any table
	• identity_insert any table	• truncate any table
	• identity_update any table	• update any table
触发器	• create trigger *	
	• create any trigger	
	• drop any trigger	
视图	• create view *	
	• create any view	
	• drop any view	

特权列表

表 1-23 按字母顺序列出所有可授予的特权和权限。用 “*” 标识的特权不需要启用细化权限。

表 1-23: 特权的字母排序列表

特权名称	特权类型	管理方（启用细化权限时）	含义
allow exceptional login	服务器	manage server 权限	
alter any object owner	数据库	manage database 权限	
alter any table	数据库	manage database 权限	
change password	服务器	manage security 权限	
checkpoint any database	服务器	manage server 权限	
checkpoint (on database)	服务器	manage server 权限	checkpoint any database
checkpoint (on sybsecurity)	服务器	manage security 权限	
connect*	服务器	manage server 权限	

特权名称	特权类型	管理方（启用细化权限时）	含义
create any default	数据库	manage database 权限	create any object
create any function	数据库	manage database 权限	create any object
create any index	数据库	manage database 权限	create any object
create any object	数据库	manage database 权限	
create any procedure	数据库	manage database 权限	create any object
create any rule	数据库	manage database 权限	create any object
create any table	数据库	manage database 权限	create any object
create any trigger	数据库	manage database 权限	create any object
create any view	数据库	manage database 权限	create any view
create database*	服务器	manage database 权限	
create default*	数据库	manage database 权限	create any default
create encryption key*	数据库	manage security 权限	manage column encryption key
create function*	数据库	manage database 权限	create any function
create index*	数据库	manage database 权限	create any index
create procedure*	数据库	manage database 权限	create procedure
create rule*	数据库	manage database 权限	create any rule
create table*	数据库	manage database 权限	create any table
create trigger*	数据库	manage database 权限	create any trigger
create view*	数据库	manage database 权限	create any view
dbcc checkalloc*	数据库	manage database 权限	dbcc checkalloc any database
dbcc checkalloc any database*	服务器	manage server 权限	
dbcc checkcatalog*	数据库	manage database 权限	dbcc checkcatalo any database
dbcc checkcatalog any database*	服务器	manage server 权限	
dbcc checkdb*	数据库	manage database 权限	dbcc checkdb any database
dbcc checkdb any database*	服务器	manage server 权限	
dbcc checkindex*	数据库	manage database 权限	dbcc checkindex any database
dbcc checkindex any database *	服务器	manage server 权限	
dbcc checkstorage*	数据库	manage database 权限	dbcc checkstorage any database
dbcc checkstorage any database *	服务器	manage server 权限	
dbcc checktable*	数据库	manage database 权限	dbcc checktable any database
dbcc checktable any database*	服务器	manage server 权限	
dbcc checkverify*	数据库	manage database 权限	dbcc checkverify any database
dbcc checkverify any database*	服务器	manage server 权限	
dbcc fix_text*	数据库	manage database 权限	dbcc fix_text any database
dbcc fix_text any database*	服务器	manage server 权限	
dbcc indexalloc*	数据库	manage database 权限	dbcc indexalloc any database

特权名称	特权类型	管理方（启用细化权限时）	含义
dbcc indexalloc any database*	服务器	manage server 权限	
dbcc reindex*	数据库	manage database 权限	dbcc reindex any database
dbcc reindex any database*	服务器	manage server 权限	
dbcc tablealloc*	数据库	manage database 权限	dbcc tablealloc any database
dbcc tablealloc any database*	服务器	manage server 权限	
dbcc textalloc*	数据库	manage database 权限	dbcc textalloc any database
dbcc textalloc any database*	服务器	manage server 权限	
dbcc tune*	服务器	manage server 权限	
decrypt*	对象（列）	manage any object permission/object owner	decrypt any table
decrypt any table	数据库	manage database 权限	
delete*	对象	manage any object permission/object owner	delete any table
delete any table	数据库	manage database 权限	
delete statistics*	对象	manage any object permission/object owner	manage any statistics
drop any default	数据库	manage database 权限	drop any object
drop any function	数据库	manage database 权限	drop any object
drop any object	数据库	manage database 权限	
drop any procedure	数据库	manage database 权限	drop any object
drop any rule	数据库	manage database 权限	drop any object
drop any table	数据库	manage database 权限	drop any object
drop any trigger	数据库	manage database 权限	drop any object
drop any view	数据库	manage database 权限	drop any object
dump any database	服务器	manage server 权限	
dump database (on database)	服务器	manage server 权限	dump any database
dump database (on sybsecurity)	服务器	manage security 权限	
execute*	对象	manage any object permission/object owner	execute any function （针对 udf） execute any function （针对系统过程）
execute any function	数据库	manage database 权限	
execute any procedure	数据库	manage database procedures	
identity_insert	对象	manage any object permission/object owner	
identity_insert any table	数据库	manage database 权限	
identity_update	对象	manage any object permission/object owner	

特权名称	特权类型	管理方（启用细化权限时）	含义
identity_update any table	数据库	manage database 权限	
insert*	对象	manage any object 权限/对象所有者	insert any table
insert any table	数据库	manage database 权限	
kill	服务器	manage server 权限	kill any process
kill any process	服务器	manage server 权限	
load any database	服务器	manage server 权限	
load database (on database)	服务器	manage server 权限	load any database
load database (on sybsecurity)	服务器	manage security 权限	
manage abstract plans	数据库	manage database 权限	
manage any database	服务器	manage server 权限	
manage any encryption key	数据库	manage security 权限	
manage any ESP	服务器	manage server 权限	
manage any execution class	服务器	manage server 权限	
manage any login	服务器	manage security 权限	
manage any login profile	服务器	manage security 权限	
manage any remote login	服务器	manage security 权限	
manage any statistics	数据库	manage database 权限	
manage any thread pool	服务器	manage server 权限	
manage any user	数据库	manage database 权限	
manage auditing	服务器	manage security 权限	
manage checkstorage	数据库	manage database 权限	
manage cluster	服务器	manage server 权限	
manage column encryption key	数据库	manage security 权限	manage any encryption key
manage data cache	服务器	manage server 权限	
manage database	数据库	manage database 权限	manage any database
manage database permissions	数据库	manage security 权限	
manage disk	服务器	manage server 权限	
manage dump configuration	服务器	manage server 权限	
manage lock promotion threshold	服务器	manage server 权限	
manage master key	数据库	manage security 权限	manage any encryption key
manage replication	服务器	manage server 权限	
manage resource limit	服务器	manage server 权限	
manage roles	服务器	manage security 权限	
manage security configuration	服务器	manage security 权限	
manage security permissions	服务器	manage security 权限	

特权名称	特权类型	管理方（启用细化权限时）	含义
manage server	服务器	manage server 权限	
manage server configuration	服务器	manage server 权限	
manage server permissions	服务器	manage server 权限	
manage service key	数据库	manage security 权限	manage any encryption key
map external file	服务器	manage server 权限	
monitor qp performance	服务器	manage server 权限	
monitor server replication	服务器	manage server 权限	
mount any database	服务器	manage server 权限	
own any database	服务器	manage server 权限	
online any database	服务器	manage server 权限	
online database (on database)	服务器	manage server 权限	online any database
online database (on sybsecurity)	服务器	manage security 权限	
own database (on database)	服务器	manage server 权限	
own database (on sybsecurity)	服务器	manage security 权限	
quiesce any database	服务器	manage server 权限	
references*	对象（列）	manage any object 权限/对象所有者	references any table
references any table	数据库	manage database 权限	
report checkstorage	数据库	manage database 权限	
reorg any table	数据库	manage database 权限	
select*	对象（列）	manage any object 权限/对象所有者	select any table （用于用户表或视图） select any audit table （用于审计表） select any system catalog （用于系统表）
select any audit table	数据库	manage database 权限	
select any system catalog	数据库	manage database 权限	
select any table	数据库	manage database 权限	
set proxy*	服务器	manage security 权限	
set switch	服务器	manage server 权限	
set tracing*	服务器	manage server 权限	set tracing （用于所有进程）
set tracing any process	服务器	manage server 权限	
setuser*	数据库	manage database 权限	
show switch	服务器	manage server 权限	
shutdown	服务器	manage server 权限	

特权名称	特权类型	管理方（启用细化权限时）	含义
transfer any table	数据库	manage database 权限	
transfer table	对象	manage any object 权限/对象所有者	transfer any table
truncate any audit table	数据库	manage database 权限	
truncate any table	数据库	manage database 权限	
truncate table*	对象	manage any object 权限/对象所有者	truncate any table
unmount any database	服务器	manage server 权限	
update*	对象（列）	manage any object 权限/对象所有者	update any table
update any security catalog	服务器	manage security 权限	
update any table	数据库	manage database 权限	
update statistics	对象	manage any object 权限/对象所有者	manage any statistics
use any database	服务器	manage server 权限	
use database (on database)	服务器	manage server 权限	use any database
use database (on sybsecurity)	服务器	manage security 权限	

- 处理一项特权可能意味着处理另一项更加细化的特权。例如，用户具有 select any table 特权意味着此用户对所有用户表均具有 select 权限。对于具有隐含关系的特权对的完整列表，请参见表 1-23。
- 授予以下数据库管理特权时，必须为每个特权指定 on *database* 子句：checkpoint、dump database、load database、online database、own database。例如，要向 smith 授予对 db1 的 dump database 特权，可以使用：

  ```
  grant dump database on db1 to smith
  ```

  可在同一授予命令中对不同的数据库授予不同的数据库管理特权。例如，要向 smith 授予对 db1 的 own database 特权和对 db2 的 load database 特权，可以使用：

  ```
  grant own database on db1, load database on db2 to
  smith
  ```

- 只能授予当前数据库中对象的权限。
- grant 和 revoke 命令须区别先后顺序。有冲突发生时，最近发出的语句生效。

- 即便用户不具有对由过程或视图引用的对象的权限，他或她也可以被授予该视图或存储过程的权限。请参见 《安全性管理指南》中的“管理用户权限”。
- 无论为 declare cursor 语句所引用的基表或视图定义了何种权限，Adaptive Server 都授予所有用户声明游标的权限。游标不是作为 Adaptive Server 对象 （如表）定义的，因此不能对游标应用任何权限。当用户打开游标时， Adaptive Server 会判断该用户对于定义该游标的结果集的对象是否具有 select 权限。每次打开游标时它都会检查权限。

  如果用户有权访问游标所定义的对象， Adaptive Server 打开该游标并允许用户使用该游标 fetch 行数据。 Adaptive Server 并不对每个 fetch 操作应用权限检查。但是，如果用户通过游标执行 delete 或 update，则会对游标结果集中所引用对象的数据的删除和更新操作应用常规权限检查。
- grant 语句为每个接受此权限的用户、组或角色在 sysprotects 系统表中添加一行。如果随后对用户或组的该权限执行 revoke 命令，Adaptive Server 会从 sysprotects 中删除相应的行。如果仅撤消了选定的组成员的该权限，但并没有撤消获得此授权的整个组的该权限， Adaptive Server 保留其初始行并为撤消的权限添加一个新行。
- 不同的授予者可为用户、组或角色授予相同的特权或权限。在此情况下，表示同一特权或权限的多个相关授权的 sysprotects 将具有多个行。如果稍后撤消一个或多个授权，则只要还有一个授权未撤消，用户、组或角色就仍会具有特权或权限。
- 如果由于用户是某组的成员而继承了特定的权限，则在明确授予此用户相同权限时，不会向 sysprotects 添加任何行。例如，如果已授予 “public”对 authors 表中 phone 列的 select 权限，然后授予 John （“public”成员）对 authors 表的所有列的 select 权限。作为对 John 的 grant 的结果所添加到 sysprotects 中的行包含对 authors 表中除 phone 列之外的所有列的引用，因为他已经具有对这一列的权限。
- 在缺省情况下，会向用户授予发出 create trigger 命令的权限。当撤消用户创建触发器的权限时，会在 sysprotects 表中为该用户添加一个撤消行。若要向该用户授予发出 create trigger 命令的权限，必须发出两个 grant 命令。第一个命令从 sysprotects 中删除撤消行；第二个命令插入一个授予行。如果撤消了创建触发器的权限，则用户甚至无法在自己拥有的表上创建触发器。撤消用户创建触发器的权限只影响从中发出 revoke 命令的数据库。
- 使用以下系统过程可显示有关权限的信息：
  - sp_helprotect 报告数据库对象、用户、组或角色的权限信息。

- sp_column_privileges 报告表或视图的一列或多列的权限信息。
- sp_table_privileges 报告表或视图的所有列的权限信息。
- sp_activeroles 显示 Adaptive Server 当前登录会话的所有活动角色，以及这些角色所包含的所有角色。
- sp_displayroles 显示授予另一角色或用户的所有角色，或以表格格式显示角色的整个层次树。

- 您可以用 sp_helprotect 来查看权限：

```
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)
```

- 不能将 grant with grant option 与 grant dbcc 一起使用。

向角色授予访问权限

授予角色的权限会替换授予用户或组的权限。例如，假定授予了 John 系统安全员角色，且 sso_role 被授予访问 sales 表的权限。如果撤消 John 访问 sales 表的个人权限，他仍可以访问 sales 表，因为他的角色权限覆盖了他的个人权限。

然而， grant execute 权限不会阻止没有指定角色的用户被独立地授予存储过程的执行权限。若要确保只有系统安全员才能被授予某个存储过程的执行权限，可在该存储过程内使用 proc_role 系统函数。该函数检查调用用户是否有执行过程的正确权限。请参见 proc_role。

***grant all* 对象创建特权**

- 在数据库中使用不带对象名的 grant all 不会授予 create encryption key 权限。仅当细化权限禁用时，才支持不带对象名的 grant all。
- 不与对象名一起使用时， grant all 指派下列权限：create database、create default、 create procedure、 create rule、 create table、 create function 和 create view。create database 权限仅由系统管理员授予，并且仅在 master 数据库内授予。

- 只有数据库所有者和系统管理员可以使用不带对象名的 grant all 语法来授予用户或组 create 命令权限。当数据库所有者使用 grant all 命令时，会显示信息性消息，声明只有系统管理员才能授予 create database 权限。而上述所有其它权限都被授予。
- 所有对象所有者都可以使用含有对象名的 grant all 授予各自对象的权限。当与表或视图名加上用户或组名一起使用时， grant all 启用对表的 delete、delete statistics、insert、select、truncate table、update 和 update statistics 权限。

grant with grant option 规则

- 不能通过 with grant option 将权限授予 “public”或者授予组或角色。
- 在授予权限的过程中，如果禁用细化权限，则会将系统管理员视为对象所有者。如果系统管理员授予操作另一用户的对象的权限，则相应的所有者名称将作为授予者在 sysprotects 和 sp_helprotect 输出中显示。启用细化权限时，授予者的名称将作为授予者出现在 sysobjects 中和 sp_helprotect 输出中。
- 不能使用 grant option 参数授予系统特权。
- 有关每个 grant 命令的信息保存在系统表 sysprotects 中，但有下列例外情况：
  - 如果同一授予者多次授予一个用户某一特定权限， Adaptive Server 会显示信息性消息。只有第一个 grant 记录被保留。
  - 如果两个 grant 相同，只是其中有一个是以 with grant option 的形式授予的，则保留 grant with grant option。
  - 如果同一授予者的两个 grant 语句向特定用户授予对某一特定表的相同权限，但针对不同的列，则 Adaptive Server 将这两个授权视作同一语句。例如，以下 grant 语句是等效的：

```
grant select on titles (price, contract)
    to keiko
grant select on titles (advance) to keiko
grant select on titles (price, contract,
    advance)
to keiko
```

使用 granted by

- 不允许使用 granted by 来授予谓词特权。
- 不要求 grantor 具有执行 grant 命令的权限。
- sysprotects.grantor 下列出的是授予者而非命令执行者。
- 使用 granted by 参数无需启用 enable granular permissions。

- 对于通过 with grant 选项收到对象的 grant 权限的用户，其不能发出 granted by 参数。所有其他用户均可发出 granted by 参数。

  例如，如果 mary 使用 grant 选项将其备忘录的 select 权限授予 john，则 john 尝试发出第二个 grant 命令时将收到错误消息。

  Mary：

  ```
  grant select on mary.books
  to john with grant option
  ```

  John：

  ```
  grant select on mary.books
  to joe granted by smith
  ```

用户和用户组

- 用户组允许用一个语句对多个用户 grant 或 revoke 权限。每个用户都可以是另一个组的成员，且始终是 “public”的成员。
- 可使用 sp_adduser 命令添加新用户并使用 sp_addgroup 命令创建组。若要允许在 Adaptive Server 上有登录名的用户以受限特权使用数据库，可使用 sp_adduser 添加一个 “guest”用户，并为 “guest”指派受限的权限。所有具有登录名的用户可作为 “guest”访问该数据库。”所有具有登录名的用户可作为 “guest”访问该数据库。
- 若要删除用户，请使用 sp_dropuser。若要删除组，请使用 sp_dropgroup。

  若要向 “public”之外的组添加新用户，请使用 sp_adduser。若要更改已建立的用户组，请使用 sp_changegroup。

  若要显示组成员，请使用 sp_helpgroup。
- 执行 sp_changegroup 以更改组成员资格时，它将通过执行以下命令清除内存中的保护高速缓存：

  ```
  grant all to null
  ```

  这样就会用 sysprotects 表中的更新信息刷新高速缓存。若要直接修改 sysprotects，请联系 Sybase 技术支持部门。

授予对系统表的缺省权限

从任意数据库发出该命令时均可授予和撤消其缺省权限的系统表有：

- sysalternates
- sysattributes
- syscolumns
- syscomments
- sysconstraints
- sysdepends
- sysindexes
- sysjars
- syskeys
- syslogs
- sysobjects
- syspartitions
- sysprocedures
- sysprotects
- sysqueryplans
- sysreferences
- sysroles
- syssegments
- sysstatistics
- systabstats
- systhresholds
- systypes
- sysusermessages
- sysusers
- sysxtypes

此命令还进行以下更改：

- 撤消 public 的 syscolumns (encrkyid) 和 syscolumns (encrkydb) 权限。
- 撤消 public 的 syscolumns (encrkydb) 和 syscolumns (encrkyid) 权限。
- 撤消 public 的 sysobjects(audflags) 权限
- 将 sysobjects 的权限授予 sso_role
- 撤消 public 对 sysencryptkeys 的任何列的 select 权限
- 将对所有 sysencryptkeys 列的 select 权限授予 sso_role
- 将 syscolumns 的权限授予 sso_role

从 master 数据库发出该命令时可授予和撤消其缺省权限的系统表有：

- sysdatabases
- sysdevices
- syslocks
- sysmessages
- sysprocesses
- systransactions
- sysusages
- sysconfigures
- syscurconfigs
- syslanguages
- syscharsets
- sysservers
- systimeranges
- sysresourcelimits
- syslogins
- sysremotelogins
- syssessions

此命令还：

- 撤消 public 对 sysdatabases(audflags) 的 select 权限
- 撤消 public 对 sysdatabases(deftabaud) 的 select
- 撤消 public 对 sysdatabases(defvwaud) 的 select
- 撤消 public 的 sysdatabases(defpraud) 的 select
- 撤消 public 对 sysdatabases(audflags2) 的 select

- 将 sysdatabases 的 select 授予 sso_role
- 撤消 public 对 syslogins(password) 的 select 权限
- 撤消 public 对 syslogins(audflags) 的 select
- 撤消 public 对 syslogins(lpid) 的 select 权限
- 将对 syslogins 的 select 授予 sso_role
- 撤消 public 对 syslisteners(net_type) 的 select
- 撤消 public 对 syslisteners(address_info) 的 select
- 将对 syslisteners 的 select 授予 sso_role
- 撤消 public 对 syssrvroles(srid) 的 select
- 撤消 public 对 syssrvroles(name) 的 select
- 撤消 public 对 syssrvroles(password) 的 select
- 撤消 public 对 syssrvroles(pwdate) 的 select
- 撤消 public 对 syssrvroles(status) 的 select
- 撤消 public 对 syssrvroles(logincount) 的 select
- 将对 syssrvroles 的 select 授予 public
- 撤消 public 对 sysloginroles(suid) 的 select
- 撤消 public 对 sysloginroles(srid) 的 select
- 撤消 public 对 sysloginroles(status) 的 select
- 将对 sysloginroles 的 select 授予 sso_role
- 撤消 public 对 sysinstances(hostname) 的 select 权限
- 将对 sysinstances 的 select 权限授予 sso_role

授予 *update statistics*、*delete statistics* 和 *truncate table* 权限

Adaptive Server 允许您向用户、角色和组授予对 update statistics、 delete statistics 和 truncate table 命令的权限。通过将 update statistics、 delete statistics 和 truncate table 添加到一个存储过程，然后将此过程的执行权限授予用户或角色，表所有者也可以通过隐式 grant 提供权限。

不能在列级授予 update statistics 权限。必须具有 sso_role 才能在 sysroles、 syssrvroles 和 sysloginroles 安全表上运行 update statistics 或 delete statistics。

缺省情况下，具有 sa_role 的用户有权在除 sysroles、 syssrvroles 和 sysloginroles 之外的系统表上运行 update statistics 和 delete statistics，并且可以将此特权转交给其他用户。

还可以发出 grant all 授予 update statistics、 delete statistics 和 truncate table 的权限。

注释 一旦授予用户执行 update statistics 的权限，其即具有执行此命令的各种变化形式的权限，如 update all statistics、 update partition statistics、update index statistics、update table statistics 等。例如，以下命令授予 Billy 在 authors 表上运行 update statistics 的各种变化形式的权限：

```
grant update statistics on authors to billy
```

如果撤消用户执行 update statistics 的权限，也就撤消了他们执行此命令的各种变化形式的能力。

不能单独授予 update statistics 的变化形式 （例如， update index statistics）。也就是说，*不能*发出以下命令：

```
grant update all statistics to harry
```

但是，可以编写存储过程来控制谁执行这些命令。例如，以下命令授予 Billy 在 authors 表上执行 update index statistics 的权限：

```
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
```

不能在列级授予和撤消 delete statistics 权限。

尽管 Adaptive Server 将 truncate table 作为一个全局、杂类审计进行审计，但是它不审计 update statistics。为了使 truncate table 和 update statistics 的审计追踪保持清晰明了， Sybase 建议您在授予用户执行权限的存储过程中包括这两条命令，如上所述。

授予代理和会话授权

- 授予执行 set proxy 或 set session authorization 的权限允许被授予者在 Adaptive Server 中充当另一个登录。 set proxy 和 set session authorization 相同，只是 set session authorization 遵循 SQL 标准，而 set proxy 是 Transact-SQL 扩展。
- 要授予 set proxy 或 set session authorization 权限，您必须处于 master 数据库中。

- 在 grant set proxy 命令中指定的名称必须是数据库中的有效用户，即必须是数据库的 sysusers 表中的名称。
- grant all 并不包括 set proxy 或 set session authorization 权限。
- 可以用 grant set proxy 逐步限制角色。例如，可以先限制 sa_role，然后再限制 sso_role：

```
grant set proxy to joe
restrict role sa_role
grant set proxy to joe
restrict role sso_role
```

- 不能对单个角色不限制。必须撤消 set proxy 才能撤消所有角色的权限。

在共享磁盘集群中授予权限

如果尝试在本地临时数据库中向用户定义的角色授予权限，则 grant 将失败。

向角色授予特权

授予角色的权限会替换授予用户或组的权限。例如，假定授予了 John 系统安全员角色，且 sso_role 被授予访问 sales 表的权限。如果撤消 John 访问 sales 表的个人权限，他仍可以访问 sales 表，因为他的角色权限覆盖了他的个人权限。

然而， grant execute 权限不会阻止没有指定角色的用户被独立地授予存储过程的执行权限。若要确保只有系统安全员才能被授予某个存储过程的执行权限，可在该存储过程内使用 proc_role 系统函数。该函数检查调用用户是否有执行过程的正确权限。请参见 proc_role。

撤消 public 或组的特权

撤消 “public”或组的特定权限同样会撤消被单独授予此权限的用户的该权限。例外情况是谓词特权的授予和撤消。请参见 《安全性管理指南》中的 “Adaptive Server 如何将谓词特权保存在 sysprotects 中”

标准

符合 ANSI SQL 的级别符合初级标准。grant dbcc 也是 Transact-SQL 扩展。

grant dbcc 以及授予组权限和授予 set proxy 都是 Transact-SQL 扩展。授予 set session authorization （功能与 set proxy 相同）符合 ANSI 标准。

权限

对 grant 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，用户通常可以使用下列特权管理特权之一来执行 grant 命令，具体视被授予的特权或权限而定。 对于服务器范围特权，您必须是具有 manage server permissions 特权或 manage security permissions 特权的用户。 对于数据库范围特权，您必须是具有 manage database permissions 特权的用户。 对于对象特权，您必须是对象所有者或具有 manage any object permission 特权的用户。 要执行 grant default，您必须是数据库所有者或对数据库具有 own database 特权的用户。 有关详细信息，请参见表 1-23 “管理方 （启用细化权限时）”一列。
细化权限已禁用	在禁用细化权限的情况下，可授予的系统特权限于 create database、 create default、 create function、 create procedure、 create rule、 create table、 create view、 connect、 set proxy 和 set tracing 命令执行 – 只有系统管理员才能授予 create database、connect 和 set tracing 权限，且只能在 master 数据库中授予这些权限。只有系统安全员可以授予 create trigger 权限。 要执行 grant default，您必须是数据库所有者或具有 sa_role 的用户。 数据库一致性检查 – 只有系统管理员才能运行 grant dbcc 命令。 数据库对象访问权授予 – 缺省情况下，数据库对象的权限授予给对象所有者。对象所有者可将其自己的数据库对象的权限授予其他用户。 函数 – 只有系统管理员才能授予对内置函数的权限。 加密列 – 只有系统安全员和密钥管理者才具有创建加密密钥的隐式权限。 代理和会话授权 – 只有系统安全员才能授予 set proxy 或 set session authorization 权限，且只能在 master 数据库中授予这些权限。授予执行 set proxy 或 set session authorization 的权限允许被授予者在服务器中充当另一个登录。 set proxy 和 set session authorization 相同，只是 set session authorization 遵循 ANSI92 标准，而 set proxy 是 Transact-SQL 扩展。 系统表 – 数据库所有者可授予对系统表的缺省权限。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
40	grant	grant	• *角色* – 当前活动角色 • *关键字或选项* – grant 语句的完整命令文本 • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

目录存储过程 sp_column_privileges, sp_table_privileges.

命令 create role, revoke, setuser, set.

函数 proc_role、 show_role.

系统过程 sp_addgroup, sp_adduser, sp_changedbowner, sp_changegroup, sp_dropgroup, sp_dropuser, sp_helpgroup, sp_helprotect, sp_helpuser, sp_role.

grant role

说明

将角色授予指定的登录名、用户、系统或用户定义的角色。

语法

```
grant role role_name
        [where pred_expression]
    to {username | rolename | login_profile_name }
```

参数

role_name

是系统安全员正授予用户或角色的系统或用户定义角色的名称。

where *pred_expression*

也称为角色激活谓词，这是激活命名的角色时必须满足的 SQL 条件。仅当向 *username* 或 *login_profile_name* 授予角色时才可使用 *pred_expression*。

如果在激活角色时 *pred_expression* 的求值结果为 FALSE， set role 命令将失败，而且 Adaptive Server 会返回一个错误消息。如果该角色为指定给自动激活的角色或是缺省角色， Adaptive Server 将不加提示地取消激活，但不会取消登录进程。

to *username* | *rolename* | *login_profile_name*

标识正在为之授予角色的登录名、角色或登录配置文件。当被授予者为登录配置文件时，具有此登录配置文件的所有用户都将被授予角色。

示例

示例 1 将 “doctor”角色授予 Mary：

```
grant role doctor_role to mary
```

示例 2 通过为 doctor_role 授予 intern_role，doctor 角色将继承 intern 角色的所有特权。

```
grant role intern_role to doctor_role
```

示例 3 具有 manage roles 特权的用户 Smith 将 nurse_role 授予用户 John，且 roleAdmin 为授予者。

```
grant role nurse_role to john
granted by roleAdmin
```

示例 4 将角色 ldap_user_role 授予登录配置文件 lp_10：

```
grant role ldap_user_role where
  get_appcontext(login_authentication) = 'LDAP'
  to login_profile lp_10
```

就以上示例而言，当分配有登录配置文件 lp_10 的用户的会话启用 ldap_user_role 时， Adaptive Server 将检查是否有使用 LDAP 连接的会话。如果已有 LDAP 连接，用户将采用 ldap_user_role；如果没有，则不会启用 ldap_user_role。通过变更登录配置文件 lp_10 并指定对 auto activated roles 属性的 ldap_user_role 来配置谓词计算，使其在登录期间自动发生。否则，当分配有 lp_10 的用户执行 set role 语句时，将会对角色激活谓词进行计算。

用法

如果 create login、 alter login、 create login profile 或 alter login profile 指定用于自动激活的角色，则当用户登录时， Adaptive Server 将 （在计算所有谓词后）自动激活授予给登录名或登录配置文件的角色。否则， Adaptive Server 将在执行 set role 时激活角色。激活相关角色时， Adaptive 将自动激活授予给其它角色的角色。

可以用 grant 命令向已被授予特定角色的所有用户授予权限。角色可以是系统角色 （如 sso_role 或 sa_role），也可以是用户定义的角色。系统安全员必须使用 create role 命令创建用户定义的角色。

权限

对 grant role 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage roles 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sso_role 的用户。要授予 sa_role，您必须是具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
85	角色	create role、 drop role、 alter role、 grant role 或 revoke role	• 角色 – 当前活动角色 • 关键字或选项 – grant role 语句的完整命令文本 • 先前值 – NULL • 当前值 – NULL • 其它信息 – NULL • 代理信息 – set proxy 有效时的初始登录名

group by 和 having 子句

说明

在 select 语句中使用，用于将表划分为组，并仅返回与 having 子句中的条件匹配的组。group by 通常与集合一起使用，以指定如何对 select 查询的未集合的列进行分组。 having 子句应用于这些组。

语法

Start of select *statement*

[group by [all] *aggregate_free_expression*
[, *aggregate_free_expression*]...]

[having *search_conditions*]

End of select *statement*

参数

group by

指定要将表划分为哪些组，并且如果集合函数包括在选择列表中，则为每个组生成一个摘要值。这些摘要值在结果中作为列出现，每一列对应一个组。在 having 子句中可以引用这些摘要列。

可在 group by 前面的选择列表中使用 avg、 count、 count_big、 max、 min 和 sum 集合函数 （该表达式通常为列名）。请参见 《参考手册：构件块》的第 2 章 “Transact-SQL 函数”。

可以按任何列的组合对表分组 即组可互相嵌套，如示例 2 所示。

all

是一个 Transact-SQL 扩展，它在结果中包含所有组，即使那些由 where 子句所排除的组也被包括进来。例如：

```
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)
```

集合列中的 “NULL” 指示了将由 where 子句排除的组。 having 子句否定 all 的含义。

aggregate_free_expression

是不包含集合的表达式。 Transact-SQL 扩展允许按不包含集合的表达式和按列名分组。

不能按列标题或别名分组。下面的示例是正确的：

```
select Price=avg (price), Pay=avg (advance),
Total=price * $1.15
from titles
group by price * $1.15
```

having

设置 group by 子句的条件，与 where 设置 select 子句的条件的方法相似。

having 搜索条件可包含集合表达式；否则， having 搜索条件与 where 搜索条件相同。下面是一个带集合的 having 子句的示例：

```
select pub_id, total = sum (total_sales)
from titles
where total_sales is not null
group by pub_id
having count(*) > 5
```

当 Adaptive Server 优化查询时，它将对 where 和 having 子句中的搜索条件求值，以确定哪些条件是可以用于选择最佳索引和查询计划的搜索参数 (SARG)。所有搜索条件都用来限定行。有关搜索参数的详细信息，请参见 《性能和调优指南：优化程序和抽象计划》。

示例

示例 1 计算每类书籍的平均预付款以及总销售额：

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

示例 2 将结果按类型分组，然后按每种类型中的 pub_id 分组：

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

示例 3 计算所有组的结果，但仅显示类型以 “p”开头的组：

```
select type, avg (price)
from titles
group by type
having type like 'p%'
```

示例 4 计算所有组的结果，但仅显示与 having 子句中的多个条件匹配的组：

```
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"
```

示例 5 连接 titles 和 publishers 表后计算每个组 （出版社）的销售额：

```
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
```

示例 6 显示预付款高于 $1000，并且价格高于所有书籍平均价格的书籍的名称：

```
select title_id, advance, price
from titles
where advance > 1000
having price > avg (price)
```

用法

- 可在 group by 后使用列名或任何表达式 （列标题或别名除外）。可以使用 group by 计算结果或显示不出现在选择列表中的列或表达式（第 438 页的 “group by 和 having 的 Transact-SQL 扩展”中介绍的 Transact-SQL 扩展）。
- 未对 group by 列 （或表达式）的最大数量明确进行限制。 group by 结果的唯一限制是 group by 列加上集合结果的宽度不超过 64K。
- group by 列中的 NULL 值将放在单个组中。
- 不能在 group by 和 having 子句中指定 text、 unitext 或 image 列。
- 不能在可更新游标的 select 语句中使用 group by 子句。
- 集合函数仅能在选择列表或 having 子句中使用。它们不能用于 where 或 group by 子句中。

  集合函数有两种类型。应用于*表中所有限定行*的集合 （每个函数为整个表产生一个值）称为*标量集合*。没有 group by 子句的选择列表中的集合函数应用于整个表；这是一个标量集合的示例。

  应用于*指定列或表达式中的一组行*的集合 （每个函数为每个组产生一个值）称为*矢量集合*。对于这两种类型的集合，集合运算的结果显示为 having 子句可以引用的一个新列。

  可将矢量集合嵌套在标量集合中。有关详细信息，请参见 《参考手册：构件块》的第 2 章 “Transact-SQL 函数”。

group by 使用优化程序

在 Adaptive Server 15.0 版中，有两种可能的算法（作为运算符实现）用于执行 group by：GroupHashing 和 GroupSorted。优化程序根据相应的因素选择要使用哪种运算符，例如这些运算符对输入数据流的要求。

GroupSorted 运算符要求要进行集合的输入行已经在组中按列排序。由于输入行必须排序，因此优化程序使用以下任一种方式：

- 使用 order by 列的索引从源表中读取行，并且 group by 列的最大宽度受到索引键最大宽度的限制，这取决于数据库页大小。
- 使用 Asort 运算符在 GroupSorted 运算符处理 group by 列中的行之前，对这些行进行排序。 group by 列和要进行集合的列必须适合放入工作表，因此 group by 列的最大宽度限制为数据库页上的最大行大小减去要进行集合的列的宽度。 group by 列的最大宽度受到数据库页大小的限制。

如果没有对 group by 列进行排序，或者超出了 GroupSorted 运算符的行大小限制，则优化程序使用 GroupHashing 运算符。 GroupHashing 运算符将一个散列函数应用于 group by 列的值，以便能够将具有相同 group by 列值的行放入相同的散列桶中。一旦输入行已全部散列到桶中，则桶中的行将进行集合以生成 group by 结果。GroupHashing 运算符的唯一限制是 group by 列和集合结果的总的行大小不能超过 64K。对于 group by 列的数量以及集合操作的数量没有限制，只对总的行宽度有限制。

使用集合的 *group by* 和 *having* 查询的工作方式

where 子句排除不符合其搜索条件的行；其功能在分组查询或非分组查询中都一样。

对于 group by 表达式中的每个唯一值， group by 子句都会将剩余的行集中到一组中。省略 group by 会为整个表创建单个组。

在选择列表中指定的集合函数计算每组的汇总值。对于标量集合来说，表只有一个值。矢量集合为不同的组计算值。

having 子句将不满足其搜索条件的组从结果中排除。尽管 having 子句仅测试行，但 group by 子句的存在与不存在会令其看起来像是在对组或行进行操作：

- 当查询中包含 group by 时，having 会排除结果组行。这就是 having 好像在对组进行操作的原因。
- 当查询不包含 group by 时， having 排除 （只有一个组的）表中的结果行。这就是 having 好像在对行进行操作的原因 （其结果与 where 子句的结果相似）。

标准 *group by* 和 *having* 查询

示例部分中的所有 group by 和 having 查询遵循 SQL 标准，该标准规定使用 group by、 having 和矢量集合函数的查询使用下列规则为每组生成一行和一个摘要值：

- 选择列表中的列必须也出现在 group by 表达式中，或者必须是集合函数的参数。
- group by 表达式只能包含选择列表中出现的列名。不过，选择列表中仅作为集合函数的参数使用的列不适用此限定。
- having 表达式中的列必须是单值的 （例如集合的参数），而且它们必须出现在选择列表或 group by 子句中。使用选择列表集合和 having 子句的查询*必须*包含 group by 子句。如果在未使用选择列表集合的查询中省略 group by，则所有未被 where 子句排除的行将被视为单个组。

在非分组查询中，“用 where 排除行”的原则是非常简单直接的。在分组查询中，此原则扩展为 “使用 where 在 group by 之前排除行，使用 having 从显示的结果中排除行”。

SQL 标准允许连接两个或更多表的查询使用 group by 和 having，只要它们遵循上述原则。指定连接或其它复杂查询时，请使用 group by 和 having 的标准语法，除非您完全理解 Transact-SQL 扩展对这两个子句的影响。

为了帮助您避免有关扩展的问题， Adaptive Server 为 set 命令提供了 fipsflagger 选项，这样每次查询中出现 Transact-SQL 扩展时都会发出一个非致命警告。有关详细信息，请参见 set。

group by 和 *having* 的 Transact-SQL 扩展

对标准 SQL 的 Transact-SQL 扩展使得数据的显示更加灵活，因为在扩展中允许引用未在创建组或摘要计算时使用的列和表达式：

- 包含集合的选择列表可以包括既不是集合函数的参数，也不包括在 group by 子句的*扩展*列。扩展列影响最终结果的显示，因为显示了附加的行。
- group by 子句可包含未列于选择列表中的列或表达式。
- group by all 子句显示所有组，甚至是那些由 where 子句从计算中排除的组。请参见 “参数”一节中关键字 all 的示例。
- having 子句可以包含未出现在选择列表以及 group by 子句中的列或表达式。

Transact-SQL 扩展将行或列添加到显示中时，或省略 group by 时，查询结果可能难以理解。以下示例可帮助您理解 Transact-SQL 扩展对查询结果的影响。

以下示例说明了使用标准 group by 和 having 子句的查询和使用 Transact-SQL 扩展的查询之间的区别：

1　标准分组查询示例：

```
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)
```

2　Transact-SQL 扩展列 price （在选择列表中，但不是集合，也不在 group by 子句中）导致所有限定行都显示在每个限定组中，即使标准 group by 子句仅为每个组生成了一个行。 group by 仍影响矢量集合，后者计算显示在每个组的每一行上的每组平均价格 （与为示例 a 计算的值相同）：

```
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 处理 Transact-SQL 扩展列的方式看起来就像一个查询忽略了 where 子句。此查询仅计算满足 where 子句的条件的行的平均价格，但它也显示与 where 子句中的条件不匹配的行。

Adaptive Server 首先使用 where 子句创建一个仅包含类型和集合值的工作表。此工作表通过分组列 type 连接回 titles 表，以在结果中包括 price 列，但连接中*没有*使用 where 子句。

titles 中唯一不会在结果中出现的行是一个 type 为“UNDECIDED”、价格为 NULL 的单独的行，也就是不会在工作表中为其输出结果的一行。如果还想在显示的结果中去除价格低于 $10.00 的行，就必须添加一个重复 where 子句的 having 子句，如示例 4 所示：

```
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　如果要在 having 子句中指定附加条件 （例如集合），则还要包括 where 子句中指定的所有条件。Adaptive Server 看起来会忽略 having 子句中所缺少的所有 where 子句条件：

```
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　这是一个使用两个表之间的连接的标准分组查询的示例。它根据 pub_id 分组，然后根据每个出版社 ID 的 type 分组，来计算每行的矢量集合：

```
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)
```

看起来好像只需要为 pub_id 和 type 列指定 group by 即可生成上述结果，下面按如下所示添加扩展列：

```
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
```

不过，上面的查询的结果与本示例中第一个查询的结果有相当大的差别。在连接这两个表以确定工作表中的矢量集合后， Adaptive Server 会将工作表与该扩展列的表 (publishers) 连接，从而生成最终结果。不同表中的每个扩展列都会调用一个附加的连接。

如您所见，在连接表的查询中使用扩展列扩展很容易产生难以理解的结果。多数情况下，请在查询中使用标准 group by 来连接表。

6 此示例使用对 group by 的 Transact-SQL 扩展来包含未出现在选择列表中的列。 pub_id 和 type 列用于对矢量集合的结果进行分组。不过，最终结果不包含每个出版社中的类型。在这种情况下，您可能只需要知道每个出版社销售多少种书籍：

```
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 此示例组合了两种 Transact-SQL 扩展效果。首先，它在选择列表中包含集合时省略 group by 子句。接着，它包含了一个扩展列。由于省略了 group by 子句：

- 整个表成为了一个组。标量集合计算得到三个限定的行。
- pub_id 成为 Transact-SQL 扩展列，因为它未出现在 group by 子句中。没有 having 子句，因此会显示组中的所有行。

```
select pub_id, count (pub_id)
from publishers

pub_id
---------- ---------
0736               3
0877               3
1389               3

 (3 rows affected)
```

8　where 子句从一个组中排除 pub_id 等于或大于 1000 的出版社，因此标量集合计算得到两个限定的行。扩展列 pub_id 显示 publishers 表中的所有限定行：

```
select pub_id, count (pub_id)
from publishers
where pub_id < "1000"

pub_id
-------------- -----------
0736                     2
0877                     2
1389                     2

 (3 rows affected)
```

9　此示例说明了不使用 group by 子句时 having 子句的效果。

- 整个表被视为一个组。没有使用 where 子句排除行，因此将对组（表）中的所有行计数。
- 此单个组表中的行通过 having 子句进行测试。
- 这些组合的效果显示了两个限定的行。

```
select pub_id, count (pub_id)
from publishers
having pub_id < "1000"

pub_id
-------------- ---------
0736                   3
0877                   3
 (2 rows affected)
```

10　此示例使用 having 的扩展，以允许使用不在选择列表和 group by 子句中的列或表达式。它确定每一书籍类型的平均价格，但会将那些总销售额未超过 $10,000 的类型排除在外，即使 sum 集合不出现在结果中也是如此。

```
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 和 *having* 及排序顺序

如果服务器的排序顺序不区分大小写， group by 将忽略分组列的大小写。例如，在不区分大小写的服务器上存在以下数据：

```
select lname, amount
from groupdemo
lname           amount
---------- ------------------
Smith               10.00
smith                5.00
SMITH                7.00
Levi                 9.00
Lé vi               20.00
```

grouping by lname 产生如下结果：

```
select lname, sum (amount)
from groupdemo

lname
---------- ----------------
Levi                9.00
Lé vi              20.00
Smith              22.00
```

相同的查询在不区分大小写和变音的服务器上产生以下结果：

```
lname
---------- -----------------
Levi                 29.00
Smith                22.00
```

标准	符合 ANSI SQL 的级别符合初级标准。 在 select 列表中使用不在 group by 列表中且不包含集合函数的列是一种 Transact-SQL 扩展。 使用 all 关键字是一种 Transact-SQL 扩展。
另请参见	**命令** compute clause, declare, select, where 子句. **文档** 《参考手册：构件块》的第 2 章 “Transact-SQL 函数”。 .

if...else

说明

施加 SQL 语句的执行条件。

语法

if *logical_expression* [plan "*abstract plan*"]
statements

[else
[if *logical_expression*] [plan "*abstract plan*"]
statement]

参数

logical_expression

是一个返回 TRUE、 FALSE 或 NULL 的表达式 （列名、常量、任何由算术运算符或逐位运算符连接起来的列名和常量的组合，也可以是子查询）。如果该表达式包含 select 语句，则必须将 select 语句用小括号括起来。

plan "*abstract plan*"

指定用于优化查询的抽象计划。它可以是用抽象计划语言指定的完整或部分计划。只能为可优化的 SQL 语句 （即访问表的 select 查询）指定计划。请参见《性能和调优指南：优化程序和抽象计划》中的 “创建和使用抽象计划”。

语句

要么是单个 SQL 语句，要么是由 begin 和 end 分隔的语句块。

示例

示例 1 如果 3 大于 2，则输出 “yes”：

```
if 3 > 2
    print "yes"
```

示例 2 if...else 条件检查邮政编码为 94705 的作者是否存在，如果存在，则将 “Berkeley author”输出到结果集：

```
if exists (select postalcode from authors
    where postalcode = "94705")
    print "Berkeley author"
```

示例 3 if...else 条件检查数据库中是否存在用户创建的对象 （所有 ID 号大于 100 的对象）。如果用户表存在， else 子句输出一条消息并选择表的名称、类型和 ID 号：

```
if (select max(id) from sysobjects) < 100
    print "No user-created objects in this database"
else
 begin
    print "These are the user-created objects"
    select name, type, id
    from sysobjects
    where id > 100
 end
```

示例 4 因为 titles 表中 PC9999 的总销售额为 NULL，所以此查询返回 FALSE。当查询的 if 部分返回 FALSE 或 NULL 时，将执行其 else 部分。有关真值与逻辑表达式的详细信息，请参见 “表达式”《参考手册：构件块》中的第 4 章 “表达式、标识符和通配符”。

```
if (select total_sales
    from titles
    where title_id = "PC9999") > 100
select "true"
else
select "false"
```

用法

- 如果 if 关键字的条件满足 （逻辑表达式返回 TRUE），则将执行此关键字及其条件后的语句。可选的 else 关键字引入一个在 if 条件不能满足时 （逻辑表达式返回 FALSE）执行的替代 SQL 语句。
- if 或 else 条件仅影响单个 SQL 语句的性能，除非语句被分组到关键字 begin 与 end 之间的块中 （请参见示例 3）。

  statement 子句可以是 execute 命令，也可以是任何其它合法的 SQL 语句或语句块。
- 如果将 select 语句用作布尔表达式的一部分，则它必须返回单个值。
- if...else 结构既可用于存储过程中 （经常用于测试特定参数是否存在），又可用于*即席* 查询中 （请参见示例 1 和 2）。
- if 测试可嵌套于另一个 if 中或 else 后。根据每个 if...else 结构所包括的 select 语句 （或其它语言结构）的复杂性，可嵌套 if 测试的最大数目也不同。

  注释　当 if...else 块中出现 alter table、 create table 或 create view 命令时， Adaptive Server 将在确定条件是否为真之前创建表或视图的模式。如果表或视图已经存在，则会导致出错。
- 如果创建具有 varchar、 nvarchar、 univarchar 或 varbinary 列的表，而列的总定义宽度大于所允许的行宽，则会出现一条警告消息，但仍然创建该表。如果试图在这样的行中插入超过最大数量的字节，或对行执行 update 命令，使其总行宽大于最大长度， Adaptive Server 将产生一条错误消息，且命令失败。

注释　当 create table 命令在 if...else 块或 while 循环中出现时， Adaptive Server 将在确定条件是否为真之前创建表模式。如果该表已经存在，则会导致错误。为避免这种情况，要么确保数据库中不存在具有相同名称的视图，要么按如下方式使用 execute 语句：

```
if not exists
     (select * from sysobjects where name="my table")
begin
execute ("create table mytable (x int)")
end
```

标准	符合 ANSI SQL 的级别Transact-SQL 扩展。
权限	使用 if...else 无需任何权限。
另请参见	**命令** begin...end, create procedure.

insert

说明

向表或视图中添加新行。

语法

```
insert [into] [database.[owner.]]{table_name | view_name}
        [(column_list)]
        {values (expression [, expression]...)
            | select_statement [plan "abstract plan"]}
```

参数

into

是可选的。

table_name | *view_name*

是要从中删除行的表或视图的名称。如果该表或视图位于另一数据库中，请指定该数据库名称；如果在数据库中有多个具有该名称的表或视图，请指定所有者的名称。 *owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

column_list

是要向其中添加数据的列的列表。请将此列表用小括号括起来。这些列可以按任何顺序列出，但传入的数据 （无论是在 values 子句中还是在 select 子句中）的顺序必须与之相同。如果一个列具有 IDENTITY 属性，则可用 syb_identity 关键字代替实际列名。

当表中仅有部分 （而不是全部）列要接收数据时，必须提供列列表。如果未给出列列表， Adaptive Server 会假定 insert 针对接收表中的所有列进行 （顺序与 create table 的顺序相同）。

有关详细信息，请参见第 450 页的 “列列表”。

values

引入一组表达式。

expression

为指定列指定常量表达式、变量、参数或空值。请将字符和日期常量用单引号或双引号引起来。

不能将子查询作为 *expression* 使用。

值列表：

- 必须用小括号括起来
- 必须与显式或隐式列列表匹配
- 可以将 “default” 作为值使用

请参见第 1 章 “系统数据类型和用户定义的数据类型”《参考手册：过程》中的。

select_statement

是标准 select 语句，用于检索要插入的值。

plan "*abstract plan*"

指定用于优化查询的抽象计划。它可以是用抽象计划语言指定的完整或部分计划。仅能为 insert...select 语句指定计划。有关详细信息，请参见 《性能和调优指南：优化程序和抽象计划》中的 “创建和使用抽象计划”。

示例

示例 1

```
insert titles
values ("BU2222", "Faster!", "business", "1389",
    null, null, null, "ok", "06/17/87", 0)
```

示例 2

```
insert titles
 (title_id, title, type, pub_id, notes, pubdate,
    contract)
values ('BU1237', 'Get Going!', 'business',
    '1389', 'great', '06/18/86', 1)
```

示例 3

```
insert newauthors
    select *
    from authors
    where city = "San Francisco"
```

示例 4

```
insert test
    select *
    from test
    where city = "San Francisco"
```

示例 5

```
insert table1 (col1, col2, col3, col4)
    values (10, 4, default, 34)
```

用法

- 仅将 insert 用于添加新行。使用 update 修改已插入的行中的列值。

列列表

列列表确定值的输入顺序。例如，假定您有一个名为 newpublishers 的表，其结构和内容与 pubs2 中的 publishers 表完全相同。在下面的示例中，newpublishers 表的列列表中的列与 publishers 表中选择列表的列匹配。

```
insert newpublishers (pub_id, pub_name)
select pub_id, pub_name
```

```
    from publishers
    where pub_name="New Age Data"
```

“New Age Data”的 pub_id 和 pub_name 被存储在 newpublishers 的 pub_id 和 pub_name 列中。

在下一个示例中，newpublishers 表的列列表中列的顺序与 publishers 表的选择列表中列的顺序不匹配。

```
insert newpublishers (pub_id, pub_name)
    select pub_name, pub_id
    from publishers
    where pub_name="New Age Data"
```

结果是：“New Age Data”的 pub_id 被存储在 newpublishers 表的 pub_name 列中，而 “New Age Data”的 pub_name 被存储在 newpublishers 表的 pub_id 列中。

可以省略列或值列表中的项，只要省略的列允许空值即可 （请参见示例 2）。

验证列值

- insert 与用 create index 命令设置的 ignore_dup_key、ignore_dup_row 和 allow_dup_row 选项交互作用。有关详细信息，请参见 create index。
- 规则或 check 约束可限制可以输入列中的合法值的域。规则用 create rule 命令创建并用 sp_bindrule 绑定。 check 约束用 create table 声明。
- 如果不显式输入值，则可提供一个缺省值。缺省值用 create default 命令创建并用 sp_bindefault 绑定，或者也可用 create table 声明。
- 如果 insert 语句违反了域或完整性规则 （请参见 create rule 和 create trigger），或者如果数据类型有误 （请参见 《参考手册：构件块》中的 create table 和第 1 章 “系统数据类型和用户定义的数据类型”），语句将失败， Adaptive Server 将显示一条错误消息。

处理空白

- 向变量字符类型或 text 列插入空字符串 ("") 时插入的是单个空格。char 列会被填补为所定义的长度。
- 所有插入到 varchar 和 univarchar 列中的数据的尾随空格都会被删除，除非字符串仅包含空格。仅包含空格的字符串会被截断为单个空格。自动截断比 char、 nchar、 unichar、 univarchar、 varchar 或 nvarchar 列指定长度长的字符串，除非将 string_rtruncation 选项设置为 on。

插入 *text*、*unitext* 和 *image* 列

将 NULL insert 到 text、ortext 或 image 列只是为文本指针分配了空间。请使用 update 为该列获取有效的文本指针。

insert 触发器

可以定义一个触发器，当对指定表发出 insert 命令时，该触发器执行指定操作。

在启用 CIS 时使用 *insert*

可将 insert 作为语言事件或参数化动态语句发送到远程服务器。

插入从其它表中选择的行

可以在单个语句中从表中选择行并将它们插入到同一表中 （请参见示例 4）。

若要用 select 将数据从某些字段为空值的表中插入不允许空值的表中，请为初始表的任何 NULL 条目提供替代值。例如，若要将数据插入不允许空值的 advances 表中，可用 “0”替代 NULL 字段：

```
insert advances
select pub_id, isnull (advance, 0) from titles
```

如果不使用 isnull 函数，此命令会将所有含非空值的行插入到 advances 表中，并对在 titles 表的 advance 列中包含 NULL 的所有行产生错误消息。

如果无法对数据进行这种替代，则不能将包含 null 值的数据插入已有 not null 规定的列。

两个表可以在结构上相同，但在某些字段是否允许有空值这一点上是不同的。使用 sp_help 可查看表中的列的空类型。

事务和 *insert*

设置链式事务模式后，如果当前没有活动的事务， Adaptive Server 将隐式地用 insert 语句启动事务。若要完成插入，则必须提交事务，或者回退更改。例如：

```
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
```

在链式事务模式下，此批处理会启动一个事务并向 stores 表中插入一个新行。如果插入与表中另一商店包含相同州和城市信息的行，则将回退对 stores 进行的更改并结束该事务。否则它提交插入并结束事务。有关链式事务模式的详细信息，请参见 《Transact-SQL 用户指南》。

向 IDENTITY 列插入值

- 向表中插入一行时，不要在列列表中包括 IDENTITY 列的名称，也不要在值列表中包括 IDENTITY 列的值。如果表仅包含一列（IDENTITY 列），应省略列列表并将值列表留空，如下所示：

  ```
  insert id_table values ()
  ```

- 首次向表中插入行时， Adaptive Server 为 IDENTITY 列指派值 1。每个新行的列值都比上一个值增加 1。此值优先于在 create table 或 alter table 语句中为列声明的或用 sp_bindefault 绑定到列的任何缺省值。

  服务器故障会使 IDENTITY 列值产生间隔。间隔的最大大小取决于 identity burning set factor 配置参数的设置。间隔也有可能来源于对 IDENTITY 列进行的手动插入、删除行或事务回退。

- 为列的基表设置 identity_insert *table_name* on 后，表所有者或对表拥有 insert 权限的用户可以向 IDENTITY 列显式插入值。只有表所有者或对表拥有 identity_insert 权限的用户可对表设置 identity_insert table_name on。用户可以在数据库中每次为一个表设置 identity_insert *table_name* on。identity_insert 为 on 时，每个 insert 语句都必须包含一个列列表，并为 IDENTITY 列指定显式值。

  通过在 IDENTITY 列中插入值，可以指定该列的源值或恢复误删除的行。除非创建了 IDENTITY 列的唯一索引，否则 Adaptive Server 不会检验插入的值的唯一性；您可以插入任何正整数。

  要向 IDENTITY 列插入显式值，表的所有者、数据库所有者或系统管理员必须为列的基表 （而不是用于插入表的视图）设置 identity_insert *table_name* on。

- 可插入 IDENTITY 列的最大值为数值 $10^{precision} - 1$。对于整数标识，最大值为其类型的最大允许值 （例如对于 tinyint 为 255，对于 smallint 为 32767）。一旦 IDENTITY 列达到此值，再执行任何 insert 语句都将返回中止当前事务的错误。

  如果发生这种情况，请使用 create table 语句创建一个与旧表相同但 IDENTITY 列的精度更大的表。一旦创建完新表，就应使用 insert 语句或 bcp 实用程序将数据从旧表复制到新表。

- 可使用 @@*identity* 全局变量检索插入 IDENTITY 列的最后一个值。如果最后一条 insert 或 select into 语句影响到不含 IDENTITY 列的表， @@*identity* 返回值 0。
- 选入结果表中的 IDENTITY 列遵循下列与 IDENTITY 属性继承相关的规则：
  - 如果多次选择 IDENTITY 列，它将在新表中定义为 not null。该 IDENTITY 列将不继承 IDENTITY 属性。
  - 如果选择 IDENTITY 列作为表达式的一部分，结果列不会继承 IDENTITY 属性。如果表达式中的任何列允许使用 null，则它创建为 null；否则，将它创建为 not null。
  - 如果 select 语句包含 group by 子句或集合函数，结果列不会继承 IDENTITY 属性。包含 IDENTITY 列的集合的列将创建为 null；其它列则创建为 not null。
  - 通过联合或连接选入表中的 IDENTITY 列不保留 IDENTITY 属性。如果表中包含 IDENTITY 列和 null 列的联合，则新列将定义为 null；否则定义为 not null。

通过视图插入数据

- 如果视图是用 with check option 创建的，则通过视图插入的每一行都必须符合视图的选择标准。

  例如， stores_cal 视图包括 stores 表中 state 值为 “CA”的所有行：

  ```
  create view stores_cal
  as select * from stores
  where state = "CA"
  with check option
  ```

  with check option 子句根据视图的选择标准检查每个 insert 语句。state 值不是 “CA”的行将被拒绝。

- 如果视图是用 with check option 创建的，则所有从*基*视图派生的视图都必须满足视图的选择标准。通过派生视图插入的每个新行都必须能通过基视图查看。

  以从 stores_cal 派生的视图 stores_cal30 为例。此新视图包含位于 California 且付款方式为 “Net 30”的商店的有关信息：

  ```
  create view stores_cal30
  as select * from stores_cal
  where payterms = "Net 30"
  ```

因为 stores_cal 是通过 with check option 创建的，所以通过 stores_cal30 插入或更新的行都必须可以通过 stores_cal 查看。 state 值不是 “CA”的任何行都会被拒绝。

stores_cal30 本身并没有 with check option 子句。这意味着可以通过 stores_cal30 插入或更新 payterms 值不是 “Net 30”的行。即使再也无法通过 stores_cal30 查看该行，以下 update 语句仍会成功：

```
update stores_cal30
set payterms = "Net 60"
where stor_id = "7067"
```

- 在用 with check option 创建的连接视图中不允许使用 insert 语句。
- 如果通过连接视图来插入或更新行，所有受影响的列都必须属于同一基表。

对表进行分区以提高插入性能

无聚簇索引的未分区表由一个双重链接的数据库页链组成，因此每次对表进行插入都使用链的最后一页。 Adaptive Server 在插入行时在最后一页上持有一个排它锁，阻止其它并发事务向表中插入数据。

使用 alter table 命令的 partition 子句对表进行分区会创建额外的页链。每个链都有自己的最后一页，可用于并发插入操作。这通过减少页争用提高了插入性能。如果表分布在多个物理设备上，则分区可减少服务器将数据从高速缓存刷新到磁盘时的 I/O 争用，从而可提高插入性能。有关对表进行分区以提高插入性能的详细信息，请参见 《性能和调优指南：基础知识》中的 “控制物理数据放置”。

标准

符合 ANSI SQL 的级别符合初级- 标准。

以下是 Transact-SQL 扩展：

- insert 语句选择部分中的 union 运算符。
- 用数据库名对表名或列名的限定。
- 通过包含连接的视图进行插入。

注释　注意：FIPS 标志程序不检查通过包含连接的视图进行的插入。

权限

对 insert 函数的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或对表拥有 insert 权限的用户。只有表所有者或拥有 insert 和 identity_insert 权限的用户可以对表的 IDENTITY 列进行插入操作。

细化权限已禁用	在禁用细化权限的情况下，您必须是表所有者或具有 sa_role 的用户。 insert 权限缺省情况下将授予表所有者。 表的 IDENTITY 列的 insert 权限仅授予表的所有者、数据库所有者和系统管理员。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息

41	insert	向表中插入数据的 insert 命令	• *Roles* – 当前活动角色
			• *关键字或选项* – if：
			  • insert – INSERT
			  • select into – INSERT INTO，后跟完全限定对象名
			• *先前值* – NULL
			• *当前值* – NULL
			• *其它信息* – NULL
			• *代理信息* – set proxy 有效时的初始登录名

42	insert	向视图中插入数据的 insert 命令	• *Roles* – 当前活动角色 • *关键字或选项* – INSERT • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 alter table, create default, create index, create rule, create table, create trigger, dbcc, delete, select, update.

数据类型 《参考手册：构件块》的第 1 章 “系统数据类型和用户定义的数据类型”。 .

系统过程 sp_bindefault, sp_bindrule, sp_help, sp_helpartition, sp_unbindefault, sp_unbindrule.

实用程序 bcp.

kill

说明　注销进程。

语法　kill *spid* with statusonly

参数

spid

是想要注销的进程的标识号。*spid* 必须是常量；它不可作为参数传送到存储过程，也不可作为局部变量使用。可用 sp_who 查看进程列表及其它信息。

with statusonly

报告处于回退状态的服务器进程 ID (spid) 的进度。它并不终止 spid。statusonly 报告显示已完成回退的百分比以及距离回退完成的估计时间长度 （以秒为单位）。

示例

示例 1 注销进程号 1378：

```
kill 1378
```

示例 2 报告 spid 号为 13 的回退过程：

```
kill 13 with statusonly
spid:13 Transaction rollback in progress.Estimated rollback completion:17%
Estimated time left:13 seconds
```

若要跟踪回滚的进度，必须多次运行 kill...with statusonly。如果在发出 kill...statusonly 时 spid 的回退已经完成，或者 Adaptive Server 没有回退指定的 spid，则 kill...statusonly 将返回以下消息：

```
Status report cannot be obtained.KILL spid:nn is not
in progress.
```

用法

- 执行 sp_who 可获得关于当前进程的报告，如：

```
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 列包含 Transact-SQL kill 命令中使用的进程标识号。blk 列包含阻塞进程的进程 ID （如果存在的话）。阻塞进程 （可能持有排它锁）是持有其它进程所需的资源的进程。在本例中，进程 10 （对表执行的 select）被进程 7 （后跟对同一表执行 insert 的 begin transaction）阻塞。

status 列报告命令的状态。表 1-24 显示了 sp_who 的状态值和效果：

表 1-24: sp_who 报告的状态值

Status	说明	kill 命令的效果
recv sleep	等待网络读取。	立即。
send sleep	等待网络发送。	立即。
alarm sleep	等待警报，例如 waitfor delay "10:00"。	立即。
lock sleep	等待获取锁。	立即。
sleeping	等待磁盘 I/O 或某种其它资源。可能说明某个进程正在运行但在执行大量磁盘 I/O。	进程通常会在 “唤醒” 时立即被注销。少数休眠进程不会唤醒，需要重新启动 Adaptive Server 才能将其清除。
runnable	在可运行进程的队列中。	立即。
running	活跃地运行在一个服务器引擎中。	立即。
infected	Adaptive Server 已检测到严重的错误情况；极少见。	不建议使用 kill 命令。可能需要重新启动 Adaptive Server 以清除进程。
background	由 Adaptive Server 而不是用户运行的进程，例如阈值进程。	立即；使用 kill 时必须格外小心。建议在注销后台进程之前，仔细检查 sysprocesses。
log suspend	到达日志的最后机会阈值时挂起进程。	立即。

- 若要获得有关当前锁及持有锁的进程的 *spid* 的报告，请使用 sp_lock。
- 在集群环境中，具有特权的 Kerberos 用户可以使用 kill 命令停止远程实例上的 DBMS 任务的 spid。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 kill 的权限检查因您的细化权限设置而异。

细化权限已启用	在已启用细化权限的情况下，如果要关闭自己的进程，您必须拥有 kill 特权，而要关闭其他用户的进程，则必须拥有 kill any process 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。 kill 特权不能移交。

另请参见

命令 shutdown.

系统过程 sp_lock, sp_who.

load database

说明

装载使用 dump database 创建的用户数据库的备份副本，包括其事务日志，并实现通过数据库转储装载的存档数据库。

load database 操作的目标平台不必与执行 dump database 操作的源平台是同一平台。可从大型平台向小型平台执行 dump database 和 load database，或者从小型平台向大型平台执行。

有关站点具有 Tivoli Storage Manager 使用许可时的 load database 语法，请参见 《将 Backup Server 与 IBM Tivoli Storage Manager 配合使用》。

语法

进行例行数据库装载：

```
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]}]]
```

返回标头或文件信息但不装载备份：

```
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}
        }]]
```

实现存档数据库：

```
load database database_name
    from dump_device
    [ [stripe on stripe_device] ...]
    [with [norecovery,][passwd=password]
```

生成一系列 load database SQL 语句，以将数据库恢复到指定时间点的状态：

```
load database database_name
    [from stripe_device]
    with listonly=[load_sql | create_sql | volume]
```

如站点已获得 Tivoli Storage Manager 许可则装载数据库的副本：

```
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]]
        } ]
```

参数

database_name

用于接收备份副本的数据库的名称。可以是通过 for load 选项创建的数据库，也可以是现有数据库。将转储的数据装载到现有的数据库将覆盖所有现有数据。接收数据库必须至少与转储数据库一样大。可将数据库名称指定为文字、局部变量或存储过程参数。

对于存档数据库，*database_name* 是要装载到其中的存档数据库的名称。

compress:

调用对存档数据库的解压缩。有关 compress 选项的详细信息，请参见 《系统管理指南》中的 “备份和恢复用户数据库”。

注释 Sybase 建议首选使用本机 "compression = *compress_level*" 选项，旧选项 "compress::*compression_level*" 其次。如果对 dump database 使用了该本机选项，则在装载数据库时无需使用 "compress::*compression_level*"。

---

from *dump_device*

指定要从其中装载转储的磁盘数据库转储的名称。

from *stripe_device*

是从中装载数据的设备。有关指定转储设备时使用何种形式的信息，请参见第 485 页的 “指定转储设备”。有关支持的转储设备的列表，请参见 Adaptive Server 安装和配置指南。

at *backup_server_name*

在转储设备附加到的计算机上运行的远程 Backup Server 的名称。对于使用 interfaces 文件的平台， *backup_server_name* 必须出现在 interfaces 文件中。

listonly = [load_sql | create_sql | volume]

生成以下命令：

- load_sql – 用于执行恢复到指定时间点操作的 load database 或 load transaction SQL 命令序列。
- create_sql – 显示 disk init/sp_cacheconfig、 disk init、 create/alter database 序列和从备份历史记录中获取的最新转储映像的 create/alter database 序列。

  在从分条设备装载数据库时，如果使用 with listonly=create_sql，则该选项将显示转储映像的 disk init/sp_cacheconfig、 disk init、 create 或 alter database 序列。
- volume – 显示转储映像的卷信息。

density = *density_value*

忽略。请参见 dump database 命令。

blocksize = *number_bytes*

替换转储设备的缺省块大小。如果在 UNIX 系统上指定块大小，它应该与用来进行转储的块大小相同。请参见 dump database 命令。

dumpvolume = *volume_name*

是 ANSI 磁带标签的卷名字段。打开磁带时， load database 会检查此标签，如果装载了错误的卷，则会生成错误消息。

注释　使用 load database 时，如果为 file=*filename* 选项提供了错误的文件名，则 dumpvolume 选项不会提供错误消息。即使安装了错误的磁带，备份服务器也会搜索整个磁带来查找该文件。

---

file = *file_name*

是磁带卷上特定数据库转储的名称。如果在生成转储时未记录转储文件名，可使用 listonly 显示有关所有转储文件的信息。

stripe on *stripe_device*

是附加的转储设备。最多可以使用 32 个设备，其中包括在 to *stripe_device* 子句中指定的设备。 Backup Server 同时从所有设备装载数据，从而减少了所需时间和卷更改次数。有关详细信息，请参见第 485 页的 “指定转储设备”。

dismount | nodismount

（在支持逻辑卸下的平台上）确定磁带是否保持装入状态。缺省情况下，在装载完成时将卸下装载所用的全部磁带。使用 nodismount 命令可使磁带供其它装载或转储使用。

nounload | unload

确定装载完成后是否回绕磁带。缺省情况下磁带不回绕，从而使您可以从同一磁带卷进行更多的装载。为要从多转储卷装载的最后一个转储文件指定 unload。这样，在装载完成后就会回绕并卸载磁带。

with [norecovery,]

实现存档数据库时指示 load database 命令将不运行恢复，以及在完成 load database 命令后自动使数据库联机。

passwd = *password*

是您提供的口令，用来防止转储文件被未经授权的用户使用。口令长度必须介于 6 到 30 个字符之间。口令不能使用变量。有关口令规则，请参见《系统管理指南：卷 1》中的“管理 Adaptive Server 登录、数据库用户和客户端连接”。

until_time = *datetime*

生成要装载 （到指定时间点）的装载序列。

listonly [= full]

显示有关磁带卷上所有转储文件的信息，但*不装载数据库*。 listonly 标识数据库和设备、转储日期和时间以及转储可被覆盖的日期和时间。 listonly = full 提供有关转储的其它详细信息。两个报告都按 ANSI 磁带标签进行排序。

列出卷上的文件之后，Backup Server 发出卷更改请求。操作员可以安装另一个磁带卷，也可以终止所有转储设备的列表操作。

在当前环境中， listonly 选项会替换 headeronly 选项。

警告！ 请不要对 1/4 英寸盒式磁带使用 load database with listonly。

with verify[only][=header | full]

在数据页复制到存档中时对数据页执行最低限度的标头或结构行检查，但是**does not load the database**。此时不会对 gam、 oam、分配页、索引、文本或日志页进行结构检查。唯一的其它检查在页码与页头匹配的页上执行。

headeronly

显示单个转储文件的标头信息，但**不装载数据库**。 headeronly 显示有关磁带上第一个文件的信息，除非使用 file = *file_name* 选项另指定一个文件名。转储标头表示：

- 转储类型 （数据库或事务日志）
- 数据库 ID
- 文件名
- 执行转储的日期
- 字符集
- 排序顺序
- 页数
- 下一个对象 ID

notify = {client | operator_console}

替换缺省的消息显示目标。

- 在提供操作员终端功能的操作系统上，始终会将卷更改消息发送到运行 Backup Server 的计算机的操作员终端上。使用 client 可将其它 Backup Server 消息发送到启动 dump database 的终端会话。
- 在不提供操作员终端功能的操作系统 （如 UNIX）上，消息将发送到启动 dump database 的客户端。使用 operator_console 将消息发送到运行 Backup Server 的终端。

override

如果数据库包含用于对其它数据库中的列进行加密的加密密钥，则必须使用 with override 才能成功装载该数据库。

syb_tsm::*object_name*

是调用 libsyb_tsm.so 模块的关键字，该模块用于实现 Backup Server 和 TSM 之间的通信。

-S *source_server_name*

源 Adaptive Server 与目标 Adaptive Server 不同时，指定源 Adaptive Server 的名称。装载操作的目标服务器与用于转储操作的源服务器不同时，该参数是必需的。

-D *source_database_name*

源数据库与目标数据库不同时，指定源数据库的名称。装载操作的目标数据库与转储操作的源数据库不同时，该参数是必需的。

示例

示例 1 从磁带设备重新装载数据库 pubs2：

```
load database pubs2
    from "/dev/nrmt0"
```

示例 2 使用 Backup Server REMOTE_BKP_SERVER 装载 pubs2 数据库。该命令指定三个设备：

```
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
```

示例 3 从位于 */opt/bin/Sybase/dumps* 上的名为 *dmp090100.dmp* 的压缩转储文件装载 pubs2 数据库：

```
load database pubs2 from
    "compress::/opt/bin/Sybase/dumps/dmp090100.dmp"
```

示例 4 装载包含加密密钥的 key_db 数据库。如果 key_db 中的加密密钥用于对其它数据库中的列进行加密，则必须使用 with override：

```
load database key_db from "/tmp/key_db.dat" with
override
```

示例 5 从 “syb_tsm::obj1.2” 装载 testdb 数据库。有关相关联的 dump 命令，请参见第 352 页的 “dump database”。

```
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"
```

示例 6 相关联 dump 命令的源数据库 (testdb) 与 load 命令的目标数据库 (pubs2) 不同时，从 TSM 备份对象 “obj1.1” 装载 pubs2 数据库。

```
load database pubs2 from "syb_tsm::-D testdb::obj1.1"
```

示例 7 在五个名为 testdata1、testdata2、testdata3、testlog4 和 testlog5 的数据库设备上创建数据库 testdb：

```
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
```

数据库 testdb 的转储是通过 dump database 获取的，此命令可在转储标头中写入附加数据库设备信息。

```
dump database testdb to "test.dmp"
go
```

将转储映像 *test.dmp* 与 with headeronly 选项配合使用来装载数据库 testdb 会导致显示转储标头内容。这将导致显示有关数据库设备的其它信息：

```
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
…..
```

数据库设备信息包括 vdevno、 devname、 path、 db_size 和 device_size。device_size 是执行 disk init 命令时所分配的设备总大小。db_size 是数据库 testdb 使用的设备大小。

将转储图像 *test.dmp* 与 create_sqlgenddlonly 选项配合使用以装载数据库 testdb 时将显示 create/alter database 命令的序列，可通过这些命令创建数据/日志段布局与执行 dump 命令时的源数据库相同的目标数据库。可将该输出发送至文件，以便生成用于创建目标数据库的 isql 命令脚本。

```
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
```

示例 8显示使用最新可用转储对特定数据库进行恢复所需的装载命令序列。读取来自转储历史记录文件的转储记录，以准备装载序列：

```
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
```

用法

- 如果在使用 sp_hidetext 之后执行跨平台的 dump 和 load，则必须手动删除并重新创建所有隐藏对象。
- listonly 和 headeronly 选项显示有关转储文件的信息，但不装载这些转储文件。
- 转储和装载通过 Backup Server 执行。
- 若要确保正确同步数据库，以使所有代理表对于您刚重装的主数据库的内容都拥有正确的模式，您可能需要在承载代理数据库的服务器上运行 alter database *dbname* for proxy_update 命令。
- 表 1-25 描述了用于从备份中恢复数据库的命令和系统过程：

表 1-25: 用于从转储恢复数据库的命令

使用此命令	要进行此操作
create database for load	创建一个用于装载转储的数据库。
load database	从转储中恢复数据库。
load transaction	对恢复的数据库应用最近的事务。
online database	在完成常规装载序列或将数据库升级到最新版本的 Adaptive Server 后，使公共用户可以访问数据库。
load {database | transaction} with {headeronly | listonly}	标识磁带上的转储文件。
sp_volchanged	响应 Backup Server 卷更改消息。

- 有关装载包含加密列的数据库的详细信息，请参见 dump database 部分中的第 368 页的 “加密列和 dump database”。

限制

- 仅限 CIS – 数据库的所有代理表都是数据库保存集的一部分。代理表的内容数据不会保存；仅保存和恢复指针。
- 如果转储操作是在其它平台上进行的，则不能装载该转储。
- 不能装载在版本低于 12.5.4 的服务器上生成的转储。
- 如果数据库具有跨数据库的参照完整性约束，则 sysreferences 系统表存储外部数据库的*名称*（而不是 ID 号）。如果您使用 load database 命令更改数据库名称或将其装载到其它服务器上，则 Adaptive Server 无法保证参照完整性。

- 每次添加或删除跨数据库约束或者删除包含跨数据库约束的表时，都请转储*上述两个*受影响的数据库。

  警告！装载这些数据库的早期转储会导致数据库损坏。在转储数据库以用不同的名称装载它或将其转移到另一个 Adaptive Server 上之前，请使用 alter table 删除所有外部参照完整性约束。

- load database 清除 master..sysattributes 中属于装载的数据库的可疑页条目。
- load database 覆盖数据库中所有现有数据。
- 装载数据库转储之后，可能需要一些时间来进行下面两种处理，然后才能将数据库置于联机状态：
  - Backup Server 清零源数据库的空间映射中的未分配页。该清零操作作为物理装载的一部分嵌入，并且在装载数据库期间发生。

    如果目标数据库大于源数据库，则在 Backup Server 完成装载后，源数据库的空间映射上限以上的空间将由 Adaptive Server 清零。
  - 恢复操作忽略在由 dump database 在其操作开始时写入的检查点之前完成的事务。事务日志活动部分中的已完成事务由恢复操作前滚。执行 online database 时，在装载序列中，对未完成事务的回退发生在该序列结束时。
- 接收数据库必须等于或大于要装载的数据库。如果接收数据库太小， Adaptive Server 会显示一条错误消息，给出所需大小。
- 不能从空设备 （UNIX 上为 /dev/null）装载。
- 不能在用户定义的事务中使用 load database。
- 一旦装载数据库，在执行 load database 和 online database 命令时，Adaptive Server 在转储文件上自动标识规模类型并执行所有必要的转换。

  Adaptive Server 转换索引行后，索引行的顺序可能不正确。在执行 online database 的过程中， Adaptive Server 将用户表上的下列索引标记为可疑索引：
  - APL （所有页锁定）表上的非聚簇索引
  - DOL （仅数据锁定）表上的聚簇索引
  - DOL 表上的非聚簇索引

  执行跨平台转储和装载操作的过程中，可疑分区的处理方式如下：

- 在跨两个字节顺序类型不同的平台执行 load database 后，首次执行 online database 命令的过程中，散列分区被标记为可疑分区。
- 对于具有使用 unichar 或 univarchar 分区键在内部生成的分区条件的循环分区，上面的所有全局聚簇索引均被标记为可疑索引。
- 数据库联机后，使用 sp_post_xpload 可修复可疑分区和索引。

注释　有关使用 sp_post_xpload 存储过程检查和重建用户表的索引的信息，请参见 《参考手册：过程》中的 “系统过程”。

- dump transaction 和 load transaction 不允许跨平台使用。
- 对远程 backupserver 执行 dump database 和 load database操作不支持跨平台进行。
- 不能跨平台装载有口令保护的转储文件。
- 如果对已分析的 XML 对象执行 dump database 和 load database，则在完成 load database 命令之后，必须重新分析该文本。
- 您无法在早于 11.9 的 Adaptive Servers 版本上跨平台执行 dump database 和 load database。
- Adaptive Server 不能转换存储为 binary、 varbinary 或 image 列的嵌入数据结构。
- 不允许对 master 数据库执行跨平台的 load database 操作。
- 执行 load database 之后，存储过程和其它编译对象在初次执行时需要从 syscomments 中的 SQL 文本重新编译。

  如果您没有从文本进行重新编译的权限，则具有该权限的人必须使用 dbcc upgrade_object 命令从文本进行重新编译以升级这些对象。

装载时将用户锁在外面

- 不能使用装载中的数据库。 load database 将数据库的状态设置为 “offline”。数据库处于 “offline”状态时，无人可使用该数据库。“offline”状态可防止用户在装载序列期间对数据库进行访问和更改。
- 在发出 online database 之前，用 load database 装载的数据库一直保持不可访问状态。

升级数据库和事务日志转储

- 要将用户数据库转储从 11.9 或更高版本的服务器恢复并升级到最新版本的 Adaptive Server，可执行如下操作：

  a　装载最近的数据库转储。

b *按顺序*装载自上次数据库转储以来生成的所有事务日志转储。

Adaptive Server 检查每个转储上的时间戳以确保按正确的顺序将其装载到正确的数据库中。

c 发出 online database 进行升级并使数据库可供公共用户使用。

d 升级后立即转储新升级的数据库，以创建与当前版本的 Adaptive Server 一致的转储。

指定转储设备

- 可以将转储设备指定为文字、局部变量或存储过程参数。
- 可将本地设备指定为：
  - sysdevices 系统表中的逻辑设备名称
  - 绝对路径名
  - 相对路径名

  Backup Server 使用 Adaptive Server 中的当前工作目录解析相对路径名。
- 通过网络装载时，请指定转储设备的绝对路径名。路径名在运行 Backup Server 的计算机上必须是有效的。如果名称包含除字母、数字或下划线 (_) 以外的字符，则必须用引号将整个名称引起来。
- 转储设备上的所有权和权限问题可能影响 load 命令的使用。
- 可以同时运行多个装载 （或转储）命令，只要每个装载命令使用不同的物理设备即可。

Backup Servers

- Backup Server 必须与 Adaptive Server 在同一台计算机上运行。Backup Server 必须在 master..sysservers 表中列出。此条目是在安装或升级期间创建的；请勿将其删除。
- 如果备份设备位于另一台计算机上而您因此要通过网络进行装载，那么在该远程计算机上也必须安装 Backup Server。

卷名

- 转储卷根据 ANSI 磁带标签标准进行标注。标签信息应包含逻辑卷号和设备在分条集中的位置。

- 装载过程中， Backup Server 使用磁带标签来检验卷的安装顺序是否正确。这使您可以从比转储时所用设备数更少的设备进行装载。

注释　通过网络进行转储和装载时，必须为每个操作指定相同数量的分条设备。

更改转储卷

如果 Backup Server 检测到当前装入的卷有问题，它会通过将消息发送到客户端或操作员控制台来请求卷更换。安装另一个卷后，操作员通过在可以与 Backup Server 通信的任何 Adaptive Server 上执行 sp_volchanged 来通知 Backup Server。

恢复系统数据库

- 仅可以将 master 数据库的转储装载到 master 数据库或存档数据库中。
- 有关从转储恢复系统数据库的分步指导，请参见 《系统管理指南》。

磁盘镜像

- 在装载开始时， Adaptive Server 将每个逻辑数据库和日志设备的主设备名传递给 Backup Server。如果主设备已取消镜像， Adaptive Server 会改为传递辅助设备名。如果任何指定的设备在 Backup Server 完成数据传输前发生故障，则 Adaptive Server 将中止装载。
- 如果在 load database 正在进行时试图取消任何指定设备的镜像，则 Adaptive Server 会显示一条消息。执行 disk unmirror 的用户可以中止装载或将 disk unmirror 推迟到装载完成后执行。
- Backup Server 将数据装载到主设备上， load database 随后将其复制到辅助设备。如果有任何数据库设备被镜像，则 load database 会花更长的时间才能完成。

实现存档数据库

存档数据库是一个占位符，仅在装载数据库转储后才有用。装载过程实际上不复制页，而是使用页映射来实现数据库。

注释　注意：在将数据库转储装载到存档数据库中时，不需要运行 Backup Server。

使用 *load database with norecovery*

load database 命令的 with norecovery 选项允许将数据库转储装载到存档数据库中而不恢复任何数据，从而可缩短装载所需的时间。许多数据库页在恢复过程中会被修改或分配，从而导致它们存储在修改页面区域中。因此，跳过恢复过程可使得在修改页面区域中占用最少的空间。 with norecovery 选项允许快速查看存档数据库。

如果使用 with norecovery，则会自动使数据库联机。

然而，若对需要恢复的数据库使用 load database with norecovery，则可能会导致此数据库在事务上和物理上不一致。对物理上不一致的数据库上运行 dbcc 检查会产生许多错误。

在使用 with norecovery 装载存档数据库后，您必须具有 sa_role 或数据库所有者权限才能使用该数据库。

将逻辑设备与存档数据库一起使用

可以使用 sp_addumpdevice 来创建可从中装载存档数据库的逻辑设备：

```
sp_addumpdevice 'archive database', 'logical_name',
    'physical_name'
```

执行此命令后，使用 *logical_name* 而不是 *physical_name* 作为 load database 命令的 *dump_device* 或 *stripe_device*。

注释 不能使用存档数据库逻辑设备作为装载到传统数据库或何时转储传统数据库的设备说明。

---

load database 与存档数据库一起使用时的限制

当 load database 与存档数据库一起使用时，具有以下限制：

- 要求存档数据库的数据库转储是安装在本地计算机上的文件系统上的磁盘转储。可以是本地存储或 NFS 存储。 `load database ... at <remote server>` 语法不受支持，磁带上的数据库转储也不受支持。
- 不支持跨体系结构装载。数据库转储和 load database 命令必须在相同的体系结构 （在字节顺序方面）上执行。
- 转储的数据库使用的页大小必须与承载存档数据库的服务器所使用的页大小相同。
- 在其上进行转储的服务器的主版本必须早于或等于承载存档数据库的服务器的主版本。
- 在其上进行数据库转储的服务器上的字符集和排序顺序必须与承载存档数据库的服务器的字符集和排序顺序相同。

load database 和加密列

如果存储密钥的数据库与这些密钥加密的列所在的数据库不是同一数据库，则必须从同时完成的转储中装载这两个数据库，以避免出现装载后加密列的密钥缺失这一问题。

装载分别包含密钥和数据的数据库后，将这两个数据库同时联机。

不应该将密钥数据库装载为具有其它名称的数据库，因为加密列及其密钥之间存在元数据依赖关系。如果必须更改密钥数据库的名称，请执行下列操作：

1　在转储包含加密列的数据库之前，使用 alter table 解密数据。

2　转储包含密钥和加密列的数据库。

3　装载数据库之后，通过 alter table 用新命名数据库中的密钥对数据重新加密。

装载压缩数据

- 不能在一个平台上创建压缩表的转储并将该转储装载到另一个平台上。
- 在包含任何形式的压缩或解压缩行的压缩表上执行的 create index 命令在 load transaction 期间将会完全恢复。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 load database 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是数据库所有者，或是对数据库拥有 load database 特权或 own database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者，或是具有以下任一角色的用户： • sa_role，或 • replication_role 或 • oper_role

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
43	load	装载数据库	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

文档 《系统管理指南》中的 “备份和恢复用户数据库”。 .

命令 alter database, dbcc, dump database, dump transaction, load transaction, online database.

系统过程 sp_helpdb, sp_helpdevice, sp_hidetext, sp_volchanged.

load transaction

说明　装载使用 dump transaction 创建的事务日志的备份副本。

语法　进行例行日志装载：

```
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}
        }]]
```

返回标头或文件信息但不装载备份日志：

```
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],
        listonly [= full],
        headeronly,
        notify = {client | operator_console}
        until_time = datetime}]]
```

将事务日志装载到存档数据库中：

```
load tran[saction] database_name
    from dump_device
    [[stripe on stripe_device] ...]
```

（仅限 Tivoli Storage Manager）您的站点许可使用 Tivoli Storage Manager 时装载事务日志的副本。

```
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
        } ]
```

参数

database_name

从转储的事务日志备份副本中接收数据的数据库的名称。接收数据库的日志段的大小至少要与转储数据库的日志段的大小相同。可将数据库的名称指定为文字、局部变量或存储过程参数。对于存档数据库，*database_name* 是要将事务日志装载到的存档数据库。

compress::
调用存档事务日志的解压缩。有关 compress 选项的详细信息，请参见 《系统管理指南》中的 “备份和恢复用户数据库”。

注释　Sybase 建议首选使用本机 "compression = *compress_level*" 选项，旧选项 "compress::*compression_level*" 其次。如果对 dump database 使用了该本机选项，则在装载数据库时无需使用 "compress::*compression_level*"。

---

from *stripe_device*
从中装载事务日志的转储设备的名称。有关指定转储设备时使用何种形式的信息，请参见第 485 页的 “指定转储设备”。有关支持的转储设备的列表，请参见 Adaptive Server 安装和配置指南。

at *backup_server_name*
在转储设备附加到的计算机上运行的远程 Backup Server 的名称。对于使用接口文件的平台， *backup_server_name* 必须出现在接口文件中。

from *dump_device*
是本地磁盘事务日志转储。

density = *density_value*
替换磁带设备的缺省密度。**忽略该选项。**

blocksize = *number_bytes*
替换转储设备的缺省块大小。如果在 UNIX 系统上指定块大小，它应该与用来进行转储的块大小相同。

dumpvolume = *volume_name*
是 ANSI 磁带标签的卷名字段。打开磁带时，load transaction 会检查此标签，如果装载了错误的卷，则会生成错误消息。

file = *file_name*
是磁带卷上特定数据库转储的名称。如果在生成转储时未记录转储文件名，可使用 listonly 显示有关所有转储文件的信息。

stripe on *stripe_device*
是附加的转储设备。最多可以使用 32 个设备，其中包括在 to *stripe_device* 子句中指定的设备。 Backup Server 同时从所有设备装载数据，从而减少了所需时间和卷更改次数。有关如何指定转储设备的信息，请参见第 485 页的 “指定转储设备”。

compression

指示您正在装载的日志已被压缩为远程服务器上的文件。无需为 load transaction 指定压缩级别。

with compression 选项不同于 compress 选项，后者用来从本地文件装载压缩的日志。

注释 Sybase 建议首选使用本机 "compression = *compress_level*" 选项，旧选项 "compress::*compression_level*" 其次。如果对 dump database 使用了该本机选项，则在装载数据库时无需使用 "compress::*compression_level*"。

---

dismount | nodismount

（在支持逻辑卸下的平台上）确定磁带是否保持装入状态。缺省情况下，在装载完成时将卸下装载所用的全部磁带。使用 nodismount 命令可使磁带供其它装载或转储使用。

nounload | unload

确定装载完成后是否回绕磁带。缺省情况下磁带不回绕，从而使您可以从同一磁带卷进行更多的装载。为要从多转储卷装载的最后一个转储文件指定 unload。这样，在装载完成后就会回绕并卸载磁带。

listonly [= full]

显示有关磁带卷上所有转储文件的信息，但**不装载事务日志**。 listonly 标识数据库和设备、转储日期和时间以及转储可被覆盖的日期和时间。 listonly = full 提供有关转储的其它详细信息。两个报告都按 ANSI 磁带标签进行排序。

列出卷上的文件之后，Backup Server 发出卷更改请求。操作员可以安装另一个磁带卷，也可以终止所有转储设备的列表操作。

在当前实现中， listonly 会替换 headeronly。

警告！请不要对 1/4 英寸盒式磁带使用 load transaction with listonly。

---

headeronly

显示单个转储文件的标头信息，但**不装载数据库**。 headeronly 显示有关磁带上第一个文件的信息，除非使用 file = *file_name* 选项另指定一个文件名。转储标头表示：

- 转储类型 （数据库或事务日志）
- 数据库 ID
- 文件名
- 执行转储的日期
- 字符集
- 排序顺序
- 页数
- 下一个对象 ID
- 日志中的检查点位置
- 最早的 begin transaction 记录的位置
- 旧的和新的序列日期

notify = {client | operator_console}

替换缺省的消息显示目标。

- 在提供操作员终端功能的操作系统上，始终会将卷更改消息发送到运行 Backup Server 的计算机的操作员终端上。使用 client 可将其它 Backup Server 消息发送到启动 dump database 的终端会话。
- 在不提供操作员终端功能的操作系统 （如 UNIX）上，消息将发送到启动 dump database 的客户端。使用 operator_console 将消息发送到运行 Backup Server 的终端。

until_time

装载事务日志中指定时间之前的事务日志。只有在指定时间前提交的事务被保存到数据库中。

syb_tsm

是调用 libsyb_tsm.so 模块的关键字，该模块用于实现 Backup Server 和 TSM 之间的通信。

object_name

是 TSM 服务器上备份对象的名称。

-S *source_server_name*

源 Adaptive Server 与目标 Adaptive Server 不同时，指定源 Adaptive Server 的名称。装载操作的目标服务器与用于转储操作的源服务器不同时，该参数是必需的。

-D *source_database_name*

源数据库与目标数据库不同时，指定源数据库的名称。装载操作的目标数据库与转储操作的源数据库不同时，该参数是必需的。

示例

示例 1 为数据库 pubs2 磁带装载事务日志：

```
load transaction pubs2
    from "/dev/nrmt0"
```

示例 2 使用 Backup Server REMOTE_BKP_SERVER 装载 pubs2 数据库的事务日志：

```
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
```

示例 3 装载 pubs2 在 2008 年 3 月 20 日上午 10:51:43:866 之前的事务日志。

```
load transaction pubs2
    from "/dev/ntmt0"
    with until_time = "mar 20, 2008 10:51:43:866am"
```

示例 4 将事务从 TSM 备份对象 “demo2.1” 装载到 testdb 数据库。源数据库和目标数据库相同。有关信息，请参见第 371 页的 “dump transaction”：

```
load transaction testdb from "syb_tsm::demo2.1"
```

示例 5 目标数据库 (pubs2) 与源数据库 (testdb) 不同时，从 TSM 备份对象 “obj1.1” 装载事务：

```
load transaction pubs2 from "syb_tsm::
    -D testdb::obj1.1"
```

用法

- 如果在使用 sp_hidetext 之后执行跨平台的 dump 和 load，则必须手动删除并重新创建所有隐藏对象。
- listonly 和 headeronly 选项显示有关转储文件的信息，但不装载这些转储文件。
- 转储和装载通过 Backup Server 执行。
- 表 1-26 描述了用于从备份中恢复数据库的命令和系统过程：

表 1-26: 用于恢复数据库的命令

使用此命令	要进行此操作
create database for load	创建一个用于装载转储的数据库。
load database	从转储中恢复数据库。
load transaction	对恢复的数据库应用最近的事务。
online database	在完成常规装载序列或将数据库升级到最新版本的 Adaptive Server 后，使公共用户可以访问数据库。
load {database | transaction} with {headeronly | listonly}	标识磁带上的转储文件。
sp_volchanged	响应 Backup Server 卷更改消息。

限制

- 不能装载在版本低于 11.9 的服务器上生成的转储。
- 数据库和事务日志必须处于同一版本级别。
- 按时间顺序装载事务日志。
- 不能从空设备 （UNIX 上为 /dev/null）装载。
- 不能在执行升级的 online database 命令之后使用 load transaction。升级数据库的正确顺序是：load database、 load transaction、 online database。
- 不到所有事务日志都装载完成，请不要发出 online database。命令序列是：

  a　Load database

  b　Load transaction （必要时重复）

  c　Online database

  然而，要在保持对数据库的只读访问的同时装载其它事务日志 （典型的 “热备份” 情况），请使用 dump tran for standby_access 选项生成事务转储。然后可以为只读访问发出 online database for standby_access。
- 不能在用户定义的事务中使用 load transaction 命令。

恢复数据库

- 若要恢复数据库，请执行以下操作：
  - 装载最近的数据库转储
  - *按顺序* 装载自上次数据库转储以来生成的所有事务日志转储
  - 发出 online database 使数据库对公共用户可用

- 每次添加或删除跨数据库约束或者删除包含跨数据库约束的表时，都请转储*上述两个*受影响的数据库。

  警告！装载这些数据库的早期转储会导致数据库损坏。

- 有关备份和恢复 Adaptive Server 数据库的详细信息，请参见 《系统管理指南》。

将数据库恢复到指定的时间

- 可对大多数能够装载或转储的数据库使用 until_time 选项。 until_time 不适用于数据和日志位于同一设备上的数据库，如 master 数据库。此外，也不能对自上次 dump database 以来拥有过截断的日志的数据库 （例如 tempdb）使用该选项。
- until_time 选项十分有用，因为：
  - 它可以使数据库与特定时间保持一致。例如，在含有决策支持系统 (DSS) 数据库和联机事务处理 (OLTP) 数据库的环境中，系统管理员可以将 DSS 数据库回退到指定的以前的时间，以在早期版本和当前版本之间比较数据。
  - 如果用户无意之间损坏了数据 （例如删除了重要的表），则可以使用 until_time 选项通过将数据库向前滚动到损坏数据之前的那个点来取消错误命令。
- 若要在数据已经损坏之后有效地使用 until_time 选项，则必须知道发生错误的确切时间。通过在发生错误后立即执行 select getdate () 命令可以查出该时间。若要获得精确到毫秒的时间，请使用 convert 函数，例如：

  ```
  select convert (char (26), getdate (), 109)
  --------------------------
  Feb 26 1997 12:45:59:650PM
  ```

- 在使用 until_time 装载事务日志之后， Adaptive Server 会重新启动数据库的日志序列。这意味着，在您再次转储该数据库之前，不能在执行 load transaction 命令后使用 until_time 装载之后的事务日志。要另外转储一个事务日志，请先转储数据库。
- 只有在指定时间之前提交的事务被保存到数据库中。然而，在某些情况下，在 until_time 指定不久之后提交的事务也应用到数据库数据。当几个事务同时提交时可能发生此种情况。事务的顺序可能不以时间顺序写到事务日志中。在这种情况下，时间顺序混乱的事务会反映在已恢复的数据中。时间差应少于 1 秒。
- 有关将数据库恢复到指定时间的详细信息，请参见 《系统管理指南》。

装载时将用户锁在外面

- 不能使用装载中的数据库。如果您正在装载数据库，则该数据库不能使用。不同于 load database， load transaction 不会更改数据库的脱机或联机状态。 load transaction 保留数据库的原有状态。 load database 将数据库的状态设置为 “offline”。数据库处于 “offline” 状态时，无人可使用该数据库。“offline”状态可防止用户在装载序列期间对数据库进行访问和更改。
- 在发出 online database 之前，用 load database 装载的数据库一直保持不可访问状态。

升级数据库和事务日志转储

要将用户数据库转储从 11.9 或更高版本的服务器恢复并升级到最新版本的 Adaptive Server，可执行如下操作：

1　装载最近的数据库转储。

2　*按顺序*装载上次数据库转储后生成的所有事务日志。

3　使用 online database 升级数据库。

4　升级后立即转储该新升级的数据库，以创建与 Adaptive Server 的当前版本一致的转储。

指定转储设备

- 可以将转储设备指定为文字、局部变量或存储过程参数。
- 从本地设备装载时，可以使用以下形式指定转储设备：
  - 绝对路径名
  - 相对路径名
  - sysdevices 系统表中的逻辑设备名称

  Backup Server 使用 Adaptive Server 中的当前工作目录解析相对路径名。
- 通过网络转储时，请指定转储设备的绝对路径名。（不能使用相对路径名或 sysdevices 系统表中的逻辑设备名。）路径名在运行 Backup Server 的计算机上必须是有效的。如果名称包括任何非字母、数字或下划线 (_) 的字符，都必须用引号将它引起来。
- 转储设备上的所有权和权限问题可能会干扰装载命令的使用。 sp_addumpdevice 将设备添加到系统表，但不保证可以从该设备装载或将文件创建为转储设备。
- 可以同时运行多个装载 （或转储），只要每个装载使用不同的物理设备即可。

Backup Servers

- Backup Server 必须与 Adaptive Server 在同一台计算机上运行。Backup Server 必须在 master..sysservers 表中列出。这个条目是在安装或升级过程中创建的，不得删除。
- 如果备份设备位于另一台计算机上而您因此要通过网络进行装载，那么在该远程计算机上也必须安装 Backup Server。

卷名

- 转储卷根据 ANSI 磁带标签标准标注。标签信息应包含逻辑卷号和设备在分条集中的位置。
- 装载过程中， Backup Server 使用磁带标签来检验卷的安装顺序是否正确。这使您可以从比转储时所用设备数更少的设备进行装载。

注释 通过网络进行转储和装载时，必须为每个操作指定相同数量的分条设备。

更改转储卷

如果 Backup Server 检测到当前装入的卷有问题，它会通过向客户机或其操作员主控台发送消息来请求更改卷。安装另一个卷后，操作员通过在可以与 Backup Server 通信的任何 Adaptive Server 上执行 sp_volchanged 通知 Backup Server。

恢复系统数据库

有关从转储中恢复系统数据库的分步指导，请参见 《系统管理指南》。

磁盘镜像

- 在装载开始时，Adaptive Server 将每个逻辑数据库和逻辑日志设备的主设备名传递给 Backup Server。如果主设备已取消镜像， Adaptive Server 会改为传递辅助设备名。如果任何指定的设备在 Backup Server 完成数据传输前发生故障，则 Adaptive Server 将中止装载。
- 如果在 load transaction 正在进行时试图取消任何指定设备的镜像，则 Adaptive Server 会显示一条消息。执行 disk unmirror 的用户可以中止装载或将 disk unmirror 推迟到装载完成后执行。
- Backup Server 将数据装载到主设备上，load transaction 随后将其复制到辅助设备上。如果有任何数据库进行了镜像，则 load transaction 会花费更长的时间完成。

将事务日志装载到存档数据库中

将事务日志装载到存档数据库时， load tran 会运行恢复撤消过程。已修改和新的数据库页会写入永久性更改段。已修改页区域必须有足够的空间容纳这些更改。如有必要，可使用 alter database 增加分配给存档数据库的正常数据库存储，从而增加已修改页区域的空间。

与传统数据库不同，存档数据库可在装载序列的中间进行联机，而不会中断装载序列。如果装载传统数据库后进行联机，而没有使用用于 standby_access 的子句，则不能再将接下来的事务日志装载到装载序列中。但是，存档数据库可以不使用用于 standby_access 的子句进行联机，并且之后装载序列还可以装载接下来的事务日志。这样，在装载序列期间可随时执行只读操作，如运行一致性检查。之所以能够这样，是因为在将事务日志装载到存档数据库时， Adaptive Server 会自动从已修改页区域中删除一次性更改段。这样可以有效地将存档数据库恢复到之前装载完成后的状态，从而允许装载序列中的下一个事务日志。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 load transaction 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是数据库所有者，或是对数据库拥有 load database 特权或 own database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者，或是具有以下任一角色的用户： • sa_role，或 • replication_role 或 • oper_role

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
44	load	load transaction	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

文档 《系统管理指南》中的 “备份和恢复用户数据库”。 .

命令 disk unmirror, dump database, dump transaction, load database, online database.

系统过程 sp_dboption, sp_helpdb, sp_helpdevice, sp_hidetext, sp_volchanged.

lock table

说明

在事务中显式锁定表。

语法

lock table *table_name* in [partition *data_partition_name*] in
{share | exclusive} mode
[wait [*numsecs*] | nowait]

参数

table_name

指定要锁定的表的名称。

partition *data_partition*

指示您要锁定数据分区。

share | exclusive

指定要应用于表或分区的锁类型 （共享或排它）。

wait *numsecs*

指定不能立即获得锁时等待的秒数。如果省略 *numsecs*，则指定 lock table 命令应一直等到锁被授予。

nowait

如果不能立即获得锁，将导致命令失败。

示例

示例 1 尝试获得 titles 表上的共享表锁。如果使用 set lock wait 设置了会话级等待，则 lock table 命令将等待这段时间；否则将使用服务器级等待时间：

```
begin transaction
lock table titles in share mode
```

示例 2 尝试获得 authors 表上的排它表锁。如果不能在 5 秒之内获得锁，该命令将返回一条信息性消息。事务中的后续命令继续按本来的方式执行，就如同没有使用 lock table 一样：

```
begin transaction
lock table authors in exclusive mode wait 5
```

示例 3 如果 5 秒之内没有获得表锁，该过程将检查用户的角色。如果由具有 sa_role 角色的用户执行，则该过程会显示一条建议性消息并在未获得表锁的情况下继续执行。如果该用户没有 sa_role 角色，则事务回退：

```
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
```

用法

- 可以对存档数据库使用 lock table。
- 如果将 lock table 用作 set chained on 命令之后的第一条语句，则将创建一个新事务。
- 只有在事务中才能使用 lock table。表锁在整个事务期间被持有。
- lock table 的行为取决于在命令中指定的或者在会话级或服务器级处于活动状态的等待时间选项。
- 若未指定 wait 和 nowait 选项，则 lock table 使用会话级或服务器级等待时间。如果已经使用 set lock wait 设置了会话级等待时间，则使用该时间；否则将使用服务器级等待时间。
- 如果在时间限制（如果有）内无法获得表锁，则 lock table 命令返回消息 12207。不回退事务。事务中的后续命令继续按本来的方式执行，就如同没有使用 lock table 命令一样。
- 不能在系统表或临时表上使用 lock table。
- 可以在同一事务中发出多个 lock table 命令。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

您必须是表所有者。要使用 lock table in share mode，您对该表必须拥有 select 访问权限。要使用 lock table in exclusive mode，您对于该表必须拥有 delete、 insert 或 update 访问权限。

另请参见

命令 set.

merge

说明

将源表中的行传送到目标表中：

- 插入位于源中且在目标中没有匹配键列的行。
- 用源行中的值更新其键列已经在目标中存在的行。

语法

```
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)
```

参数

into [[*database*.]*owner*.]*identifier* [as *table_alias*]

将目标对象指定为表或将可更新视图指定为表别名。目标可以是完全限定名称标识符或短名称标识符 – 不包括数据库名称和所有者名称 – 而且在这种情况下，Adaptive Server 使用当前数据库以及用户或数据库所有者。

还可以指定表别名作为引用目标表的替代方法。

using [[*database*.]*owner*.]*identifier* [as *table_alias*] | (*select_query*) as *alias_name* [*column_list*]

将源对象指定为表、视图或派生表。如果源对象为：

- 表或视图 – 使用 [[*database*.]*owner*.]*identifier*。
- 派生表 – 通过 select 查询引用它，采用的形式为别名名称以及用于定义派生表的可选列列表。

merge_search_conditions

检查源表中的行是否与目标表中的行匹配，由一个谓词 （如 “*col_name* = *col_name*”）列表组成。

search_conditions

是组织良好的布尔表达式，用在 matched/not matched 子句中。

update set {*col_name* = *expression*} | delete

这两个选项都始终位于 matched 子句中。 update 向匹配的行分配新值，而 delete 删除当前匹配的行。

insert [(*column_list*)] values (*value_list*)

始终出现在 not matched 子句中，在目标表中插入不匹配的行。

示例

示例 1 将 DailySales 表合并到 GlobalSales 中：

```
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
```

示例 2 通过动态参数标记将派生表用作源表：

```
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
```

用法

- 目标表不能是任何参照完整性约束的一部分。
- 对于含有 on 子句的 merge 查询，如果其引用常量布尔表达式 （如 (1=0) 或 (1=1)），则没有特定的优化。
- 在 on 子句中引用的目标列不能位于 update 操作的 set 子句中。
- 虽然您可以从存储过程内调用 merge 语句，但 update 和 insert 语句不允许出现在标量 SQL 函数中。
- 目标表不能是具有 instead of 触发器的可更新视图。
- merge 语句是可以高速缓存的，而且 update 操作的 set 子句中的文字以及 insert 操作的 insert 值列表中的文字是文字参数化过程的目标。
- 目标表不能是代理表。
- 对于源表中的每行，如果该行：
  - 在目标表中有匹配的行，并且搜索条件的求值结果为 true – 在目标表中或目标表的相应行中执行 update。

- 在目标表中没有匹配的行，并且搜索条件的求值结果为 true – 在目标表中插入该行。
- 在目标表中有多个匹配的行 – 出现错误。这是标准的 SQL-2003 行为。

merge 语句可包含多个具有不同搜索条件的 when matched 和 when not matched 子句。when 子句中的第一个满足条件的 when 会运行相应的操作；其余的将被忽略。

权限

任何对源对象具有 select 权限，对目标对象具有 insert、 update 或 delete 权限的用户都可以使用 merge。

mount

说明

将数据库附加到目标或辅助 Adaptive Server。

mount 命令对**清单文件**中的信息进行解码并使该数据库集可用。 mount 与 bcp 批量复制实用程序之类的其它复制过程不同，它执行所有必需的支持操作，包括添加数据库设备并在必要情况下激活这些设备、为新的数据库创建目录条目以及恢复它们。

装入数据库时，如果要在目标 Adaptive Server 上使用其它设备名称，请使用 mount with listonly 并修改目标服务器上的设备路径名，然后使用 mount 实际装入数据库。

注释　对于每个可以访问原始 Adaptive Server 上的数据库的登录名，相同 suid 在目标 Adaptive Server 上具有相应登录名更加方便，因为这样可以避免用户 ID 调和问题。

为了使权限保持不变，目标 Adaptive Server 上的登录映射必须与源 Adaptive Server 上的登录映射完全相同。有关登录映射的详细信息，请参见 《系统管理指南，卷 1》中的 “管理远程服务器”。

语法

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*

参数

manifest_file

清单文件是描述位于一组数据库设备上的数据库的二进制文件。

对文件内容执行字符转换的操作 （例如 ftp）会损坏清单文件，除非以二进制模式执行。

示例

示例 1 查找在源 Adaptive Server 的清单文件上列出的路径名：

```
mount database all from "/data/sybase2/mfile1" with listonly
go

[database]
```

```
  mydb
[device]
  "/data/sybase1/d0.dbs" = "1dev1"
  "/data/sybase2/d14.dbs" = "1dev13"
```

如果使用的路径名不同于源路径名，请检验或修改它们以使其符合目标 Adaptive Server 上的标准。

示例 2 将数据库设备复制到辅助 Adaptive Server 上之后，就可以将其装入：

```
mount database all from "/data/sybase2/mfile1" using
    "/data/sybase2/d0.dbs" = "1dev1",
    "/data/sybase2/d14.dbs" = "1dev13"
```

当完成 mount 过程之后，数据库仍然处于脱机状态。使用 online database 命令使数据库进入联机状态。无需重新启动服务器。

示例 3 目标服务器可以与源服务器相同。在这种情况下，必须将数据库名称映射为其它名称，并且逻辑设备名称将在内部重命名。

1 在同一服务器中创建数据库 mydb 的精确副本：

```
11> quiesce database mydb_tag hold mydb for external dump to
   "/data/mydb.manifest"
2> go
```

2 复制 OS 文件：

```
$ cp /data/sybase2/mydb.dbs /data/sybase2/mydb_copy.dbs
```

3 现在可以将其作为副本装入：

```
11> 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
```

将自动为物理设备 *//data/sybase2/mydb_copy.dbs/* 指派计算机生成的逻辑名，逻辑名采用的格式为 *Cccc$<mydb_dev>*，其中：

- *C* – 为 [A – Z]
- *c* – 为 [AZ, 09]，并且是指编码逻辑设备号
- *mydb_dev* – 最多只能包含旧逻辑设备名的 26 个字符。

目标 Adaptive Server 上不能存在被传输的数据库的数据库 ID。由于数据库已装入到同一服务器上，因此必须更改数据库 ID。装入设备中的分配页保留原始数据库 ID，并且 disk refit 命令使用该信息。运行 mount database 后使用 dbcc checkalloc 命令调整 dbid，以便 disk refit 可在装入设备上起作用。如果正在载入的数据库不是用于临时用途，请运行 checkalloc。

用法

- using 子句允许您通过 “=” 符号或 “as” 子句定义映射。
- 如果有多个设备，可以一个映射使用 “=”，另一个映射使用 “as”。
- 在数据库和设备中，都可以按名称 （指定逻辑和物理名称）和按顺序映射设备。如果一个数据库按名称映射，则所有数据库都必须按名称映射，反之亦然。这一规则同样适用于设备。
- 不能在目标 Adaptive Server 上 mount 所传送的一组数据库中的一部分；清单文件中的所有数据库及其设备必须一起装入。
- 在目标 Adaptive Server 上执行 mount 命令时：
  - 目标 Adaptive Server 上的页大小必须等于源 Adaptive Server 上的页大小。
  - 目标 Adaptive Server 上必须配置足够的设备才能成功添加属于所装入数据库的所有设备。
  - 如果从源 Adaptive Server 装入的逻辑设备与目标 Adaptive Server 上的逻辑设备名称相同，则这些设备会自动重命名，除非您在 mount 命令中包含别名。

    如果物理设备名在目标 Adaptive Server 上已经存在，则必须在源 Adaptive Server 上从操作系统级别重命名物理设备名，并使用 mount 命令提供新物理设备名。
  - 日志版本在源 Adaptive Server 和目标 Adaptive Server 中必须相同。
  - 装入数据库时，Adaptive Server 的主要版本号不能高于数据库。例如，不能在 12.5.x 版本的 Adaptive Server 上装入 15.0 版本的数据库。
  - 源 Adaptive Server 和目标 Adaptive Server 的平台必须相同。
  - 排序顺序和字符集不同的问题由 load database 之后的规则解决。只有排序顺序为二进制时，才能装入字符集不同的数据库。

Cluster Edition

Cluster Edition 中支持 mount database 和 unmount database。如果当这些命令中的一个命令正在执行时实例出现故障，则命令可能中止。这种情况下，当实例故障切换恢复完成时，用户必须重新发出 mount database 或 unmount database。

目标更改

在目标 Adaptive Server 上装入数据库之后，某些设置在装入的数据库上将被清除：

- 关闭复制。
- 清除并关闭审计设置。
- 清除 CIS 选项、缺省远程位置和类型。
- 为载入的数据库及其对象删除高速缓存绑定。
- 装入的数据库的恢复顺序将被删除，变为缺省 dbid 顺序。

系统方面的考虑

- 不能在事务中使用 mount 命令。
- 不可在配置为高可用性的服务器上 mount （装入）数据库。

性能考虑事项

将数据库 mount 到 Adaptive Server 时，如果更改装入的数据库的 dbid，则会对所有过程进行标记，以在该数据库中进行重新编译。这将增加在目标服务器上恢复数据库所用的时间，并会延长该过程的首次执行时间。

重命名设备

清单文件中含有创建该清单文件的源 Adaptive Server 所知的设备路径。如果目标 Adaptive Server 使用不同的路径访问设备，可以在 mount 命令上指定新的路径。

1 将 mount 命令与 with listonly 一起使用来显示旧路径：

```
mount database all from "/work2/Mpubs_file" with listonly
go

[database]
  mydb
[device]
  "/work2/Devices/pubsdat.dat" = "pubs2dat"
```

2 如果设备 pubs2dat 的新路径是 */work2/Devices/pubsdevice.dat*（Windows 中的设备路径），则在 mount 命令中指定这一新设备：

```
mount database all from "/work2/Mpubs_file" using
```

```
"/work2/datadevices/pubsdevice.dat" = "pubs2dat"
```

如果逻辑设备名已存在于目标服务器中，将使用自动生成的唯一名称对其进行重命名。

标准　符合 ANSI SQL 的级别Transact-SQL 扩展。

权限　对 mount 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 mount any database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 或 oper_role 的用户。

审计　sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
101	mount	mount database	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见　**命令** unmount, quiesce database.

文档　《系统管理指南，卷 2》中的 “装入和卸下数据库” .

online database

说明

在常规装载序列完成后，将数据库标记为可供公共用户使用；如果需要，将装载的数据库升级到当前版本的 Adaptive Server；在装载了使用 for standby_access 选项转储的事务日志后使数据库联机。还可以使用 online database 使存档数据库联机。

语法

online database *database_name* [for standby_access]

参数

database_name

指定要联机的数据库的名称。

for standby_access

假定数据库中不包含打开的事务时将数据库联机。

示例

示例 1 装载序列完成后，将 pubs2 数据库提供给公共用户使用：

```
online database pubs2
```

示例 2 将数据库 inventory_db 联机在使用通过 dump tran...with standby_access 获得的事务日志转储装载 inventory_db 之后使用：

```
online database inventory_db for standby_access
```

用法

- 在常规数据库或事务日志装载序列完成后，online database 将数据库联机以供常规使用。
- 发出 load database 时，会将数据库状态设置为 “脱机”。脱机状态在 sysdatabases 系统表中设置，并且保持该设置直到 online database 完成。
- 不到所有事务日志都装载完成，请*不要*发出 online database。命令序列是：
  - load database
  - load transaction （可能有多个 load transaction）
  - online database
- 如果对当前联机数据库执行 online database ， 则不进行处理，也不会生成错误消息。
- online database...for standby_access 只可用于使用 dump transaction...with standby_access 转储的事务日志。如果在装载未使用 dump transaction...with standby access 转储的事务日志之后，使用 online database...for standby_access， online database 将生成错误消息并失败。
- 可以使用 sp_helpdb 查看数据库当前是处于联机、联机以用于备用访问还是脱机状态。

升级数据库

- online database 在必要时会启动载入数据库和事务日志转储的升级过程，以使该数据库与当前版本的 Adaptive Server 兼容。升级完成之后，数据库被提供给公共用户使用。如果处理过程中发生错误，数据库将保持脱机状态。
- 仅在数据库或事务日志装载序列之后才需要使用 online database。新的安装或升级不需要。如果将 Adaptive Server 升级到新的版本，则所有与此服务器关联的数据库都将自动升级。
- online database 只升级 12.5.4 或更高版本的用户数据库。
- 使用 online database 升级数据库之后，请转储新升级的数据库以创建与当前版本的 Adaptive Server 相一致的转储。必须先转储升级的数据库，然后才能发出 dump transaction 命令。

存档数据库

online database *database_name* 命令执行撤消恢复操作，在此期间，已修改或分配的页可能会重新映射到修改页面区域。

若装载数据库时使用了 with norecovery，则您不需要将此数据库联机，因为装载过程会自动使数据库联机，而不用运行恢复撤消过程。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 online database 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是数据库所有者，或是对数据库具有 online database 特权或 own database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者，或是具有以下任一角色的用户： • sa_role 或 • replication_role 或 • oper_role

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
83	security	online database	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 dump database, dump transaction, load database, load transaction.

系统过程 sp_helpdb.

open

说明

打开要进行处理的游标。

语法

open *cursor_name*

参数

cursor_name
是要打开的游标的名称。

示例

打开名为 authors_crsr 的游标：

```
open authors_crsr
```

用法

- open 可以打开一个游标。游标用于分别修改或删除各行。要使用 fetch、 update 和 delete 语句，必须先打开一个游标。有关游标的详细信息，请参见 《Transact-SQL 用户指南》。
- 如果游标已经打开或还没有用 declare cursor 语句创建游标，Adaptive Server 将返回错误消息。
- 打开游标时， Adaptive Server 对定义游标的 select 语句 （在 declare cursor 语句中指定）求值，并使游标结果集可供处理。
- 在第一次打开游标时，其位置在游标结果集第一行的前面。
- 可以对存档数据库使用 open。
- 设置链式事务模式时，如果当前没有活动的事务，则 Adaptive Server 隐式地用 open 语句启动事务。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

open 权限缺省情况下授予所有用户。

另请参见

命令 close, declare cursor, fetch.

order by 子句

说明

按排序顺序返回指定列中的查询结果。

语法

[*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*

参数

order by
按列对结果进行排序。

asc
按升序顺序排列结果。如果未指定 asc 或 desc，则使用 asc。

desc
按降序顺序排列结果。

示例

示例 1 选择价格高于 $19.99 的书名，并将书名按字母顺序与价格一起列出：

```
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
```

示例 2 按类型的降序字母顺序列出 titles 表中的书，并计算每种类型的平均价格和平均预付款：

```
select type, price, advance
from titles
```

```
order by type desc
compute avg (price), avg (advance) by type
```

示例 3 列出 titles 表中的 title ID 和预付款除以总销售额得到的值，按计算得到的最低值到最高值顺序排列：

```
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
```

示例 4 按类型顺序列出书名和类型，并在输出中重命名各个列：

```
select title as BookName, type as Type
from titles
order by Type
```

用法

- order by 按顺序在指定列中返回查询结果。order by 属于 select 命令的一部分。
- 在 Transact-SQL 中，可以使用 order by 对未出现在 select 列表中的项进行排序。可以按照列标题、列名、表达式、别名（如果已在 select 列表中指定）或表示项目在 select 列表中位置的数字 (*select_list_number*) 进行排序。
- 如果按 *select_list_number* 排序，则 order by 子句引用的列必须包括在 select 列表中，且 select 列表不能为 *（星号）。

- 使用 order by 以有意义的顺序显示查询结果。如果没有 order by 子句，则无法控制 Adaptive Server 返回结果的顺序。

存在相同的表列名和列别名时 *order by* 的行为

存在以下三个条件时，Adaptive Server 会将 order by 子句中的列名解释为别名：

- order by 子句包含对限定列名的引用 （即 order by *table.column*）。
- 表列名和别名同时存在。
- 表列名和别名都与 order by 子句中的列名相同。

在以下示例中，尽管 order by 子句相同，但两个查询的结果集也是不同的；而且 order by 子句在两种情况中引用的列也不同。

```
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)
```

出现这种行为的原因是，Adaptive Server 允许 order by 子句引用由表名限定的别名列名。当存在列名与别名列相同的基表列时， Adaptive Server 会为别名列提供更高的优先级。

限制

- order by 子句所允许的最大列数是 31 列。
- 不能对 text、 unitext 或 image 数据类型的列使用 order by。

- 子查询和视图定义中不能包含 order by 子句，也不能包含 compute 子句和 into 关键字。反之，在 order by 列表中也不能使用子查询。
- 如果服务器或语言类型游标的 select 语句中包含 order by 子句，则不能更新该游标的结果集。有关应用于可更新游标的限制的详细信息，请参见 《Transact-SQL 用户指南》。
- 如果使用 compute by，必须同时使用 order by 子句。在 compute by 之后列出的表达式必须与 order by 后面列出的表达式相同或是其子集，它们从左向右的顺序必须一致，必须以同一表达式开始且不可跳过任何表达式。例如，假设 order by 子句为：

  ```
  order by a, b, c
  ```

  则 compute by 子句可以是下列任何一种或全部形式：

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

  compute 关键字也可以不与 by 连用，以生成总和、总计数等等。在这种情况下， order by 是可选的。

归类序列

- 利用 order by，可以将空值排在其它所有值之前。
- Adaptive Server 上的排序顺序 （归类序列）决定了数据的排序顺序。排序顺序可以选择二进制、字典、不区分大小写、具有优先级的区分大小写以及不区分大小写和变音。还可以提供特定于本国/地区语言的排序顺序。

表 1-27: 排序顺序选择的影响

Adaptive Server 排序顺序	对 *order by* 结果的影响
二进制顺序	根据字符集中每个字符的字节数字值对所有数据进行排序。二进制排序顺序将所有大写字母排在小写字母之前。二进制排序顺序是多字节字符集的唯一选项。
字典顺序	将大写字母排在相应的小写字母之前 （区分大小写）。字典排序顺序识别字母的各种变音形式，并将它们排在非变音形式之后。
字典顺序，不区分大小写	按字典顺序排序数据但不区分大小写。大写字母与对应的小写字母等效，其排序方式如接下来的 “排序规则”中所述。
字典顺序，不区分大小写，具有优先级	将大写字母排在优先的位置，放在其相应小写字母之前。执行比较时（例如在 where 子句中）不区分大小写。
字典顺序, 不区分大小写和变音	按字典顺序对数据排序，但不区分大小写；将变音格式的字母与相关联的未变音字母同等对待。此排序顺序在排序结果中混合变音字符和非变音字符。

- sp_helpsort 报告 Adaptive Server 上安装的排序顺序。

排序规则

当按 Adaptive Server 的排序顺序有两行等值时，将使用以下规则对行进行排序：

- 比较在 order by 子句中指定的列中的值。
- 如果两行有相等的列值，则逐字节比较整行的二进制值。该比较在行上以列的内部排序顺序执行，而不是以列在查询或原始的 create table 子句中指定的顺序执行。简而言之，数据先按顺序存储在所有固定长度列中，然后再按顺序存储在所有可变长度列中。
- 如果行相等，则比较行 ID。

  给定表：

  ```
  create table sortdemo (lname varchar (20),
                         init char (1) not null)
  ```

  和数据：

  ```
  lname      init
  ---------- ----
  Smith      B
  SMITH      C
  smith      A
  ```

  当按 *lname* 排序时得到结果如下：

  ```
  lname      init
  ---------- ----
  smith      A
  Smith      B
  SMITH      C
  ```

  因为在内部首先存储固定长度的 char 数据 （init 列），所以 order by 基于 “Asmith”、“BSmith”和 “CSMITH”的二进制值对这些行进行排序。

  但是，如果 init 是 varchar 类型，则首先排列 *lname* 列，然后再排列 init 列。对 “SMITHC”、“SmithB”和 “smithA”的二进制值进行比较，且行以该顺序返回。

降序扫描

- 在 order by 子句中使用 desc 关键字为查询优化程序提供了一种不需要工作表和排序步骤就能以降序返回结果的策略。此优化按每个索引页上的前一页指针，反向扫描该索引的页链。

若要使用这种优化方式， order by 子句中的列必须与索引顺序相匹配。这些列可以是键的子集，但必须是前缀子集，也就是说，它们必须包括前面的键。如果在 order by 子句中指定的列是索引键的超集，则不能使用降序扫描优化。

如果查询中涉及连接，则可以按降序的键顺序扫描所有表，只要满足键的前缀子集要求即可。也可以将降序扫描优化用于连接中的一个或多个表，而其它表仍按升序顺序进行扫描。

- 如果其它用户进程正向前扫描以执行更新或删除操作，执行降序扫描可能导致死锁。在执行页面拆分和收缩时也可能遇到死锁。可以在服务器上使用 sp_sysmon 跟踪死锁，或者使用配置参数 print deadlock information 将死锁信息发送到错误日志。
- 如果应用程序必须以降序顺序返回结果，但降序扫描优化产生了死锁问题，则可能的解决方法有：
  - 对降序扫描使用 set transaction isolation level 0 扫描。有关隔离级别 0 读取的效果的详细信息，请参见 set 命令以及 《性能和调优指南：锁定》中的 “使用锁定命令”。
  - 使用 allow backward scans 配置参数禁用降序扫描优化，以便使用 desc 的所有查询都以升序顺序扫描表并以降序顺序对结果集进行排序。请参见 《系统管理指南》。
  - 将有问题的降序扫描分为两步：首先，按升序顺序将所需行选入临时表；然后，按降序顺序从临时表中进行选择。
- 如果后向扫描由于重复键值的存在而使用了包含溢出页的聚簇索引，则降序扫描所返回的结果集可能不与升序扫描所返回的结果集的顺序完全相反。指定的键值按顺序返回，但溢出页上相同键的行的顺序可能不同。有关聚簇索引中溢出页存储方式的说明，请参见 《性能和调优指南：基础知识》中的 “索引”。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

使用 union 运算符时在 select 语句的 order by 子句中指定新的列标题是一种 Transact-SQL 扩展。

存在相同的表列名和列别名时，order by 的行为是 ANSI SQL 标准的特定于供应商的扩展。

另请参见

命令 compute clause, declare, group by 和 having 子句, select, where 子句.

系统过程 sp_configure, sp_helpsort, sp_lock, sp_sysmon.

prepare transaction

说明	由 DB-Library 在两阶段提交应用程序中使用，以查看服务器是否已准备好提交事务。
语法	prepare tran[saction]
用法	• 请参见 Open Client DB-Library Reference Manual（《Open Client DB-Library 参考手册》）。
标准	符合 ANSI SQL 的级别Transact-SQL 扩展。
另请参见	**命令** begin transaction, begin transaction, rollback, save transaction.

print

说明

在用户屏幕上输出用户定义的消息。

语法

print
{*format_string* | *@local_variable* |
@@global_variable}
[, *arg_list*]

参数

format_string

可以是一个变量或字符串。 *format_string* 的最大长度是 1023 字节。

格式字符串最多可包含 20 个任意顺序的唯一占位符。将消息文本发送到客户端时，使用 *format_string* 后接参数的格式化内容替换这些占位符。

为了在用其它语法结构将格式字符串转换为某种语言时可以对参数重新排序，要对占位符进行编号。参数所用占位符的显示形式如下：“ %*nn* !”一个百分号 (%)，后接一个 1 到 20 之间的整数，后面再接一个感叹号 (!)。整数表示参数列表中字符串中的参数编号。“%1!”“%1!”是原始版本中的第一个参数，“%2!”是第二个参数，依此类推。

用这种方法指示参数的位置可以使转换正确，即使参数出现在目标语言中的顺序与其在原始语言中的顺序不同也是如此。

例如，假定以下是一条英文消息：

```
%1! is not allowed in %2!.
```

此消息的德语版本是：

```
%1! ist in %2!
```

此消息的日语版本是：

```
%2! の中で %1! は許されません。
```

在此示例中，“%1!”在所有三种语言中都表示同一参数，“%2!”也是如此。此示例显示了翻译格式中有时需要对参数重新排序。

@local_variable

必须是 char、nchar、varchar 或 nvarchar 类型，且必须在使用它的批处理或过程中予以声明。

@@global_variable

必须是 char 或 varchar 类型或可以自动转换成这些类型的类型 （例如 *@@version*）。当前， *@@version* 是唯一的字符类型的全局变量。

arg_list

可以是一系列由逗号分隔的变量或常量。*arg_list* 是可选的，除非提供了包含“*%nn*!”形式的格式字符串。在这种情况下，*arg_list* 所包含的参数数目必须至少等于占位符的最大编号。参数可以是除 text 或 image 以外的任何数据类型；在出现在最终的消息中之前，它将被转换为字符数据类型。

示例

示例 1 如果在 authors 表中有住在邮政区号 94705 的作者，则输出“Berkeley author”：

```
if exists (select postalcode from authors
where postalcode = '94705')
print "Berkeley author"
```

示例 2 声明一个变量，为该变量赋值，然后输出变量值：

```
declare @msg char (50)
select @msg = "What's up, doc?"
  print @msg

What's up, doc?
```

示例 3 说明变量和占位符在消息中的用法：

```
declare @tabname varchar (30)
select @tabname = "titles"

declare @username varchar (30)
select @username = "ezekiel"

print "The table '%1!'is not owned by the user '%2!'.",
@tabname, @username

The table 'titles' is not owned
by the user 'ezekiel.'
```

用法

- 替换后，*format_string* 加上所有参数的最大输出字符串长度是 1023 个字节。
- 如果在格式字符串中使用占位符，请记住：对于字符串中的每个占位符 *n*，占位符 1 到 *n*- 1 也必须存在于同一字符串中，不过它们不必以数值顺序出现。例如，不允许将占位符 1 和 3 放在一个格式字符串中，却不把占位符 2 也放在此字符串中。如果在格式字符串中省略了一个数字，则执行 print 时会生成错误消息。
- *arg_list* 必须为 *format_string* 中的每个占位符各包括一个参数，否则事务将被中止。可以使用比占位符多的参数。

- 要在错误消息中包括实际的百分号，请在 *format_string* 中使用两个百分号 （“%%”）。如果 *format_string* 中包括不是用作占位符的单个百分号 （“%”）， Adaptive Server 将返回错误消息。
- 如果参数求值结果为 NULL，它将被转换成零长度的字符串。如果不希望输出中出现零长度字符串，可使用 isnull 函数。例如，如果 *@arg* 为空，下列语句将输出 `I think we have nothing here.`：

```
declare @arg varchar (30)
select @arg = isnull (col1, "nothing") from
table_a where ...
print "I think we have %1! here", @arg
```

- 可以将用户定义的消息添加到系统表 sysusermessages 中供所有应用程序使用。使用 sp_addmessage 向 sysusermessages 中添加消息；使用 sp_getmessage 检索消息以供 print 和 raiserror 使用。
- 使用 raiserror （而不是 print）输出用户定义的错误消息并将错误号存储在 *@@error* 中。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 print 无需任何权限。

另请参见

命令 declare, raiserror.

系统过程 sp_addmessage, sp_getmessage.

quiesce database

说明

挂起和恢复对指定的一系列数据库进行更新。

语法

quiesce database *tag_name* hold *database_list* [for external dump]
[to *manifest_file* [with override]]

或：

quiesce database *tag_name* release

参数

tag_name

是一个用户定义的名称，它指定要 hold 或 release 的数据库列表。*tag_name* 必须遵循标识符的规则。

hold

与 to manifest_file 子句一起使用时，挂起数据库并创建清单文件。

警告！ 因为清单文件是二进制文件，所以执行文件内容的字符转换的操作 （例如 ftp）将导致文件损坏，除非操作以二进制模式执行。

---

database_list

是 quiesce database hold 命令中包含的数据库列表。

for external dump

指定当挂起对列表中的数据库的更新时，将使用 Adaptive Server 外部的某种工具物理地复制所有受影响的数据库设备。复制操作可以替代 dump database 和 load database 的组合。

manifest_file

描述位于一组数据库设备上的数据库的二进制文件。仅当占用这些设备的那组数据库在这些设备上处于隔离和自我包含状态时才能创建清单文件。

因为清单文件是二进制文件，所以可以对文件内容执行字符转换的操作 （例如 ftp）将导致文件损坏，除非操作以二进制模式执行。

with override

覆盖妨碍成功对数据库执行 quiesce database 操作的任何限制。

示例

示例 1 挂起 salesdb 和 ordersdb 上的更新活动：

```
quiesce database report_dbs hold salesdb, ordersdb
```

示例 2 恢复标有 report_dbs 标签的数据库上的更新活动：

```
quiesce database report_dbs release
```

示例 3 挂起对 pubs2 数据库的更新活动，并指示制作此数据库的外部副本：

```
quiesce database pubs_tag hold pubs2 for external dump
```

示例 4 将数据库置于挂起状态并为要复制到另一 Adaptive Server 上的数据库生成清单文件：

```
quiesce database pubs_tag hold pubs2 for external dump to
    "/work2/sybase1/mpubs_file with override
```

该命令完成后，控制将返回到用户。

示例 5 复制数据库设备，使用 mount database with listonly 列出所有要复制的设备进行查看：

```
1> mount database all from "/data/sybase2/mfile1" with listonly
2> go

"/data/sybase1/d0.dbs" = "1dev1"
```

如果被抑制的那组数据库包含对该组数据库之外的数据库的引用，则无法创建清单文件。使用 with override 选项可绕过此限制：

```
quiesce database pubs2_tag release for external dump to Mpubs_file
```

示例 6 *key_db* 包含用于对 *col_db* 中的列进行加密的加密密钥时，这些命令可以成功执行：

```
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"
```

用法

- 与 hold 关键字一起使用的 quiesce database 将挂起对指定数据库的所有更新。事务不能更新已挂起的数据库中的数据，后台任务（例如检查点进程和管家进程）会跳过所有处于挂起状态的数据库。
- 与 release 关键字一起使用的 quiesce database 允许恢复以前挂起的数据库上的更新。
- 与 for external dump 子句一起使用的 quiesce database 指示制作数据库的外部副本。
- 如果执行以下操作，恢复此数据库可能会失败或者数据库可能不一致：
  - 停顿减弱持久性数据库（包括 tempdb）
  - 复制设备

  - 将此数据库装入到另一 Adaptive Server 上

  发出 quiesce database 命令之前，使用 alter database 更改数据库的持久性。

- quiesce database hold 和 release 命令不需要从同一个用户会话执行。
- 如果在 quiesce database hold 命令中指定的数据库中含有处于就绪状态的分布式事务或多数据库事务， Adaptive Server 会在超时期间等待以完成这些事务。如果这些事务在超时期限内未完成，则 quiesce database hold 会失败。
- 如果 Adaptive Server 正在对 quiesce database hold 中指定的数据库执行 dump database 或 dump transaction 命令，则该数据库只有在 dump 命令完成后才会被挂起。
- 如果在对数据库的更新处于挂起状态时执行 dump database 或 dump transaction 命令，则 Adaptive Server 会阻塞这两个命令，直到使用 quiesce database release 释放该数据库。
- 如果尝试对受抑制的数据库运行查询， Adaptive Server 会发出错误消息 880：

  ```
  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
  ```

  一旦数据库不再处于抑制状态，便会运行查询。

- 若要复制数据库，请使用 quiesce database 命令和用于创建清单文件的扩展。 quiesce database 通过阻塞写入数据库的操作实现 quiesce hold，然后创建清单文件。然后，该命令将对数据库的控制权返回给用户。现在可以使用一种实用程序将数据库复制到另一 Adaptive Server。对于复制操作，必须遵循 quiesce database hold 的以下规则：
  - 复制操作在 quiesce database hold 过程完成后才能开始。
  - 必须复制 quiesce database 命令中的每个数据库的全部设备。
  - 在调用 quiesce database release 之前，必须完成复制进程。

加密列和 *quiesce database*

- 如果数据库包含加密密钥，则可以使用 quiesce database。
- 如果数据库中的列是使用其它数据库中存储的密钥加密的，则必须使用 with override 对该数据库执行 quiesce。
- 允许 quiesce database *key_db, col_db*，其中 *key_db* 是包含加密密钥的数据库， *col_db* 是表中含有用 *key_db* 中的密钥加密的列的数据库。

在集群环境中停顿数据库

- 如果发出 shutdown instance 或 shutdown cluster，则集群将中止所有 quiesce database 命令。
- 如果正在执行 shutdown instance 或 shutdown cluster 命令，则集群将拒绝用户发出的所有 quiescedb 命令。
- 如果正在进行实例故障切换恢复，则集群将中止所有 quiesce database 命令。
- 如果正在进行实例故障切换恢复，则集群将拒绝用户发出的所有 quiesce database 命令。
- 如果正在执行的 quiesce database hold 命令涉及 master 数据库，则不能向集群添加新实例。

权限

对 quiesce database 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 quiesce any database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
96	quiesce	quiesce database	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 dump database、 dump transaction、 mount、 unmount.

系统过程 sp_helpdb, sp_who.

raiserror

说明

在用户屏幕上输出用户定义的错误消息并设置系统标记来记录发生了错误这一情况。

语法

raiserror *error_number*
[{*format_string* | @*local_variable*}] [, *arg_list*]
[with errordata *restricted_select_list*]

参数

error_number

是值大于 17,000 的局部变量或整数。如果 *error_number* 介于 17,000 和 19,999 之间，并且 *format_string* 缺失或为空 ("")，则 Adaptive Server 从 master 数据库的 sysmessages 表中检索错误消息文本。这些错误消息主要供系统过程使用。

如果 *error_number* 等于或大于 20,000，且 *format_string* 缺失或为空，则 raiserror 从最初发出该查询或存放该存储过程的数据库中的 sysusermessages 表中检索消息文本。 Adaptive Server 尝试用 @@*langid* 的当前设置所定义的语言从 sysmessages 或 sysusermessages 检索消息。

format_string

是一个最大长度为 1024 字节的字符串。可以选择用局部变量声明 *format_string*，并在 raiserror 中使用该变量 （请参见 @*local_variable*）。

raiserror 可识别要输出的字符串中的占位符。格式字符串最多可包含 20 个任意顺序的唯一占位符。将消息文本发送到客户端时，使用 *format_string* 后接参数的格式化内容替换这些占位符。

为了在用其它语法结构将格式字符串转换为某种语言时，可以对参数重新排序，要对占位符进行编号。参数所用占位符的显示形式如下：“%nn!”— 一个百分比符号 (%)，后接一个 1 到 20 之间的整数，后面再接一个感叹号 (!)。整数表示参数列表中字符串中的参数编号。“%1!”“%1!”是原始版本中的第一个参数，“%2!”是第二个参数，依此类推。

用这种方法指示参数的位置可以使转换正确，即使参数出现在目标语言中的顺序与其在原始语言中的顺序不同也是如此。

例如，假定以下是一条英文消息：

```
%1! is not allowed in %2!.
```

此消息的德语版本是：

```
%1! ist in %2!
```

此消息的日语版本是：

```
%2! の中で %1! は許されません。
```

在此示例中，“%1!”在所有三种语言中都表示同一参数，“%2!”也是如此。此示例显示了翻译格式中有时需要对参数重新排序。

@local_variable

是一个包含 *format_string* 值的局部变量。它必须是 char 或 varchar 类型，且必须在使用它的批处理或过程中予以声明。

arg_list

是一系列由逗号分隔的变量或常量。*arg_list* 是可选的，除非提供了包含“%*nn*!”形式的占位符的格式字符串。参数可以是除 text 和 image 以外的任何数据类型；在出现在最终的字符串中之前，它将被转换为 char 数据类型。

如果参数求值结果为 NULL， Adaptive Server 会将它转换成零长度的 char 字符串。

with errordata

为 Client-Library™ 程序提供扩展错误数据。

restricted_select_list

包含一个或多个下列项目：

- “*”，表示按 create table 顺序排列的所有列。
- 按您希望的查看顺序排列的列名称列表。当选择现有的 IDENTITY 列时，可使用由表名限定的 syb_identity 关键字在必要的地方替代实际的列名。
- 向结果表中添加新 IDENTITY 列的规范：

  ```
  column_name = identity (precision)
  ```

- 缺省列标题 （列名）的替代内容，格式如下：

  ```
  column_heading = column_name
  column_name column_heading
  ```

column_name as *column_heading*

在所有这些格式中，都可以将列标题用引号引起来。如果标题不是有效的标识符 （即标题是保留字、标题以特殊字符开始或者标题包含空格或标点符号），就必须用引号引起来。

- 表达式 （列名、常量、函数、任何由算术运算符或逐位运算符连接起来的列名、常量和函数的组合，也可以是子查询）。
- 内置函数或集合。
- 上面所列项目的任意组合。

restricted_select_list 也可以按如下形式执行变量赋值：

```
@variable = expression
[, @variable = expression ...]
```

restricted_select_list 的限制包括：

- 不能将变量赋值与任何其它 *restricted_select_list* 选项合并。
- 不能在 *restricted_select_list* 中使用 from、where 或其它 select 子句。
- 不能在 *restricted_select_list* 中使用 “*” 表示所有列。

请参见 《Transact-SQL 用户指南》。

示例

示例 1 如果没有找到用 *@tabname* 参数提供的表，则该示例存储过程将返回一个错误：

```
create procedure showtable_sp @tabname varchar (18)
as
if not exists (select name from sysobjects
    where name = @tabname)
    begin
        raiserror 99999 "Table %1! not found.",
        @tabname
    end
else
    begin
        select sysobjects.name, type, crdate, indid
        from sysindexes, sysobjects
        where sysobjects.name = @tabname
        and sysobjects.id = sysindexes.id
    end
```

示例 2 本示例在 sysusermessages 中添加一条消息，然后使用 raiserror 测试该消息，并提供替换参数：

```
sp_addmessage 25001,
"There is already a remote user named '%1!'
```

```
for remote server '%2!'."

raiserror 25001, jane, myserver
```

示例 3 本示例使用 with errordata 选项将扩展错误数据 *column* 和 *server* 返回给客户端应用程序，以说明涉及的列和所用的服务器：

```
raiserror 20100 "Login must be at least 5
    characters long" with errordata "column" =
    "login", "server" = @@servername
```

用法

- 用户定义的消息可以即席生成 （如示例 1 和示例 3），或者可以将其添加到系统表 sysusermessages 中供所有应用程序使用 （如示例 2）。可使用 sp_addmessage 将消息添加到 sysusermessages 中；使用 sp_getmessage 检索消息以供 print 和 raiserror 使用。
- 用户定义的错误消息的错误号必须大于 20,000。最大值为 2,147,483,647 ($2^{31}-1$)。
- 所有用户定义错误消息的严重级都是 16。这表示用户已经犯了非致命错误。
- 替换后， *format_string* 加上所有参数的最大输出字符串长度是 1024 个字节。
- 如果在格式字符串中使用占位符，请记住：对于字符串中的每个占位符 *n*，占位符 *1* 到 *n-1* 也必须存在于同一字符串中，不过它们不必以数值顺序出现。例如，不允许将占位符 1 和 3 放在一个格式字符串中，却不把占位符 2 也放在此字符串中。如果在格式字符串中省略了一个数字，则执行 raiserror 时会生成错误消息。
- 如果相对于 *format_string* 中的占位符来说参数太少，Adaptive Server 会显示一条错误消息并中止当前执行的语句，但不中止任何打开的事务。但是，如果此错误发生在存储过程中， Adaptive Server 会继续在名为 raiserror 的行执行下一个语句，任何打开的事务仍然打开。如果此过程发生在 SQL 代码批处理中， Adaptive Server 会中止该批处理，任何打开的事务仍然打开。
- 若要在错误消息中包括实际的百分号，请在 *format_string* 中使用两个百分号 （“%%”）。如果 *format_string* 中包括不是用作占位符的单个百分号 （“%”）， Adaptive Server 将返回错误消息。
- 如果参数求值结果为 NULL，它将被转换成零长度的 char 字符串。如果不希望输出中出现零长度字符串，可使用 isnull 函数。
- 当执行 raiserror 时，错误号放置在全局变量 @@*error* 中，该变量存储系统最新生成的错误号。

- 如果希望将错误号存储在 @@*error* 中，请使用 raiserror，而不要使用 print。
- 若要在 raiserror 中包括 *arg_list*，请在 *error_number* 后加一个逗号，或者在第一个参数前加上 *format_string*。若要包括扩展错误数据，请用空格 （不是逗号）将 *extended_value* 与 *error_number*、 *format_string* 或 *arg_list* 隔开。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 raiserror 无需任何权限。

另请参见

命令 declare, print.

系统过程 sp_addmessage, sp_getmessage.

readtext

说明

从指定偏移位置开始，读取指定字节数或字符数的 text、unitext 和 image 值。

语法

```
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]}]
```

参数

table_name.column_name

是 text、unitext 或 image 列的名称。必须包括表名称。如果该表位于另一数据库中，请指定数据库名；如果数据库中有多个具有该名称的表，请指定所有者的名称。 *owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

text_pointer

是一个 varbinary (16) 值，该值存储指向 text、unitext 或 image 数据的指针。请使用 textptr 函数确定该值。text、unitext 和 image 数据不与其它表列存储在同一组链接页中。它存储在单独的一组链接页中。指向实际位置的指针和数据存储在一起；textptr 返回该指针。

offset

指定在开始读取 text、unitext 或 image 数据之前要跳过的字节数或字符数。

size

指定要读取的数据的字节数或字符数。

holdlock

导致文本值被锁定以进行读取，直到事务结束。其他用户可以读取该值，但不能修改该值。

noholdlock

禁止服务器持有执行该语句期间获得的任何锁，无论当前处于何种事务隔离级别。在查询中不能同时指定 holdlock 和 noholdlock 选项。

readpast

指定 readtext 自动跳过带排它锁的行，不必等待也不生成消息。

using
指定 readtext 将 *offset* 和 *size* 参数解释为字节数 (bytes) 还是 textptr 字符数 （chars 与 characters 同义）。当用于单字节字符集或用于 image 值（readtext 逐字节读取 image 值）时，该选项无效。如果未提供 using 选项，则 readtext 将 *size* 和 *offset* 参数解释为字节数。

at isolation
指定查询的隔离级别（0、1 或 3）。如果忽略该子句，查询将使用执行它的会话的隔离级别 （缺省隔离级别为 1）。如果在指定了 at isolation read uncommitted 的查询中同时还指定 holdlock，则 Adaptive Server 会发出警告并忽略 at isolation 子句。对于其它隔离级别，holdlock 优先于 at isolation 子句。

read uncommitted
将查询的隔离级别指定为 0。可以在 at isolation 子句中指定 0 来代替 read uncommitted。

read committed
将查询的隔离级别指定为 1。可以在 at isolation 子句中指定 1 来代替 read committed。

可重复的读取
将查询的隔离级别指定为 2。可以在 at isolation 子句中指定 2 来代替 serializable。

serializable
将查询的隔离级别指定为 3。可以在 at isolation 子句中指定 3 来代替 serializable。

示例

示例 1 选择 copy 列的第二个到第六个字符：

```
declare @val varbinary(16)
select @val = textptr(copy) from blurbs
where au_id = "648-92-1872"
readtext blurbs.copy @val 1 5 using chars
```

示例 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
```

用法

- textptr 函数将 16 字节的二进制字符串（文本指针）返回到指定行的 text、 unitext 或 image 列，或者如果查询返回了多个行，则返回到最后一行的 text、 unitext 或 image 列。声明一个局部变量来容纳文本指针，然后用 readtext 使用此变量。

- 全局变量 @@*textsize* 中的值是返回数据的字节数限制，如果它小于为 readtext 指定的大小，则会取代后者。用 set textsize 可更改 @@*textsize* 的值。
- 在使用字节数作为偏移和大小时， Adaptive Server 可能会在要返回的 text 数据的开头或末尾找到部分字符。如果是这样，且字符集转换被打开，则服务器在将文本返回到客户端之前，
- Adaptive Server 必须确定要发送到客户端的字节数，来响应 readtext 命令。当 *offset* 和 *size* 用字节数表示时，要确定返回文本中的字节数很简单。当偏移和大小用字符数表示时，服务器必须计算返回到客户端的字节数。因此，在使用字符数表示 *offset* 和 *size* 时，执行速度可能会较慢。只有在 Adaptive Server 使用多字节字符集时，using characters 选项才有用：它确保 readtext 不会返回部分字符。
- 不能对视图中的 text、 unitext 或 image 列使用 readtext。
- 如果试图在更改为多字节字符集后在 text 值上使用 readtext，且没有运行 dbcc fix_text，则命令会失败，并会有一条错误消息指示您在表上运行 dbcc fix_text。

对 *unitext* 列使用 *readtext*

在对定义为 unitext 数据类型的列发出 readtext 时，readtext *offset* 参数指定在开始读取 unitext 数据之前要跳过的字节数或 Unicode 值。readtext *size* 参数指定要读取的字节数或 16 位 Unicode 值。如果指定 using bytes （缺省值），则在必要时, *offset* 和 *size* 值将被调整为始终以 Unicode 字符边界开始和结束。

如果 enable surrogate processing 为 on，则 readtext 只在代理边界上截断，开始/结束位置也相应调整，并返回整个 Unicode 字符。因此，如果对定义为 unitext 的列发出 readtext，则所返回的字节数可能会比指定的字节数少。

在下面的示例中， unitext 列 ut 包含字符串 U+0101U+0041U+0042U+0043：

```
declare @val varbinary(16)
select @val = textptr (ut) from unitable
where i = 1
readtext foo.ut @val 1 5
```

该查询将返回值 U+0041U+0042。

offset 位置被调整为 2，因为 readtext 不能从 Unicode 字符的第二个字节处开始读取。 Unicode 字符总是由偶数个字节组成。从第二个字节开始（或以奇数字节结束）会使结果偏移一个字节，从而导致结果集不正确。

在上面的示例中，*size* 值被调整为 4，因为 readtext 无法读取第四个字符 U+0043 的部分字节。

在下面的查询中，启用了 enable surrogate processing，并且 ut 列包含字符串 U+d800dc00U+00c2U+dbffdeffU+d800dc00：

```
declare @val varbinary(16)
select @val = textptr (ut) from unitable
where i = 2
readtext foo.ut @val 1 8
```

该查询将返回值 U+00c2U+dbffdeff。起始位置被重新设置为 2，并且实际的结果大小为 6 个字节而不是 8 个字节，原因是 readtext 未拆分代理对。代理对 （在此示例中为范围 d800..dbff 中的第一个值和范围 dc00..dfff 中的第二个值）需要 4 字节边界，但遵从 UTF-16 的 Unicode 规则不允许拆分这些 4 字节字符。

使用 *readpast* 选项

- readpast 选项只适用于仅数据锁定表。如果将该选项指定给所有页锁定表，它将被忽略。
- readpast 选项与 holdlock 选项不兼容。如果在一个命令中指定了两者，将产生错误并终止命令。
- 如果 readtext 指定了 at isolation read uncommitted， readpast 将产生警告但不终止命令。
- 如果语句隔离级别设置为 3， readpast 将产生错误并终止命令。
- 如果会话范围的隔离级别为 3，将不加提示地忽略 readpast 选项。
- 如果会话范围的隔离级别是 0， readpast 将产生警告但不终止命令。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

readtext 需要具有对表的 select 权限。

另请参见

命令 set, writetext.

系统过程 text、 image 和 unitext 数据类型.

reconfigure

说明	reconfigure 命令目前不起作用，包含它是为了使现有脚本不进行任何修改就可以运行。
语法	reconfigure
用法	如果您有包括 reconfigure 的脚本，应尽快更改它们。虽然当前版本仍包括 reconfigure，但后续版本可能不再支持它。
标准	符合 ANSI SQL 的级别Transact-SQL 扩展。
权限	reconfigure 权限缺省情况下授予系统管理员，而且是不可移交的。
另请参见	**系统过程** sp_configure.

refresh

说明

刷新指定的预计算结果集。

语法

refresh precomputed result set *prs_name*

参数

prs_name

预先计算的结果集的名称。完全限定的 *prs_name* 不得含有服务器或数据库名。

示例

刷新 au_prs 预先计算的结果集：

```
refresh precomputed result set au_prs
```

用法

- refresh precomputed result set 权限在缺省情况下将授予预先计算的结果集的所有者，获得授权的所有者可将此权限移交给其他用户。

标准

refresh precomputed result set 命令属于 Transact-SQL 扩展，不包含在 SQL 标准内。

权限

您必须拥有 create table 特权才能刷新预先计算结果集。

审计

不对预先计算结果集刷新进行审计。

remove java

说明

当数据库中安装 Java 类后从数据库中删除一个或多个 Java-SQL 类、软件包或 JAR。

语法

```
remove java
        class class_name[, class_name]...
            | package package_name[, package_name]...
            | jar jar_name[, jar_name]...[retain classes]
```

参数

class *class_name*

要从数据库中删除的一个或多个 Java 类的名称。类必须安装在当前数据库中。

package *package_name*

要删除的一个或多个 Java 包的名称。包必须存储在当前数据库中。

jar *jar_name*

是 SQL 标识符或包含有效 SQL 标识符的可多达 30 字节的字符串值。每个 *jar_name* 必须与当前数据库中保留的 JAR 的名称相等。

retain classes

指定所指定的 JAR 不再保留在数据库中，且保留的类没有相关联的 JAR。

用法

- 如果存储过程中包含 remove java 语句，则当前数据库是过程创建时的当前数据库而不是过程被调用时的当前数据库。

  如果存储过程中不包含 remove java 语句，则当前数据库是 remove 语句执行时的当前数据库。

- 如果指定了 class 或 package 且任何已删除的类有相关联的 JAR，那么将会引发例外。
- 如果任何存储过程、表或视图包含对已删除类的引用，如列、变量或参数的数据类型，那么将会引发例外。
- 所有删除的类都会：
  - 从当前数据库中删除。
  - 从当前连接的 Java 虚拟机 (Java VM) 卸载。删除的类不会从其它连接的 Java VM 中卸载。
- 如果在 remove java 执行期间引发了任何例外，则 remove java 的所有操作都会被取消。
- 不能删除被 SQLJ 存储过程或函数直接引用的 Java-SQL 类。
- 若要从数据库中删除 Java-SQL 类，您必须：

a 使用 drop procedure 和 drop function 删除所有直接引用此类的 SQLJ 存储过程或函数。

b 使用 remove java 从数据库中删除 Java-SQL 类。

锁

- 使用 remove java 时，会在 sysxtypes 上放置一个排它表锁。
- 如果指定了 jar，则会在 sysjars 上放置一个排它表锁。

权限

只有系统管理员或数据库所有者才能使用 remove java。

对 remove jar class 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	**审计选项**	**审计的命令或访问权限**	**extrainfo 中的信息**
94	remove	remove java	• *角色* – 当前活动角色 • *关键字或选项* – NULL • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

文档 《Adaptive Server Enterprise 中的 Java》 .

系统过程 sp_helpjava.

系统表 sysjars、 sysxtypes.

实用程序 extractjava, installjava.

reorg

说明

根据使用的选项，回收页上未使用的空间、删除行转移或将表中所有行重新写入到新页中。

语法

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}]

参数

compact

组合 reorg reclaim_space 和 reorg forwarded_rows 的功能以在同一步骤回收空间和撤消行转移。

forwarded_rows

删除行转移。

在配置了压缩的表中取消转移或重新插入数据行，会根据表的压缩级别对该行进行压缩。

index_partition_name

是正在对其运行 reorg 的索引分区的名称。update statistics 会执行检查来验证 *index_partition_name* 是不是索引分区。如果指定了一个索引分区，则仅重建该索引分区

indexname

指定要重新组织的索引的名称。

partition_name

是正在对其运行 reorg 的分区的名称。

tablename

指定要重新组织的表的名称。如果指定了 *indexname*，则只重新组织索引。

rebuild

如果指定了表名，则将表中的所有行重新写入到新页中，这样就会根据表的聚簇索引（如果存在）安排表，使得所有页符合当前空间管理设置、没有转移的行且页的行间没有间隔。如果表有索引，则删除并重新创建所有索引。如果指定了索引名， reorg 在表可进行读取和更新活动时重建该索引。

新行采用分区或表的压缩级别，无论原来的表或分区中的数据是什么压缩级别。

注释 系统目录不支持 reorg rebuild。

with online

允许在 reorg rebuild 运行时对表执行并发访问。

reclaim_space

回收删除和更新后留下的未使用空间。对表中的每个数据页来说，如果有因提交的删除或缩短行更新而产生的未使用空间， reorg reclaim_space 将连续重写当前行，而将所有未使用的空间留在页尾。如果页上没有行，则页会被释放。

如果表标记了压缩，则 reclaim_space 会压缩数据。

注释 reorg reclaim_space 只影响具有可变长度行的表，并且只释放页中的空间。若要减少使用页数，请使用 reorg rebuild 命令。

with resume

从上一个 reorg 命令终止的点启动重组。当上一个 reorg 命令指定了时间限制 (with time = *no_of_minutes*) 时使用。

with time = *no_of_minutes*

指定 reorg 命令运行的分钟数。

with compress

允许您压缩受 reorg 操作影响的行。

示例

示例 1 回收 titles 表中未使用的页空间：

```
reorg reclaim_space titles
```

示例 2 回收索引 titleind 中未使用的页空间：

```
reorg reclaim_space titles titleind
```

示例 3 在 titles 表上启动 reorg compact。 reorg 从表的起始处开始并持续 120 分钟。如果 reorg 在时间限制内完成，它将返回到表的起始处继续执行直到全部时间耗尽：

```
reorg compact titles with time = 120
```

示例 4 在前一个 reorg compact 结束的点启动 reorg compact 并持续 30 分钟：

```
reorg compact titles with resume, time = 30
```

示例 5 对 titles 表的 smallsales 分区运行 reorg forwarded_rows：

```
reorg forwarded_rows titles partition smallsales
```

示例 6 对 authors 表运行 reorg forwarded_rows：

```
reorg forwarded_rows authors
```

示例 7 对 titles 表的 bigsales 分区运行 reorg reclaim_space：

```
reorg reclaim_space titles partition bigsales
```

示例 8 对 titles 表的 bigsales 分区运行 reorg compact：

```
reorg compact titles partition bigsales
```

示例 9 对 titles 表运行 reorg compact，并压缩受影响的行：

```
reorg compact titles with compress
```

示例 10 对 sales 表的索引为 local_idx 的索引分区 idx_p2 运行 reorg rebuild：

```
reorg rebuild sales local_idx partition idx_p2
```

用法

- 在 reorg （reorg rebuild 除外）中指定的表必须有数据行或数据页锁定方案。
- 运行 reorg 之后，索引扫描速度更快。
- 对表运行 reorg 将会对并行查询的性能产生负面影响。
- 如果不包含索引或分区名称，则重建整个表。
- 重建表的索引之后，可以在表中执行 dump tran。不过，如果整个表都被重建，则不能执行 dump tran。
- 尽管在位置索引中允许联机索引重建，但它仅重建索引页。数据页保持不变，这意味着数据行既不被排序也不被重新写入到刷新页中。可以通过删除位置索引然后重新创建它来重建数据页。
- 可以重建 systabstats 的索引，但是不能对表本身运行 reorg rebuild。
- 低于 15.0 的 Adaptive Server 版本限制您对所有页锁定表使用 reorg rebuild。Adaptive Server 15.0 版和更高版本允许您对使用所有页锁定的整个表运行 reorg rebuild。reorg rebuild 会将数据复制到一组新页中并重建全部索引，从而重建整个表。
- 不能对所有页锁定表使用 reorg rebuild 的子命令 （例如 compact、reclaim_space 和 forwarded_rows）。

- 不能对所有页锁定表使用 reorg rebuild *table_name index_name*。
- 运行 reorg rebuild *table_name* 会更新所有前导索引列的统计信息。但是，运行 reorg rebuild *table_name index_name* 不会自动更新统计信息。不过，如果更新中包含的数据变更足以影响其计划选择和性能，则运行 reorg rebuild *index_name* 时， Adaptive Server 会自动更新索引统计信息。
- writetext 可与 online 参数同时运行。
- reorg 对分配给 text 或 image 列的空间没有影响。
- 不能在事务内发出 reorg。
- reorg rebuild 需要将数据库选项 select into/bulkcopy/pllsort 设置为 true 并在数据库中运行 checkpoint。
- reorg rebuild 需要与表的大小和索引一样大小的额外磁盘空间。使用 sp_spaceused 可以找出表当前占用空间的大小。可以使用 sp_helpsegment 检查可用空间的容量。
- 运行 reorg rebuild 后，必须在转储事务日志前转储数据库。
- 对使用索引中的 reorg rebuild 的要求不如对在表中使用该命令的要求严格。有以下一些规则：
  - 不需要设置 select into 来重建索引。
  - 重建表需要有一个可容纳完整的该表副本的空间。重建索引在小事务中进行，并且在页复制之后马上释放这些页，因此，进程仅需要可容纳在每个事务中复制的页的空间。
  - 可以在事务级扫描 （脏读）处于活动状态时重建表中的索引。

reclaim_space、 *forwarded_rows* 和 *compact* 参数

- 通过使用多个短期小事务使得与其它活动的冲突减至最小。每个事务被限制为八页 reorg 处理。
- 重写单个分区的空间。
- 提供 resume 和 time 选项，可以设置 reorg 运行时间的长度限制，并能从前一个 reorg 命令停止处继续运行 reorg。例如，这允许用户在非高峰时间使用一系列部分重组操作来对大表运行 reorg 命令。

碎片收集和锁

- 对于数据行表 – Adaptive Server 使用闩锁来执行碎片收集，并在移至下一页前释放页上的闩锁。除非碎片收集遇到转移的行时获得排它表锁 （它会将其保留到该事务结束为止），否则不会获得任何锁。后续的事务使用闩锁，直到遇到转移的行。

- 对于数据页表 – Adaptive Server 使用页锁来执行碎片收集，但在移至下一页前释放页锁。当碎片收集遇到转移的行时，它会获得排它表锁 （它会将其保留到该事务结束为止）。后续的事务使用页锁，直到遇到其它转移的行。
- 如果碎片收集遇到分配给对象的 OAM 页，但不引用分配 （运行 reorg compact 需要共享表锁），请使用 reorg compact。

使用 resume 和 time 参数

使用 resume 和 time 参数时应考虑以下问题：

- 如果只指定了 resume 选项，则 reorg 在前一 reorg 命令停止的地方开始，一直到表的末尾时结束。
- 如果只指定了 time 选项，则 reorg 命令从表的起始处开始执行，一直持续到所指定的分钟数时结束。
- 如果指定了这两个选项，则 reorg 从前一 reorg 命令停止的地方开始执行，一直持续到指定的分钟数时结束。

对压缩表运行 reorg

- reorg rebuild – 根据位置索引 （如果存在）来排序行，根据当前有效的空间管理设置将行写入新数据页。根据各个分区的压缩级别对新行进行压缩和解压缩，无论原来的表或分区中的数据压缩状态如何。
- reorg reclaim_space – 压缩数据行以节省更多空间 （如果表标记了压缩）。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 reorg 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或拥有 reorg any table 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是表所有者或具有 sa_role 的用户。

另请参见

文档 请参见 《系统管理指南》。

系统过程 sp_chgattribute.

return

说明

无条件退出批处理或过程，并提供可选返回状态。 return 之后的语句不会执行。

语法

return [*integer_expression*] [plan "*abstract_plan*"]

参数

integer_expression

是由过程返回的整数值。存储过程可以向调用过程或应用程序返回一个整数值。

plan "*abstract_plan*"

指定用来优化查询的抽象计划。抽象计划可以是用抽象计划语言指定的完整或部分计划。只能为可优化的 SQL 语句 （即访问表的查询）指定计划。有关详细信息，请参见 《性能和调优指南：优化程序和抽象计划》中的 “创建和使用抽象计划”。

示例

示例 1 如果没有指定用户名参数，则 return 命令将导致退出此过程 （将消息发送给用户屏幕之后）。如果指定了用户名，则从相应的系统表中检索当前数据库中该用户创建的规则名称。

```
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
```

示例 2 如果更新导致商业类图书的平均价格超过 $15， return 命令在对 titles 执行更多更新之前终止批处理：

```
print "Begin update batch"
update titles
    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'
```

示例 3 此过程创建两个用户定义的状态代码：如果 contract 列包含 1，则返回值 1；所有其它情况返回 2（例如， contract 上是 0 值或 title_id 与行不匹配）：

```
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
```

用法

- 返回状态值可用于执行当前过程的批处理或过程的后续语句，但必须以下列形式给出：

  ```
  execute @retval = procedure_name
  ```

  有关详细信息，请参见 execute。

- Adaptive Server 保留 0 来指示返回成功，而用 -1 到 -99 之间的负数指示失败的各种原因。如果未提供用户定义的返回值，则使用 Adaptive Server 的值。用户定义的返回状态值不能与 Adaptive Server 保留的值冲突。当前使用 0 和从 -1 到 -14 的值：

值	含义
0	过程执行时没有发生错误
-1	缺失对象
-2	数据类型错误
-3	进程被选作死锁牺牲品
-4	权限错误
-5	语法错误
-6	杂类用户错误
-7	资源错误，如空间不足
-8	非致命内部问题
-9	达到系统限制
-10	致命内部不一致性

值	含义
-11	致命内部不一致性
-12	表或索引损坏
-13	数据库损坏
-14	硬件错误

值 -15 至 -99 留作 Adaptive Server 将来使用。

- 如果在执行时发生了多个错误，将返回绝对值最高的状态。用户定义的返回值始终优先于 Adaptive Server 提供的返回值。
- return 命令可在想退出批处理或过程的任何地方使用。返回立即执行并且是完全的：return 之后的语句不会执行。
- 存储过程不能返回 NULL 返回状态。如果过程试图返回空值 （例如，使用 return *@status*，而其中 *@status* 是 NULL），将产生警告消息，并返回一个 0 到 -14 之间的值。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 return 无需任何权限。

另请参见

命令 begin...end, execute, if...else, while.

revoke

说明

撤消用户、角色或组的权限。

语法

撤消访问数据库对象的权限：

```
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]
```

撤消选择内置函数的权限：

```
revoke select
    on [builtin] builtin
    from {name_list | role_list}
    [granted by grantor]
```

撤消系统特权：

```
revoke {all [privileges] | privilege_list}
    from {public | name_list | role_list}
    [granted by grantor]
```

撤消运行 set proxy 的权限

```
revoke set proxy
    from {public | name_list | role_list}
    [granted by grantor]
```

撤消 dbcc 特权：

```
revoke {dbcc_privilege [on database]
        [, dbcc_privilege [on database], ...]}
    from {user_list | role_list}
    [granted by grantor]
```

撤消 public 的缺省权限：

```
revoke default permissions on system tables
```

参数

all

指派访问数据库对象的权限时 （请参见 “撤消访问数据库对象的权限”的语法）， all 指定撤消所有适用于指定对象的权限（decrypt 权限除外）。所有对象所有者都可以使用含有对象名的 revoke all 撤消各自对象的权限。 decrypt 权限必须单独撤消。

未启用细化权限时，系统管理员或数据库所有者可以使用 revoke all 撤消创建数据库对象的权限 （请参见 “撤消系统特权”的语法）。由系统管理员使用时， revoke all 将撤消所有 create 特权 （create database、 create default、 create procedure、 create rule、 create table、 create function 和 create view）。数据库所有者使用 revoke all 或在 master 数据库外执行 revoke all 时， Adaptive Server 将撤消除 create database 之外的所有 create 特权，并显示一条信息性消息。

当启用细化权限时，不支持使用 revoke all 撤消所有 create 特权。有关详细信息，请参见 《安全性管理指南》中的 “使用细化权限”。

不能将 all 用于包含 where 子句的 revoke 语句。

permission_list

是要撤消的权限的列表。如果列出的权限不止一个，可用逗号将它们隔开。下表说明每种类型的对象可以授予和撤消的访问权限。

对象	permission_list 可包括
表	select、 insert、 delete、 update references、 update statistics、 delete statistics、 truncate table、 decrypt、 identity_insert *、 identity_update *
视图	select、insert、delete、update、decrypt、identity_insert *、identity_update *
列	select、 update、 references、 decrypt 列名可以在 *permission_list* 或 *column_list* 中指定。
存储过程	execute
加密密钥	select
函数	execute *

注释 标有星号 (*) 的权限仅在启用细化权限时才能授予。

builtin

是内置函数。通过指定内置函数，可将表和同名的可撤消内置函数区分开来。这些函数是 set_appcontext、 get_appcontext、 list_appcontext、 authmech 和 rm_appcontext。

privilege_list

是可以撤消的一组系统特权。系统特权包括服务器范围和数据库范围特权。请参见表 1-21 和表 1-22 查看可以撤消的系统特权的列表。另请参见 “用法”部分了解有关如何撤消系统特权的详细信息。使用逗号分隔多个命令。

table_name

是您在撤消其权限的表的名称。该表必须在当前数据库中。

column_list

是由逗号分隔的该权限的适用于列的列表。如果指定了列，则只能撤消 select、 reference、 decrypt 和 update 权限。

view_name

是您在撤消其权限的视图的名称。该视图必须位于当前数据库中。

stored _procedure_name

是您在撤消其权限的存储过程的名称。该存储过程必须位于当前数据库中。

function_name

是您要撤消其权限的函数的名称。此函数必须存在于当前数据库中。

keyname

是您在撤消其权限的密钥的名称。加密密钥必须位于当前数据库中。对于每个 revoke 语句只能列出一个对象。只能撤消对密钥的 select 权限。

[with {*pred_name* | {all | no} predicates}]

可以后跟命名的谓词、双重关键字 all predicates 或双重关键字 no predicates。

- *pred_name* - 您要撤消的带谓词授权的名称。 sp_helprotect 显示用于标识行过滤授权的谓词名称。
- no predicates – 指示 Adaptive Server 仅删除命名被授予者的给定访问权限的不带谓词授权。
- no predicates – 指示 Adaptive Server 仅删除命名被授予者的给定访问权限的所有带谓词授权。保留所有不带谓词的授权。

如果省略 with 子句，则带谓词和不带谓词的 grant 都会被撤消 （缺省行为）。

public

指所有用户。对于对象访问权限， public 不包括对象所有者。对于对象创建权限或 set proxy 授权， public 不包括数据库所有者。不能使用 with grant option 为 “public”或者其它组或角色授予权限。

name_list
是由逗号分隔的用户名和组名列表。

set proxy
撤消用户充当另一用户的特权。

grant option for
撤消 with grant option 权限，以便 *name_list* 中指定的用户不能再将指定的权限授予其他用户。如果这些用户已将权限授予另一些用户，则必须使用 cascade 选项撤消其权限。 *name_list* 中指定的用户保留对象访问权限，但不能再将访问权限授予其他用户。grant option for 仅适用于对象访问权限，而不适用于对象创建权限。

cascade
从撤消者早先授予了权限的所有用户撤消指定的对象访问权限。仅适用于对象访问权限，不适用于对象创建权限。不带 grant option for 选项使用 revoke 时，由撤消者授予给其他用户的权限也被撤消：级联自动发生。

granted by *grantor*
指示撤消由 *grantor* 而非执行 revoke 命令的用户所授予的权限或特权。

grantor
当前数据库中的有效用户名。

dbcc_privilege
您要撤消的 dbcc 特权的名称。它不能是一个变量。表 1-21 和表 1-22 列出了 dbcc 特权。

database
是您正在撤消其权限的数据库的名称。它与特定于数据库的 dbcc 特权一起使用，仅对目标数据库撤消权限。被撤消者必须是目标数据库中的有效用户。 *database* 应符合标识符的规则，而且不能是变量。

如果在同一命令中存在多个撤消的操作，则 *database* 必须是唯一的。

user_list
是您在撤消其权限的用户的列表，而且不能是变量。

role_list
是您在撤消其权限的系统角色或用户定义角色的名称列表，而且不能是变量。

注释 不能向 public 或组授予或撤消 dbcc 特权。

---

all [privileges]
使用 all 或 all privileges 可撤消所有授予和拒绝的特权。

column_list

如果与 with *pred_name* 一起使用，则带谓词的行级访问权限会从命名列中删除。如果仍存在针对该行级特权引用的其它列，则对于减少的列列表，特权及其相关的命名谓词仍位于 sysprotects 中。

系统表的缺省权限

指定撤消第 547 页的 “撤消系统表的缺省权限”中列出的系统表的缺省权限。

示例

示例 1 撤消 Mary 和 “sales”组对表 titles 的 insert 和 delete 权限：

```
revoke insert, delete
on titles
from mary, sales
```

示例 2 撤消 “public”（包括所有用户）对 get_appcontext 函数的 select 权限：

```
revoke select on builtin get_appcontext from public
```

将其与下列命令比较，如果存在一个名为 get_appcontext 的表，下列命令撤消对该表的 select 权限：

```
revoke select on get_appcontext from public
```

示例 3 有两种方法可以撤消 “public”对 titles 表的 price 和 advance 列的 update 权限：

```
revoke update
on titles (price, advance)
from public
```

或：

```
revoke update (price, advance)
on titles
from public
```

示例 4 撤消 Mary 和 John 使用 create database 和 create table 命令的权限。因为正在撤消 create database 权限，所以此命令只能在 master 数据库中执行。仅撤消 Mary 和 John 对 master 数据库的 create table 权限：

```
revoke create database, create table from mary, john
```

示例 5 撤消 Harry 和 Billy 执行 set proxy 或 set session authorization 从而可以在服务器上充当另一个用户的权限：

```
revoke set proxy from harry, billy
```

示例 6 撤消具有 sso_role 的用户执行 set proxy 或 set session authorization 的权限：

```
revoke set session authorization from sso_role
```

示例 7 撤消 Mary 在当前数据库中的所有对象创建权限（create encryption key 和 create trigger 除外）：

```
revoke all from mary
```

示例 8 撤消 Mary 对 titles 表的所有对象访问权限 （decrypt 权限除外）：

```
revoke all on titles from mary
```

示例 9 有两种方式撤消 Tom 在其它表 （引用 titles 表的 price 和 advance 列的表）上创建参照完整性约束的权限：

```
revoke references
on titles (price, advance)
from tom
```

或：

```
revoke references (price, advance)
on titles
from tom
```

示例 10 撤消所有被授予 “operator”角色的用户执行 new_sproc 的权限：

```
revoke execute on new_sproc from oper_role
```

示例 11 撤消 John 的以下权限：授予其他用户对 authors 表的 insert、update 和 delete 权限。同样撤消其他用户的 John 已授予的任何此类权限：

```
revoke grant option for
insert, update, delete
on authors
from john
cascade
```

示例 12 撤消 Frank 执行 dbcc 的特权：

```
revoke dbcc checkdb on all from frank
```

示例 13 撤消 Harry 在 authors 表上的 truncate table 和 update statistics 特权：

```
revoke truncate table on authors from harry
revoke update statistics on authors from harry
```

用户 Billy 和 Harry 不能再在 authors 上运行这些命令。

示例 14 撤消所有具有 oper_role 的用户的 truncate table、 update 和 delete statistics 特权：

```
revoke truncate table on authors from oper_role
revoke update statistics on authors from oper_role
revoke delete statistics on authors from oper_role
```

示例 15 撤消 public 的 decrypt 权限：

```
revoke decrypt on customer from public
```

示例 16 撤消用户 joe 的 create encryption key 权限：

```
revoke create encryption key from joe
```

示例 17 撤消数据库所有者对 ssn_key 的 select 权限。

```
revoke select on ssn_key from dbo
```

示例 18 以下示例假定已向 user1 授予以下权限，以便从表 t1 中进行选择：

- 无条件授权，可在所有行中查看 t1.col1 和 t1.col4：

  ```
  grant select on t1 (col1, col4) to user1
  ```

- 行过滤授权，在选择 t1.col2 或 t1.col3 时适用：

  ```
  grant select on t1 (col2, col3)
    where col1 = 1 as pred1
    to user1
  ```

删除对 t1.col2 的、带有 pred1 的 select 权限：

```
revoke select on t1 (col2) with pred1
  from user1
```

注释　执行此 revoke 命令后， user1 选择 t1.col3 时， pred1 仍适用。

如果授予者发出了以下命令之一，则 user1 对 t1 的所有使用 pred1 的权限都将被删除：

```
revoke select on t1 (col2, col3) with pred1
  from user1
```

或

```
revoke select on t1 with pred1
```

示例 19 在上一示例中所描述的授权之后，以下代码将删除对 t1.col2 和 t1.col3 的带谓词 pred1 的授权 （all predicates 将撤消给定访问权限和被授予者的所有行过滤带谓词授权）：

```
revoke select on t1 with all predicates
  from user1
```

示例 20 删除 user1 对 t1 的所有 select 访问权限；即，针对 t1 上的 select 访问，向 user1 授予的 t1 上的所有带谓词和不带谓词的授权：

```
revoke select on t1 from user1
```

示例 21 仅应用于不带谓词的授权：

```
revoke select on t1 with no predicates
```

示例 22 撤消 mary 向 john 授予的对 mary.books 表的 select 权限：

```
revoke select on mary.books from john granted by mary
```

示例 23 撤消用户 smith 的系统特权 manage any login：

```
use master
revoke manage any login from smith
```

用法

- 有关权限的详细信息，请参见 grant 命令。
- 只能撤消当前数据库中对象的权限。
- 只能撤消您未使用 granted by *grantor* 选项授予的权限或特权。如果使用了 granted by *grantor* 选项，您可以撤消由其他用户授予的权限或特权。
- grant 和 revoke 命令区分先后顺序。发生冲突时，最近发出的命令生效。但带谓词的特权的授予和撤消则例外。请参见 《安全性管理指南》中的 “Adaptive Server 如何将带谓词的特权保存在 sysprotects 中”
- 在 revoke 语法中，可以用单词 to 替代 from。
- 如果在 revoke 语句中没有指定 grant option for，则 with grant option 权限与指定的对象访问权限都将从用户撤消。另外，如果用户已授予了任何其他用户指定的权限，所有那些权限都会被撤消。也就是说， revoke 级联发生。
- 不同的授予者可为用户、组或角色授予相同的特权或权限。在此情况下，表示同一特权或权限的多个相关授权的 sysprotects 将具有多个行。如果稍后撤消一个或多个授权，则只要还有一个授权未撤消，用户、组或角色就仍会具有特权或权限。
- grant 语句为每个接受此权限的用户、组或角色在 sysprotects 系统表中添加一行。如果随后对用户或组的该权限执行 revoke 命令，Adaptive Server 会从 sysprotects 中删除相应的行。如果仅撤消了选定的组成员的该权限，但并没有撤消获得此授权的整个组的该权限， Adaptive Server 保留其初始行并为撤消操作添加一个新行。

- 在缺省情况下，授予用户发出 create trigger 的权限。当撤消用户创建触发器的权限时，会在 sysprotects 表中为该用户添加一个撤消行。若要授予发出 create trigger 的权限，必须发出两个授权命令。第一个命令从 sysprotects 中删除撤消行；第二个命令插入一个授予行。如果撤消了创建触发器的权限，则用户甚至无法在自己拥有的表上创建触发器。撤消用户创建触发器的权限只影响从中发出 revoke 命令的数据库。

使用 *cascade* 选项

revoke grant option for 撤消用户授予其他用户指定权限的能力，但不撤消该用户的此权限。如果用户已将此权限授予其他用户，则必须使用 cascade 选项，否则，会收到错误消息并且 revoke 失败。

例如，假定要撤消用户 Bob 在 titles 上的 with grant option 权限，使用这个语句：

```
revoke grant option for select
on titles
from bob
cascade
```

- 如果 Bob 没有将此权限授予其他用户，此命令会撤消他授予的能力，但他仍然具有对 titles 表的 select 权限。
- 如果 Bob 已将此权限授予其他用户，则必须使用 cascade 选项。如果不使用此选项，您将收到错误消息并且 revoke 失败。 cascade 撤消已被 Bob 授予 select 权限的所有用户的该权限，同时撤消他们将其授予其他人的能力。

不能使用带有 cascade 选项的 revoke 命令来撤消由表所有者授予的权限。例如，表的所有者 (UserA) 可以用如下语句将权限授予另一用户 (UserB)：

```
create table T1 (...)
grant select on T1 to UserB
```

不过，系统管理员不能用带有 cascade 选项的 revoke 特权命令用如下语句撤消 UserB 的特权：

```
revoke select on T1 from UserA cascade
```

该语句撤消了表所有者的 select 特权，但并不撤消 UserB 的那些特权。

缺省情况下，会隐式撤消表所有者以外用户的所有数据操作语言 (DML) 操作 （启用 restricted decrypt permission 时的解密权限除外。请参见 User Guide for Encrypted Columns （《加密列用户指南》））。因为 sysprotects 表中不包含指示表所有者已授权而后又撤消了特权的记录，所以 cascade 选项未被调用。

必须明确地撤消 UserB 的 select 特权。

使用 granted by

- 不允许使用 granted by 来撤消带谓词的特权。
- 不要求 grantor 具有执行 grant 命令的权限。
- 将撤消授予者 （由 sysprotects.grantor 表示）而非命令执行者所授予的权限。
- 使用 granted by 参数无需启用 enable granular permissions。
- 对于通过 with grant 选项收到对象的 grant 权限的用户，其不能发出 granted by 参数。所有其他用户均可发出 granted by 参数。

  例如， John 通过 with grant 选项获得 mary 的表的权限。当他试图使用 granted by 选项发出 revoke 语句时，将会返回一个错误。

  Mary：

  ```
  grant select on mary.books to john
  with grant option
  ```

  Mary：

  ```
  grant select on mary.books to smith
  ```

  John：

  ```
  revoke select on mary.books from smith
  granted by mary
  ```

撤消 *set proxy* 和 *set session authorization*

- 要撤消 set proxy 或 set session authorization，您必须处于 master 数据库中。
- set proxy 与 set session authorization 基本相同，只有一点区别：set session authorization 遵循 SQL 标准。要了解只采用 SQL 标准的命令和语法，请使用 set session authorization。
- revoke all *并不*包括 set proxy 或 set session authorization 权限。

数据库用户组

- 数据库用户组允许一次对多个用户 grant 或 revoke 权限。用户始终是缺省组和 “public” 的成员，还可以是一个其它组的成员。Adaptive Server 安装脚本为 “public” 指定了一组权限。

  使用 sp_addgroup 创建组，使用 sp_dropgroup 删除组。使用 sp_adduser 向组中添加新用户。使用 sp_changegroup 更改用户的组成员资格。若要显示组成员，请使用 sp_helpgroup。

撤消系统表的缺省权限

revokes default permissions on all system tables from “public.”

从任何数据库发出该命令时均可撤消其缺省权限的系统表有：

- sysalternates
- sysattributes
- syscolumns
- syscomments
- sysconstraints
- sysdepends
- sysindexes
- sysjars
- syskeys
- syslogs
- sysobjects
- syspartitions
- sysprocedures
- sysprotects
- sysqueryplans
- sysreferences
- sysroles
- syssegments
- sysstatistics
- systabstats
- systhresholds
- systypes
- sysusermessages
- sysusers
- sysxtypes

从 master 数据库发出此命令时可撤消其缺省权限的系统表有：

- sysdatabases
- sysdevices
- syslocks
- sysmessages
- sysprocesses
- systransactions
- sysusages
- sysconfigures
- syscurconfigs
- syslanguages
- syscharsets
- sysservers
- systimeranges
- sysresourcelimits
- syslogins
- sysremotelogins

撤消 *update statistics*、 *delete statistics* 和 *truncate table* 的权限

Adaptive Server 允许您撤消用户、角色和组对 update statistics、 delete statistics 和 truncate table 命令的权限。通过将 update statistics、 delete statistics 和 truncate table 添加到一个存储过程，然后将此过程的执行权限授予用户或角色，表所有者也可以通过隐式 grant 提供权限。

不能在列级撤消 update statistics 的权限。必须具有 sso_role 才能在 sysroles、 syssrvroles 和 sysloginroles 安全表上运行 update statistics 或 delete statistics。

缺省情况下，数据库所有者有权在除 sysroles、syssrvroles 和 sysloginroles 之外的系统表上运行 update statistics 和 delete statistics，并且可以将此特权转交给其他用户。

还可以发出 grant all 授予 update statistics、 delete statistics 和 truncate table 的权限。

注释 撤消用户执行 update statistics 的权限后，他们也就丧失了执行此命令的各种变化形式的权限，如 update all statistics、update partition statistics、update index statistics、 update statistics *table* 等。例如，下面的命令撤消 Billy 在 authors 表上运行 update statistics 的各种变化形式的权限：

```
revoke update statistics on authors to billy
```

如果撤消用户执行 update statistics 的权限，也就撤消了他们执行此命令的各种变化形式的能力。

不能单独撤消执行 update statistics 的变化形式 （例如， update index statistics）的权限。也就是说，*不能*发出以下命令：

```
revoke update all statistics from harry
```

不能在列级授予和撤消 delete statistics 权限。请参见第 401 页的“grant”的 “用法”部分。

在集群环境中撤消

如果尝试在本地临时数据库中撤消用户定义的角色的权限，则 revoke 将失败。

标准 符合 ANSI SQL 的级别Transact-SQL 扩展。

权限 对 revoke 函数的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下， revoke 命令通常可由具有下列特权管理特权之一的用户执行，具体视要撤消的特权或权限而定。 对于服务器范围特权，您必须是具有 manage server permissions 特权或 manage security permissions 特权的用户。 对于数据库范围特权，您必须是具有 manage database permissions 特权的用户。 对于对象特权，您必须是对象所有者或具有 manage any object permission 特权的用户。 要执行 revoke default，您必须是数据库所有者或对数据库具有 own database 特权的用户。 有关详细信息，请参见表 1-23 “管理方 （启用细化权限时）”一列。
细化权限已禁用	在启用细化权限的情况下： 命令执行 - 只有系统管理员可以撤消 create database 权限，且只能从 master 数据库撤消。只有系统安全员可以撤消 create trigger 和 create encryption key 权限。 数据库一致性检查 – 只有系统管理员可以运行 revoke dbcc 命令。数据库所有者不能运行 revoke dbcc。 数据库对象访问 – 缺省情况下将数据库对象撤消权限授予对象所有者。对象所有者可撤消其他用户对其自己的数据库的权限。 函数 - 只有系统管理员可以撤消内置函数的权限。 代理和会话授权 – 只有系统安全员可以撤消 set proxy 或 set session authorization 权限，且只能从 master 数据库撤消。 角色 - 只能从 master 数据库撤消角色。只有系统安全员可以撤消用户或角色的 sso_role 或用户定义的角色。只有系统管理员可以撤消用户或角色的 oper_rolesa_role。只有同时拥有 sa_role 和 sso_role 的用户可以撤消包含 sa_role 的角色。 表 - 数据库所有者可以撤消系统表的缺省权限。表所有者和系统安全员可撤消对表或表中一组列的 decrypt 权限。 缺省权限 - 数据库所有者或具有 sa_role 角色的登录用户可以撤消缺省权限。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
47	revoke	revoke	• *角色* – 当前活动角色 • *关键字或选项* – revoke 语句的完整命令文本 • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 grant, setuser, set.

函数 proc_role.

系统过程 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

说明

撤消组、登录名、登录配置文件或角色的角色：

语法

```
revoke role {role_name [, role_list ...]} from
    {grantee [, grantee ...]}
```

参数

角色

是系统角色或用户定义的角色的名称。使用 revoke role 撤消授予角色、登录名或登录配置文件的角色。

role_name

是系统角色或用户定义的角色的名称。撤消被授予者的角色时，您将撤消该被授予者通过角色成员资格所获得的所有权限。

grantee

是您要撤消其角色的系统角色、用户定义角色、登录配置文件的名称或用户登录名。

role_list

是您要撤消的系统或用户定义角色的名称列表，不能是变量。

示例

示例 1 撤消 “specialist_role”的 “doctor_role”：

```
revoke role doctor_role from specialist_role
```

示例 2 撤消 “specialist_role”和 “intern_role”以及用户 Mary 和 Tom 的 “doctor_role”和 “surgeon_role”：

```
revoke role doctor_role, surgeon_role from
specialist_role, intern_role, mary, tom
```

示例 3 具有 manage roles 特权的用户 Smith 撤消 doctor_role 的 nurse_role（最初由 roleAdmin 授予）：

```
revoke role nurse_role from doctor_role
granted by roleAdmin
```

注释　manage roles 特权只有在启用细化权限时才可用。

示例 4 撤消系统操作员所假定的登录配置文件的系统角色 oper_role。

```
revoke role oper_role from lp_operator
```

用法

- 可以在用户已登录的情况下撤消此用户的角色。 Adaptive Server 会在执行访问检查之前验证用户的已激活角色。
- 如果您撤消登录配置文件的角色， Adaptive Server 会撤消分配给该配置文件的所有用户 （包括当前登录到 Adaptive Server 的用户）的角色。

标准 符合 ANSI SQL 的级别Transact-SQL 扩展。

权限 对 revoke role 函数的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage roles 特权的用户。
细化权限已禁用	在启用细化权限的情况下： 角色 - 只能从 master 数据库撤消角色。只有系统安全员可以撤消用户或角色的 sso_role、oper_role 或用户定义的角色。只有系统管理员可以撤消用户或角色的 sa_role。只有同时拥有 sa_role 和 sso_role 的用户可以撤消包含 sa_role 的角色。

审计 sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
85	角色	create role、 drop role、 alter role、 grant role 或 revoke role	• *角色* - 当前活动角色 • *关键字或选项* - revoke role 语句的完整命令文本 • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 grant, setuser, set.

函数 proc_role.

系统过程 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

说明

将用户定义的事务回退到事务内的指定保存点或事务的起始点。

语法

rollback [tran | transaction | work]
[*transaction_name* | *savepoint_name*]

参数

tran | transaction | work

指定您想要回退事务或工作。如果您指定 tran、 transaction 或 work，您也可以指定 *transaction_name* 或 *savepoint_name*。

transaction_name

是指派给最外层的事务的名称。它必须符合标识符的规则。

savepoint_name

是在 save transaction 语句中指派给保存点的名称。此名称必须符合标识符的规则。

示例

回退事务：

```
begin transaction
delete from publishers where pub_id = "9906"
rollback transaction
```

用法

- 不带 *transaction_name* 或 *savepoint_name* 的 rollback transaction 会将用户定义的事务回退到最外层事务的起始处。
- rollback transaction *transaction_name* 会将用户定义的事务回退到指定事务的起始处。尽管可以嵌套事务，但只能回退最外层事务。
- rollback transaction *savepoint_name* 会将用户定义的事务回退到匹配的 save transaction *savepoint_name*。

限制

- 如果当前没有事务处于活动状态，则 commit 或 rollback 语句不起作用。
- rollback 命令必须出现在事务中。输入 commit 后，就不能再回退事务。

回退整个事务

- 不带保存点名称的 rollback 将取消整个事务。事务的所有语句或过程都被撤消。
- 如果 rollback 命令不带 *savepoint_name* 或 *transaction_name* 选项，则事务将回退到此批处理中的第一个 begin transaction。这同样包括那些使用链式事务模式隐式地从 begin transaction 开始的事务。

回退到保存点

要取消部分事务，请使用带 *savepoint_name* 的 rollback。保存点是用户用 save transaction 命令在事务内设立的标记。保存点和 rollback 之间的所有语句或过程都将被撤消。

事务回退到保存点之后，可以使用 commit 继续完成 （执行 rollback 后的任何 SQL 语句），或不带保存点地使用 rollback 从而将其全部撤消。事务中保存点的数目没有限制。

在触发器和存储过程内回退

- 在触发器或存储过程中，不带事务或保存点名称的 rollback 语句使所有语句回退到调用过程或引发触发器的批处理中的第一个显示或隐式的 begin transaction。
- 当触发器包含不带保存点名称的 rollback 命令时，回退中止整个批处理。批处理中回退之后的任何语句都不会执行。
- 远程过程调用 (RPC) 独立于任何包含它的事务来执行。在标准事务（即不使用 Open Client™ DB-Library 两阶段提交的事务）中，由远程服务器通过 RPC 执行的命令不能通过 rollback 回退，并且执行时不依赖 commit。
- 有关使用事务管理语句和 rollback 对存储过程、触发器和批处理的影响的完整信息，请参见 《Transact-SQL 用户指南》。

标准

符合 ANSI SQL 的级别符合初级标准。

Transact-SQL 扩展
rollback transaction 和语句的 rollback tran 形式以及事务名称的使用。

权限

使用 rollback 无需任何权限。

另请参见

命令 begin transaction, commit, create trigger, save transaction.

rollback trigger

说明

回退触发器中所做的工作，包括导致触发器引发的数据修改，并发出可选的 raiserror 语句。

语法

rollback trigger
[with *raiserror_statement*]

参数

with *raiserror_statement*

指定 raiserror 语句，它输出用户定义的错误消息并设置系统标志来记录所发生的错误情况。在执行 rollback trigger 时，该语句能够将错误提交到客户端，这样该错误包含的事务状态就能够反映出回退的情况。有关定义 raiserror_statement 的语法和规则的详细信息，请参见 raiserror 命令。

示例

回退触发器并且发出用户定义的错误消息 25002：

```
rollback trigger with raiserror 25002
    "title_id does not exist in titles table."
```

用法

- 执行 rollback trigger 时， Adaptive Server 将中止当前正在执行的命令并暂停执行触发器的其余部分。
- 如果发出 rollback trigger 的触发器嵌套在其它触发器内，则 Adaptive Server 将回退在这些触发器中完成的所有工作 （最多还包含引起第一个触发器引发的那个更新操作）。
- Adaptive Server 忽略在触发器外执行的 rollback trigger 语句，并且不发出与该语句关联的 raiserror。但在触发器外、事务内执行的 rollback trigger 语句会产生一个错误，该错误导致 Adaptive Server 回退此事务，并中止当前的语句批处理。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 rollback trigger 无需任何权限。

另请参见

命令 create trigger, raiserror, rollback.

save transaction

说明

在事务内设置一个保存点。

语法

save transaction *savepoint_name*

参数

savepoint_name

是指派给保存点的名称。它必须符合标识符的规则。

示例

更新两位作者的 royaltyper 条目后，插入保存点 percentchanged，然后确定书籍涨价 10% 将如何影响作者的版税收入。使用 rollback transaction 将事务回退到保存点：

```
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
```

用法

- 保存点是事务内的允许回退部分事务的用户定义标记。rollback *savepoint_name* 回退到指定的保存点；保存点与 rollback 之间的所有语句或过程都被撤消。

保存点之前的语句不被撤消 – 但也不会提交。回退到保存点之后，事务继续执行语句。不带保存点的 rollback 将取消整个事务。commit 允许事务继续进行直到完成。

- 如果嵌套了事务， save transaction 仅在最外层事务中创建保存点。
- 事务中保存点的数目没有限制。
- 如果 rollback 命令中没有指定 *savepoint_name* 或 *transaction_name*，在批处理中第一个 begin transaction 之前的所有语句将被回退，并且整个事务被取消。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 save transaction 无需任何权限。

另请参见

命令 begin transaction, commit, rollback.

文档 《Transact-SQL 用户指南》中有关使用事务语句的说明。 .

select

说明

从数据库对象中检索行。

语法

```
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 ::=
```

注释 注意：有关 *select_list*的详细信息，请参见 “参数”部分。

```
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 | ANSI_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]

            ANSI_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"

注释 有关 select...for_xml_clause 的语法、示例和用法的信息，请参见《XML 服务》一书。

参数

all

在结果中包括所有行， all 是缺省值。

distinct

只在结果中包括唯一行。 distinct 必须是选择列表的第一个词。在浏览模式下将忽略 distinct。

由于关键字 distinct 的关系，NULL 值被认为是相等的：无论遇到多少 NULL，只选择一个。

top *unsigned_integer*

与 select...into 语句一起使用，可以限制插入到目标表中的行数。这与 set rowcount 不同，在执行 select...into 时，set rowcount 命令会被忽略。

- 当与 delete、 update 一起使用或用在视图中时，不能指定顺序。如果表上有聚簇索引形成的隐含顺序，则会应用该顺序；否则，结果可能以任何顺序排列，排列顺序不可预料。
- *n* 为 0 到 $2^{32}-1$（4GB-1 或 4,294,967,295）之间的无符号 32 位值。零表示没有行。
- 与游标一起使用时， top n 会限制结果集的总体大小。指定 set cursor rowcount 则会限制单次读取的结果。
- 如果视图定义包含 select top *n*，而带有 where clause 子句的查询也使用该命令，则二者的结果可能不一致。

select_list

包含一个或多个下列项目：

- “*”，表示按 create table 顺序排列的所有列。
- 按您希望的查看顺序排列的列名称列表。当选择现有的 IDENTITY 列时，可使用由表名限定的 syb_identity 关键字在必要的地方替代实际的列名。
- 向结果表中添加新 IDENTITY 列的规范：

  column_name = identity (int | smallint | tinyint | precision)

  如果您指定 int、 smallint 或 tinyint，则结果列为整数。如果您指定 precision，则结果为数值数据类型。

- 缺省列标题 （列名）的替代内容，形式如下：

```
column_heading = column_name
column_name column_heading
```

```
column_name as column_heading
```

在所有这些格式中，都可以将列标题用引号引起来。如果标题不是有效的标识符 （即标题是保留字、标题以特殊字符开始或者标题包含空格或标点符号），就必须用引号引起来。

- 表达式 （列名、常量、函数、任何由算术运算符或逐位运算符连接起来的列名、常量和函数的组合，也可以是子查询）。
- 内置函数或集合。
- 上面所列项目的任意组合。

select_list 也可以给变量赋值，形式如下：

```
@variable = expression
    [, @variable = expression ...]
```

不能将变量赋值与任何其它 *select_list* 选项组合。

into

除非与 existing table 一起使用，否则将根据在选择列表中指定的列以及在 where 子句中选择的行来创建新表。请参见第 580 页的 “使用 select into”。

colname encrypt

在目标表中，指定对 *colname* 进行加密。缺省情况下， Adaptive Server 会对从源表中选择的数据进行解密。必须使用 encrypt 关键字才能保留数据加密或者在目标数据库中对源数据库中未加密的列进行加密。

compression = *compression_level* | not compressed

指示行中的大对象 (LOB) 数据是否压缩以及压缩级别。

compression_level | not compressed

指示行的压缩级别：

- 0 – 不压缩行。
- 1 到 9 - Adaptive Server 使用 ZLib 压缩。通常，压缩级别数字越大， Adaptive Server 压缩 LOB 数据的程度就更大，压缩和非压缩数据之间的比率就越大 （也就是说，压缩数据与非压缩数据大小相比，节省的空间量就越大 （以字节为单位））。

  但是，压缩量取决于 LOB 内容，压缩级别越高，进程占用 CPU 越大。也就是说，级别 9 的压缩率最高，但其 CPU 占用率也最高。
- 100 - Adaptive Server 使用 FastLZ 压缩。这是使用最低 CPU 使用率的压缩率；通常用于较小量数据。
- 101 - Adaptive Server 使用 FastLZ 压缩。与值为 100 时相比，值为 101 时的 CPU 占用率略高，但压缩率也略高。

压缩算法忽略不使用 LOB 数据的行。

column_list

是一个由逗号分隔的列列表。

with database...key

指定用于源数据的密钥或指定其它密钥。

in row [(*length*)]

设置或更改目标表中的 LOB 列的行内特性。如果您不指定 *length*，Adaptive Server 会使用所配置的缺省行内长度。

缺省情况下，目标表中的 LOB 列继承 select 列表中的相应 LOB 列的存储属性。如果表达式 （如 convert(text, *column*) 内置函数）生成目标表的 LOB 列，则该列会自动使用行外存储，除非您通过指定 in row [(*length*)] 更改设置。

off row

将列的存储格式从行内更改为行外。

external [[*table*] | *directory* | *file*]

表示外部对象的类型是表、目录或文件。如果您未指明文件、目录或表，则 select into 假定您使用的是表。

注释　在使用 *partition_clause* 的任何部分时，不能指定外部位置。只能在当前服务器和数据库中的表上创建分区。

'*server_name*.[*database*].[*owner*].*object_name*'
表示您将所选择的数据添加到远程 *server_name* 上所找到的表或视图中。

dml_logging
确定 insert、update 和 delete 操作以及某些形式的批量插入的日志记录量。以下值之一：

- full – Adaptive Server 记录所有事务
- minimal - Adaptive Sever 不记录行或页更改

at ′ *path_name*′
表示特定于操作系统的完整路径名称，该路径包含您将所选择的数据添加到的外部文件或目录。 Adaptive Server 必须能访问 *path_name* 中的所有目录。

column delimeter '*string*'
表示将列的数据转换为字符串格式后，用来分隔列的分隔符。 *string* 最多可以有 16 个字符。如果没有指定分隔符， select into 使用制表符。

existing table *table_name*
表示您正在选择数据到代理表中。除代理表外，不能对任何其它表类型使用 select into。 select 列表中的列列表必须与代理表中的列列表在类型、长度和数量上匹配。

on *segment_name*
指定要在已命名段上创建表。在使用 on *segment_name* 选项之前，必须使用 disk init 初始化设备，并使用 sp_addsegment 将段添加到数据库。要获得数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。

partition by range
指定要根据一个或多个分区列中的值对记录分区。将每个分区列值与用户提供的几组上下限相比较，以确定分区分配。

column_name
在 *partition_clause* 中使用时，指定一个分区键列。

partition_name
指定要在其上存储表记录的新分区的名称。分区名称在表或索引上的分区组内必须唯一。如果使用 set quoted_identifier on，则分区名称可以是分隔标识符。否则，分区名称必须是有效的标识符。

如果省略 *partition_name*，则 Adaptive Server 将创建一个格式为 *table_name_partition_id* 的名称。 Adaptive Server 将截断超过允许的最大长度的分区名称。

values <= *constant* | MAX

指定命名分区的上限值（包括边界值）。为最高分区上限指定常量值将在表上施加隐式完整性约束。关键字 MAX 指定给定数据类型的最大值。

on *segment_name*

在 *partition_clause* 中使用时，指定将在其上放置分区的段的名称。使用 on *segment_name* 时，必须已经用 create database 或 alter database 将逻辑设备指派给了数据库，且必须已经用 sp_addsegment 在数据库中创建了段。要获得数据库中可用段名的列表，请咨询系统管理员或使用 sp_helpsegment。

partition by hash

指定要按系统提供的散列函数对记录分区。该函数计算分区键的散列值，这些分区键指定将记录分配到的分区。

partition by list

指定要按命名列中指定的实际值对记录分区。分区键只包含一个列。最多可以列出 250 个常量作为每个列表分区的分区值。

partition by roundrobin

指定要按顺序方式对记录分区。循环分区表没有分区键。用户和优化程序都不知道特定记录位于哪个分区中。

lock datarows | datapages | allpages

指定要用于由 select into 命令创建的表的锁定方案。缺省值是对配置参数 lock scheme 的全服务器范围的设置。

max_rows_per_page

限制由 select into 创建的表的数据页的行数。与 fillfactor 不同，max_rows_per_page 值在插入和删除数据时保持不变。仅数据锁定表不支持 max_rows_per_page。

exp_row_size = *num_bytes*

指定由 select into 命令创建的表的期望行宽。只对数据行、数据页锁定方案和具有可变长度行的表有效。有效值可以是 0、 1 以及介于表的最小和最大行长度之间的任何值。缺省值为 0，表示使用服务器范围的缺省值。

reservepagegap = *num_pages*

指定填充页与空白页的比率，要保留该比率的空白页作为 select into 分配扩充来存储数据。此选项只对 select into 命令有效。对于每个指定的 *num_pages*，都会预留一个空白页供表将来扩展之用。有效值为 0 到 255。缺省值为 0。

readpast

指定查询自动跳过带排它锁的行，不必等待也不生成消息。

identity_gap
指定表的标识间隔。该值只替换此表的系统标识间隔设置。

compression =
表示要用于表或分区的压缩级别。新的压缩级别应用到新插入或更新的数据：

- none – 不压缩该表或分区中的数据。对于分区， none 表示始终不压缩该分区中的数据，即使将表压缩修改为 row 或 page 压缩也是如此。
- row - 压缩单个行中的一个或多个数据项。只有在压缩形式比非压缩形式更节省空间时， Adaptive Server 才会用 row 压缩形式存储数据。在分区级或表级设置 row 压缩。
- page - 当页面已满时，则会使用页级压缩来压缩现有的采用行压缩形式的数据行，从而创建页级字典、索引和字符编码条目。在分区级或表级设置 page 压缩。

  Adaptive Server 只有在已经在行级压缩数据后才会在页级压缩数据，因此，将压缩设置为 page 意味着 page 和 row 压缩。

lob_compression = *compression_level*
决定表的压缩级别。

value
是标识间隔量。

如果用 select into 语句从具有特定标识间隔设置的表创建表，则新表不继承父表的标识间隔设置。相反，新表使用 identity burning set factor 设置。若要为新表指定一个特定的 identity_gap 设置，则应在 select into 语句中指定标识间隔。可以为新表指定一个与父表相同或不同的标识间隔。

from
表示 select 语句中要使用哪些表和视图。 from 是必需的，除非 select 列表不包含列名 （即只包含常量和算术表达式）：

```
select 5 x, 2 y, "the product is", 5*2 Result

x       y                       Result
------- ------- ----------------- -----------
        5       2 the product is   10
```

一个查询最多可以引用 50 个表和 14 个工作表（如由集合函数创建的那些表）。 50 个表的限制包括：

- from 子句中列出的表 （或表上的视图）。
- 对同一个表的多个引用的每个实例 （自连接）
- 在子查询中引用的表
- 用 into 创建的表
- from 子句中列出的视图所引用的基表

view_name, *table_name*

列出 select 语句中使用的表和视图。如果该表或视图位于另一数据库中，请指定该数据库名称；如果在数据库中有多个具有该名称的表或视图，请指定所有者的名称。 *owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

如果列表中有多个表或视图，用逗号分隔它们的名称。表和视图在关键字 from 后的顺序不影响结果。

可在同一语句中查询不同数据库中的表。

为清晰起见，或为区分表和视图在自连接或子查询中充当的不同角色，可给出表名和视图名的相关名 （别名）。若要指派相关名，首先给出表或视图的名称，接着一个空格，然后是相关名，如下所示：

```
select pub_name, title_id
    from publishers pu, titles t
    where t.pub_id = pu.pub_id
```

对该表或视图的所有其它引用 （例如在 where 子句中）必须使用该相关名。相关名不能以数字开头。

index *index_name*

指定访问 *table_name* 要使用的索引。从视图中选择时不能使用此选项，但可将它用作 create view 语句中 select 子句的一部分。

parallel

如果 Adaptive Server 配置为允许并行处理，则指定并行分区或索引扫描。

degree_of_parallelism

指定将以并行方式扫描表或索引的工作进程数。如果设置为 1，查询将以串行方式执行。

prefetch *size*

为绑定到配置了大 I/O 的高速缓存的表指定 I/O 大小 （单位为千字节）。当从视图中选择时，不能使用这一选项，但可以作为 create view 语句的 select 子句的一部分使用。 sp_helpcache 显示对象绑定到的高速缓存或缺省高速缓存的有效大小。若要配置数据高速缓存大小，请使用 sp_cacheconfigure。

在使用 prefetch 并指定预取大小 (*size*) 时，最小值是 2K 或逻辑页大小与 2 的任意次方的乘积 （最大为 16K）。 prefetch 大小选项如下 （单位为千字节）：

逻辑页大小	预取大小选项
2	2, 4, 8 16
4	4, 8, 16, 32
8	8, 16, 32, 64
16	16, 32, 64, 128

在查询中指定的 prefetch 大小仅仅是一个建议。要使用指定的大小，应将数据高速缓存按该大小进行配置。如果未将数据高速缓存配置为特定的大小，则使用缺省的 prefetch 大小。

如果启用了 CIS，则不能对远程服务器使用 prefetch。

lru | mru

指定对表使用的缓冲区替换策略。使用 lru 强制优化程序将表读入 MRU/LRU （最近使用最多的/最近使用最少的）链上的高速缓存。使用 mru 可放弃高速缓存中的缓冲区并将其替换为该表的下一个缓冲区。从视图中选择时不能使用此选项，但可将它用作 create view 语句中 select 子句的一部分。

holdlock

通过持有共享锁直到事务完成 （而不是无论事务是否完成，一旦请求的数据页不再需要，就释放共享锁）来加强对指定表或视图的共享锁的限制。

holdlock 选项仅适用于指定了该选项的表或视图，且仅在使用该选项的语句所定义的事务的持续期间适用。设置 set 命令的 transaction isolation level 3 选项隐式地为事务中的每个 select 语句应用 holdlock。在包括 for browse 选项的 select 语句中，不允许使用关键字 holdlock。在查询中不能同时指定 holdlock 和 noholdlock 选项。

如果启用了 CIS，则不能对远程服务器使用 holdlock。

noholdlock

禁止服务器持有执行此 select 语句期间获得的任何锁，无论当前处于何种事务隔离级别。在查询中不能同时指定 holdlock 和 noholdlock 选项。

shared

指示 Adaptive Server 在指定的表或视图上使用共享锁 （而非更新锁）。这允许其它客户端获得表或视图的更新锁。 shared 关键字只能与 declare cursor 语句中包含的 select 子句一起使用。例如：

```
declare shared_crsr cursor
for select title, title_id
from titles shared
where title_id like "BU%"
```

在每一表或视图名称后，可联合 shared 使用关键字 holdlock。但 holdlock 必须在 shared 之前。

ANSI join

使用 ANSI 语法的内部或外部连接。 from 子句指定要连接的表。

inner

只包含满足 on 子句条件的内部和外部表的行。对于不满足 on 子句条件的外部表中的行，包含内部连接的查询所得到的结果集中不包含任何空值行。

outer

包含外部表的所有行，无论是否满足 on 子句条件。如果某行不满足 on 子句条件，则内部表的值将作为空值存储在连接表中。 ANSI 外连接中的 where 子句将限制查询结果中的行。

left

左连接将保留在连接子句左侧列出的表引用的所有行。左表引用称为外部表或行保留表。

在下面的查询中， T1 是外部表， T2 是内部表：

```
T1 left join T2
T2 right join T1
```

right

右连接将保留在连接子句右侧列出的表引用的所有行 （请参见上例）。

search_conditions

用于为检索的行设置条件。搜索条件可以包括列名、表达式、算术运算符、比较运算符、关键字 not、like、is null、and、or、between、in、exists、 any 和 all、子查询、条件表达式或这些项目的任意组合。有关详细信息，请参见第 697 页的 “where 子句”。

group by
查找每个组的值。这些值在结果中作为新列、而不是新行出现。

当 group by 与标准 SQL 一起使用时，选择列表中的每一项必须在组中的每一行有固定值或与集合函数 （为每个组生成一个值）一起使用。Transact-SQL 对选择列表中的项目没有这种限制。此外，Transact-SQL 还允许按任何表达式 （列别名除外）分组；对于标准 SQL，则只能按列来分组。

可以将表 1-28 中列出的集合 （*expression* 几乎总是一个列名）与 group by 一起使用：

表 1-28: 将集合与 group by 一起使用的结果

集合函数	结果
sum([all | distinct] *expression*)	数值列中的值的总和。
avg([all | distinct] *expression*)	数值列中的值的平均值。
count([all | distinct] *expression*)	以 integer 形式返回的列中 （不同）非空值的数量。
count_big ([all | distinct] *expression*)	以 bigint 形式返回的列中不同非空值的数量。
count(*)	以 integer 形式返回的选定行数。
count_big(*)	以 bigint 形式返回的选定行数。
max (*expression*)	列中的最大值。
min (*expression*)	列中的最小值。

有关详细信息，请参见第 434 页的 “group by 和 having 子句”。

可以按任何列组合来对表分组 – 即组可以互相嵌套。不能按列标题分组；必须使用列名、表达式或代表项目在选择列表中的位置的编号。

group by all
在结果中包括所有组，即使是那些没有任何行满足搜索条件的组也包括在内。有关示例，请参见第 434 页的 “group by 和 having 子句”。

aggregate_free_expression
是不包含集合的表达式。

having

设置 group by 子句的条件，方式与用 where 为 select 子句设置条件的方式相似。可以包含的条件的数目没有限制。

可以使用不带 group by 子句的 having 子句。

如果选择列表中的任何列没有应用集合函数且不包括在查询的 group by 子句中（在标准 SQL 中为非法），则 having 和 where 的意义稍有不同。

在这种情况下， where 子句限制集合计算中包括的行，但不限制查询返回的行。相反， having 子句限制查询返回的行，但不影响集合的计算。有关示例，请参见第 434 页的 “group by 和 having 子句”。

order by

按列对结果进行排序。在 Transact-SQL 中，可以对选择列表中没有的项目使用 order by。可以按列名、列标题 （或别名）、表达式或代表项目在**选择列表**中位置的编号 (*select_list_number*) 来排序。如果按选择列表编号排序，则 order by 子句引用的列必须包括在选择列表中，且选择列表不能是 * （星号）。

使用带 order by 的 select max 可以在结果集中返回多行。

asc

按升序排序结果 （缺省值）。

desc

按降序排序结果。

compute

与行集合 （sum、 avg、 min、 max、 count 和 count_big）一起使用以生成控制中断摘要值。摘要值在查询结果中作为附加行出现，从而允许用一个语句查看明细行和摘要行。

select into 子句不能与 compute 一起使用。

如果使用 compute by，也必须同时使用 order by 子句。在 compute by 后面列出的列必须与 order by 后面列出的列相同或是其子集，它们从左向右的顺序必须一致，以同一表达式开始且不跳过任何表达式。

例如，如果 order by 子句为 `order by a, b, c`，则 compute by 子句可以是以下任意 （或所有）形式：

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

```
compute by a
```

不带 by 的 compute 关键字可用于生成总和、总计数等。如果使用不带 by 的 compute，order by 就是可选的。有关详细信息和示例，请参见第 88 页的 “compute clause”。

如果启用了 CIS， compute 不会被转发给远程服务器。

for {read only | update}

指定游标结果集是只读的或可更新的。

对于低于 15.7 的 Adaptive Server 版本，只能在存储过程中使用此选项，而且只有在过程定义游标查询时才可以。在这种情况下， select 是唯一被允许在过程中使用的语句。它定义 for read only 或 for update 选项 （代替 declare cursor 语句）。这种声明游标的方法提供了在读取行时进行页级锁定这一好处。

另外，对于低于 15.7 的 Adaptive Server 版本，如果存储过程中的 select 语句不用于定义游标， Adaptive Server 会忽略 for read only | update 选项。有关使用存储过程声明游标的详细信息，请参见嵌入式 SQL™ 文档。

对于 Adaptive Server 15.7 和更高版本，如果设置了 select for update，您就可以在存储过程之外在语言级使用 for update 选项，但必须在事务之内。这种 select 不需要引用游标。当您在数据行锁定表中使用 select for update 时，选定的行会在事务持续时间内被以排它方式锁定。

有关只读或可更新游标的信息，请参见 《Transact-SQL 用户指南》。

of *column_name_list*

是来自被定义为可更新的游标结果集 （用 for update 选项定义）的列的列表。

at isolation

指定查询的隔离级别 （0、 1、 2 或 3）。如果忽略该子句，查询将使用执行它的会话的隔离级别（缺省隔离级别为 1）。at isolation 子句仅对单个查询或在 declare cursor 语句内有效。如果使用 at isolation，则 Adaptive Server 返回语法错误：

- 用于使用 into 子句的查询
- 在子查询内
- 用于 create view 语句中的查询
- 用于 insert 语句中的查询
- 使用 for browse 子句的查询

如果查询中有 union 运算符，则必须在最后一个 select 后指定 at isolation 子句。如果在指定了 at isolation read uncommitted 的查询中同时指定 holdlock、noholdlock 或 shared，则 Adaptive Server 会发出警告并忽略 at isolation 子句。对于其它隔离级别， holdlock 优先于 at isolation 子句。有关隔离级别的详细信息，请参见 《Transact-SQL 用户指南》。

如果启用了 CIS，则不能对远程服务器使用 at isolation。

read uncommitted | 0

将查询的隔离级别指定为 0。

read committed | 1

将查询的隔离级别指定为 1。

repeatable read | 2

将查询的事务隔离级别指定为 2。

serializable | 3

将查询的隔离级别指定为 3。

for browse

必须附加到在 DB-Library 浏览应用程序中被发送到 Adaptive Server 的 SQL 语句的结尾。有关详细信息，请参见 Open Client DB-Library Reference Manual （《Open Client DB-Library 参考手册》）。

plan "*abstract plan*"

指定用来优化查询的抽象计划。它可以是用抽象计划语言指定的完整或部分计划。有关详细信息，请参见 《性能和调优指南》中的 “创建和使用抽象计划”。

示例

示例 1 从 publishers 表中选择所有行和列：

```
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
```

示例 2 从 publishers 表的特定列中选择所有行：

```
select pub_id, pub_name, city, state from publishers
```

示例 3 从 publishers 表的特定列中选择所有行，替换一个列名并向输出中添加字符串：

```
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
```

示例 4 从 titles 表的特定列中选择所有行，并替换列名：

```
select type as Type, price as Price
from titles
```

示例 5 为 select into 指定锁定方案和保留页间距：

```
select title_id, title, price
into bus_titles
lock datarows with reservepagegap = 10
from titles
where type = "business"
```

示例 6 选择数据添加到 bigspenders 表时对 creditcard 列进行加密：

```
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
```

示例 7 只选择未以排它方式锁定的行。如果任何其他用户在某一限定行上有排它锁，则不返回该行：

```
select title, price
from titles readpast
    where type = "news"
    and price between $20 and $30
```

示例 8 选择特定列和行，将结果放入临时表 #advance_rpt 中：

```
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
```

示例 9 从 authors 表中选择 au_lname 的前 3 行：

```
select top 3 au_lname from authors
```

示例 10 并置两列并将结果放入临时表 #tempnames 中：

```
select "Author_name" = au_fname + " " + au_lname
    into #tempnames
    from authors
```

示例 11 选择特定列和行，返回按类型由高到低排序的结果，并计算摘要信息：

```
select type, price, advance from titles
order by type desc
compute avg (price), sum (advance) by type
compute sum(price), sum(advance)
```

示例 12 选择特定列和行，并计算 price 和 advance 列的总和：

```
select type, price, advance from titles compute sum (price), sum (advance)
```

示例 13 创建 coffeetabletitles 表，该表是 titles 表的副本，其中只包括价格高于 $20 的书：

```
select * into coffeetabletitles from titles
where price > $20
```

示例 14 创建 newtitles 表，该表是 titles 表的空副本：

```
select * into newtitles from titles
where 1 = 0
```

示例 15 给出优化提示：

```
select title_id, title
    from titles (index title_id_ind prefetch 16)
    where title_id like "BU%"
```

示例 16 使用 syb_identity 关键字从 sales_east 和 sales_west 表中选择 IDENTITY 列：

```
select sales_east.syb_identity,
```

```
sales_west.syb_identity
from sales_east, sales_west
```

示例 17 创建 newtitles 表，该表是具有 IDENTITY 列的 titles 表的副本：

```
select *, row_id = identity (10)
into newtitles from titles
```

示例 18 指定查询的事务隔离级别。

```
select pub_id, pub_name
from publishers
at isolation read uncommitted
```

示例 19 使用 repeatable read 隔离级别从 titles 中选择。在事务完成之前，其他用户将不能更改受影响行中的值或删除受影响的行：

```
begin tran
select type, avg(price)
    from titles
    group by type
at isolation repeatable read
```

示例 20 给出查询并行度的优化提示：

```
select ord_num from salesdetail
     (index salesdetail parallel 3)
```

示例 21 在 title_id 列上连接 titleauthor 和 titles 表。结果集中仅包含 price 大于 15 的行：

```
select au_id, titles.title_id, title, price
from titleauthor inner join titles
on titleauthor.title_id = titles.title_id
and price > 15
```

示例 22 结果集包括 authors 表中的所有作者。当作者与他们的出版者不居住同一城市时， pub_name 列中的值就为空。只有 Cheryl Carson 和 Abraham Bennet 与他们的出版者位于同一城市，这两位作者在 pub_name 列中产生非空值：

```
select au_fname, au_lname, pub_name
from authors left join publishers
on authors.city = publishers.city
```

示例 23 从现有表 (newtable) 创建带标识间隔的新表 (oldtable)，可在 select into 语句中指定：

```
select identity into newtable
with identity_gap = 20
from oldtable
```

有关标识间隔的详细信息，请参见 《Transact-SQL 用户指南》的 “创建数据库和表”中的 “管理表中的标识间隔”。

示例 24 创建一个具有行级压缩的表 bay_area_authors，并用有关居住在旧金山海湾地区的作者的信息来填充该表：

```
select * into bay_area_authors
with compression = row
from authors
where postalcode like ′ 94%′
```

示例 25 创建一个名为 titles_2 的新表，该表压缩除 title 和 advance 外的所有列：

```
select * into titles_2
(title not compressed,
advance not compressed)
with compression = page
from titles
```

示例 26 设置 select for update 配置参数，执行 select for update，然后在同一事务内执行 update：

```
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 basetbl1 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 basetbl1 set c_bigint = 5000 where c_int > 90
(10 rows affected)
commit tran
go
```

示例 27 从现有表 sales_detail 创建一个新表 sales_report。新表按 qty 列的范围分区。

```
select * into sales_report partition by range (qty)
(smallorder values <= (500) on seg1,
bigorder values <= (5000) on seg2)
from sales_detail
```

示例 28 使用此查询查找产生过多 IO 的语句作为调优候选项。

```
select lio_avg, qtext from sysquerymetrics order by
lio_avg
```

示例 29 选择 titles 表添加到 pubs3 数据库：

```
select title_id, title, price
into bus_titles
with dml_logging = minimal
from titles
```

用法

- select 语句中的关键字 （与在所有其它语句中一样）必须按语法语句中显示的顺序使用。
- select 语句中表达式的最大数量是 4096。
- 可在 select 后使用关键字 all 以与其它 SQL 实现兼容。all 是缺省值。在此环境中使用时， all 与 distinct 正好相反。结果中包含检索到的所有行，不管是否有重复。
- 除了在 create table、 create view 和 select into 语句中外，如果列标题用引号引起来，则可以包含任何字符，包括空格和 Adaptive Server 关键字。如果列标题没有用引号引起，则它必须符合标识符规则。
- like 表示的字符串不能超过 255 字节。
- 对列数超过 255 的表不能使用 select...for browse 选项。
- create table、create view 和 select into 语句中的列标题以及表的别名都必须遵循标识符的规则。

- 若要使用 select 将数据从某些字段为空值的表中插入不允许空值的表中，必须为初始表中的所有 NULL 条目提供替代值。例如，为了将数据插入不允许空值的 advances 表中，以下示例用 “0”替代 NULL 字段：

  ```
  insert advances
  select pub_id, isnull (advance, 0) from titles
  ```

  如果不使用 isnull 函数，此命令会将带有非空值的所有行插入 advances 表，并对在 titles 表的 advance 列中包含 NULL 的所有行生成错误消息。

  如果无法对数据进行这种替代，则不能将包含空值的数据插入具有 NOT NULL 规定的列。

  两个表可以在结构上相同，但在某些字段是否允许有空值这一点上是不同的。使用 sp_help 可查看表中的列的空类型。

- 使用 select 语句返回的 text、unitext 或 image 数据的缺省长度是 32K。可使用 set textsize 更改此值。当前会话的大小存储在全局变量 *@@textsize* 中。在登录到 Adaptive Server 时，某些客户端软件可能发出 set textsize 命令。

- 远程 Adaptive Server 中的数据可通过使用远程过程调用来检索。有关详细信息，请参见 create procedure 和 execute。

- 游标定义中使用的 select 语句（通过 declare cursor）必须包含 from 子句，但不能包含 compute、for browse 或 into 子句。如果 select 语句包含以下任何结构，则游标被视为是只读且不可更新的：

  - distinct 选项
  - group by 子句
  - 集合函数
  - union 运算符

  如果使用包含 order by 子句的 select 语句在存储过程中声明游标，则该游标也被认为是只读的。即使它被认为是可更新的，也不能用包含两个或多个表的连接的 select 语句定义的游标删除行。有关详细信息，请参见 declare cursor。

- 如果为变量赋值的 select 语句返回多个行，则最后返回的值被赋给此变量。例如：

  ```
  declare @x varchar (40)
  select @x = pub_name from publishers
  print @x
   (3 rows affected)
  ```

```
Algodata Infosystems
```

使用 ANSI 连接语法

- 使用 ANSI 内部和外部连接语法编写查询之前，请阅读《Transact-SQL 用户指南》中的 “连接：从几个表中检索数据”中的 “外部连接”。

使用 *select into*

- select into 是两步式操作。第一步创建新表，第二步将指定行插入新表。

  注释 可以对 CIS 现有表执行 select into。

  因为用 select into 操作插入的行不被记录，所以不能在用户定义的事务中发出 select into 命令，即使 ddl in tran 数据库选项设置为 true 亦如此。select into 操作期间的页分配会被记录，因此大的 select into 操作可能会填满事务日志。

  如果 select into 语句在创建新表后失败， Adaptive Server 将*不会*自动删除此表或释放此表的第一个数据页。这意味着，发生错误之前在第一页插入的所有行都将保留在该页上。检查 select into 语句后的全局变量 *@@error* 的值，以确保没有发生错误。使用 drop table 语句删除新表，然后重新发出 select into 语句。

- 新表的名称在数据库中必须是唯一的，并且必须符合标识符规则。也可以 select into 临时表 （参见示例 7、 8 和 11）。

- 与基表相关联的任何规则、约束或缺省值不会转到新表中。可使用 sp_bindrule 和 sp_bindefault 把规则或缺省值绑定到新表。

- select into 不结转基表的 max_rows_per_page 值，并且使用值为 0 的 max_rows_per_page 创建新表。可使用 sp_chgattribute 设置 max_rows_per_page 的值。

- 为了 select into （插入）一个永久表， select into/bulkcopy/pllsort 选项必须设置为 true （通过执行 sp_dboption）。无须为了 select into （插入）一个临时表而将 select into/bulkcopy/pllsort 选项设置为 true，因为从不恢复临时数据库。

  在数据库中使用 select into 之后，必须先执行完全数据库转储才能使用 dump transaction 命令。select into 操作只记录页的分配而不记录数据行的更改。因此，不能从事务日志恢复更改。在这种情况下，发出 dump transaction 语句会产生错误消息，指示您改用 dump database。

缺省情况下，在新创建的数据库中，select into/bulkcopy/pllsort 选项设置为 false。若要更改缺省情况，请在 model 数据库中将此选项设置为 true。

- select into 可用于存档数据库。
- 当 dump database 进行时， select into 的运行速度更慢。
- 可以通过在 where 子句中使用“假”条件来用 select into 创建没有数据的复制表 （请参见示例 12）。
- 必须为选择列表中包含集合函数或任何表达式的列提供列标题。选择列表中任何常数、算术或字符表达式、内置函数的使用或并置都需要被影响的项目的列标题。列标题必须是有效的标识符或必须用引号引起来 （请参见示例 7 和 8）。
- 当使用 select into 时，数据类型和可为空性被隐式地赋给实际值，例如：

  ```
  select x = getdate () into mytable
  ```

  这产生一个不可为空的列，不论 allow nulls by default 打开还是关闭均如此。它取决于如何使用 select 命令以及在语法中与其它哪些命令一起使用。

  convert 语法允许显式指定结果列的数据类型和可为空性，而不用缺省值。

  将 getdate 包含在确实得到空值的函数中，例如：

  ```
  select x = nullif (getdate (), "1/1/1900") into
  mytable
  ```

  或者，使用 convert 语法：

  ```
  select x = convert (datetime null, getdate ()) into
  mytable
  ```

- 不能在用户定义的事务或具有 compute 子句的语句中使用 select into。
- 要选择一个 IDENTITY 列并添加到结果表中，应在 select 语句的 *column_list* 中包括该列名 （或 syb_identity 关键字）。新列遵守下列规则：
  - 如果多次选择某一 IDENTITY 列，它将在新表中定义为 NOT NULL。该 IDENTITY 列将不继承 IDENTITY 属性。
  - 如果选择 IDENTITY 列作为表达式的一部分，结果列不会继承 IDENTITY 属性。如果表达式中的任何列允许使用空值，则将它创建为 NULL；否则，将它创建为 NOT NULL。

  - 如果 select 语句包含 group by 子句或集合函数，结果列不会继承 IDENTITY 属性。包含 IDENTITY 列的集合的列将被创建为 NULL，其它列则为 NOT NULL。
  - 通过联合或连接选入表中的 IDENTITY 列不保留 IDENTITY 属性。如果表中包含 IDENTITY 列和 NULL 列的联合，新列将定义为 NULL。否则，定义为 NOT NULL。
- 不能使用 select into 创建带有多个 IDENTITY 列的新表。如果 select 语句包括现有 IDENTITY 列和格式为 *column_name* = identity *(precision)* 的新 IDENTITY 规范，则语句会失败。
- 如果启用 CIS，而且 into 表驻留在 Adaptive Server 上，则 Adaptive Server 使用批量复制例程将数据复制到新表中。在对远程表使用 select into 前，请将 select into/bulkcopy 数据库选项设置为 true。
- 有关嵌入式 SQL 命令 select into *host_var_list* 的信息，请参见 *Open Client 嵌入式 SQL 参考手册*。

使用 *select...into* 转换目标列的 NULL 属性

- 使用 convert 命令更改要将所选择的数据添加到的目标列的可为空性。例如，下面的命令可从 titles 表中选择数据，添加到名为 temp_titles 的目标表中，但会将 total_sales 列由 null 转换为 not null：

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

指定压缩级别

- 您用 select into 创建的目标表不继承源表中的任何配置。也就是说，如果从中拉出数据的表配置了行级压缩，则从 select into 生成的表并不配置压缩，除非您显式声明了压缩级别。
- 您可以为目标表指示列级、分区级和表级压缩 （分区的压缩级别会覆盖表的压缩级别）。
- 当您压缩目标表时， Adaptive Server 会选择列的压缩级别 （如果合格），无论源表中的对应列的压缩级别如何都是如此。
- select into 命令可以包括在目标表中不压缩的列的列表。
- 不能在目标表上加密压缩列。

参数与数据压缩的交互

- max_rows_per_page 只适用于所有页锁定压缩表；不能将其用在仅数据锁定表中。

- 可以对具有可变长度列的所有页锁定表和仅数据锁定表使用 exp_row_size。但是，该表必须能够在同一页上扩展更新而不导致所有页锁定表上发生页面拆分或仅数据锁定表上发生行转移。

  exp_row_size 采用相同方式计算压缩表和非压缩表的空间限制。但是，因为压缩可生成具有大量空间的页，所以，为 exp_row_size 留出的空间可能会导致只有极少行不超出页面范围 （与不设置 exp_row_size 相比）。

  不能对具有固定长度列的表使用 exp_row_size，即使 Adaptive Server 可能会在压缩过程中将某些固定长度列转换为可变长度列。

- fillfactor 用于确定压缩后在数据页上使用的空间。这仅在创建聚簇索引 （它是需要对数据页进行排序的操作）时才有意义。

使用 *select for update*

- 对于 Adaptive Server 15.7 和更高版本，如果配置参数 select for update 设置为 1，select for update 选择的行就会被以排它方式锁定，前提条件是该命令是在事务上下文内在数据行锁定表上执行的，或者在链式模式中进行的。如果 select for update 在游标上下文内运行，游标 open 和 fetch 语句就必须位于事务上下文内。
- 通过 select for update 选择的行 （在游标上下文之内或之外）会保留排它锁，直到事务完成为止。
- select for update 的限制：
  - select for update 在子查询块中无效。
  - select for update 仅在 select 直接从基表中返回，而不是从工作表中返回时才适用。 select for update 不能和具有集合或 group by、computed、 union、 having 或 distinct 子句的查询一起使用。
  - 比起 select for update，有资格进行实际事务 update 的行可能更多。这些行可能出现在更新集内。可使用隔离级别 3 来防止这种 “幻像” 行。
  - 在 select 处理期间，并发 select for update 任务可能会尝试以不同顺序锁定一组同样的行，从而导致应用程序死锁。但是，一旦 select for update 完成，对这组行的后续更新不会被阻止，而且不会遇到死锁。
  - 所有现有的针对可更新游标的限制都适用于游标上下文之内或之外的 select for update。唯一不同之处是，对于 select for update， order by 子句通过可更新游标得到支持。对可更新游标的限制适用于语言和执行游标。
  - select for update 引用的所有表必须来自同一个数据库。

- select for update 仅在其数据行锁定的表上得到支持。

指定行内 LOB 列

- 缺省情况下，目标表中的 LOB 列继承 select 列表中的相应 LOB 列的存储属性。如果表达式 （如 convert(text, *column*) 内置函数）生成目标表的 LOB 列，则该列会自动使用行外存储。

使用 *select...into* 指定锁定方案

- 将 lock 选项和 select...into 一起使用时，可以为通过该命令创建的表指定锁定方案。如果不指定锁定方案，则应用缺省锁定方案 （由配置参数 lock scheme 设置）。
- 使用 lock 选项时，也可以指定空间管理属性 max_rows_per_page、exp_row_size 和 reservepagegap。

  可以使用 sp_chgattribute 更改用 select into 创建的表的空间管理属性。

使用 *select...into* 指定分区策略

- partitions_clause 与 select...into 一起使用时，可以为通过该命令创建的表指定分区属性。（有关详细信息，请参见 create table。）如果未指定分区类型， Adaptive Server 将创建未分区表。如果要插入的任何行不满足目标表中任何分区的条件， select...into 将失败。

使用 *index*、 *prefetch* 和 *lru* | *mru*

- index、 prefetch 和 lru | mru 选项指定用于查询执行的索引、高速缓存以及 I/O 策略。这些选项会覆盖 Adaptive Server 优化程序作出的选择。请慎用这些语句，并始终用 set statistics io on 检查性能影响。有关使用这些选项的详细信息，请参见 《性能和调优指南》。

使用加密列

- 如果使用 encrypt 子句时未指定密钥名称，则 Adaptive Server 将使用数据库缺省密钥对目标列中的数据进行加密。
- 如果源表中的列已经加密，但是未对目标列指定 encrypt 子句，则 Adaptive Server 将对源表中的数据进行解密，并在目标列中插入纯文本数据。
- 如果指定对目标列使用源列数据所用的密钥进行加密，并且如果该密钥未使用初始化矢量或随机填充，则 Adaptive Server 会将源列中的数据以密文形式复制到目标列，而无需执行中间的解密和重新加密操作。
- 然而，如果指定对目标列使用与源列所用的密钥不同的密钥进行加密，或者该密钥在加密过程中使用了初始化矢量或填充，则 Adaptive Server 将对加密列的每个所选行执行解密和加密操作。

使用 *parallel*

- parallel 选项将减少 Adaptive Server 优化程序可用于并行处理的工作线程的数目。 *degree_of_parallelism* 不能大于配置的 max parallel degree。如果指定一个大于配置的 max parallel degree 的值，则优化程序会忽略 parallel 选项。
- 当多个工作进程合并其结果时， Adaptive Server 返回的行的顺序可能因执行顺序而异。若要以一致的顺序从分区表中获取行，请使用 order by 子句，或通过在查询的 from 子句中使用 parallel 1 来替换并行查询执行。
- 出现下列任何情况时，将忽略指定 parallel 的 from 子句：
  - 为更新或插入使用了 select 语句。
  - 在游标定义中使用了 from 子句。
  - parallel 用于子查询的所有内部查询块中的 from 子句。
  - select 语句创建了一个视图。
  - 该表是外部连接的内部表。
  - 查询指定了表的 min 或 max 并指定了索引。
  - 指定了未分区的聚簇索引，或者该索引是唯一的 parallel 选项。
  - 查询指定表的 exists。
  - 配置参数 max scan parallel degree 的值为 1，并且查询指定了索引。
  - 包括了非聚簇索引。有关索引覆盖的信息，请参见 《性能和调优指南》中的 “索引”。
  - 此表为系统表或虚拟表。
  - 用 OR 策略处理了查询。有关 OR 策略的解释，请参见 《性能和调优指南》。
  - 查询为用户返回了大量的行。

使用 *readpast*

- readpast 选项允许 select 命令访问指定的表而不会被其它任务持有的不兼容锁阻塞。 readpast 查询只能在仅数据锁定表上执行。
- 如果对所有页锁定表指定了 readpast 选项，此 readpast 选项将被忽略。此命令在为此命令或会话指定的隔离级别下执行。如果隔离级别是 0，则执行脏读，命令从锁定行返回值并且不会阻塞。如果隔离级别为 1 或 3，则在必须读取具有不兼容锁的页时，命令将阻塞。

- 会话级隔离级别与 select 命令中的表的 readpast 的交互作用如表 1-29 所示。

表 1-29: 会话级隔离级别的影响和 readpast

会话隔离级别	效果
0， read uncommitted（脏读）	将忽略 readpast，向用户返回包含未提交事务的行。输出警告消息。
1， read committed	将跳过带有不兼容锁的行或页；在被读取的行或页上不持有锁。 使用 readpast 可能会产生重复，且增加 distinct 子句亦无法清除此问题。 若要解决这一问题，在使用 readpast 命令时，*除了* distinct 子句以外还应使用 group by 子句以避免重复。
2， repeatable read	跳过持有不兼容锁的行或页；所读取的所有行或页上都持有共享锁，直到语句或事务结束为止；语句所读取的所有页上都持有锁，直到事务完成为止。
3， serializable	将忽略 readpast，命令在级别 3 执行。命令将在任何带有不兼容锁的行或页上阻塞。

- 如果指定 readpast 的 select 命令还包括以下任何内容，这些命令将失败并显示错误消息：
  - at isolation 子句，指定 0 或 read uncommitted
  - at isolation 子句，指定 3 或 serializable
  - 同一表上的 holdlock 关键字
- 如果在指定 readpast 的 select 查询中指定 at isolation 2 或 at isolation repeatable read，则 readpast 表会持有共享锁，直至语句或事务完成为止。
- 如果带有 readpast 选项的 select 命令遇到带有不兼容锁的文本列，则 readpast 锁定将检索行，但返回值为 null 的文本列。在这种情况下，由于列被锁定，因此包含空值的文本列和返回的空值之间没有区别。

扩展的 *select* * 语法

当存储过程或触发器的源文本存储在系统表 syscomments 中时，使用 select * 的查询将存储在 syscomments 中，同时扩展在 select * 中引用的列列表。

例如，包含列 col1 和 col2 的表中的 select * 将存储为：

```
select <table>.col1, <table>.col2 from <table>
```

在 Adaptive Server 版本 12.5.4 中，增强了列-列表扩展功能，以检查标识符 （表名、列名等）是否符合标识符规则。

例如，如果一个表包括列 col1 和 2col，第二个列名以数字开头，该列名只能用括号括起来才能用在 create table 语句中。

当在存储过程或触发器中对此表执行 select * 时， syscomments 中的文本与以下内容类似：

```
select <table>.col1, <table>[2col] from <table>
```

对于用于扩展 select * 的文本中的所有标识符，当标识符不符合标识符规则时，将会添加括号。

必须用括号括起标识符，以确保 Adaptive Server 在升级至更新版本时可以使用该 SQL 文本。

标准

符合 ANSI SQL 的级别符合初级标准。

Transact-SQL 扩展包括：

- 创建新表的 select into
- lock 子句
- compute 子句
- 全局变量与局部变量
- index 子句、 prefetch、 parallel 和 lru | mru
- holdlock、 noholdlock 和 shared 关键字
- "*column_heading* = *column_name*"
- 限定表名和列名
- for browse 子句中的 select
- 在选择列表中使用不在 group by 列表中并且也未使用集合函数的列
- at isolation repeatable read | 2 选项

权限

对 select 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表或视图的所有者。 您必须是对表或视图具有 select 权限的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是表或视图的所有者或拥有 sa_role 的用户。 您必须是对表或视图具有 select 权限的用户。 select 权限缺省情况下将授予表或视图的所有者，获得授权的所有者可将此权限移交给其他用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
62	select	从表中选择数据的 select 命令	• *角色* – 当前活动角色 • *关键字或选项* – select、 select into 或 readtext • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名
63	select	从视图中选择数据的 select 命令	• *角色* – 当前活动角色 • *关键字或选项* – select、 select into 或 readtext • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 compute clause、 create index、 create trigger、 delete、 group by 和 having 子句、insert、order by 子句、set、union 运算符、update、where 子句.

函数 avg、 count、 isnull、 max、 min、 sum.

系统过程 sp_cachestrategy, sp_chgattribute, sp_dboption.

set

说明

在用户的工作会话期间设置 Adaptive Server 查询处理选项；在触发器或存储过程内设置某些选项。

语法

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}

参数

set advanced_aggregation

在会话级别启用和禁用高级集合。

set @*variable* = *expression*

允许在一个语句中对多个变量赋值。set @*variable* = *expression* 命令与 Transact-SQL 中的 select @*variable* = *expression* 相同 （其替代方法）。

expression 可以是常量、函数、常量的任意组合以及由算术运算符或逐位运算符连接的函数，也可以是子查询。

set ansinull {on | off}

影响集合行为和比较行为。有关集合行为和比较行为的详细信息，请参见第 628 页的 “集合行为”。

set ansi_permissions {on | off}

决定是否检查 delete 和 update 语句的 ANSI SQL 权限要求。缺省值是 off。表 1-30 总结了权限要求：

表 1-30: update 和 delete 所需的权限

命令	set ansi_permissions 的权限要求：	
	Off	**On**
update	• 针对要设置值的列的 update 权限	• 针对要设置值的列的 update 权限 • 针对出现在 where 子句中的所有列的 select 权限 • 针对set 子句右侧的所有列的 select 权限
delete	• 表的 delete 权限	• 表的 delete 权限 • 针对出现在 where 子句中的所有列的 select 权限

set arithabort [arith_overflow | numeric_truncation] {on | off}

决定出现算术错误时， Adaptive Server 如何工作。两个 arithabort 选项（arithabort arith_overflow 和 arithabort numeric_truncation）可处理不同类型的算术错误。您可以单独地设置每一个选项，或者用单个的 set arithabort on 或 set arithabort off 语句来设置这两个选项。

- arithabort arith_overflow – 指定在出现被零除错误、在显式或隐式数据类型转换过程中出现范围溢出错误或出现域错误后 Adaptive Server 的行为。这种类型的错误是很严重的。缺省设置为 arithabort arith_overflow on，它将回退发生错误的整个事务。如果错误发生在不包含事务的批处理中，则 arithabort arith_overflow on 不回退批处理中以前的命令，但 Adaptive Server 也不执行批处理中生成错误的语句之后的任何语句。

  将 arith_overflow 设置为 on 指的是执行时间，而不是指将 Adaptive Server 设置成的规范化级别。

  如果设置了 arithabort arith_overflow off，Adaptive Server 将中止导致错误的语句，但会继续处理事务或批处理中的其它语句。

- arithabort numeric_truncation – 指定精确数值类型在隐式数据类型转换过程中发生标度损失后 Adaptive Server 的行为。（当显式转换导致标度损失时，将截断结果而不发出任何警告。）缺省设置 arithabort numeric_truncation on 将中止导致错误的语句，但 Adaptive Server 会继续处理事务或批处理中的其它语句。如果设置了 arithabort numeric_truncation off，Adaptive Server 就会截断查询结果并继续进行处理。

set arithignore [arith_overflow] {on | off}

确定在出现被零除错误或精度损失后 Adaptive Server 是否显示消息。缺省情况下，将 arithignore 选项设置为 off。这会使 Adaptive Server 在任何导致数字溢出的查询之后显示警告消息。若要使 Adaptive Server 忽略溢出错误，请使用 set arithignore on。忽略可选的 arith_overflow 关键字不会有任何影响。

set bulk array size *number*

建立在使用批量复制接口传送之前在本地服务器内存中放入缓冲区的行数。

请将此选项仅与 CIS 一起使用，以通过 select into 将行传送到远程服务器。

可使用 @@*bulkarraysize* 全局变量查看当前设置。

number – 指示放入缓冲区的行数。如果传送的行包含 text、 unitext、 image 或 java ADT，则批量复制接口将忽略数组大小的当前设置，而使用值 1。同时，实际使用的数组大小不应超过 @@*bulkbatchzise* 的值。如果 @@*bulkbatchsize* 的值小于数组大小，则使用较小的值。

新连接将继承来自配置属性 cis bulk insert array size 当前设置的初始数组大小，缺省值为 50。将此值设置为 0 将其重设为缺省值。

set bulk batch size *number*

建立使用批量接口时通过 select into *proxy_table* 传送到远程服务器的行数。批量接口适用于所有 Adaptive Server，就像 DirectConnect 适用于 Oracle 版本 12.5.1 一样。

请将此选项仅与 CIS 一起使用，以通过 select into 将行传送到远程服务器。

可使用 @@*bulkbatchsize* 全局变量查看当前设置。

批量接口允许在传送指定的行数之后执行 commit。这样远程服务器就可以释放由批量传送操作占用的任何日志空间，并可启用在两个服务器之间的大数据组的传送，而不会填满事务日志。

新连接将继承来自配置属性 cis bulk insert batch size 当前设置的初始批处理大小，缺省值为 0。值为 0 表示传送完所有行之前不应该提交任何行。

set builtin_date_strings *number*

如果给定的字符串作为参数代替按时间顺序的值，则服务器会将其解释为 datetime 值 （而不考虑其外观精度）。这是缺省行为，由 builtin_date_strings 值 0 表示。

如果将 builtin_date_strings 的值更改为 1，服务器会将参数字符串解释为 bigdatetime。这将影响按时间顺序的 builtin 的结果。

set {chained, close on endtran, nocount, noexec, parseonly, self_recursion, showplan, sort_resources} {on | off}

- chained – 恰好在会话起点的第一个数据检索或数据修改语句之前，以及在事务结束之后开始事务。在链式模式中， Adaptive Server 在下列语句之前会隐式执行一个 begin transaction 命令：delete、fetch、insert、lock table、open、select 和 update。不能在事务中执行 set chained。
- close on endtran – 导致 Adaptive Server 在事务结束时关闭该事务内所有打开的游标。可使用 commit 或 rollback 语句结束事务。不过，仅影响在设置此选项 （存储过程、触发器等等）的范围内声明的游标。有关游标范围的详细信息，请参见 《Transact-SQL 用户指南》。

  有关已评估配置的详细信息，请参见 《系统管理指南》。
- nocount – 控制受语句影响的行的显示。set nocount on 禁用行的显示；set nocount off 重新启用行计数。
- noexec – 编译每个查询但不执行。noexec 通常与 showplan 一起使用。设置 noexec on 之后，不执行任何后续命令 （包括其它 set 命令），直到设置 noexec off 为止。
- parseonly – 检查每个查询的语法并返回所有错误消息，但不编译或执行查询。不在存储过程或触发器内使用 parseonly。
- self_recursion – 确定 Adaptive Server 是否允许触发器再次自行引发（这称为**自递归**）。缺省情况下， Adaptive Server 不允许触发器自递归。可以仅在当前客户端会话期间打开此选项，其作用受设置此选项的触发器的范围的限制。例如，如果设置了 self_recursion on 的触发器返回或引起其它触发器引发，则此选项将还原为 off。此选项仅能在触发器内工作，对用户会话没有影响。
- showplan – 生成查询处理计划的说明。 showplan 的结果在性能诊断中是有用的。在存储过程或触发器中使用时 showplan 不输出结果。对于并行查询， showplan 还输出运行时调整的查询计划 （如果适用）。请参见 《性能和调优指南》。
- sort_resources – 生成 create index 语句的排序计划的说明。 sort_resources 的结果在决定排序操作是串行还是并行时非常有用。当 sort_resouces 为 on 时， Adaptive Server 会输出排序计划，但不会执行 create index 语句。请参见 《性能和调优指南》中的“并行排序”。

set char_convert {off | on [with {error | no_error}] | *charset* [with {error | no_error}]}

启用或禁用 Adaptive Server 和客户端之间的字符集转换。如果客户端使用 Open Client DB-Library 版本 4.6 或更高版本，而客户端和服务器使用不同的字符集，则登录进程中会打开转换，并基于客户端使用的字符集将转换设置为缺省值。也可以使用 set char_convert *charset* 启动服务器字符集和不同的客户端字符集之间的转换。

charset 可以是 syscharsets 中 type 值小于 2000 的字符集 ID 或名称。

set char_convert off 会关闭转换，这样发送和接收字符时都不会进行转换。若转换已关闭，则 set char_convert on 会打开转换。如果字符集转换未在登录进程中打开，也未通过 set char_convert 命令打开，则 set char_convert on 会生成一条错误消息。

如果请求使用 set char_convert *charset* 进行字符集转换，而 Adaptive Server 不能执行请求的转换，则转换状态与请求前的转换状态保持一致。例如，如果在 set char_convert *charset* 命令之前将转换设置为 off，则如果请求失败，转换仍然是关闭的。

当包含了 with no_error 选项时，如果来自 Adaptive Server 的字符不能转换为客户端的字符集时， Adaptive Server 不会通知应用程序。当客户端与 Adaptive Server 连接时，错误报告最初是打开的：如果您不需要错误报告，就必须使用 set char_convert {on | charset} with no_error 关闭每个会话的错误报告。若要在会话内打开已关闭的错误报告，请使用 set char_convert {on | charset} with error。

无论错误报告是否打开，不可转换的字节都会替换为 ASCII 问号 (?)。

有关字符集转换中的错误处理的详细信息，请参见 《系统管理指南》。

set cis_rpc_handling {on | off}

将 CIS 设为集群环境中用于处理 RPC 的缺省机制。

set [clientname *client_name* | clienthostname *host_name* | clientapplname *application_name*]

为客户端指派名称。

- clientname *client_name* – 为客户端指派单独的名称。在一个系统中，多个客户端使用相同的客户端名称与 Adaptive Server 连接时，该命令对区分这些客户端是有用的。为用户指派新名称后，这些客户端会以此新名称显示在 sysprocesses 表中。

  client_name 是指派给用户的新名称。

- clienthostname *host_name* – 为主机指派单独的名称。在一个系统中，多个客户端使用相同的主机名与 Adaptive Server 连接时，该命令对区分这些客户端是有用的。为主机指派一个新名称后，它会以此新名称显示在 sysprocesses 表中。

  host_name 是指派给主机的新名称。

- clientapplname *application_name* – 为应用程序指派单独的名称。在一个系统中，多个客户端使用相同的应用程序名与 Adaptive Server 连接时，该命令对区分这些客户端是有用的。为应用程序指派一个新名称后，它会以此新名称显示在 sysprocesses 表中。

  application_name 是指派给应用程序的新名称。

set compression {on | off | default}

针对会话启用或禁用压缩：

- on – 对配置了压缩的表和分区的新数据启用数据压缩。Adaptive Server 会压缩所有符合条件的行。
- off – 插入并更新所有未压缩的新数据。触发页压缩的插入会忽略未压缩的行。更新的行会保持不压缩。在 update 期间压缩的解压缩行 （作为 uncompressed 插入的行会保持不压缩，直到显式指定压缩为止）。
- default – 将压缩级别重新设置为缺省设置 （插入和更新是根据表或分区设置进行压缩的）。

注释　与大多数 set 参数不同，如果您在嵌套过程中发出 set compression 之前执行 set export_options，Adaptive Server does 就不会将压缩级别导出到父过程的上下文中。

set cursor rows *number* for *cursor_name*

导致 Adaptive Server 为客户端应用程序中的每个游标的 fetch请求返回 *number* 行。 *number* 可以是无小数点的数值文字，也可以是 integer 类型的局部变量。如果 *number* 小于或等于零，则将此值设置为 1。您可以为游标设置 cursor rows 选项，无论它是打开的还是关闭的。不过，此选项不影响包含 into 子句的 fetch 请求。 *cursor_name* 指定要为其设置 返回的行数的游标。

set {datefirst *number*, dateformat *format*, language *language*}

指定下列设置：

- datefirst *number* – 使用数值设置指定一周的第一天。us_english 语言中该缺省值为 Sunday。若要设置一周的第一天，请使用下面的设置：

将一周的第一天设置为	使用设置
Monday	1
Tuesday	2
Wednesday	3
Thursday	4
Friday	5
Saturday	6
Sunday （us_english 语言中的缺省值）	7

注释 无论您将哪一天设置为一周的第一天，这天的值都为 1。该值与您在 set datefirst n 中使用的数值设置不同。例如，如果将“Sunday”设置为一周的第一天，则其值为 1。如果将“Monday”设置为一周的第一天，则“Monday”的值变为 1。如果将“Wednesday”设置为一周的第一天，则“Wednesday”的值变为 1，依此类推。

- dateformat *format* – 设置输入 datetime、 smalldatetime、 date 或 time 数据时的日期分量 *month*/*day*/*year* 的顺序。有效参数为 *mdy*、*dmy*、*ymd*、 *ydm*、 *myd* 和 *dym*。 us_english 语言中该缺省值为 *mdy*。
- language *language* – 是显示系统消息的语言的正式名称。该语言必须安装在 Adaptive Server 上。缺省值为 us_english。

set deferred_name_resolution

设置仅适用于当前会话的延迟名称解析。

set delayed_commit {on | off | default}

确定将日志记录写入磁盘的时间。当 delayed_commit 参数设置为 true 时，日志记录将以异步方式写入磁盘并且控制将返回客户端，而不等待 IO 完成。

会话级设置将覆盖任何现有的数据库级设置。将 delayed_commit 更改为其缺省值可恢复数据库级设置。

注释 注意：在使用 delayed_commit 之前，须仔细考虑应用的注意事项。

set dml_logging {minimal | default}

确定 insert、 update 和 delete (DML) 操作的记录量。有效值包括：

- minimal – Adaptive Server 尝试不记录对 DML 语句所做的任何更改。在大多数情况下， Adaptive Server 很少或根本不记录到 syslogs。
- default – Adaptive Server 禁用特定于会话的最少日志记录，并使用基于特定于表的日志记录级别和数据库范围的日志记录级别为单个表启用的记录模式。

对日志记录进行的更改仅应用于此会话中的用户所拥有的对象 （如果适用）。另外， set dml_logging 只影响会话所有者拥有的表：

- 任何用户都可以执行 set dml_logging 以进行 minimal 记录以及恢复为 default 记录模式。此 set 成功后，将为当前会话最低限度记录在任何数据库中执行语句的用户拥有的所有表上的 DML，直到执行 set dml_logging default。
- 当为会话启用了 minimal 记录但 DML 在不是由会话用户拥有的表上操作时， DML 记录缺省为数据库和表级别的设置。
- 会话或过程中的 DML 记录设置由过程继承，但只影响运行会话的用户拥有的表。

set encryption passwd '*password_phrase*'
for {key | column} {*keyname* | *column_name*}

创建加密密钥的口令以在 insert、update、delete、select、alter table 或 select into 语句中加密或解密数据。

- *password_phrase* – 是使用 create encryption key 或 alter encryption key 命令指定的显式口令，用于保护密钥。
- key – 表示当访问由命名密钥加密的任何列时， Adaptive Server 将使用此口令解密密钥
- *keyname* – 可作为完全限定名提供。例如：

  [[*database*.][*owner*.]*keyname*
- column – 指定 Adaptive Server 只在加密或解密命名列的上下文中使用此口令。最终用户不必知道加密给定列的密钥的名称。
- *column_name* – 要设置加密口令的列的名称。 *column_name* 提供为：

  [[*database*.][*owner*].]*table_name*.*column_name*

set export_options [on | off]

Adaptive Server 的缺省行为是在触发器或系统过程运行完成后重置它们设置的任何设置参数更改。通过启用 set export_options 可在会话期间保留由系统过程或触发器设置的会话设置。

例如，以下语句将启用 set export_options：

```
set export_options on
```

以下语句将禁用 set export_options 并使 Adaptive Server 恢复缺省行为：

```
set export_options off
```

set fipsflagger {on | off}

决定使用初级 ANSI SQL 的 Transact-SQL 扩展时 Adaptive Server 是否显示警告消息。缺省情况下，使用非标准 SQL 时 Adaptive Server 不会通知您。此选项不会禁用 SQL 扩展。发出非 ANSI SQL 命令时，处理完成。

set flushmessage {on | off}

确定 Adaptive Server 何时向用户返回消息。缺省情况下，在生成消息的查询完成之前或者缓冲区达到容量上限之前，消息存储在缓冲区中。使用 set flushmessage on 可在消息生成时立即将它们返回给用户。

set fmtonly {on | off}

在存储过程中捕获计划，但不实际执行。

set forceplan {on | off}

导致查询优化程序将查询的 from 子句中表的顺序用作查询计划的连接顺序。 forceplan 通常用于优化程序不能选择好的计划时。强制执行不正确的计划会对 I/O 和性能产生严重的负面影响。请参见 《性能和调优指南》。

注释 查询优化程序忽略强制对外部连接使用非法连接顺序的尝试，如以下示例所示：

```
1> set forceplan on
2> select * from table1, table2
    where table2.id *= table1.id
```

---

set identity_insert [*database*.[*owner*.]]*table_name* {on | off}

确定是否允许显式插入到表的 IDENTITY 列中。此选项只能用于基表。它不能用于视图，也不能在触发器内进行设置。

为表设置 identity_insert on 之后，表所有者或对列具有 insert 权限的用户可手动插入任何大于 5 的合法值。例如，如果插入值 55，则会使 IDENTITY 列值产生很大间隔：

```
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)
```

如果随后将 identity_insert 设置为 off，则 Adaptive Server 在下一次插入时为 IDENTITY 列指派值 55 + 1 （即 56）。如果回退包含 insert 语句的事务， Adaptive Server 将放弃值 56，并在下一次插入时使用值 57。

除非创建了 IDENTITY 列的唯一索引，否则 Adaptive Server 不会检验插入的值的唯一性；您可以插入任何正整数。

设置 identity_insert *table_name* off 可通过禁止显式插入到 IDENTITY 列而恢复缺省行为。无论何时，都可以在会话中对单一数据库表使用 set identity_insert *table_name* on。

对 set identity_insert table_name on|off 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或是对表拥有 identity_insert 权限的用户。表所有者或具有 manage any object permission 特权的用户可以将权限授予其他用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者、表所有者或是具有 sa_role 的用户。 identity_insert 权限不能移交。

set identity_update *table_name* {on | off}

将 set identity_update 设置为 on 时，可以显式地更新表上的 IDENTITY 列的值。 identity_update 更改限定行的标识列值。启用 identity_update 时，可以更新任何大于 0 的标识值。不过，如果输入值大于 identity burn max 值，就会分配一组新的 ID 值，而且会相应更新 OAM 页上的 identity burn max 值。如果事务中包含 update，则不能回退新的 identity burn max 值。可以使用 syb_identity 指向标识列来进行 update。例如：

```
update table_name set syb_identity = value
```

where 子句

Adaptive Server 不检查是否有重复条目，也不检验条目是否唯一。可将现有值更新为列的声明精度所允许范围内的任何正整数。可通过在标识列上创建唯一索引来检查是否有重复条目。

对 set identity_update table_name on|off 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或是对表拥有 identity_update 权限的用户。表所有者或具有 manage any object permission 特权的用户可以将权限授予其他用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是数据库所有者、表所有者或是具有 sa_role 的用户。 identity_update 权限不能移交。

set index_union on | off

启用时，设置索引联合以使用 or 子句限制表扫描。

索引联合 （也称为 or 策略）用于包含 or 子句的查询。例如：

```
select * from titleauthor where au_id = "409-56-7008" or title_id = "PC8888"
```

如果 index_union 为：

- 已启用 则此示例使用 au_id 上的索引来查找所有 au_id = "409-56-7008" 的 titleauthor 元组的行 ID (RID)；并使用 title_id 上的索引来查找所有 title_id = "PC8888" 的 titleauthor 元组的 RID。Adaptive Server 然后对所有 RID 执行联合以删除重复项。结果 RID 将与 RidJoin 连接以访问数据元组。
- 已禁用 – Adaptive Server 在查询中不使用索引联合策略来限制表扫描，而是使用表上的其它访问路径（在上面的示例中，它对表 titleauthor 使用表扫描），并在扫描运算符中将 or 子句作为过滤器进行应用。

set literal_autoparam on | off

缺省情况下为 on。如果 literal_autoparam 的服务器级设置为 on，则此选项可启用和禁用该功能的使用。如果服务器级设置是 off，则此设置不起作用。

set lock {wait [*numsecs*] | nowait}

指定锁的设置。

- wait – 指定命令在中止和返回错误之前，等待获取锁的时间长度。
- *numsecs* – 指定命令等待获取锁的秒数。有效值介于 0 和 2147483647 之间，后者是最大的整数值。
- lock nowait – 指定如果命令不能立即获取锁，则返回一个错误并失败。 set lock nowait 等同于 set lock wait 0。

set logbulkcopy {on | off }

为会话配置快速记录 bcp。

set materialized_view_optimization {disable | fresh | stale}

确定查询优化过程中要考虑的预先计算结果集。以下各项之一：

- disabled - （缺省值） Adaptive Server 在查询优化中不使用任何预先计算结果集。
- fresh – 只有使用 immediate refresh 策略时，Adaptive Server 才会考虑预先计算结果集。
- stale – 查询优化时， Adaptive Server 会考虑所有启用的预先计算结果集，即使这些结果集已失效也是如此。

set metrics_capture {on | off}

在会话级启用查询处理 (QP) 指标的捕获，将该捕获设置为 on。QP 指标用于标识和比较查询执行中的经验指标值。执行查询时，查询与一组作为 QP 指标比较基础的已定义指标关联。

set mon_stateful_history on | off

禁用后，对历史监控表 （monSysStatement、 monErrorLog、 monSysSQLText、 monSysPlanText 和 monDeadLock）的查询将返回表缓冲区中的所有行。

启用后，对历史监控表的查询将仅返回自 mon_stateful_history 禁用后添加到表中的行。

set nodata

指定查询完成后不将任何数据传送到客户端。如果指定 set nodata on，则只将 TDS 格式流发送到客户端，查询的行为就像没有符合条件的任何行。

set offsets {select, from, order, compute, table, procedure, statement, param, execute} {on | off}

返回 Transact-SQL 语句中指定的关键字的位置 （相对于查询的起始位置）。关键字列表是一个用逗号分隔的列表，可以包括下面的任何 Transact-SQL 结构：select、from、order、compute、table、procedure、statement、 param 和 execute。如果没有错误， Adaptive Server 返回偏移量。

此选项仅适用于 Open Client DB-Library。

set option *show_option* {normal | brief | long | on | off}

以文本格式生成诊断输出。

有效的 *show_option* 值包括：

- show – 显示所有模块通用的基本语法
- show_lop – 显示所用的逻辑运算符 （扫描、连接等）
- show_managers – 显示优化期间所用的数据结构管理器
- show_log_props – 显示计算出的逻辑属性 （行计数、选择性等）
- show_parallel – 显示并行查询优化的详细信息
- show_histograms – 显示与 SARG/Join 列关联的直方图的处理
- show_abstract_plan – 显示抽象计划的详细信息
- show_search_engine – 显示连接顺序算法的详细信息
- show_counters – 显示优化计数器
- show_best_plan – 显示优化程序选定的最佳查询计划的详细信息
- show_pio_costing – 显示物理输入/输出 （读写磁盘）评估值
- show_lio_costing – 显示逻辑输入/输出 （读写内存）评估值
- show_elimination – 显示分区排除
- show_missing_stats – 显示 SARG/Join 列中缺失的有用统计信息的详细信息

有关详细信息，请参见 Query Optimizer （《查询优化程序》）中的“Displaying Query Optimization Strategies and Estimates”（显示查询优化策略和评估）。

对 set option *show_option* 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 set tracing 特权或 monitor qp performance 特权的用户。

细化权限已禁用	在禁用细化权限的情况下，您必须是具有 set tracing 特权的用户或是具有 sa_role 的用户。

set opttimeoutlimit

设置优化程序的超时限制。 opttimeoutlimit 值的有效范围为 0 到 4000 毫秒，其中 0 表示没有优化限制。

set parallel_degree *number*

指定在一个查询的并行执行中可使用的工作进程数上限。此数目应小于或等于每个查询中的工作进程数目，后者由 max parallel degree 配置参数设置。 *@@parallel_degree* 全局变量存储当前设置。

set plan {dump | load} [*group_name*] {on | off}

引入一个抽象计划命令。

- dump – 启用或禁用为当前连接捕获抽象计划。如果不指定 *group_name*，则计划存储在缺省组 ap_stdout 中。
- load – 启用或禁用为当前连接装载抽象计划。如果不指定 *group_name*，则从缺省组 ap_stdin 中装载计划。
- *group_name* – 是用于装载或存储计划的抽象计划组的名称。

有关详细信息，请参见 《性能和调优指南》中的 “创建和使用抽象计划”。

set plan exists check {on | off}

当和 set plan load 一起使用时，可为多达 20 个查询存储散列键，这些查询来自每个用户的高速缓存中的抽象计划组。

set plan for *show*

为诊断输出生成一个 XML 文档。 *show* 的有效值为：

- show_exec_xml – 以 XML 形式获取编译的计划输出，显示每个查询计划运算符。
- show_execio_xml – 获取计划输出以及估计的和实际的 IO。这也包括查询文本。
- show_opt_xml – 获取优化程序诊断输出 （显示各种不同的组件，如逻辑运算符）、管理器输出、某些搜索引擎诊断输出和最佳查询计划输出。
- show_lop_xml – 以 XML 形式获取输出逻辑运算符树。
- show_managers_xml – 显示查询优化程序准备阶段中各组件管理器的输出。
- show_log_props_xml – 显示给定等效类 （查询中的一个或多个关系组）的逻辑属性。
- show_parallel_xml – 显示生成并行查询计划时与优化程序有关的诊断。
- show_histograms_xml – 显示与直方图和直方图合并有关的诊断。
- show_abstract_plan_xml – 显示 AP 生成/应用。
- show_search_engine_xml – 显示与搜索引擎相关的诊断。
- show_counters_xml – 显示计划对象构造/析构计数器。
- show_best_plan_xml – 以 XML 形式显示最佳计划。
- show_pio_costing_xml – 以 XML 形式显示实际的 PIO 开销计算。
- show_lio_costing_xml – 以 XML 形式显示实际的 LIO 开销计算。
- show_elimination_xml – 以 XML 形式显示分区排除。
- client – 如果指定，则将输出传送到客户端。
- message – 如果指定，则将输出传送到内部消息缓冲区。

有关详细信息，请参见 《性能和调优系列：查询处理和抽象计划》 中的 “显示查询优化策略和估计值”。

对 set plan for *show* 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 set tracing 特权或 monitor qp performance 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 set tracing 特权的用户或是具有 sa_role 的用户。

set plan optgoal {allrows_oltp | allrows_mix | allrows_dss | user_defined_goal_identifier}

设置优化目标。

- allrows_mix – 缺省优化目标，以及混合查询环境中最有用的目标。它均衡考虑了 OLTP 和 DSS 查询环境的需要。
- allrows_dss – 中等到高度复杂性可操作 DSS 查询的最有用目标。目前，在实验基础上提供该目标。

有关优化计划的详细信息，请参见 《性能和调优系列：查询处理和抽象计划》中的 “了解 Adaptive Server 中的查询处理”。

set plan optlevel *value*

设置会话的优化级别。每个 Adaptive Server 版本或 ESD 都可能包括一个新优化级别。例如：

- ase_current – 启用当前及之前版本中的所有优化程序更改。
- ase_default – 禁用自 15.0.3 ESD #1 版本以来的所有优化程序更改。
- ase1503esd2 – 启用 15.0.3 ESD #2 版本及之前版本中的所有优化程序更改。
- ase1503esd3 – 启用 15.0.3 ESD #3 版本及之前版本中的所有优化程序更改。

请参见 《性能和调优系列：查询处理和抽象计划》中的 “控制优化”。

set plan opttimeoutlimit *number*

在会话级设置超时，其中 *n* 是 0 和 1000 之间的任意整数。有关优化计划的详细信息，请参见 《性能和调优系列：查询处理和抽象计划》中的 “了解 Adaptive Server 中的查询处理”。

set plan replace {on | off}

启用或禁用在计划捕获模式期间替换现有抽象计划。缺省情况下，计划替换是关闭的。

set prefetch {on | off}

启用或禁用数据高速缓存的大 I/O。

set print_minlogged_mode_override

向会话输出生成跟踪信息，从而就已经由其它规则为其覆盖表的最少记录模式的语句进行报告。这些跟踪信息包括：是否存在参照完整性约束；延迟模式选择；受影响的表的名称；产生影响的规则的说明等。

set proc_output_params {on | off}

控制将存储过程生成的输出参数发送回客户端。 set proc_output_params off 禁止将输出参数发送回客户端。此参数缺省值为 on。

set proc_return_status {on | off}

控制将返回状态 TDS 标志发送回客户端。 set proc_return_status off 禁止将返回状态标志发送回客户端，isql 客户端不显示 (return status = 0) 消息。此参数缺省值为 on。

警告！ 如果执行过程的客户端应用程序依赖于基于返回状态的过程的成功与否，则不要使用 set proc_return_status off 选项。

---

set process_limit_action {abort | quiet | warning}

指定当没有足够的工作进程可用时， Adaptive Server 是否执行并行查询。在这些情况下，如果：

- process_limit_action 设置为 quiet，则 Adaptive Server 会自动调节计划以使用不超过可用进程数的并行度。
- 没有足够的工作进程可用时， process_limit_action 设置为 warning，则 Adaptive Server 会在调节计划时发出警告消息
- process_limit_action 设置为 abort， Adaptive Server 会中止查询，并发出消息说明没有足够的工作进程可用。

set proxy *login_name*

允许您使用 *login_name* 的权限、登录名和 suid （服务器用户 ID）。对于 *login_name*，指定来自 master..syslogins 的有效登录，并用引号引起来。若要还原到初始登录名和 suid，请使用 set proxy （在该命令中使用初始 *login_name*）。

注释 要使用 set proxy *login_name*，用户 （包括系统安全员）必须具有显式授予的权限。不具有显式权限时，“sa_role”和 “sso_role”都不能发出 set proxy login_name 命令。

有关详细信息，请参见第 633 页的 “使用代理”。

set quoted_identifier {on | off}

确定 Adaptive Server 是否识别用双引号引起的分隔标识符。缺省情况下， quoted_identifier 设置为 off，且所有标识符都必须满足下列条件之一：

- 符合有效标识符的规则。
- 用中括号括起。

如果使用 set quoted_identifier on，则双引号的行为与中括号相同，通过将标识符用双引号引起来，还可以使用以非字母字符开头的表名、视图名和列名，包括其它情况下不能使用的字符或保留字字符。分隔标识符不能超过 28 字节，可能不能被所有前端产品识别，而且当用作系统过程的参数时还会产生意外的结果。

当 quoted_identifier 为 on 时，用双引号引起来的所有字符串都被视作标识符。对字符或二进制字符串使用单引号。

set repartition_degree *number*

是出于语义目的而对任何中间数据流重新分区的最大程度。有关为会话设置 max repartition degree 值的详细信息，请参见 《性能和调优系列：查询处理和抽象计划》中的 “并行查询处理”。

set repthreshold *number*

在会话级别设置 SQL 复制阈值。如果在存储过程中调用 set repthreshold，则其作用域为该过程。如果在用户会话中调用，则其作用域为该会话。

用户可以更改阈值的作用域并使用登录触发器设置会话阈值，在这种情况下，无需在登录时显式设置会话阈值。

例如，

```
create proc myproc
as
    set repthreshold 777
---------------------

alter login sa modify login script 'myproc'
----------------------
option changed.
(Return status = 0
```

每次用户 sa 登录时都会调用 “myproc”过程，并确保整个会话的复制阈值设置为 777。

更改阈值作用域的另一种方法是使用 set export_options：

```
create proc p2
```

```
as
    set repthreshold222
    set export_options on
--------------------------
```

执行 p2 后，阈值仍设置为 222。阈值的层次为：会话 > 表 > 数据库

如果按上述顺序指定，则会话阈值设置为 0 时，仍然有效的复制阈值仅包括表和数据库阈值。如果未设置阈值，则阈值缺省为 50 行。

要设置表级别阈值，请参见 sp_setrepdbmode；要设置数据库级别的阈值，请参见 sp_setrepdefmode。有关这两个存储过程的信息，请参见 《参考手册：过程》。

可以导出会话阈值；存储过程可以设置阈值并将 export options 设置配置为 ON。 Adaptive Server 在调用过程或会话中强制执行新阈值。

对 set repthreshold 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 manage replication 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 replication_role 的用户。

set resource_granularity *number*

覆盖全局值 max resource granularity 并将它设置为会话特定的值，这将对 Adaptive Server 是否使用内存密集型操作产生影响。有关详细信息，请参见 Query Processor （《查询处理器》）中的 “Parallel Query Processing”（并行查询处理）。

set role {*role_name* [with passwd "*password*"]} {on | off}

使用 set role *role_name* off 关闭角色，而在需要时可使用 set role *role_name* on 打开角色。

- *role_name* 可以是系统角色或用户定义角色的名称。当您登录时，已授予您的所有角色都会自动激活。如果已向您授予 sa_role 或 sso_role，您不能禁用这些角色，除非您是数据库中的指定用户或别名用户，或者存在 “guest”用户。
- *role_name* – 由系统安全员创建的任何用户定义角色的名称。缺省情况下，用户定义的角色不启用。要将用户定义的角色设置为在登录时激活，系统安全员必须在创建或者变更登录名或登录配置文件时将此角色指定为缺省角色或自动激活的角色。
- with *passwd* – 指定要激活角色的口令。如果用户定义的角色带有口令，则必须指定此口令才能激活角色。

要使用 set role，必须已向您授予此角色。如果仅因为具有某个角色而获得访问数据库的权利，则使用数据库期间不能关闭此角色。

如果已使用激活谓词向您授予角色，则 Adaptive Server 会在您执行 set role on 时评估该谓词。如果该谓词的评估结果为 false，则 Adaptive Server 将返回一条错误消息，并且不会激活该角色。

set {rowcount *number*, textsize *number*}

导致 Adaptive Server 在影响指定数目的行后停止处理查询 （select、insert、 update 或 delete）。 *number* 可以是无小数点的数值文字，也可以是 integer 类型的局部变量。若要关闭该选项，请使用：

```
set rowcount 0
```

您可用 *@@setrowcount* 全局变量确定 set rowcount 的当前值。例如：

```
select @@setrowcount

_____________
           37
```

set scan_parallel_degree *number*

指定基于散列的扫描 （未分区表上的并行索引扫描和并行表扫描）的最大特定于会话的并行度。这个数值必须小于或等于 max scan parallel degree 配置参数的当前值。 *@@scan_parallel_degree* 全局变量可存储当前设置。

set send_locator {on | off }

指定 Adaptive Server 在发送到客户端的结果集中是发送 LOB 还是发送引用 LOB 的定位符。当此选项为 off （缺省值）时， Adaptive Server 发送 LOB。

set session authorization *login_name*

与 set proxy 相似，不同的是：set session authorization 遵循 SQL 标准，而 set proxy 是 Transact-SQL 扩展。

set show_exec_info

在命令执行时生成其它信息。

- on – 生成有关 DML 语句的记录模式的额外诊断。显示当前语句和会话的所选记录模式以及执行 DML 的用户。
- off – 禁用 show_exec_info。

set show_permission_source, {on|off}

以表形式显示被授予者、被授予者的类型、授予者、操作、对象以及谓词。被授予者可以是 *user_name*、*role_name* 或 *group_name*。`Type of grantee` 列将显示被授予者是用户、角色还是组。如果没有谓词， Adaptive Server 将返回 NULL。

在使用 set show_permission_source, {on|off} 时，请考虑：

- 如果将权限授予属于某个层次的角色，那么不会直接将该权限授予用户。例如，如果您将 role1 的权限授予 role2，进而又授予 role3，然后该权限被授予某个用户。 set show_permission_source 会在 `grantee` 列中显示 “role1”，而不是 “role3”，因为是向 role1 而不是 role3 授予了该权限。可使用 sp_displayroles ... expand_up 查看有关 role3 的信息。
- 如果有多个授予者授予特定操作的权限， set show_permission_source 会显示与具有最大用户 ID 的授予者相关联的权限。
- set show_permission_source 显示对象级而非列级的信息，因为同一对象的多个行可能来自不同列的不同授予者。
- 如果对于操作和授予者的组合，对象的不同谓词存在多个权限，则 set show_permission_source 会显示所有谓词名称，每个谓词对应单独一行。

set show_sqltext {on | off}

用于为即席查询、存储过程、游标和动态准备的语句输出 SQL 文本。

在执行查询以收集 SQL 会话的诊断信息之前，无需启用 set show_sqltext （与 set showplan on 等命令的处理方式相同）。相反，可以在命令运行时启用它，以帮助确定未正常执行的查询并诊断其存在的问题。

在启用 show_sqltext 之前，必须先启用 dbcc traceon，以使输出显示为标准输出：

```
dbcc traceon(3604)
```

set show_sqltext 的语法为：

```
set show_sqltext {on | off}
```

例如，以下语句将启用 show_sqltext：

```
set show_sqltext on
```

一旦启用 set show_sqltext， Adaptive Server 便会将您输入的每个命令或系统过程的所有 SQL 文本都输出为标准输出。根据运行的命令或系统过程，此输出可能非常详尽。

若要禁用 show_sqltext，请输入：

```
set show_sqltext off
```

set statement_cache on | off

缺省情况下为 on。如果 statement_cache 的服务器级设置为 on，则此选项可启用和禁用该功能的使用。如果服务器级设置是 off，则此设置不起作用。

set show_transformed_sql, {on|off}

在 Adaptive Server 查询处理期间将查询的中间形式显示为 SQL 文本 - 即，在对视图、谓词特权、加密等执行查询转换之后，但在执行子查询的查询转换之前。

set statistics {io, subquerycache, time, plancost, simulate} {on | off}

显示各种类型的统计信息

- io – 显示有关语句中引用的每个表的统计信息：
  - 访问表的次数 （扫描计数）
  - 逻辑读取次数 （内存中访问的页）
  - 物理读取次数 （数据库设备访问）

  statistics io 为每个命令显示写入的缓冲区数目。

  如果 Adaptive Server 已配置为强制资源限制，则 statistics io 还显示总的 I/O 开销。
- subquerycache – 为每个子查询显示子查询高速缓存中的高速缓存命中次数、未命中次数和行数。
- time – 显示 Adaptive Server 分析和编译每个命令所用的时间。在命令的每一个步骤，statistics time 显示 Adaptive Server 执行此命令所需的时间。时间以毫秒或时钟周期为单位指定，其确切值与计算机有关。
- plancost – 以树格式显示查询统计信息。

  注释 启用 set statistics plancost 后，Adaptive Server 将 `lio`、`pio` 和 `row` 的名称分别缩写为 `l`、 `p` 和 `r`。
- simulate – 指定优化程序应使用模拟统计信息来优化查询。

请参见 《性能和调优系列：利用统计分析改进性能》中的 “使用 set statistics 命令”。

set strict_dtm_enforcement {on | off}

决定服务器是否将事务传播到不支持 Adaptive Server 事务协调服务的服务器。缺省值继承自 strict dtm enforcement 配置参数的值。

set string_rtruncation {on | off}

决定 insert 或 update 命令截断 char、unichar、varchar 或 univarchar 字符串时 Adaptive Server 是否引发 SQLSTATE 例外。如果截断的字符均为空格，则不会引发例外。缺省设置 off 不会引发 SQLSTATE 例外，而且会不加提示地截断字符串。

set system_view {instance | cluster | clear}

（仅限集群）为会话指定系统视图，并控制影响存储过程 （例如 sp_who）的输出的虚设表的实现。

- instance – 为当前实例设置系统视图。
- cluster – 为集群设置系统视图。
- clear – 清除任何会话级别设置，返回到托管该 spid 的逻辑集群的 system_view 设置。输入 `select @@system_view` 来检查当前值。

set switch [serverwide] {on | off} *trace_flag*[,*trace_flag*] [,with *option* [, *option*]>

用于在本地和服务器范围内设置跟踪标志和开关名称。

- serverwide – 可选项，将 serverwide 开关设置为 ON 或 OFF。缺省设置因会话而异。
- on – 跟踪标志已打开。
- off – 跟踪标志已关闭。
- *trace_flag* – 编号 （原来的跟踪标志编号）和/或开关名称构成的序列。
- *option* – 可选的开关选项序列。有效值包括：
  - override – 此选项是启用未说明的开关名称或跟踪标志所必需的
  - no_info – 此选项用来阻止任何信息性警告出现

对 set switch 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下： • 要设置跟踪标志 3604/3605，您必须具有 monitor qp performance 特权或 set switch 特权。 对于所有其它跟踪标志，您必须具有 set switch 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须具有 sa_role。

set textsize [*number*]

指定 select 语句返回的 text、unitext 或 image 类型数据的最大大小（以字节为单位）。 *@@textsize* 全局变量存储当前设置。

isql 中的缺省设置为 32K。有些客户端软件可设置其它缺省值。若要将 textsize 重新设置为缺省大小 (32K)，请使用：

```
set textsize 0
```

set tracefile [*filename*] [off] [for *spid*]

一旦启用，便会将当前会话的所有 SQL 文本都保存至指定的文件，每个 SQL 文本批处理都会附加到上一个批处理之后。

启用跟踪的语法为：

set tracefile *file_name* [off] [for *spid*

禁用跟踪的语法为：

set tracefile off [for *spid*]

其中：

- *file_name* – 保存 SQL 文本的文件的完整路径。如果未指定目录路径，Adaptive Server 将在 *$SYBASE* 中创建该文件。

  注释 如果 *file_name* 包含数字和字母之外的特殊字符 （“:”、“/”等），则必须用引号将 *file_name* 引起来。例如，以下 *file_name* 必须用引号引起来，因为目录结构中包含 “/”：

  ```
  set tracefile '/tmp/mytracefile.txt' for 25
  ```

  如果 *file_name* 不包含特殊字符，并且要将其保存到 *$SYBASE*，则不需要用引号引起来。例如，以下 *file_name* 不需要用引号引起来：

  ```
  set tracefile mytracefile.txt
  ```

- off – 为此会话或 spid 禁用跟踪。
- *spid* – SQL 文本要保存至跟踪文件的服务器进程 ID。只有具有 SA 或系统安全员角色的用户才能为其它 spid 启用跟踪。不能保存系统任务 （如管家或端口管理器）的 SQL 文本。

注释 对特定会话使用 set tracefile 后，会将所有后续 set 命令或 DBCC 跟踪的诊断输出重定向到跟踪文件。

确保在发出 set tracefile off 之前关闭已打开的所有诊断命令，否则本应转到跟踪文件的输出会转到客户端。

对 set tracefile 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，要为您自己的会话设置跟踪文件，您必须具有 set tracing 特权或 monitor qp performance 特权。要为其他用户的会话设置跟踪文件，您必须具有 set tracing any process 特权或 monitor qp performance 特权。
细化权限已禁用	在禁用细化权限的情况下，要为您自己的会话设置跟踪文件，您必须具有 set tracing 特权；要为其他用户的会话设置跟踪文件，您必须具有 sa_role 或 sso_role。

set transaction isolation level {[read uncommitted | 0] | [read committed | 1] | [repeatable read | 2] | [serializable | 3]}

设置会话的事务隔离级别。设置该选项后，当前或以后的所有事务都会以此隔离级别操作。

- read uncommitted | 0 – 以隔离级别 0 执行的扫描不要求任何锁。因此，扫描过程中级别为 0 的扫描的结果集可能更改。如果扫描位置因基础表中的更改而丢失，必须使用唯一索引才能重新启动该扫描。如果没有唯一索引，该扫描可能会中止。

  缺省情况下，如果对位于只读数据库之外的表进行 0 级扫描，就需要使用唯一索引。可以强制 Adaptive Server 选择非唯一索引或表扫描来覆盖此要求，如下所示：

  ```
  select * from table_name (index table_name)
  ```

  基础表上的活动可能会导致扫描在完成之前就中止。

- read committed | 1 – 缺省情况下，Adaptive Server 的事务隔离级别是 read committed 或 1，此级别允许持有数据上的共享读取锁。
- repeatable read | 2 – 禁止非重复读取。
- serializable | 3 – 指定隔离级别 3，Adaptive Server 将 holdlock 应用到事务中的所有 select 和 readtext 操作，该事务将持有查询的读取锁，直到该事务结束为止。如果同时还设置了链式模式，那么对于所有隐式开始事务的数据检索或修改语句，此隔离级别将继续有效。

set transactional_rpc {on | off}

控制远程过程调用的处理。如果此选项设置为 on，则事务等待处理时，Adaptive Server 会协调 RPC。如果此选项设置为 off，则远程过程调用由 Adaptive Server 节点处理器处理。缺省值继承自 enable xact coordination 配置参数的值。

示例

示例 1 通知 Adaptive Server 对等于 (=) 和不等于 (!=) 比较计算和集合函数中的 NULL 值操作数求值时要符合初级 ANSI SQL 标准：

```
set ansinull on
```

使用 set ansinull on 时，如果 Adaptive Server 在一个或多个列或行中发现空值，则集合函数和行集合会引发下面的 SQLSTATE 警告：

```
Warning - null value eliminated in set function
```

如果等于操作数或不等于操作数中有值为 NULL，则比较的结果为 UNKNOWN。例如，下面的查询在 ansinull 模式下不会返回任何行：

```
select * from titles where price = null
```

如果使用 set ansinull off，同样的查询将返回 price 为 NULL 的行。

示例 2 激活字符集转换，根据客户端所用的字符集将转换设置为缺省设置。当无法将字符转换为客户端字符集时， Adaptive Server 还会通知客户端或应用程序：

```
set char_convert on with error
```

示例 3 指定在缺省情况下 CIS 处理外发 RPC 请求：

```
set cis_rpc_handling on
```

示例 4 为此用户指派客户端名称 alison、主机名 money1 以及应用程序名称 webserver2：

```
set clientname 'alison'
set clienthostname 'money1'
set clientapplname 'webserver2'
```

示例 5 使用 test_cursor 为客户端请求的每个后续 fetch 语句返回 5 行：

```
set cursor rows 5 for test_cursor
```

示例 6 通知 Adaptive Server 在会话期间保留由系统过程或触发器设置的会话设置。

```
set export_options on
```

若要禁用 set export_options 并使 Adaptive Server 恢复为缺省行为，请使用：

```
set export_options off
```

可以使用 set export_options on 导出这些优化设置。

注释 缺省情况下，为登录触发器启用 set export_options。

示例 7 如果使用 Transact-SQL 扩展，通知 Adaptive Server 显示一条警告消息：

```
set fipsflagger on
```

然后，如果使用非标准 SQL，如下：

```
use pubs2
go
```

Adaptive Server 显示：

```
SQL statement on line number 1 contains Non-ANSI text.
The error is caused due to the use of use database.
```

示例 8 将值 100 插入到 stores_south 表的 IDENTITY 列，然后禁止对此列进行进一步的显式插入。注意 syb_identity 关键字的使用；Adaptive Server 使用 IDENTITY 列名替换此关键字：

```
set identity_insert stores_south on
go
insert stores_south (syb_identity)
values (100)
go
set identity_insert stores_south off
go
```

示例 9 启用 idenity_update 并分别用值 1 和 10 更新表，然后禁用 identity_update：

```
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
```

示例 10 如果会话或存储过程中的后续命令无法立即获取请求的锁，则返回一个错误并失败：

```
set lock nowait
```

示例 11 当前会话或存储过程中的后续命令会无限期地等待直到获取锁：

```
set lock wait
```

示例 12 会话或存储过程中的后续命令将等待 5 秒以获取锁，若无法获取锁，则生成错误消息且命令失败：

```
set lock wait 5
```

示例 13 对 dev_plans 组启用捕获抽象计划：

```
set plan dump dev_plans on
```

示例 14 对当前会话中的查询启用从 dev_plans 组中装载抽象计划：

```
set plan load dev_plans on
```

示例 15 禁止输出参数信息：

```
1> 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
```

如果未执行 set proc_output_params off，(return status = 0) 后的输出将包含以下行：

```
Return parameters:

 -----------
           2
```

示例 16 禁止参数和返回状态 TDS 标记的输出：

```
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)
```

另外，还可以通过在运行此批处理之前使用 set nocount on 选项禁止报告受影响行数的行，从而生成没有额外消息的输出。

示例 17 执行此命令的用户现在可以使用登录名“mary”及 Mary 的服务器用户 ID 在服务器上进行操作：

```
set proxy "mary"
```

示例 18 对于每个 insert、update、delete 和 select 语句，Adaptive Server 会在查询影响前四行后停止查询。例如：

```
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
```

示例 19 通知 Adaptive Server 将所有用双引号引起来的字符串作为标识符处理。表名 “!*&strange_table”和列名 “emp’ s_name”在 quoted_identifier 为 on 时是合法的标识符名称：

```
set quoted_identifier on
go
create table "!*&strange_table"
      ("emp's_name" char (10), age int)
go
set quoted_identifier off
go
```

示例 20 将用中括号括起来的字符串视为标识符。即使 quoted_identifier 为 off，表名 [!*&strange_table] 和列名 [emp’ s_name] 也是合法的标识符名称，因为这些名称是用中括号括起来的：

```
set quoted_identifier off
go
create table [!*&strange_table]
      ([emp’ s_name] char (10), age int)
go
```

有关带中括号的标识符用法的信息，请参见第 630 页的 “分隔标识符”。

示例 21 激活 “doctor”角色。用户可使用此命令指定他们想激活的角色：

```
set role doctor_role on
```

示例 22 停用用户在当前会话中的系统管理员角色：

```
set role "sa_role" off
```

示例 23 用户输入口令激活 “doctor”角色：

```
set role doctor_role with passwd "physician" on
```

示例 24 停用 “doctor”角色：

```
set role doctor_role off
```

示例 25 指定未分区表上的并行索引扫描和并行表扫描的最大并行度为 4：

```
set scan_parallel_degree 4
```

示例 26 另一种声明示例 5 的方法：

```
set session authorization "mary"
```

示例 27 为每个查询返回处理计划的说明，但并不执行处理计划：

```
set showplan, noexec on
go
select * from publishers
go
```

示例 28 以树格式显示查询的统计信息：

```
set statistics plancost on

select * from authors

au_id       au_lname     au_fname     phone        address
city    state       country          postalcode
----------- -----------  -----------  ------------ --------------------
------------ -----  ----------      ------------------------------------
172-32-1176  White        Johnson      408 496-7223 10932 Bigge Rd.
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)
```

示例 29 导致 Adaptive Server 在截断 char、 unichar 或 nchar 字符串时生成例外：

```
set string_rtruncation on
```

如果 insert 或 update 语句会截断字符串，则 Adaptive Server 显示：

```
string data, right truncation
```

示例 30 将用 select 语句返回的 text、unitext 或 image 数据大小上限设置为 100 个字节：

```
set textsize 100
```

示例 31 将 serverwide 开关设置为 on，以便为未说明的跟踪标志 110 设置跟踪标志，并且不显示其它信息性警告：

```
set switch serverwide on 110 with override, no_info
```

示例 32 为当前会话打开一个名为 *sql_text_file* 的跟踪文件：

```
set tracefile '/var/sybase/REL1502/text_dir/sql_text_file'
```

来自 set showplan、 set statistics io 和 dbcc traceon(100) 的后续输出将保存到 *sql_text_file* 中。

示例 33 不指定目录路径，所以该跟踪文件将保存到 *$SYBASE/sql_text_file* 中：

```
set tracefile 'sql_text_file' for 11
```

在 spid 11 上运行的任何 SQL 都将保存到此跟踪文件中。

示例 34 为 spid 86 保存 SQL 文本：

```
set tracefile '/var/sybase/REL1502/text_dir/sql_text_file' for 86
```

示例 35 指定事务等待执行时，由 CIS 访问方法而不是 Adaptive Server 节点处理器处理 RPC：

```
set transactional_rpc on
```

示例 36 会话中的所有后续查询都将在可重复读取事务隔离级别运行：

```
set transaction isolation level 2
```

示例 37 在事务期间对该事务中的每个 select 语句执行读取锁：

```
set transaction isolation level 3
```

示例 38 此示例演示当表最初在事务中是使用最少记录操作时，同一表上的多个语句的 DML 日志记录模式如何保持不变。

1 开始事务，并将 DML 日志记录设置为最少：

```
begin tran
set dml_logging minimal
```

2 运行 insert 命令：

```
insert into tab1 values(1)
```

3 将 DML 日志记录设置回缺省值：

```
set dml_logging default
```

虽然将 DML 日志记录重新设置为缺省值，但因为 t1 以前在此事务中是使用最少记录运行的，所以将使用最少记录执行此 insert：

```
insert into tab1 values(1)
```

错误日志包括覆盖日志记录模式选择的原因。

示例 39 此示例在同一会话中将 show_exec_info 从 minimal 更改为 full：

1 登录到 Adaptive Server：

```
isql -Ubob -Pbob123
use myimdb
```

2 创建表 tab1：

```
create table tab1(col1 int)
```

3 启用 show_exec_info 并将 DML 日志记录设置为 minimal：

```
set show_exec_info on
set dml_login minimal
```

4 将值插入到 tab1 中：

```
insert into tab1 values(1)
```

5 Adaptive Server 显示表名和数据库名、运行命令的用户 ID 以及所用的日志记录模式：

```
Operating on the table 'tab1', database 'myimdb' (owner ID 3) in
'minimal' logging mode by user ID 3.
```

6 将 DML 日志记录设置回缺省值：

```
set dml_logging default
```

7　向 tab1 中插入更多值：

```
insert into tab1 values(1)
```

8　Adaptive Server 显示表名和数据库名、运行命令的用户 ID 以及所用的日志记录模式：

```
Operating on the table 'tab1', database 'myimdb' (owner ID 3) in 'full'
logging mode by user ID 3.
```

用法

fipsflagger、 string_rtruncation、 ansinull、 ansi_permissions、 arithabort 和 arithignore 影响 Adaptive Server 的错误处理以及与 SQL 标准的遵从性。

- 仅在启用 CIS 时才可使用 cis_rpc_handling 和 transactional_rpc 选项。
- async log service 选项和 delayed_commit 互斥。如果 async log service 设置为 “true”，则 delayed_commit 将不起作用。
- 如果将 Adaptive Server 配置为并行， parallel_degree 和 scan_parallel_degree 将限制查询的并行度。使用这些选项时，应提示优化程序限制并行查询，使其使用的工作进程少于配置参数允许的数目。将这些参数设置为 0 可恢复全服务器范围的配置值。

  如果指定值大于配置参数的允许值， Adaptive Server 会发出警告消息并使用由配置参数设置的值。
- 如果在触发器或存储过程内使用 set 命令，则执行触发器和存储过程后，大部分 set 选项都会回复到它们先前的设置。

  下面的选项在过程或触发器执行后不会回复到其先前设置，但它们会在整个 Adaptive Server 会话期间保持不变，除非您显式地重设它们：

  - datefirst
  - dateformat
  - identity_insert
  - language
  - quoted_identifier
- 如果指定多个 set 选项，第一个语法错误会导致忽略后面的选项。不过，在错误前指定的选项会得到执行，并会设置新的选项值。
- 如果为用户指派客户端名称、主机名或应用程序名，则这些指派仅在当前会话中起作用。用户下次登录时您必须重新指派这些值。虽然新名称出现在 sysprocesses 中，但它们并不用于权限检查，而 sp_who 仍将客户端连接显示为初始登录。有关设置用户进程的详细信息，请参见 《系统管理指南》。

- 除 showplan 和 char_convert 之外的所有 set 选项可立即生效。showplan 在后面的批处理中生效。这里有两个使用 set showplan on 的示例：

```
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)
```

但是：

```
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 会自动在 clientname、clienthostname 和 clientapplname 列中存储一个或多个空格。因此，使用包含 “is null” 的这三列中任一列的查询不会返回预期结果集。
- 如果启用了 set fipsflagger 选项，则在发出 set proxy 命令时该命令会发出以下警告：

```
SQL statement on line number 1 contains Non-ANSI
```

```
text.The error is caused due to the use of DBCC.
```

- 如果使用登录触发器设置当前执行属性，则在登录触发器中启用或禁用的任何可导出的 set 选项在当前进程中生效。
- 有些 set 选项可以分成一组：
  - parseonly、noexec、prefetch、showplan、rowcount 和 nocount 控制着查询的执行方式。将 parseonly 和 noexec 都设置为 on 是毫无意义的。rowcount 的缺省设置是 0（返回所有行），其它的缺省为 off。
  - statistics 选项在每次查询后显示性能统计信息。statistics 选项的缺省设置是 off。有关 noexec、 prefetch、 showplan 和 statistics 的详细信息，请参见《性能和调优指南》。
  - 可以使用从子查询返回的文字、变量或表达式在 set 子句中最多更新 1024 列。
  - offsets 用于 DB-Library，以解读来自 Adaptive Server 的结果。此选项的缺省设置为 on。
  - datefirst、dateformat 和 language 影响日期函数、日期顺序和消息显示。如果是在触发器或存储过程内使用，则这些选项不会还原为其以前的设置。

    在缺省语言 us_english 中， datefirst 是 1 （星期天）， dateformat 是 *mdy*，而消息是以 us_english 显示的。有的语言缺省 （包括 including us_english）产生 Sunday=1、 Monday=2，等等；其它一些产生 Monday=1、 Tuesday=2，等等。

    set language 暗指 Adaptive Server 应该使用所指定语言的第一种星期和日期格式，但不会覆盖当前会话的早些时候发出的显式 set datefirst 或 set dateformat 命令。
  - cursor rows 和 close on endtran 影响 Adaptive Server 处理游标的方式。所有游标的 cursor rows 的缺省设置都是 1。 close on endtran 的缺省设置是 off。
  - chained 和 transaction isolation level 允许 Adaptive Server 以符合 SQL 标准的方式处理事务。

对某些 set 参数的编译期更改

在使用抽象计划来创建存储过程或在 Transact-SQL 批处理中运行这些抽象计划时，Adaptive Server 15.0.2 版及更高版本更改了其中部分 set 参数的编译期行为。

在 Adaptive Server 的早期版本中，set 参数在执行或重新编译存储过程后生效。Adaptive Server 15.0.2 允许您在编译时使用优化程序 set 参数来影响存储过程或批处理中的优化程序。

注释 这一更改的行为可能会影响结果集的组成。Sybase 建议您在生产系统中使用 15.0.2 版的 set 参数之前，先查看这些参数创建的结果集。

在从存储过程返回之前，必须先重置 set 参数，否则，后续存储过程的执行可能会受到影响。如果计划将此更改传播给后续存储过程，请使用 export_options 参数。

表 1-31 显示使用 set export_options on 时可导出的优化程序选项。

表 1-31: 可使用set export_options on 导出的优化程序选项

optgoal	store_index	showmanagers
opttimeout	bushy_space_search	showlogprops
merge_join	parallel_query	showparallel
hash_join	replicated_partitioning	showhistograms
nl_join	basic_optimization	showabstractplan
distinct_sorted	index_intersection	showsearchengine
distinct_sorting	index_union	showcounters
distinct_hashing	multi_gt_store_index	showbestplan
group_sorted	opportunistic_grouping	showfinalplan
group_hashing	opportunistic_distinct	showcodegen
group_inserting	auto_query_tuning	showpiocosting
order_sorting	streaming_sort	showliocosting
addend_union_all	nary_nl_join	showelimination
merge_union_all	query_tuning_mem_limit	showpllcosting
merge_union_distinct	query_tuning_time_limit	shownostats
hash_union_distinct	showlop	showexecio

集合行为

ansinull 决定聚合函数中 NULL 值操作数的求值是否符合 ANSI SQL 标准。如果使用 set ansinull on，则当聚合函数从计算中消除 NULL 值操作数时， Adaptive Server 会生成警告消息。

例如，如果设置为 set ansinull off （缺省值），并对 titles 表执行下面的查询：

```
select avg (total_sales) from titles
```

Adaptive Server 返回：

```
-----------
       6090
```

但是，如果在设置为 set ansinull on 的情况下执行相同的查询， Adaptive Server 会返回以下结果：

```
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)
```

该消息表示 total_sales 中的某些条目包含 NULL 值，没有实际的数量，因此您无法得到该表中所有书籍的总销售额的全部数据。然而，返回值在获得的数据中是最高的。

比较行为

SQL 标准要求：如果一个等同性比较的两个操作数中有一个为 NULL 值，则结果为 UNKNOWN。 Transact-SQL 对 NULL 值的处理是有区别的。如果一个操作数是列、参数或变量，而另一个操作数是 NULL 常量，或者是其值为 NULL 的参数或变量，则结果是 TRUE 或 FALSE：

- Sybase NULL 模式 -- “`val = NULL`”在 “`val`”为 NULL 时为 true
- ANSI NULL 模式 -- “`val = NULL`”在 “`val`”为 NULL 时为 unknown

ANSI 规则的 where 和 on 子句返回值为 true 的行，拒绝值为 false 和 unknown 的行。

ANSI 规则的 check 约束拒绝 false 值。因此，不会拒绝为 unknown 或 true 的结果。

如果：

- 启用 ansinull 模式 – 不要使用 Sybase NULL 比较 （`val = NULL` 或 `val != NULL`）。

- 希望在 insert 和 update 时使用 ANSI-null 模式 – 不要在 check 约束中使用 Sybase NULL 比较。

而应该使用 ANSI IS NULL 或 IS NOT NULL 语法以避免生成预料之外的结果。

分隔标识符

在 quoted_identifier 选项设置为 on 时，如果语句的语法要求引用的字符串包含一个标识符，则不需要用双引号将标识符括起来。例如：

```
set quoted_identifier on
create table "1one" (c1 int)
```

然而， object_id 需要一个字符串，因此您必须将表名用引号引起来以选择信息：

```
select object_id('1one')
-----------------------
  896003192
```

您可以通过使引号加倍出现，从而在带引号的标识符中包括一个嵌入的双引号：

```
create table "embedded""quote" (c1 int)
```

但是，在语句语法要求对象名表示为字符串时，无需使引号加倍出现：

```
select object_id('embedded"quote')
```

带中括号的标识符 Adaptive Server 支持一种可替代带引号的标识符的方法，即使用中括号括起标识符。带中括号的标识符的行为与带引号的标识符的行为相同，不同之处在于无需借助 set quoted_identifier on 就可使用带中括号的标识符。

在使用带中括号的标识符代替带引号的标识符来创建对象时，对象名应至少包含一个有效字符，例如：

```
create table [table name]

create database [database name]
```

对象名中的所有尾随空格都被删除，因此以下语句均视为是相同的：

```
[tab1<space><space>]
[tab1<space><space>]
[tab1]
[tab1<space><space><space>]
tab1
```

这项规则适用于可使用带中括号的标识符创建的所有对象。

下面列出了在 Adaptive Server 中使用分隔标识符的限制：

- 标识符名称中都不能包含圆点 (.)
- 作为存储过程参数的对象名 – Adaptive Server 存储过程对象名可视为字符串，因此不需要分隔符。例如，下面给出了实际存在名为 table 的表时的正确结果：

  ```
  exec sp_help 'dbo.table'
  ```

  但在下面的语句中，对象名中的中括号没有去掉：

  ```
  exec sp_help 'dbo.[table]'
  ```

角色与 *set* 选项

- 您登录到 Adaptive Server 时，所有授予您的系统定义角色都会自动激活。授予您的用户定义角色不会自动激活。要自动激活授予您的用户定义角色，请使用 sp_modifylogin。请参见 《参考手册：过程》 中的 sp_modifylogin。请使用 set role *role_name* on 或 set role *role_name* off 打开或关闭角色。

  例如，如果授予了您系统管理员角色，您就可使用当前数据库中的数据库所有者标识 （或用户 ID）。若要使用您真正的用户 ID，请执行以下命令：

  ```
  set role "sa_role" off
  ```

  如果您不是当前数据库中的用户，而且没有 “guest” 用户，则无法设置 sa_role off。
- 如果要激活的用户定义角色带有口令，则必须指定此口令才能打开角色。因此，您应输入：

  ```
  set role "role_name" with passwd "password" on
  ```
- 在 set role 期间，如果失败的角色激活尝试次数达到 max failed_logins 中设置的数值， Adaptive Server 会锁定该角色。发生这种情况时，locksuid、 locdate 和 lockreason 会在 syssrvroles 中更新。

内存数据库和宽松持久性数据库

- 将日志记录级别设置为 minimal 只会影响当前用户拥有的对象的日志记录模式。但是，如果用户拥有系统管理员特权，则将日志记录设置为 minimal 会影响用户会话中所有对象的日志记录模式。
- 将最低限度记录复制到使用 select into 创建的表中的数据。 with dml_logging = minimal 指定以后对此表执行的 DML 操作的记录模式
- show_exec_info 不会显示 Adaptive Server 覆盖用户选择的最少记录模式的原因。可使用 set switch print_minlogged_mode_override 查看此覆盖原因。

- 特定于会话的日志记录模式设置将覆盖在表和数据库级别设置的日志记录选项，并包括以下限制：
  - 数据库范围的设置
  - 基于日志记录模式禁用当前会话的 DML 日志记录：
    - 数据库范围的日志记录模式设置
    - 特定于表的日志记录模式设置
    - 更新的表的所有权
- 如果将特定于会话的 DML 记录设置为 minimal，则运行 set dml_logging default 将会根据表和数据库范围的设置，使受影响的表的日志记录模式恢复为其缺省日志记录模式。
- 如果数据库所有者或表所有者已经将表配置为以最少日志记录模式运行，则不能使用 set dml_logging 执行完全记录 DML。

针对会话设置压缩

- 对会话启用或禁用压缩不会更改现有数据的压缩级别。
- 更新的行会保持不压缩。 Adaptive Server 会在更新期间将所有压缩行解压缩。
- 如果 set compression 设置为 off，则需要数据复制的命令 （例如，reorg rebuild 和 alter table）就会将数据行解压缩。
- 配置 set compression on 后，后续的更新会使用分区或表的压缩级别。

使用谓词特权

- 使用 set show_transformed_sql 并使 set noexec 运行但并不执行，查询可显示 SQL 文本。
- 为会话启用 show_transformed_sql 时，DML 或 select 命令会显示以下各项的 SQL 文本：
  - 谓词文本 （如果 SQL 命令中的表上存在谓词）。如果不存在谓词， show_transformed_sql 显示 NULL。
  - 用户查询文本
  - 执行集合处理、视图处理、加密和谓词合并后查询的 SQL 文本。对于不访问任何谓词的查询， SQL 文本表示在执行视图处理、加密等操作之后的文本。
  - 执行子查询处理后查询的 SQL 文本。

分布式事务、 CIS 和 *set* 选项

- cis rpc handling 配置属性和 set transactional_rpc 命令的行为已随着 ASTC 的引入而发生了改变。在 12.0 以前的版本中，启用 cis rpc handling 会通过 CIS 的 Client-Library 连接来传递*所有* RPC。这样，只要启用 cis rpc handling，无论是否明确设置 transactional_rpc，其行为都会发生。对于 Adaptive Server 12.0，此行为已经更改。如果已启用 cis rpc handling 且 transactional_rpc 为 off，事务内的 RPC 将通过节点处理器来传递。在事务外执行的 RPC 则通过 CIS 的 Client-Library 连接来发送。
- 启用 Adaptive Server 分布式事务管理服务时，可以将 RPC 放入事务中。这些 RPC 称为**事务性 RPC**。事务性 RPC 是其工作可以包含在当前事务中的 RPC。远程工作单元可以和本地事务执行的工作一起提交或回退。

  若要使用事务性 RPC，请使用 sp_configure 启用 CIS 和分布式事务管理，然后发出 set transactional_rpc 命令。当 set transactional_rpc 为 on 且事务正等待处理时， Adaptive Server （而不是 Adaptive Server 节点处理器）会协调 RPC。

  set transactional_rpc 命令缺省为 off。 set cis_rpc_handling 命令会覆盖 set transactional_rpc 命令。如果您设置 cis_rpc_handling on，则所有外发 RPC 都由 CIS 处理。
- 有关使用 set transactional_rpc、 set cis_rpc_handling 和 sp_configure 的讨论，请参见 《组件集成服务用户指南》。

使用代理

注释　不具有显式权限时，“sa_role”和 “sso_role”都不能发出 set proxy login_name 命令。若要使用 set proxy login_name，所有用户 （包括系统安全员）都必须具有系统安全员显式授予的权限。

- 您必须具有 master 数据库的 set proxy 特权或 set session authorization 特权，才能使用 set proxy 或 set session authorization 命令。
- 可以使用以下命令将服务器用户标识切换成任何其它的服务器登录名，并根据目标登录角色来限制其使用：

  ```
  grant set proxy to user_or_role_list
  [restrict role role_list | all | system]
  ```

  有关详细信息，请参见第 401 页的 “grant”。
- 以初始 *login_name* 执行 set proxy 或 set session authorization 会恢复先前的标识。

- 不得在事务内执行 set proxy 或 set session authorization。
- Adaptive Server 仅允许一个级别的登录标识更改。因此，使用 set proxy 或 set session authorization 更改标识后，在再次更改前必须恢复为初始标识。例如，假定您的登录名为 “ralph”。若要以 “mary” 标识创建表，以 “joe” 标识创建视图，然后返回到您自己的登录标识，请使用以下语句：

```
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"
```

- 如果用户发出 set proxy 来取得另一个用户的权限、登录名和 suid，则 Adaptive Server 会检查代理用户对数据库对象的访问权，而不是原始用户的访问权。

  Adaptive Server 使用已登录用户的名称和口令信息，通过使用登录凭据检查对加密密钥的自动访问。 Adaptive Server 不具有对代理用户口令的访问权。通过登录口令对密钥的访问代表登录的用户，而不是代表通过别名、 set proxy 或 setuser 假定的用户。如果加密密钥的副本是为登录关联而设置，但仍然由系统加密口令或主密钥进行加密，则对这些加密密钥副本的访问会进行类似处理。

使用 *lock wait*

- 缺省情况下，不能立即获取锁的 Adaptive Server 任务会等待，直到不兼容锁被释放，然后继续处理。这等同于未在 *numsecs* 参数中指定值的 set lock wait 。
- 可以使用带 lock wait period 选项的 sp_configure 来设置服务器范围的锁等待时间。
- lock wait period （会话级设置为 set lock wait nnn）仅适用于用户定义的表。这些设置对系统表毫无影响。
- 在会话级或在存储过程中使用 set lock 命令定义的锁等待时间会覆盖服务器级别的锁等待时间。

- 如果 set lock wait 为其自身使用，且未为 *numsecs* 设置值，则当前会话中的所有后续命令会无限期等待直到获取请求的锁。
- sp_sysmon 报告等待锁的任务在等待期间未能获取锁的次数。

可重复读取的事务隔离级别

- 可重复读取的隔离级别 （也称为事务隔离级别 2）会持有语句读取的所有页上的锁，直到事务完成为止。
- 当一个事务从表中读取行，而另一个事务可以修改相同的行，并在第一个事务完成前提交更改时，会发生不可重复读取。如果第一个事务重新读取这些行，这些行就具有了不同的值，因此初始读取是不可重复的。可重复读取会在事务期间持有共享锁，阻止更新锁定行或更新锁定页上的行的事务。

使用模拟统计信息

可以使用 optdiag 实用程序的 simulate 模式将模拟统计信息装载到数据库中。如果在会话中发出了 set statistics simulate on，则查询会使用模拟统计信息 （而不是使用表的实际统计信息）进行优化。

受 *set* 选项影响的全局变量

表 1-32 列出了包含有关 set 命令控制的会话选项的信息的全局变量。

表 1-32: 包含会话选项的全局变量

全局变量	说明
@@char_convert	如果字符集转换无效，则包含 0。如果字符集转换有效，则包含 1。
@@client_csexpansion	返回从服务器字符集转换为客户端字符集时使用的扩展因子。例如，如果 *@@client_csexpansion* 包含一个值 2，则服务器字符集中的字符在转换为客户端字符集后最多可以占用原来字节数的两倍。
@@cursor_rows	为可滚动游标专门设计的全局变量。该变量显示游标结果集中的总行数。返回值 -1：
@@datefirst	使用 set datefirst n 进行设置，其中， n 是介于 1 和 7 之间的值。返回 *@@datefirst* 的当前值，它指示为每个星期指定的第一天，表示为 tinyint。 在 Adaptive Server 中，缺省值为星期日 （基于 us_language 缺省值），您可以通过指定 `set datefirst 7` 来进行设置。有关这些设置和值的详细信息，请参见 set 命令的 datefirst 选项。
@@isolation	包含 Transact-SQL 程序的当前隔离级别。 *@@isolation* 采用活动级别的值 （0、 1 或 3）。
@@lock_timeout	使用 set lock wait n 进行设置。返回当前的 *lock_timeout* 设置，单位为毫秒。*@@lock_timeout* 返回值 n。缺省值为无超时。如果在会话开始时没有执行 set lock wait n， *@@lock_timeout* 将返回 -1。
@@options	包含以十六进制形式表示的会话的 set 选项。
@@parallel_degree	包含当前的最大并行度设置。

全局变量	说明
@@rowcount	包含最后的查询所影响的行数。任何不返回行的命令 （如 if、 update 或 delete 语句）都会将 *@@rowcount* 设置为 0。对于游标，*@@rowcount* 表示从游标结果集返回到客户端 （直到最后一个 fetch 命令）的累积行数。 *@@rowcount* 即使在 nocount 设置为 on 时也可更新。
@@scan_parallel_degree	包含非聚簇索引扫描的当前最大并行度设置。
@@textsize	包含 select所返回的 text、 unitext 或 image 数据的字节数的限制。 isql 的缺省限制为 32KB；缺省值取决于客户端软件。它可以用 set textsize 为某个会话进行更改。 如果使用 enable surrogate processing，则 Unicode 代理对 （两个 16 位值）将作为单个字符返回，即使实际返回大小可能小于 *@@text* 大小值也是如此。
@@tranchained	包含 Transact-SQL 程序的当前事务模式。*@@tranchained* 为非链接模式则返回 0，为链式模式则返回 1。

表 1-33 列出了用于 *@@options* 的 set 选项和值。

表 1-33: @@options 的 set 选项和值

数字值	十六进制值	*set* 选项
4	0x04	showplan
5	0x05	noexec
6	0x06	arithignore
8	0x08	arithabort
13	0x0D	control
14	0x0E	offsets
15	0x0F	statistics io 和 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

对数据库中的 Java 使用 *fipsflagger*

- 当 fipsflagger 为 on 时，如果使用下面的扩展， Adaptive Server 会显示一条警告消息：
  - installjava 实用程序
  - remove java 命令
  - 引用 Java 类作为数据类型的列和变量声明

    - 将 Java-SQL 表达式用于成员引用的语句
- fipsflagger 的状态不影响 Java 方法执行的算术表达式。
- 有关数据库中的 Java 的详细信息，请参见 *Adaptive Server Enterprise 中的 Java*。

使用 set tracefile 的限制

- 不能保存系统任务 （如管家或端口管理器）的 SQL 文本。
- 必须具有 sa 或 sso 角色，或被授予 set tracing 权限，才能运行启用或禁用跟踪。
- 不允许 set tracefile 将现有文件打开以作为跟踪文件。
- 在 SA 或系统安全员会话期间，如果为指定 spid 启用 set tracfile，则后续执行的所有跟踪命令都将对该 spid 而不是对系统管理员或系统安全员 spid 生效。
- 如果 Adaptive Server 在写入跟踪文件时文件空间不足，它将关闭文件并禁用跟踪。
- 如果 isql 会话为 spid 启动跟踪，但 isql 会话在未禁用跟踪的情况下退出，则另一个 isql 会话可以开始跟踪此 spid。
- 跟踪只在为其启用跟踪的会话中进行，而不在启用跟踪的会话中进行。
- 不能从一个 sa 或 sso 会话中同时跟踪多个会话。如果试图为已打开一个跟踪文件的会话再打开一个跟踪文件， Adaptive Server 将发出错误消息：`tracefile is already open for this session.`
- 不能从多个 sa 或 sso 会话中跟踪同一会话。
- 当被跟踪的会话退出或禁用跟踪时，存储跟踪输出的文件会关闭。
- 在为跟踪分配资源之前，请记住，所有跟踪都要求每个引擎具有一个文件描述符。

设置将诊断信息保存到跟踪文件的选项

可将 set tracefile 与其它可提供诊断信息以便更好地了解慢速查询的 set 命令和选项一起使用。以下是用于将诊断信息保存到文件的 set 命令和选项：

- set show_sqltext [on | off]
- set showplan [on | off]
- set statistics io [on | off]
- set statistics time [on | off]

- set statistics plancost [on | off]

以下是 set 选项：

- 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_lio_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]

对 show_sqltext 的限制

- 必须具有 sa 或 sso 角色才能运行 show_sqltext。
- 不能使用 show_sqltext 为触发器输出 SQL 文本。
- 不能使用 show_sqltext 显示绑定变量或视图名。

从登录触发器导出 *set* 选项

Adaptive Server 可以使登录触发器内部的 set 选项对整个用户会话保持有效。将自动导出以下 set 选项：

- 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

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

ANSI SQL 标准指定与低于 15.7 的 Adaptive Server 版本中的 Transact-SQL 行为不同的行为。缺省情况下，会为所有 Embedded-SQL 预编译应用程序启用一致行为。其它需要匹配此标准行为的应用程序可以使用以下 set 选项：

表 1-34: 符合初级 ANSI SQL 的 set 选项

选项	设置
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

权限

权限检查可能因您的细化权限设置而异。通常， set 权限缺省授予所有用户，使用它不需要特殊的权限。但 set identity_insert、 set identity_update、 set option show_option、 set plan for show、 set proxy、 set repthreshold、set role、set session authorization、set tracefile 和 set switch 例外。有关每个例外情况的权限要求，请参见上文的命令说明。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
88	security	set proxy 或 set session authorization	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *以前值* - 以前的 suid • *当前值* - 新的 suid • *其它信息* - NULL • *代理信息* - 当 set proxy 或 set session authorization 无参数时为初始登录名；否则为 NULL

另请参见

命令 create trigger、 fetch、 grant、 insert、 lock table、 revoke、 set.

函数 convert.

实用程序 isql、 optdiag.

存储过程 sp_setrepdbmode、 sp_setrepdefmode.

setuser

说明

允许数据库所有者充当其他用户。

语法

setuser ["*user_name*"]

示例

数据库所有者在数据库中临时使用 Mary 的标识来为 Joe 授予 authors（Mary 拥有的表）的权限：

```
setuser "mary"
go
grant select on authors to joe
setuser
go
```

用法

- 数据库所有者使用 setuser 来采用其他用户的标识，从而可以使用其他用户的数据库对象、授予权限、创建对象或用于其它目的。
- 除了登录帐户 “sa”运行的会话以外，当数据库所有者使用 setuser 命令时， Adaptive Server 将检查被模拟的用户的权限，而并非数据库所有者的权限。被模拟的用户必须在数据库的 sysusers 表中列出。
- setuser 仅能影响本地数据库中的权限。它不会影响远程过程调用或其它数据库中的访问对象。
- setuser 会一直有效，直到发出另一个 setuser 命令，或使用 use 命令更改当前数据库为止。
- 创建数据库时， setuser 无效。
- 如果执行 setuser 时不使用用户名，则可恢复数据库所有者的初始标识。
- 系统管理员可以使用 setuser 创建将由另一用户拥有的对象。不过，因为系统管理员是在系统管理员权限系统之外进行操作，他或她就不能使用 setuser 获取其他用户的权限。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 setuser 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须具有 setuser 特权才能运行 setuser。缺省情况下，会为数据库所有者授予 setuser 特权。
细化权限已禁用	如果禁用细化权限， setuser 特权缺省情况下授予数据库所有者，并且不能移交。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
84	setuser	setuser	• *角色* – 当前活动角色 • *关键字或选项* – NULL • *先前值* – NULL • *当前值* – NULL • *其它信息* – 设置的用户的名称 • *代理信息* – set proxy 有效时的初始登录名

另请参见 **命令** grant, revoke, use.

shutdown

说明

关闭发出 shutdown 命令的 Adaptive Server、其本地 Backup Server 或远程 Backup Server。

语法

shutdown [*srvname*] [with {wait [="*hh:mm:ss*"] | nowait}]]

用于集群的语法：

shutdown {cluster | [*instance_name*]} [with {wait | nowait}]

参数

srvname

是 Backup Server 在 Adaptive Server 的 sysservers 系统表中的逻辑名。关闭本地 Adaptive Server 时不需要此参数。

with wait

为缺省值。这会正常关闭 Adaptive Server 或 Backup Server。

hh:mm:ss

为可选设置，用于指定服务器等待所有运行或休眠进程完成其工作的最长时间。

with nowait

立即关闭 Adaptive Server 或 Backup Server，而不用等待当前正在执行的语句完成。

警告！ 仅在极端情况下使用 shutdown with nowait。在 Adaptive Server 中，请在执行 shutdown with nowait 之前发出 checkpoint 命令。使用 shutdown with nowait 可导致 IDENTITY 列值出现间隔。

---

cluster

（仅限共享磁盘集群） – 指定要关闭集群中的所有实例。

instance_name

（仅限共享磁盘集群） – 是要关闭的特定实例的名称。

示例

示例 1 关闭发出 shutdown 命令的 Adaptive Server：

```
shutdown
```

示例 2 立即关闭 Adaptive Server：

```
shutdown with nowait
```

示例 3 关闭本地 Backup Server：

```
shutdown SYB_BACKUP
```

示例 4 关闭远程 Backup Server REM_BACKUP：

```
shutdown REM_BACKUP
```

示例 5 关闭当前集群：

```
shutdown cluster
```

示例 6 关闭实例 “ase1”，但保持集群为运行状态：

```
shutdown ase1
```

用法

- 除非使用 nowait 选项，否则 shutdown 会试图以下列方式正常关闭 Adaptive Server：
  - 禁用登录 （系统管理员除外）
  - 在每个数据库中执行检查点
  - 等待当前正在执行的 SQL 语句或存储过程结束

  注释 在数据服务器上执行正常关闭不会终止包含无限循环的 SQL 批处理，以及包含 waitfor delay 命令的 Transact-SQL。

  关闭服务器而不使用 nowait 选项，可将自动恢复过程必须做的工作量降至最少。

  正常关闭会等待当前执行的语句完成，假定条件是许多语句是在完成时会执行提交操作的原子事务。但是，如果会话将在事务中进入 “awaiting command” 状态，则正常关闭不会等待较长事务进行提交。而是会执行关机，并且事务会在恢复时回退。

  Sybase 建议您对所有长期运行的打开事务检查 master..syslogshold，并在发出 shutdown 命令之前对其进行处理。您可以通过以下方式处理此类事务：联系运行此类事务的用户，看看他们能否可以提交这些事务或终止它们并等待回退。在关闭之前回退活动事务要比恢复后回退速度更快，因为许多页面可能仍然处于缓存中，因此涉及到的日志扫描会比较少。如果存在多个长期运行的事务，请反复检查 syslogshold，直到您不再看到非常旧的事务为止。

  注释 当服务器上存在长期运行的打开事务时，执行关闭可能会导致恢复时间变得很长。

- 除非使用 nowait 选项，否则 shutdown *backup_server* 会等待活动转储和装载完成。一旦向 Backup Server 发出 shutdown 命令，就不能启动使用此 Backup Server 的新的转储或装载。
- 使用 shutdown 仅能中断本地 Adaptive Server；不能中断远程 Adaptive Server。
- 仅当出现下列情况时才能中断 Backup Server：

- 它在 sysservers 表中列出。使用 sp_addserver 向 sysservers 添加条目。
- 它在执行命令的 Adaptive Server 的 interfaces 文件中列出。

- 使用 sp_helpserver 确定 Backup Server 在 Adaptive Server 中的名称。将 Backup Server 的 name （而非其 *network_name*）指定为 *srvname* 参数。例如：

```
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
```

若要关闭名为 WHALE_BACKUP 的远程 Backup Server，请使用：

```
shutdown REM_BACKUP
```

指定等待时间

服务器准备关闭时，它将：

1　对所有数据库执行 checkpoint

2　阻止任何新用户登录

3　等待所有运行或休眠进程完成其作业

4　再次对数据库执行checkpoint，这次带有通知您需要刷新以下内容的标志：

- 混合日志数据数据库中的所有动态阈值
- 所有对象统计信息
- 用以避免恢复后产生缺陷的标识字段的值

如果在使用 with wait 时带上 *hh:mm:ss* 选项，则指定的时间不是 Adaptive Server 关闭其自身所花费的最长总时间。 Adaptive Server 会计算第一次执行checkpoint所花费的时间，并自动从指定的时间中减去此时间。

例如，如果指定的最长等待时间为 20 分钟，第一次检查点操作时间为 3 分钟，则 Adaptive Server 允许过程完成的时间为 17 分钟。但是，如果由于某种原因第二次执行checkpoint的时间较长，则不将此时间计算到指定的 with wait *hh:mm:ss* 参数中。

Adaptive Server 也允许checkpoint花费的时间超过 with wait *hh:mm:ss* 中指定的时间。例如，如果指定的等待时间为 10 分钟，而第一次checkpoint需要 20 分钟才能完成，则 Adaptive Server 不会中途中断checkpoint，而是等待checkpoint完成。当出现这种情况时， Adaptive Server 将在经过指定的时间段并完成checkpoint后立即开始关闭，并运行最后一次checkpoint （带有用于通知必须执行的刷新的标志）。

在集群环境中关闭

- 不带任何选项的 shutdown 命令在集群环境中无效，例如：

```
shutdown
go
```

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 shutdown 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下： • 要关闭 SMP 服务器、集群实例或备份服务器，您必须是具有 shutdown 特权的用户。 • 要关闭集群，您必须是具有 shutdown 特权和 manage cluster 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
51	security	服务器关闭	• *角色* - 当前活动角色 • *关键字或选项* - shutdown • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter database.

系统过程 sp_addserver, sp_helpserver.

transfer table

说明

启动增量表传输。

语法

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}]

参数

table

Adaptive Server 中的任何有效表。transfer table 需要 sa_role 或表的所有权。表所有者可以授予自己拥有的表的 transfer table 权限。

to | from

指示传输方向。不能对 transfer table...from 使用用于 transfer table...to 的所有参数。 from 参数的参数是：

- column_order=*option* （不适用于使用 for ase 进行的装载；保留以备后用）
- column_separator=*string* （不适用于使用 for ase 进行的装载；保留以备后用）
- encryption={true | false}（不适用于使用 for ase 进行的装载；保留以备后用）
- progress=*nnn*
- row_separator=*string* （不适用于使用 for ase 进行的装载；保留以备后用）

destination_file

对操作系统有效的、 Adaptive Server 可以访问的任何文件或路径名。如果文件是相对文件路径，则 Adaptive Server 提供其绝对路径。如果 Adaptive Server 无法打开 *destination_file*，它将发出错误消息。

如果满足以下所有条件， Adaptive Server 将删除目标文件：

- 文件是常规文件，不是命名管道、符号链接或其它特殊文件。
- Adaptive Server 在此传输过程中打开文件。
- 传输失败，或没有发送任何行。

for 子句

指定目标数据格式之一。如果省略 for 子句，则指定表的第一次传输的缺省值为 for ase。后续传输的缺省值为以前的成功传输使用的格式，除非命令指定格式 with resend = id，在这种情况下，缺省值是以前指定的传输格式。

- ase – 用于将数据导入 Adaptive Server 的格式。此输出格式是许可的功能，可供 RAP 客户和具有内存数据库许可证的客户使用。不应用任何数据转换。此文件格式包括用于描述表的标头，其中包括源计算机的字节顺序、字符集和缺省排序顺序。这是以前没有成功传输的表的缺省值。
- bcp – 用于将数据作为 bcp 二进制格式数据导入的格式。将行作为可使用 bcp 装载的二进制数据输出。不应用任何数据转换。在传输过程中， Adaptive Server 创建一个 bcp 用于描述数据的格式文件，并与输出文件显示在同一目录中。

  不能使用 for bcp 传输命名管道。

  如果输出文件是命名管道以外的任何文件类型，则 Adaptive Server 对格式文件使用以下命名约定：

{*table_name*},{*database_id*},{*object_id*}.fmt

- iq – 采用适合使用 IQ 的 load table 命令装载到 Sybase IQ 中的格式写入数据。 Adaptive Server 采用二进制格式将数据写入到文件中，并应用任何必要的数据转换以将其数据类型转换为与 IQ 兼容的版本。除非包括 with fixed_length='true' 或 with null_byte='true' 修饰符，否则 for iq 会采用缺省格式写入数据：
  - 缺省格式 – 可为空的数据包括一个后续的 “空字节”，这个单字节标识符包含：
    - 0 （如果列非空）
    - 1 （如果列为空）

    不可为空的数据不包括此空字节 （请参见 IQ 文档的 load table）。可变长度的字符串前面有指示字符串长度的一个或两个字节，其中前缀字节数由列的最大长度决定：一个字节代表字符串最多有 255 个字节，而两个字节代表字符串有 256 个字节或更长（Adaptive Server 支持字符串最多有 16000 个字节左右）。除了字符串以外，将每个列作为固定宽度进行传输，如有必要，将进行填补以将其扩展到此固定宽度大小。
  - 使用以下修饰符确定数据格式：
    - with fixed_length='true' – 将所有列 （包括字符串）填补到其完整宽度。使用空白填补字符串；使用 <NUL> 或 0x00 填补其它列。没有任何列具有长度指示符
    - with null_byte='true' – 所有列都必须具有空字节，而不管列是否可为空。此修饰符强制 for iq 使用 fixed_length='true’ 修饰符，而不管命令指定的是什么
- csv – 字符编码值格式。将行作为字符编码数据输出。用指定的列分隔符分隔列，用指定的行终结符终结行。分隔符和终结符由用户定义。

with 子句
提供用于修改命令操作的选项。

column_separator = *string*
声明在 csv 格式的输出列之间写入的字符串，替换缺省值。为后续传输写入的字符串缺省为以前指定的 column_separator。

column_order = *option*

声明将列数据写入到输出的顺序。这些选项有：

- id – 按 syscolumns 中提供的列 ID 排序。这是传输为 for bcp 时唯一可接受的列顺序，并且是这些传输的缺省值。
- name – 使用 Adaptive Server 当前字符集和排序顺序，按 syscolumns 中提供的列名排序。
- name_utf8 – 按 syscolumns 中提供的列名排序，在排序之前将列名转换为 UTF8 字符。
- offset – 按数据行中的列偏移排序。这是传输为 for ase 时唯一可接受的列顺序，并且是这些传输的缺省值。

如果使用的 column_order 与 for 子句格式不匹配，则 Adaptive Server 会发出错误消息。列顺序有：

- for ase – 使用 offset 列顺序
- for bcp – 使用 id 列顺序

encryption = *option*

指定命令如何处理加密列。选项有：

- true – 在传输前解密列。该值为缺省值。用户必须有权解密任何加密列。
- false – 完全按照加密列在数据行中的显示传输加密列。

注释 若要恢复数据，接收方必须知道加密密钥和用于加密数据的算法。如果 Adaptive Server 将加密数据写入文件，则会按照数据第一次存储在表中时的加密方式写入数据。传输数据不会更改该方式。若要恢复该数据，接收方必须知道加密数据的密钥和加密算法的任何特殊功能（例如，是否使用初始化矢量）。

---

progress = *sss*

指示传输应在操作期间每 *sss* 秒生成一次进度消息。缺省设置为省略进度消息。

row_separator = *string*

声明要在 csv 格式的每个输出行末尾写入的字符串，替换缺省值。此选项无效，除非传输为 for csv。与 column_separator 一样，第一次之后的 csv 模式的所有传输的缺省值为最近成功的传输的缺省值。缺省行分隔符因平台而异：在 Linux 和 UNIX 上为换行符 (Ctrl+J)，在 Windows 上为回车和换行符 (Ctrl+M Ctrl+J)。

resend =*id*

标识其序列 ID 列获取此数据传输的开始时间戳的表的历史记录条目。此选项重新发送以前发送的数据。除非命令中指定的表标记为增量传输，否则将忽略 resend =*id*。如果此表中不存在指定的 sequence ID，则 Adaptive Server 将重新发送整个表。

Adaptive Server 选择指示的条目的开始时间戳作为此传输的开始时间戳，并选择指示的条目的目标类型 （ase、 bcp 等）作为传输的缺省目标类型。

id 的负值会检索指定表的以前成功完成的传输的历史记录条目。- 1 指定最近成功完成的传输，- 2 指定下一个最近的，依次类推。传输历史记录表同时存储成功的和失败的传输的条目。

tracking_id =*nnn*

指定帮助跟踪给定传输的可选整数标识符。使用 spt_TableTransfer.tracking_id 列确定 *nnn* 的值并在查询中使用该值。此示例返回跟踪 ID 号 123 的编辑状态和序列 ID，以及输出数据文件的完整路径 （如果这些值不存在，则返回 NULL）：

```
select end_code, sequence_id, pathname from spt_TableTransfer
where id = object_id('mytable') and tracking_id = 123
```

Adaptive Server 不控制 tracking_id 或要求它是唯一的。

注释　此跟踪 ID 不是用于 resend =*id* 的序列 ID。

sync = true | false

确定传输如何与事务交互。选项有：

- true – 将同步传输，以便将传输中包括的表中的行作为组进行捕获。 transfer 等待影响此表的所有事务结束后才开始。在 transfer 等待开始期间，影响此表的新事务将无法修改此表。它们将等待 transfer 开始。在 transfer 进行过程中，事务在经过 transfer 检查之前无法修改此表中的行。
- false – 不同步传输。 transfer 发送所选时间戳范围内的行，而不管是否可以发送表中的其它行。这是缺省行为。

注释 sync 只影响要传输的表。 transfer 不考虑跨表约束。

fixed_length = true | false

确定 transfer...for iq 是否传输输出文件中作为固定长度字段的所有列。通常， Adaptive Server 传输具有 1 或 2 个字节前缀长度的可变长度字符串。将 fixed_length 设置为 true 会导致 Adaptive Server 使用空白填补字符串，直到它们到达列的最大宽度。必须将此参数与 for iq 参数结合使用。将 fixed_length 设置为：

- true – Adaptive Server 将字符串填补到其完整宽度，而不是使用前缀长度。
- false – Adaptive Server 使用缺省行为发送字符串，即，使用前缀长度发送字符串。

null_byte = true | false

确定 transfer...for iq 是否在每个传输列末尾附加一个字节，以指示列是否为空。通常， Adaptive Server 只为可为空的列提供此字节。选项有：

- true – Adaptive Server 在所有列末尾包括一个空字节，如果列为空，则为 0，否则为 1，而不管列是否可为空。true 强制 for iq 使用 fixed_length='true’ 修饰符，而不管您使用 transfer 命令指定的是什么。
- false – Adaptive Server 只为可为空的列提供空字节。

注释 注意：不管 null_byte 是设置为 true 还是 false，它只适用于包括 for iq 子句的传输。

示例

示例 1 为用户 “john” 授予传输表 mytable 的权限：

```
grant transfer table on mytable to john
```

示例 2 将 mytable 传输到输出文件，其格式设置为用于装载到 Sybase IQ。如果此示例未包括 name_utf8，则缺省顺序将为列 ID 顺序：

```
transfer table mytable to '/path/to/file' for iq
with column_order = 'name_utf8'
```

示例 3 传输 mytable，其格式设置为 Adaptive Server 文件格式，使用 offset 的列输出顺序。此示例请求从不存在的历史记录条目执行 resend；因此，将传输整个表：

```
transfer table mytable to '/path/to/file3/'for ase
with resend=10
```

此示例更改 for ase 传输的缺省列顺序。传输后，缺省接收方为 ase，列顺序为偏移，列和行分隔符为空。

用法

- transfer table 只发送上一次传输后更改的提交数据。
- with 子句中的 column_separator 和 row_separator 的 *string* 参数最多可为 64 个字节长，并且可以包含格式设置指令：
  - “\b”指示退格 <BS> (Ctrl+H)。
  - “\n”指示换行符 <LF> (Ctrl+J)。
  - “\r”指示回车 <CR> (Ctrl+M)。
  - “\t”指示 <TAB> (Ctrl+I)。
  - “\\”指示反斜杠。
  - 字符串中不属于这些序列之一的任何 “\” 是实际的反斜杠，并在字符串中显示为反斜杠。
- transfer table .. from 不会在更新或插入期间引发触发器。
- 当 transfer table 在运行中发生错误（如重复键）时，Adaptive Server 会仅显示错误号，很难了解错误原因。例如：

```
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.
```

若要检索错误消息，可手动查询 master..sysmessages。例如，如果 2601 是错误号，请输入：

```
select * from master..sysmessages where error = 2601
```

有关错误 2601 的详细信息，请参见 《故障排除指南》。

传输未标记为增量传输的表

可以对未标记为增量传输的表使用 transfer table，并具有以下限制：

- 并非总是传输所有行。如果用户在传输进行过程中更新表，则可能不会传输更新的行。
- 传输不是增量的；您只能传输整个表，并且不会向以后的传输通知此传输。
- 不会向 spt_TableTransfer 写入任何历史记录条目。在传输持续期间，传输显示在 monTableTransfer 中，但一旦传输完成，记录便消失。

权限

传输表的权限不会自动授予解密该表中数据的权限。若要解密任何加密列，必须从表所有者获得特定权限。

下文说明了基于您的细化权限设置的 transfer table 的权限检查。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或是对表拥有 tranfer table 权限的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是表所有者、具有 transfer table 权限的用户或具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
136	transfer table	transfer table	

另请参见

命令 create table、 alter table.

truncate lob

说明　将 LOB 截断成指定长度。

语法　truncate lob *locator_descriptor* [(*result_length*)]

参数

locator_descriptor

是有效的定位符：宿主变量、局部变量或定位符的实际二进制值。

result_length

是 text 和 unitext 定位符的字符数长度，以及 image 定位符的字节数长度。

示例　将文本定位符 *@w* 引用的 LOB 截断成 20 个字符：

```
truncate lob @w (20)
```

用法　如果未指定 *result_length* 或它为 0（零），Adaptive Server 会释放 LOB 内存，并将定位符指向空 LOB。

权限　任何用户都可以执行 truncate lob。

另请参见　**命令** deallocate locator.

Transact-SQL 函数 locator_valid、 return_lob、 create_locator.

truncate precomputed result set

说明

截断预计算结果集中的数据。

语法

truncate {precomputed result set | materialized view}
[*owner_name*.]*prs_name*

参数

precomputed result set | materialized view
指定是否截断物化视图或预计算结果集中的数据。

prs_name
预计算结果集的名称。完全限定的 *prs_name* 不得含有服务器或数据库名称。

示例

截断 authors_prs 预计算结果集：

```
truncate precomputed result set authors_prs
```

用法

truncate precomputed result set 会自动将预计算结果集更改为 disable。要重新启用预计算结果集，可发出 refresh precomputed result set 命令。

权限

只有预计算结果集的所有者才能运行 truncate precomputed result set。

truncate table

说明

从表或分区中删除所有行。

语法

truncate table [[*database*.]*owner*.]*table_name*
[partition *partition_name]*

参数

table_name

是要截断的表的名称。如果该表位于另一数据库中，请指定数据库名；如果数据库中有多个具有该名称的表，请指定所有者的名称。 *owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

partition_name

指定要截断的分区的名称。

示例

示例 1 删除 authors 表中的所有数据：

```
truncate table authors
```

示例 2 删除 titles 表的 smallsales 分区中的所有数据：

```
truncate table titles partition smallsales
```

用法

- truncate table 删除表中的所有行。表结构及所有索引仍然存在，直到发出 drop table 命令为止。绑定到列的规则、缺省值、约束继续保持绑定，而触发器也继续有效。
- Adaptive Server 不再使用分布页；统计信息现在存储在表 sysstatistics 和 systabstats 中。

  在 truncate table 运行期间，不再删除 （释放）统计信息，因此在添加数据后不必运行 update statistics 命令。

  truncate table 不删除表的统计信息。
- truncate table 等同于 （但快于）不含 where 子句的 delete 命令。delete 一次删除一行，并将每个删除的行作为一个事务记入日志；truncate table 释放整个数据页并制作较少的日志条目。 delete 和 truncate table 都可回收由数据及其相关索引所占用的空间。
- 截断某一分区不会影响其它分区中的数据。
- 一次只能截断一个分区。
- 截断表会锁定整个表，直到截断进程完成。
- 因为删除的行不是单独记入日志的，因此 truncate table 无法引发触发器。
- 如果其它表具有引用该表的行，则不能使用 truncate table。先删除外表的行，或截断外表，然后再截断主表。

- 可以使用 grant 授予用户或角色对表使用 truncate table 的权限，也可以使用 revoke 命令撤消这些权限。

标准

符合 ANSI SQL 的级别符合初级标准。

权限

对 truncate table 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是表所有者或是对表拥有 truncate table 权限的用户。要截断审计表，您必须具有 manage auditing 特权或 truncate any audit table 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须是表所有者、具有表的 truncate table 权限的用户、具有 replication_role 的用户或具有 sa_role 的用户。要截断审计表，您必须是具有 sso_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
64	truncate	truncate table	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 alter table, create table, create trigger, delete, drop table, grant, revoke.

union 运算符

说明

返回由两个或多个查询的结果组成的一个结果集。除非指定 all 关键字，否则重复行会从结果集中删除。

语法

```
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]
```

参数

top *unsigned_integer*

最多行数限制应用于组成联合的各个 select，而不应用于整个联合。

into

根据选择列表中指定的列以及在 where 子句中选择的行来创建新表。联合运算中的第一个查询是唯一能够包含 into 子句的查询。

union

创建由两个 select 语句指定的数据的联合。

all

包括结果中的所有行；不删除重复行。

示例

示例 1 结果集包含 sales 和 sales_east 表的 stor_id 和 stor_name 列中的内容：

```
select stor_id, stor_name from sales
union
select stor_id, stor_name from sales_east
```

示例 2 第一个查询中的 into 子句指定 results 表持有 publishers、 stores 和 stores_east 表的指定列的联合的最终结果集：

```
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
```

示例 3 首先，生成 sales 和 sales_east 表的指定列的 union。然后生成此结果和 publishers 的 union。最后，生成第二个结果和 authors 的 union：

```
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)
```

示例 4 返回六行。最多行数限制应用于组成联合的各个 select，而不应用于整个联合：

```
select top 3 au_lname from authors
union all
select top 3 title from titles
```

用法

- 在 union 运算符的一侧最多可以有 50 个子查询。
- 在 union 查询的所有各侧最多可以有 256 个表。
- 可以在 select 语句中使用 union，例如：

```
create view
select * from Jan1998Sales
union all
select * from Feb1998Sales
union all
```

- 只能在 union 语句的末尾使用 order by 和 compute 子句，以定义最终结果的顺序或计算摘要值。
- group by 和 having 子句仅能在单独的查询中使用，不能用于影响最终结果集。
- 包含 union 运算符的 SQL 语句的缺省计算顺序是从左到右。
- 因为 union 是二元运算，在表达式涉及两个以上查询时必须添加小括号以指定运算顺序。
- union 语句中的第一个查询可以包含一个 into 子句，该子句创建一个用于存放最终结果集的表。into 语句必须位于第一个查询中，否则您将收到错误消息 （见示例 2）。
- union 运算符可用于 insert...select 语句中。例如：

```
insert into sales.overall
  select * from sales
  union
  select * from sales_east
```

- SQL 语句中的所有选择列表必须具有相同数目的表达式 （列名、算术表达式、集合函数，等等）。例如，下面的语句无效，因为第一个选择列表中的表达式比第二个中的多：

```
/* 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 语句的选择列表中的相应列必须以相同顺序出现，因为 union 会以各个查询中给出的顺序一对一地比较列。
- union 生成的结果表的列名来自 union 语句中的*第一个*单独的查询。若要为结果集定义新的列标题，请在第一个查询中进行。另外，若要用新名称引用结果集中的一列（例如在 order by 语句中），应在第一个 select 语句中用该新名称进行引用。例如，下面的查询是正确的：

```
select Cities = city from stores
union
select city from stores_east
order by Cities
```

- 作为 union 操作的一部分的列的说明不必相同。表 1-35 列出了数据类型和结果表中相应列的规则。

表 1-35: union 运算中的结果数据类型

union 运算中列的数据类型	结果表中相应列的数据类型
数据类型不兼容（数据转换未由 Adaptive Server 隐式处理）	Adaptive Server 返回的错误。
二者均为固定长度字符，长度为 L1 和 L2	固定长度字符，长度等于 L1 和 L2 中之较大者。
二者均为固定长度二进制，长度为 L1 和 L2	固定长度二进制，长度等于 L1 个 L2 中之较大者。
其中之一或两者均为可变长度字符	可变长度字符，长度等于为联合中的列指定的长度中的最大值。
其中之一或两者均为可变长度二进制	可变长度二进制，长度等于为联合中的列指定的长度中的最大值。
二者均为数值数据类型（例如 smallint、 int、 float、 money）	数据类型等于两列的最大精度。例如，如果表 A 中的一列类型为 int，而在表 B 中的相应列的类型为 float，于是结果表的相应列的数据类型为 float，因为 float 比 int 更精确。
两个列说明均指定 NOT NULL	指定 NOT NULL。

限制

- 不能在子查询中使用 union 运算符。
- 不能将 union 运算符和 for browse 子句一起使用。

标准

符合 ANSI SQL 的级别符合初级

以下是 Transact-SQL 扩展：

- 在 insert 语句的 select 子句中使用 union

- 当 select 语句中有 union 运算符时，在 select 语句的 order by 子句中指定新的列标题

另请参见

命令 compute clause, declare, group by 和 having 子句, order by 子句, select, where 子句.

函数 convert .

unmount

说明

关闭数据库并从 Adaptive Server 中删除它。设备也会失效并被删除。卸下时，数据库及其页面不会改变。数据库页保留在 OS 设备上。在 unmount 命令完成后，即可断开连接并移动源 Adaptive Server 上的设备（如果必要）。可以使用 *manifest_file* 扩展创建用于辅助 Adaptive Server 的清单文件。

unmount 命令将单个命令中的数据库数限制为 8 个。

警告！ unmount 命令从 Adaptive Server 删除数据库及其所有信息。仅在要将数据库从一台 Adaptive Server 移动到另一台 Adaptive Server 上时，才使用 unmount 命令。

语法

unmount database *dbname_list* to *manifest_file*

参数

dbname_list

正在卸下的数据库。可以 unmount 多个数据库。

manifest_file

描述位于一组数据库设备上的数据库的二进制文件。仅当占用这些设备的那组数据库在这些设备上处于隔离和自我包含状态时才能创建清单文件。

因为清单文件是二进制文件，所以对文件内容执行字符转换的操作（例如 ftp）会损坏该文件，除非以二进制模式进行。

示例

示例 1 从 Adaptive Server 中 unmount 数据库并为数据库创建清单文件：

```
unmount database pubs2 to "/work2/Devices/Mpubs2_file"
```

示例 2 已将在 key_db 中创建的加密密钥用于加密 col_db 中的列。以下命令可成功卸载指定的数据库：

```
unmount database key_db, col_db
unmount database key_db with override
unmount database col_db with override
```

用法

您不能：

- 卸载系统数据库。但可以卸下 sybsystemprocs。
- 卸下代理数据库或用户创建的临时数据库。
- 在事务中使用 unmount 命令。
- 在配置了 HA 的服务器上卸下数据库。

Cluster Edition

Cluster Edition 中支持 mount database 和 unmount database。如果当这些命令中的一个命令正在执行时实例出现故障，则命令可能中止。这种情况下，当实例故障切换恢复完成时，用户必须重新发出 mount database 或 unmount database。

加密列和 *unmount database*

- 当使用其它数据库中的密钥加密列时，应将所有相关的数据库作为一个集合卸下。包含加密列的数据库和包含密钥的数据库之间的相互依赖性类似于使用参照完整性的数据库的相互依赖性。
- 如果数据库中包含的列是使用其它数据库中的密钥加密的，则应使用 override 选项对该数据库执行 unmount （Adaptive Server 发出警告消息，但操作成功）。
- 如果没有包括 with override，命令将失败并显示一条错误消息。

以下命令未使用 override，将会失败，并显示一条错误消息：

```
unmount database key_db

unmount database col_db
```

标准

符合 ANSI SQL 的级别符合初级标准。

权限

对 unmount 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是具有 unmount any database 特权的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
102	unmount	unmount database	• *角色* - 当前活动角色 • *关键字或选项* - NULL • *先前值* - NULL • *当前值* - NULL • *其它信息* - NULL • *代理信息* - set proxy 有效时的初始登录名

另请参见

命令 mount, quiesce database.

update

说明

通过添加数据或者修改现有数据，更改现有行中的数据。

语法

```
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
```

参数

table_name | *view_name*

是要更新的表或视图的名称。如果该表或视图位于另一数据库中，请指定该数据库名称；如果在数据库中有多个具有该名称的表或视图，请指定所有者的名称。*owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

top *unsigned_integer*

将 top *n* 子句紧接在关键字之后插入，并限制更新的行数。

set
指定列名或变量名并赋予其新值。该值可以是表达式或 NULL。列出多个列名或变量名和值时，必须将其用逗号分隔。

from
使用其它表或视图中的数据修改正在更新的表或视图中的行。

readpast
使 update 命令只修改数据行锁定表上未锁定的行或数据页锁定表上未锁定页上的行。update...readpast 自动跳过锁定的行或页而不等待锁被释放。

where
是标准的 where 子句 （请参见 where 子句）。

index {*index_name* | *table_name*}
index_name 指定用于访问 *table_name* 的索引。更新视图时不能使用此选项。

prefetch *size*
为绑定到配置了大 I/O 的高速缓存的表指定 I/O 大小 （单位为千字节）。更新视图时不能使用此选项。 sp_helpcache 显示对象绑定到的高速缓存或缺省高速缓存的有效大小。若要配置数据高速缓存大小，请使用 sp_cacheconfigure。

在使用 prefetch 并指定预取大小 (*size*) 时，最小值是 2K 或逻辑页大小与 2 的任意次方的乘积 （最大为 16K）。 prefetch 大小选项如下 （单位为千字节）：

逻辑页大小	预取大小选项
2	2, 4, 8, 16
4	4, 8, 16, 32
8	8, 16, 32, 64
16	16, 32, 64, 128

在查询中指定的 prefetch 大小仅仅是一个建议。要使用指定的大小，应将数据高速缓存按该大小进行配置。如果未将数据高速缓存配置为特定的大小，则使用缺省的 prefetch 大小。

如果启用了 CIS，则不能对远程服务器使用 prefetch。

lru | mru
指定对表使用的缓冲区替换策略。使用 lru 强制优化程序将表读入 MRU/LRU （最近使用最多的/最近使用最少的）链上的高速缓存。使用 mru 可放弃高速缓存中的缓冲区并将其替换为该表的下一个缓冲区。更新视图时不能使用此选项。

where current of

导致 Adaptive Server 更新 *cursor_name* 的当前游标位置指示的表或视图中的行。

index_name

是要更新的索引的名称。如果未指定索引名，则更新指定表中所有索引的分布统计信息。

plan "*abstract plan*"

指定用来优化查询的抽象计划。它可以是用抽象计划语言指定的完整或部分计划。有关详细信息，请参见 《性能和调优系列：查询处理和抽象计划》中的 “创建和使用抽象计划”。

示例

示例 1 authors 表中的所有 McBaddens 现在成为 MacBaddens：

```
update authors
set au_lname = "MacBadden"
where au_lname = "McBadden"
```

示例 2 修改 total_sales 列以反映在 sales 和 salesdetail 表中记录的最新销售。在此假定只在给定日期记录了给定标题的一组销售，且更新是当前更新：

```
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)
```

示例 3 将 titles 表中当前由 title_crsr 指向的书的价格更改为 $24.95：

```
update titles
set price = 24.95
where current of title_crsr
```

示例 4 查找 IDENTITY 列等于 4 的行并将该书的价格更改为 $18.95。Adaptive Server 用 IDENTITY 列的名称替换 syb_identity 关键字：

```
update titles
set price = 18.95
where syb_identity = 4
```

示例 5 使用一个已声明的变量更新 titles 表：

```
declare @x money
select @x = 0
update titles
```

```
    set total_sales = total_sales + 1,
    @x = price
    where title_id = "BU1032"
```

示例 6更新另一任务不对其持有锁的行：

```
update salesdetail set discount = 40
        from salesdetail readpast
    where title_id like "BU1032"
        and qty > 100
```

用法

- 使用 update 更改已经插入的行中的值。使用 insert 添加新行。
- 可在一条 update 语句中引用多达 15 个表。
- update 与用 create index 命令设置的 ignore_dup_key、 ignore_dup_row 和 allow_dup_row 选项交互作用。有关详细信息，请参见 create index。
- 可以定义一个触发器，从而在对指定的表或表中指定的列发出 update 命令时执行指定的操作。
- 在 Adaptive Server 12.5.2 版之前的版本中，有时不用工作表就能解析对视图执行的带有 union all 子句的 update 和 delete 查询，这偶尔会导致不正确的结果。在 Adaptive Server 12.5.2 中，始终使用 tempdb 中的工作表解析带 union all 子句对视图执行 update 和 delete 的查询。

在 *update* 语句中使用变量

- 可在 update 语句的 set 子句中对变量赋值，与在 select 语句中对它们赋值的方式类似。
- 在 update 语句中使用某个变量前，必须用 declare 声明该变量，并用 select 进行初始化 （如示例 5 所示）。
- 更新中每个限定的行都会进行变量赋值。
- 在 update 语句中赋值的右侧引用变量时，该变量的当前值在更新每行时更改。**当前值** 是当前行更新之前该变量的值。以下示例说明更新每行时当前值如何更改。

  假设有以下语句：

  ```
  declare @x int
  select @x=0
  update table1
      set C1=C1+@x, @x=@x+1
      where column2=xyz
  ```

  更新开始之前 C1 的值是 1。下表说明 *@x* 变量的当前值在每次更新之后如何更改：

行	C1 的初始值	@x 的初始值	计算：C1+@x= 更新的 C1	更新后的 C1 值	计算：@x+1= 更新的 @x	更新值
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

- 当在同一 update 语句中提供多个变量赋值时，赋予变量的值可能取决于它们在赋值列表中的顺序，但并不总是如此。为了获得最佳结果，请不要依赖于位置来确定所赋的值。
- 如果返回多行并发生将非集合列赋予某个变量的情况，则该变量的最终值是处理的最后一行，因此可能没有用。
- 赋值给变量的 update 语句不需要设置任何限定行的值。
- 如果没有符合更新的行，则不对变量赋值。
- 在 update 语句中赋值的变量不能在同一 update 语句的子查询中引用，无论该子查询在该 update 语句的何处出现。
- 在 update 语句中赋值的变量不能在同一 update 语句的 where 或 having 子句中引用。
- 在由连接驱动的更新中，在 update 语句的右侧赋值的变量使用不是正在更新的表中的列。结果值取决于为更新选择的连接顺序和连接的表限定的行数。
- 因为已更新变量的值不是存储在磁盘上，所以更新变量不受 update 语句回退的影响。

对事务使用 *update*

设置了 chained transaction mode on 且当前没有活动的事务时， Adaptive Server 将隐式地用 update 语句启动事务。若要完成更新，必须 commit 事务或 rollback 更改。例如：

```
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
```

该批处理启动一个事务 （使用链式事务模式），并更新 stores 表中的某一行。如果更新与表中另一商店包含相同州和城市信息的行，则将回退对 stores 表进行的更改并结束该事务。否则，它提交更新并结束事务。

Adaptive Server 允许在给定事务中多次发出更新单个行的 update 语句。例如，由于其类型 ID 是 “mod_cook”，所以以下两个更新都会影响 title_id 为 MC2022 的书的价格：

```
begin transaction
update titles
set price = price + $10
where title_id = "MC2222"
update titles
set price = price * 1.1
where type = "mod_cook"
```

在更新中使用连接

在 update 的 from 子句中执行连接是为了更新而对 ANSI 标准的 SQL 语法所做的 Transact-SQL 扩展。update 语句的处理方式决定了单个语句进行的更新不累计。也就是说，如果 update 语句包含一个连接，而此连接中的另一个表在连接列中有多个匹配值，则第二个更新不基于第一个更新的新值，而是基于其初始值。结果是不可预知的，因为它们取决于处理的顺序。考虑以下连接：

```
update titles set total_sales = total_sales + qty
    from titles t, salesdetail sd
    where t.title_id = sd.title_id
```

对 titles 中的每个 title_id、对 salesdetail 的匹配行中的一行，total_sales 值只更新一次。每次的结果可能随查询的连接顺序、表分区或可用索引的不同而不同。但是每次只从 salesdetail 向 total_sales 添加一个单值。

如果希望返回与连接列匹配的值的总和，则以下使用子查询的查询将返回正确结果：

```
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
```

对字符数据使用 *update*

- 用空字符串 ("") 更新可变长度的字符数据或 text 或 unitext 列会插入单个空格。固定长度的字符列会填补到定义的长度。

- 从可变长度列数据中删除所有尾随空格，字符串只包含空格时例外。仅包含空格的字符串会被截断为单个空格。自动截断比 char、nchar、unichar、varchar、univarchar 或 nvarchar 列指定长度长的字符串，除非将 string_rtruncation 设置为 on。
- 对 text 或 unitext 列执行 update 会初始化 text 或 unitext 列，赋予它一个有效的文本指针，并至少分配一个文本页。

在游标中使用 *update*

- 不能更新可滚动游标。
- 若要更新使用游标的行，请使用 declare cursor 定义游标，然后将其打开。游标名不能是 Transact-SQL 参数或局部变量。游标必须可更新，否则 Adaptive Server 将返回错误。对游标结果集进行的任何更新操作也会影响派生该游标行的基表行。
- 用 update...where current of指定的 *table_name* 或 *view_name*必须是在定义该游标的 select 语句的第一个 from 子句中指定的表或视图。如果该 from 子句使用连接引用多个表或视图，可仅指定正在被更新的表或视图。

  更新后，游标位置保持不变。只要另一 SQL 语句没有移动该游标的位置，就可继续更新该游标位置上的行。
- Adaptive Server 允许更新游标的 *select_statement* 列列表中没有指定的列，但这些列必须是 *select_statement* 中指定的表的一部分。但是，如果用 for update 指定了 *column_name_list*，且正在声明该游标，则只能更新那些指定的列。

更新 IDENTITY 列

不能更新带有 IDENTITY 属性的列，无论是通过其基表更新还是通过视图更新。若要确定列是否定义有 IDENTITY 属性，请在列的基表上使用 sp_help。

选入结果表中的 IDENTITY 列遵循与 IDENTITY 属性的继承性有关的下列规则：

- 如果多次选择某一 IDENTITY 列，它将在新表中定义为 NOT NULL。该 IDENTITY 列将不继承 IDENTITY 属性。
- 如果选择 IDENTITY 列作为表达式的一部分，结果列不会继承 IDENTITY 属性。如果表达式中的任何列允许使用 NULL 值，则将它创建为 NULL；否则，将它创建为 NOT NULL。
- 如果 select 语句包含 group by 子句或集合函数，结果列不会继承 IDENTITY 属性。包含 IDENTITY 列的集合的列将被创建为 NULL，其它列则被创建为 NOT NULL。

- 通过联合或连接选入表中的 IDENTITY 列不保留 IDENTITY 属性。如果表中包含 IDENTITY 列和 NULL 列的联合，新列将定义为 NULL。否则，定义为 NOT NULL。

通过视图更新数据

- 不能 update 用 distinct 子句定义的视图。
- 如果视图是用 with check option 创建的，则通过视图更新的每一行都必须可以通过视图查看。例如，stores_cal 视图包括 stores 表中 state 值为“CA”的所有行。with check option 子句根据视图的选择标准检查每个 update 语句。

```
create view stores_cal
as select * from stores
where state = "CA"
with check option
```

  如果将 state 更改为 “CA”之外的值，则 update 语句 （例如下面的语句）将失败：

```
update stores_cal
set state = "WA"
where store_id = "7066"
```

- 如果视图是用 with check option 创建的，则所有从基视图派生的视图都必须满足视图的选择标准。通过*派生*视图更新的每一行都必须能通过基视图查看。

  以从 stores_cal 派生的视图 stores_cal30 为例。此新视图包含位于 California 且付款方式为 “Net 30”的商店的有关信息：

```
create view stores_cal30
as select * from stores_cal
where payterms = "Net 30"
```

  因为 stores_cal 是用 with check option 创建的，所以通过 stores_cal30 更新的所有行都必须可以通过 stores_cal 查看。任何将 state 更改为 “CA”之外的值的行都被拒绝。

  请注意，stores_cal30 本身并不具有 with check option 子句。所以，可以更新 stores_cal30 中 *payterms* 值不是 “Net 30”的行。例如，以下 update 语句会成功执行，不过再也无法通过 stores_cal30 查看该行了：

```
update stores_cal30
set payterms = "Net 60"
where stor_id = "7067"
```

- 不能在将两个或多个表的列连接起来的视图中更新行，除非下列两个条件都为真：

  - 该视图没有 with check option 子句，且
  - 所有被更新的列都属于同一基表。
- 允许对包含 with check option 子句的连接视图执行 update 语句。如果任何受影响的列出现在 where 子句中，或出现在包含来自多个表的列的表达式中，更新将会失败。
- 如果通过连接视图来更新行，所有受影响的列都必须属于同一基表。

使用 *index*、*prefetch* 或 *lru* | *mru*

index、 prefetch 和 lru | mru 可覆盖 Adaptive Server 优化程序作出的选择。请慎用这些语句，并始终用 set statistics io on 检查性能影响。有关使用这些选项的详细信息，请参见 《性能和调优指南》。

使用 *readpast*

- readpast 选项只适用于仅数据锁定表。如果为所有页锁定表指定 readpast，则会忽略该选项。
- readpast 选项与 holdlock 选项不兼容。如果在同一 select 命令中指定了两者，将产生错误并终止命令。
- 如果会话范围的隔离级别为 3，将忽略 readpast 选项。
- 如果会话的事务隔离级别是 0，使用 readpast 的 update 命令将不会发出警告消息。对于数据页锁定表，这些命令将修改所有没有被不兼容锁锁定的所有页中的所有行。对于数据行锁定表，它们影响所有没有被不兼容锁锁定的行。
- 如果将带有 readpast 选项的 update 命令应用于两个或多个文本列，并且检查的第一个文本列含有不兼容锁，则 readpast 锁定将跳过此行。如果此列没有不兼容锁，则此命令将获取一个锁并修改此列。然后，如果行中的任何后续文本列带有不兼容锁，则此命令将阻塞，直到它可以获取锁并修改此列为止。
- 有关 readpast 的详细信息，请参见 《性能和调优指南》。

标准

符合 ANSI SQL 的级别符合初级标准。

以下是 Transact-SQL 扩展：

- 使用 from 子句、限定表或列名是可以由 FIPS 标志程序检测的 Transact-SQL 扩展。连接视图的更新或目标列表包含表达式的视图的更新是直到运行期间才可检测且不可用 FIPS 标志程序进行标志的 Transact-SQL 扩展。
- 变量的使用。
- readpast

权限

如果 set ansi_permissions 为 on，则需要具有正在更新的表的 update 权限。此外，必须具有所有在 where 子句中出现的列以及下列 set 子句的所有列上的 select 权限。缺省情况下， ansi_permissions 为 off。

以下说明了基于您的细化权限设置的 update 权限检查。

细化权限已启用	在启用细化权限的情况下，你必须是表或者视图的所有者，或是具有 update 权限的用户。
细化权限已禁用	在禁用细化权限的情况下，您必须是表或视图的所有者、具有 update 权限的用户或具有 sa_role 的用户。

审计

sysaudits 的 event 和 extrainfo 列中的值如下：

事件	审计选项	审计的命令或访问权限	extrainfo 中的信息
70	update	对表执行的 update 命令	• *角色* – 当前活动角色 • *关键字或选项* – update 或 writetext • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名
71	update	对视图执行的 update 命令	• *角色* – 当前活动角色 • *关键字或选项* – update 或 writetext • *先前值* – NULL • *当前值* – NULL • *其它信息* – NULL • *代理信息* – set proxy 有效时的初始登录名

另请参见

命令 alter table, create default, create index, create rule, create trigger, insert, where 子句.

系统过程 sp_bindefault, sp_bindrule, sp_help, sp_helpartition, sp_helpindex, sp_unbindefault, sp_unbindrule.

update all statistics

说明

更新给定表的所有统计信息（包括所有列的直方图），不管是否已对其建立索引。可以对单个数据分区运行 update all statistics。

语法

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*]

参数

table_name

是正在更新其统计信息的表的名称。

data_partition_name

是要更新的分区的名称。更新数据分区上的每个本地索引分区的统计信息。不更新全局索引的统计信息。

using *step* values

指定直方图梯级数。对于不存在统计信息的列，缺省值是 20。要更改缺省值，请使用 sp_configure 来修改 *number of histogram steps* 参数。如果 sysstatistics 中已经有某个列的统计信息，则该列的缺省值是当前梯级数。

这些梯级适用于分区表的每个分区；例如， update index statistics 对更新统计信息的扫描中涉及的每个数据和索引分区都使用缺省梯级值 20。如果通过全局索引的索引扫描生成全局统计信息，则缺省情况下应用 20 个梯级。如果通过数据扫描或本地索引扫描生成分区统计信息，则缺省情况下对每个分区应用 20 个梯级。

如果通过 using *step* values 指定的直方图梯级数为 M，且 histogram_tuning_factor 参数为 N，则 update index statistics 使用 0 和 M*N 之间的梯级数，具体取决于 update index statistics 隔离的频率单元数以及是否存在任何范围单元。

with consumers = *consumers*

指定在提供了 *column_list* 并启用了并行查询处理的情况下用于排序的消耗程序进程数。 consumers 选项指定应用于排序的并行度，排序是为单个数据分区上的统计信息更新所执行的。例如，如果带有列列表的 update statistics 应用于包含三个数据分区的表，则会单独对每个分区中的数据进行排序，并在每次排序期间应用 consumers 选项。三次排序本身不并行执行。

注释 max parallel degree 配置参数的值必须大于 with consumers 的值。例如，如果 with consumers 设置为 2，则 max parallel degree 不得小于 3。

---

with sampling = *N* percent

指定为收集统计信息而对列进行随机采样的百分比。 *N* 值是 1 和 100 之间的任意数字。

[no_hashing | partial_hashing | hashing

表示 update all statistics 收集的基于散列的统计信息的级别。以下各项之一：

- no_hashing – updates all statistics 使用 15.7 之前版本 Adaptive Server 的算法收集基于排序的统计信息。
- partial_hashing – updates all statistics 使用针对包含的唯一值少于 65536 的列的算法。如果 updates all statistics 遇到的唯一列计数大于或等于 65536 阈值，则会额外使用排序扫描。
- hashing – updates all statistics 使用低域和高域散列创建直方图。

这些参数的缺省值是 update statistics hashing 的配置值。

max_resource_granularity = *N* percent

限制与 update all statistics 和散列结合使用的 tempdb 缓冲区高速缓存量。

histogram_tuning_factor = *integer*

确定 update all statistics 的分配粒度。

print_progress = *int*

确定 update all statistics 是否显示进度消息。

- 0 – （缺省值）命令不显示任何进度消息
- 1 - 命令显示进度消息

示例

示例 1 更新 salesdetail 表的所有统计信息：

```
update all statistics salesdetail
```

示例 2 更新 salesdetail 表上 smallsales 分区的所有统计信息：

```
update all statistics salesdetail partition smallsales
```

示例 3 更新 authors 表的基于散列的统计信息：

```
update all statistics authors with hashing
```

用法

- update all statistics 更新给定表的所有统计信息。Adaptive Server 保存有关表中页分配的统计信息，考虑在分区表的查询处理中是否使用并行扫描以及查询处理中使用哪个或哪些索引时，将会用到这些统计信息。查询优化取决于存储的统计信息的精确性。
- 为每个列创建直方图统计信息，方法是：对前导列执行索引扫描，将前导列投影到工作表中并执行排序，或者使用散列针对各项扫描的多个列同时生成直方图。
- 为正在更新其统计信息的索引列的所有前缀子集创建密度统计信息。例如，如果索引位于 c1、 c2 和 c3 列上，则前缀子集为 (c1,c2) 和 (c1, c2, c3)。
- 在对单个数据分区运行 update all statistics 时，将使用索引扫描为本地索引的每个前导列生成直方图。如果启用基于散列的统计信息收集，则还会在索引扫描期间生成有关所有次要属性的统计信息。对于所有其它列 （包括全局索引的前导列）， update all statistics 将执行数据扫描并随后执行排序，或执行使用基于散列的统计数据收集的数据扫描。

  基于散列的统计信息的优势在于一次数据扫描就可以收集所有列的直方图，而基于排序的统计信息则要针对每个列单独进行扫描。
- update statistics 命令创建特定于分区的统计信息。在创建分区统计信息时，隐式创建全局统计信息。分区统计信息充当创建全局统计信息的输入并启用每分区 DDL 操作。全局统计信息由优化程序使用。
- update all statistics 为表的每个数据和索引分区重新生成并更新存储在 systabstats 中的表统计信息。如果为特定的数据分区运行 update all statistics 命令，则只为该数据分区和所有本地索引分区生成并更新表统计信息。跳过全局索引。

使用基于散列的统计信息

- 仅当散列方法生成的直方图与排序方法生成的直方图精确度及质量相同时 （低域情况）， partial_hashing 参数才会将散列用于列，否则， partial_hashing 参数会针对高域情况使用排序方法。
- 某些情况下 （高域情况），散列方法生成的直方图可能没有排序方法生成的直方图精确。
- 尽管基于散列的统计信息不需要 tempdb 磁盘空间或排序所使用的过程高速缓存，但它可能会使用大量的 tempdb 缓冲区高速缓存。

- 用于 no_hashing 参数的大规模排序可能会清除语句高速缓存中的语句和存储过程，从而释放过程高速缓存以支持排序。
- max_resource_granularity 限制用于 hashing 或 partial_hashing 的 tempdb 缓冲区高速缓存量。不会影响 no_hashing 参数或排序使用的内存量。
- 如果包括 partial_hashing 参数，并且存在指示高域列的先前的列直方图，则 Adaptive Server 会假定此列需要基于排序的统计信息。如果不存在先前的列直方图，则 Adaptive Server 会假定此列为低域，直到达到高域限制为止。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

要运行 update all statistics，您必须是表所有者或具有表的 update statistics 权限的用户。

另请参见

命令 update statistics, update index statistics, update statistics, update table statistics.

update index statistics

说明

更新索引中所有列的统计信息。

语法

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*]

参数

table_name

与 update statistics 一起使用时， *table_name* 是与索引相关联的表的名称。因为 Transact-SQL 不要求索引名在数据库中唯一，所以 *table_name* 是必需的。

data_partition_name

是要更新的分区的名称。更新数据分区上的每个本地索引分区的统计信息。不更新全局索引的统计信息。

index_name

是要更新的索引的名称。如果未指定索引名，则更新指定表中所有索引的分布统计信息。

index_partition_name

是要更新的索引分区的名称。

using *step* values

指定直方图梯级数。对于不存在统计信息的列，缺省值是 20。如果需要更改此参数的缺省值，请使用 sp_configure 修改 *number of histogram steps* 参数。如果 sysstatistics 中已经有某个列的统计信息，则该列的缺省值是当前梯级数。

这些梯级适用于分区表的每个分区；例如， update index statistics 对更新统计信息的扫描中涉及的每个数据和索引分区都使用缺省梯级值 20。如果通过全局索引的索引扫描生成全局统计信息，则缺省情况下应用 20 个梯级。如果通过数据扫描或本地索引扫描生成分区统计信息，则缺省情况下对每个分区应用 20 个梯级。

如果通过 using *step* values 指定的直方图梯级数为 M，且 histogram_tuning_factor 参数为 N，则 update index statistics 使用 0 和 M*N 之间的梯级数，具体取决于 update index statistics 隔离的频率单元数以及是否存在任何范围单元。

with consumers = *consumers*

指定在提供了 *column_list* 并启用了并行查询处理的情况下用于排序的消耗程序进程数。 consumers 选项指定应用于排序的并行度，排序是为单个数据分区上的统计信息更新所执行的。例如，如果带有列列表的 update statistics 应用于包含三个数据分区的表，则会单独对每个分区中的数据进行排序，并在每次排序期间应用 consumers 选项。三次排序本身不并行执行。

注释 max parallel degree 配置参数的值必须大于 with consumers 的值。例如，如果 with consumers 设置为 2，则 max parallel degree 不得小于 3。

with sampling = *N* percent

指定为收集统计信息而对列进行随机采样的百分比。 *N* 值是 1 和 100 之间的任意数字。

注释 散列当前不支持采样。

[no_hashing | partial_hashing | hashing]

指示 update index statistics 收集的基于散列的统计信息的级别。以下各项之一：

- no_hashing – （缺省值）update index statistics 使用 15.7 之前版本 Adaptive Server 的算法收集基于排序的统计信息。
- partial_hashing – update index statistics 使用针对包含的唯一值少于 65536 的列的算法。如果 updates index statistics 遇到的唯一列计数大于或等于 65536 阈值，则会额外使用排序扫描。
- hashing – updates index statistics 使用低域和高域散列创建直方图。

max_resource_granularity = *N* percent

限制与 update index statistics 和散列结合使用的 tempdb 缓冲区高速缓存量。

with histogram_tuning_factor = *integer*

确定 update index statistics 的分配粒度。

print_progress = *int*

确定 update index statistics 是否显示进度消息。

- 0 – （缺省值）命令不显示任何进度消息
- 1 - 命令显示进度消息

示例

示例 1 生成 authors 表的所有索引中的所有列的统计信息：

```
update index statistics authors
```

示例 2 生成 authors 表的 au_names_ix 索引中的所有列的统计信息：

```
update index statistics authors au_names_ix
```

示例 3 使用采样率 20% 生成 au_names_ix 索引的所有内部列的统计信息。

```
update index statistics authors au_names_ix
    with sampling = 20 percent
```

使用索引页的完全扫描收集 au_names_ix 的前导列的统计信息；不对该列进行采样。

示例 4 生成索引分区的所有列的统计信息：

```
update index statistics publishers publish1_idx
    partition p1
```

用法

- 当与表名和索引名一起使用时，update index statistics 将为指定索引中的所有列更新统计信息。如果只与表名一起使用，则 update index statistics 为表的所有索引中的所有列更新统计信息。
- 如果对大表运行 update index statistics，则在 tempdb 不够大，无法处理该命令时，该命令会失败并生成错误号 1105。
- 指定未建索引列的名称或索引的非前导列的名称会生成该列的统计信息而不创建索引。
- 为正在更新其统计信息的每个索引列创建直方图统计信息。
- 为正在更新其统计信息的索引列的所有前缀子集创建密度统计信息。
- 如果对特定分区使用 update index statistics，也将隐式更新全局统计信息。
- 分区统计信息充当创建全局统计信息的输入并启用每分区 DDL 操作。全局统计信息由优化程序使用。
- update index statistics 也为该命令所更新的表的每个数据和索引分区重新生成和更新存储在 systabstats 中的表统计信息。如果为特定的数据分区运行 update index statistics 命令，则只为该数据分区和所有本地索引分区生成和更新表统计信息。跳过全局索引。如果对特定的索引分区运行 update index statistics，则只更新该索引分区的表统计信息。
- with consumers 子句设计用于 RAID 设备上的分区表，这些设备对 Adaptive Server 显示为单个 I/O 设备，但可以提供并行排序所需的高吞吐量。请参见 《性能和调优指南》中的 “并行排序”。

- update index statistics 命令可生成一系列更新统计信息操作，这些操作与索引级和列级等效命令使用相同的锁定、扫描和排序。例如，如果 salesdetail 表在 salesdetail (stor_id, ord_num, title_id) 上有名为 sales_det_ix 的非聚簇索引，则 update index statistics salesdetail 命令将执行以下 update statistics 操作：

```
update statistics salesdetail sales_det_ix
update statistics salesdetail (ord_num)
update statistics salesdetail (title_id)
```

使用基于散列的统计信息

- 仅当散列方法生成的直方图与排序方法生成的直方图精确度及质量相同时 （低域情况）， partial_hashing 参数才会将散列用于列，否则， partial_hashing 参数会针对高域情况使用排序方法。
- 某些情况下 （高域情况），散列方法生成的直方图可能没有排序方法生成的直方图精确。
- 尽管基于散列的统计信息不需要 tempdb 磁盘空间或排序所使用的过程高速缓存，但它可能会使用大量的 tempdb 缓冲区高速缓存。
- 用于 no_hashing 参数的大规模排序可能会清除语句高速缓存中的语句和存储过程，从而释放过程高速缓存以支持排序。
- max_resource_granularity 限制用于散列或 partial_hashing 的 tempdb 缓冲区高速缓存量。不会影响 no_hashing 参数或排序使用的内存量。
- 如果包括 partial_hashing 参数，并且存在指示高域列的先前的列直方图，则 Adaptive Server 会假定此列需要基于排序的统计信息。如果不存在先前的列直方图，则 Adaptive Server 会假定此列为低域，直到达到高域限制为止。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

要运行 update index statistics，您必须是表所有者或具有表的 update statistics 权限的用户。

另请参见

命令 delete statistics, update all statistics, update statistics, update table statistics.

文档 《性能和调优指南》 .

update statistics

说明

为索引、表或分区中的所有列更新指定索引中键值的分布信息，并重置全局非聚簇索引的数据更改计数器。

语法

update statistics *table_name*
[[partition *data_partition_name*]
[(*column1*, *column2*, ⋯) | (*column1*), (*column2*), ⋯] |
index_name [partition *index_partition_name*]]
[using step values | [out_of_range [on | off| default]]]
[with consumers = *consumers*][, sampling=*N* percent]
[, no_hashing | partial_hashing | hashing]
[, max_resource_granularity = *N* [percent]]
[, histogram_tuning_factor = *int*]
[, print_progress = *int*]

参数

table_name

与 update statistics 一起使用时， *table_name* 是与索引相关联的表的名称。因为 Transact-SQL 不要求索引名在数据库中唯一，所以 *table_name* 是必需的。

index_name

是要更新的索引的名称。如果未指定索引名，则更新指定表中所有索引的分布统计信息。

data_partition_name

是要更新的分区的名称。更新数据分区上的每个本地索引分区的统计信息。不更新全局索引的统计信息。

index_partition_name

是要更新的索引分区的名称。

(column1, column2, ...)

创建对此列元组创建并包括本地索引时会创建的相同统计信息。也就是，对 column1 创建直方图，然后对所有列前缀创建多属性分布 (column1, column2), (column1, column2, column3)。

(*column1*), (*column2*)

运行 update statistics 的列的逗号分隔列表。通过单个 update statistics 命令可对列表中的各个列创建直方图，但不会创建任何多属性分布。

using *step* values

指定直方图梯级数。对于不存在统计信息的列，缺省值是 20。如果需要更改此参数的缺省值，请使用 sp_configure 修改 *number of histogram steps* 参数。如果 sysstatistics 中已经有某个列的统计信息，则该列的缺省值是当前梯级数。

这些梯级适用于分区表的每个分区；例如， update statistics 对更新统计信息扫描中涉及的每个数据和索引分区都使用缺省梯级值 20。如果通过全局索引的索引扫描生成全局统计信息，则缺省情况下应用 20 个梯级。如果通过数据扫描或本地索引扫描生成分区统计信息，则缺省情况下对每个分区应用 20 个梯级。

如果通过 using *step* values 指定的直方图梯级数为 M，且 histogram_tuning_factor 参数为 N，则 update statistics 使用 0 和 M*N 之间的梯级数，具体取决于 update statistics 隔离的频率单元数以及是否存在任何范围单元。

out_of_range [on | off | default]

update statistics 完成时，迅速增长的表的列统计信息可能会过期，这会导致 SARG （搜索子句）越界，其选择的取值范围要大于列的直方图所指示的范围。越界 SARG 的选择性为 0。out_of_range] 直方图调整功能可调整列的直方图，并可对此类 SARG 分配合适的选择性值。

缺省情况下，会在服务器范围内启用越界 SARG 的直方图调整功能。

out_of_range [on | off | default] 可指定列级别的越界直方图调整。以下各项之一：

- on – 启用 column_name 的越界直方图调整。
- off – 禁用 column_name 的越界直方图调整。
- default – 根据跟踪标志 15355 的值更改越界直方图调整：
  - 跟踪标志 15355 打开时，禁用越界直方图调整。
  - 跟踪标志 15355 关闭时，启用越界直方图调整。

with consumers = *consumers*

指定在提供了 *column_list* 并启用了并行查询处理的情况下用于排序的消耗程序进程数。 consumers 选项指定应用于排序的并行度，排序是为单个数据分区上的统计信息更新所执行的。例如，如果带有列列表的 update statistics 应用于包含三个数据分区的表，则会单独对每个分区中的数据进行排序，并在每次排序期间应用 consumers 选项。三次排序本身不并行执行。

注释　max parallel degree 配置参数的值必须大于 with consumers 的值。例如，如果 with consumers 设置为 2，则 max parallel degree 不得小于 3。

---

with sampling = *N* percent

指定为收集统计信息而对列进行随机采样的百分比。*N* 值是 1 和 100 之间的任意数字。采样适用于下列所有 update statistics 类型：

- update statistics *table_name* (*col_name*)
- update index statistics
- update all statistics

索引

指定要更新的索引中所有列的统计信息。

[no_hashing | partial_hashing | hashing]

指示 update statistics 收集的基于散列的统计信息的级别。以下各项之一：

- no_hashing – （缺省值） update statistics 使用 15.7 之前版本 Adaptive Server 的算法收集基于排序的统计信息。
- partial_hashing – update statistics 使用针对包含的唯一值少于 65536 的列的算法。如果 update statistics 遇到的唯一列计数大于或等于 65536 阈值，则会额外使用排序扫描。
- hashing – updates statistics 使用低域和高域散列创建直方图。

max_resource_granularity = *N* percent

限制与 update statistics 和散列结合使用的 tempdb 缓冲区高速缓存量。

with histogram_tuning_factor = *integer*

确定 update statistics 的分配粒度。

print_progress = *int*

确定 update statistics 是否显示进度消息。

- 0 – （缺省值）命令不显示任何进度消息
- 1 - 命令显示进度消息

示例

示例 1 生成 titles 表的 price 列的统计信息：

```
update statistics titles (price) using 40 values
```

示例 2 更新数据分区 smallsales 上的统计信息。Adaptive Server 将为该数据分区的每个本地索引的前导列和组合列分别创建直方图和密度。不更新全局索引的统计信息：

```
update statistics titles partition smallsales
```

示例 3 更新数据分区 smallsales 上的统计信息。 Adaptive Server 将为列 col1 创建直方图，为组合列 col1 和 col2 创建密度：

```
update statistics titles partition smallsales (col1,
col2)
```

示例 4 为列检测到 out_of_range SARG 时，优化程序会调整列的直方图并对越界子句分配合适的选择性值。

```
update statistics TOFO_FUOP_ORD(OrdDt) using
out_of_range on
```

示例 5 如果打开跟踪标志 15355，则不会针对越界 SARG 调整列的直方图：

```
update statistics TOFO_FUOP_ORD(OrdDt) using
out_of_range default
```

示例 6 对 authors 表运行 update statistics， histogram_tuning_factor 为 5%。

```
update index statistics authors with
histogram_tuning_factor = 5
```

用法

- Adaptive Server 保存有关每个索引中键值分布的统计信息，并在决定查询处理中使用哪个或哪些索引时使用这些统计信息。
- 在包含数据的表上创建非聚簇索引时，对新索引自动运行 update statistics。在包含数据的表上创建聚簇索引时，对所有索引自动运行 update statistics。
- 在空表上运行 update statistics 不会影响系统表。
- 查询优化取决于统计信息的准确性。如果索引中的键值有重大更改，应该重新在该索引或列上运行 update statistics。如果索引列中已添加、更改或删除 （即怀疑键值的分布已更改）大量数据，请使用 update statistics 命令。
- 还应对包含大量行的系统表运行 update statistics。如果有权对用户表运行此命令，则运行过程与对系统表运行此命令没有任何区别。如果没有统计信息，系统存储过程的执行效果有可能会很差。
- 在对数据分区运行 update statistics 时，该命令将跳过全局索引。

- 当与表名和索引名一起使用时，update statistics 将更新索引的前导列的统计信息。如果只与表名一起使用，则 update statistics 将更新表上所有索引的前导列的统计信息。
- 如果使用逗号分隔的列表 (col1), (col2)... 并启用散列，则在不超过资源粒度的情况下，可使用一次扫描来收集统计信息。如果启用排序，则要对每一列分别使用一次扫描。如果使用部分散列，则可针对低域列使用一次扫描，在不超过资源粒度的情况下，会针对每个高域列的排序使用一次扫描 （也就是说，如果有三列，则有三项排序）。
- 指定未建索引列的名称或索引的非前导列的名称会生成该列的统计信息而不创建索引。
- 在列列表中指定多个列（例如 (col1, col2, ...)）将为第一列生成或更新直方图，为列列表的所有前缀子集生成或更新密度统计信息。这种情况下，不能使用基于散列的统计信息。
- 如果使用 update statistics 为聚簇索引的非前导列和非索引列生成统计信息，则 update statistics 必须扫描表并执行排序。

  除非指定列列表 （如 (col1), (col2)... 或 (col1, col2, col3)），否则 Adaptive Server 会忽略对 update statistics 的采样。如果不是针对列列表进行采样，则使用 update all statistics 或 update index statistics。如果指定列列表，并且这些列是聚簇索引的非前导列和非索引列，则 update statistics 必须扫描表并执行排序，或使用基于散列的算法。
- 如果对特定分区使用 update statistics，也将隐式更新全局统计信息。
- 如果试图将 out_of_range 选项与诸如 consumers 或 sampling 的其它选项一起用于 update statistics，则 Adaptive Server 会引发错误 16015。

  如果为当前没有列级别统计信息的列指定 out_of_range 选项，则 Adaptive Server 会引发错误 16016。
- update statistics 为该命令所更新的表的每个数据和索引分区重新生成和更新存储在 systabstats 中的表统计信息。如果为特定数据分区运行 update statistics 命令，则只为该数据分区和所有本地索引分区生成和更新表统计信息。跳过全局索引。如果对特定的索引分区运行 update statistics，则只更新该索引分区的表统计信息。
- with consumers 子句设计用于 RAID 设备上的分区表，这些设备对 Adaptive Server 显示为单个 I/O 设备，但能够提供并行排序所需的高吞吐量。请参见 《性能和调优系列：查询处理和抽象计划》中的“利用统计信息来提高性能”。

- 表 1-36 显示了在 update statistics 期间执行的扫描类型、获取的锁类型以及何时需要排序。

表 1-36: update statistics 期间的锁定、扫描和排序

update statistics 指定	执行的扫描和排序	锁定
表名		
所有页锁定表	表扫描，加上每个非聚簇索引的叶级扫描	级别 1；共享的意图表锁，当前页的共享锁
DOL 锁定表	表扫描，加上每个非聚簇索引和聚簇索引的叶级扫描 （如果存在）	级别 0；脏读
表名和聚簇索引名		
所有页锁定表	表扫描	级别 1；共享的意图表锁，当前页的共享锁
DOL 锁定表	叶级索引扫描[1]	级别 0；脏读
表名和非聚簇索引名		
所有页锁定表	叶级索引扫描[1]	级别 1；共享的意图表锁，当前页的共享锁
DOL 锁定表	叶级索引扫描[1]	级别 0；脏读
表名和列名		
所有页锁定表	表扫描；创建工作表和排序工作表	级别 1；共享的意图表锁，当前页的共享锁
DOL 锁定表	表扫描；创建工作表和排序工作表	级别 0；脏读

[1] 请参见 《性能和调优系列：查询处理和抽象计划》第 10 章 “利用统计数据改善性能”中的 “Adaptive Server 何时执行扫描和排序”，确定如果进行统计信息采样的列存在于两个或两个以上的索引中，要对哪些索引进行扫描。

- update index statistics 命令可生成一系列更新统计信息操作，这些操作与索引级和列级等效命令使用相同的锁定、扫描和排序。例如，如果 salesdetail 表在 salesdetail (stor_id, ord_num, title_id) 上有名为 sales_det_ix 的非聚簇索引，则 update index statistics salesdetail 命令将执行以下 update statistics 操作：

```
update statistics salesdetail sales_det_ix
update statistics salesdetail (ord_num)
update statistics salesdetail (title_id)
```

- 在从早期版本升级期间，不在 master 数据库的系统表上运行 update statistics。大多数系统过程查询的列上都有索引，不需要在这些表上运行 update statistics 来用于常规用途。但是，允许在所有数据库中的所有系统表 （除了那些非常规表）上运行 update statistics。这些查询时根据内部结构构建的表包括 syscurconfigs、 sysengines、 sysgams、 syslisteners、 syslocks、 syslogs、 syslogshold、 sysmonitors、 sysprocesses、 syssecmechs、 systestlog 和 systransactions。

  无需对 Replication Server RSSD 表运行 update statistics。如果在 Replication Server 尝试访问 RSSD 表时对这些表运行 updates statistics，则会导致 Replication Server 错误。 RSSD 表及其格式只适用于 Replication Server 处理过程。

- 这些 update statistics 参数会保留在所有受影响的列上的值并覆盖所有配置设置，直到列的统计信息被删除，或对列运行 sp_modifystats ... REMOVE_STICKINESS 为止：

  - using step values
  - out_of_range
  - no_hashing
  - partial_hashing
  - hashing
  - histogram_tuning_factor
  - sampling = *N* percent

  注释　consumers 和 max_resource_granularity 不保留其值。

  例如，如果发出：

  ```
  update statistics table_name(column1) with
  no_hashing
  ```

  此表随后的 update statistics 命令会将缺省配置值用于除 column1（继续使用 no_hashing，直到删除 column1 统计信息为止）之外的所有列的 update statistics hashing。

- with consumers 和 with sampling 适用于排序而不适用于散列。可将 partial_hashing 参数与 with consumers 及 with sampling 参数一起使用，但 with consumers 和 with sampling 仅适用于高域列的排序。如果将 with hashing 参数与 with consumers 或 with sampling 参数显式结合使用，则 Adaptive Server 会忽略那些参数。

使用基于散列的统计信息

- 仅当散列方法生成的直方图与排序方法生成的直方图精确度及质量相同时 （低域情况）， partial_hashing 参数才会将散列用于列，否则， partial_hashing 参数会针对高域情况使用排序方法。
- 某些情况下 （高域情况），散列方法生成的直方图可能没有排序方法生成的直方图精确。
- 尽管基于散列的统计信息不需要 tempdb 磁盘空间或排序所使用的过程高速缓存，但它可能会使用大量的 tempdb 缓冲区高速缓存。
- 用于 no_hashing 参数的大规模排序可能会清除语句高速缓存中的语句和存储过程，从而释放过程高速缓存以支持排序。
- max_resource_granularity 限制用于散列或 partial_hashing 的 tempdb 缓冲区高速缓存量。不会影响 no_hashing 参数或排序使用的内存量。
- 如果包括 partial_hashing 参数，并且存在指示高域列的先前的列直方图，则 Adaptive Server 会假定此列需要基于排序的统计信息。如果不存在先前的列直方图，则 Adaptive Server 会假定此列为低域，直到达到高域限制为止。

更新统计信息并进行采样

Adaptive Server 扫描数据页的示例。如果在 update statistics 中指定索引，如下所示：

```
update statistics table_name [index_name] with sampling = N percent
```

该命令将为指定表上的所有索引的前导列或指定索引的前导列创建和更新统计信息。

在使用带有 using *steps* value 的 sampling = *N* percent 选项时，必须最后指定 sampling = *N* percent 选项：

```
update statistics titles (type)
    using 40 value
    with sampling = 10 percent
```

如果不这样做，则会收到错误消息：

```
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'.
```

创建索引和存储过程

Adaptive Server 在执行完 update statistics 语句后自动重新编译存储过程。虽然在执行 update statistics 之前开始的即席查询仍继续进行，但它们不使用新统计信息。

在 Adaptive Server 12.5 和更早版本中，高速缓存的存储过程忽略 update statistics。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

要运行 update statistics，您必须是表所有者或具有表的 update statistics 权限的用户。

另请参见

命令 delete statistics, update all statistics, update index statistics, update table statistics.

文档 《性能和调优指南》 .

有关 optdiag 语法和使用，请参见 《性能和调优指南：监控和分析》中的 “统计信息表和使用 optdiag 显示统计信息”。

update table statistics

说明

update table statistics 更新存储在 systabstats 表中的统计信息，如行计数、集群比等。 update table statistics 不会影响存储在 sysstatistics 中的列统计信息。

语法

update table statistics *table_name*
[partition *data_partition_name*]
[*index_name* [partition *index_partition_name*]]

参数

table_name
是正在更新其统计信息的表的名称。

data_partition_name
是正在更新其统计信息的数据分区的名称。如果未包含此名称，则将更新所有数据分区的表统计信息。

index_name
是与分区相关的索引的名称。

index_partition_name
是索引分区的名称。

示例

示例 1 对 smallsales 分区执行表统计信息更新：

```
update table statistics titles partition smallsales
```

示例 2 对 titles 表上的所有分区执行表统计信息更新：

```
update table statistics titles
```

用法

- update table statistics 不更新索引分区的统计信息。要生成索引分区的表级统计信息，请使用 update statistics。
- 由于运行 update table statistics 时会产生运行 update statistics 的 I/O 开销，因此，应该使用 update statistics 来生成列和表统计信息。

  您可以创建全局索引来生成全局统计信息，然后删除全局索引。

  在单个分区上运行 update statistics 时，可通过合并分区统计信息来创建全局统计信息。但是，这些合并的全局统计信息不如在创建全局索引时附带创建的全局统计信息准确。应避免生成列统计信息，它会覆盖以前更准确的列统计信息版本。

  如果指定：

  - *index_name* – update table statistics 将更新索引的所有索引分区的统计信息。
  - *index_partition* – update table statistics 将更新特定索引分区的统计信息。

标准	符合 ANSI SQL 的级别Transact-SQL 扩展。
权限	要运行 update table statistics，您必须是表所有者或具有表的 update statistics 权限的用户。
另请参见	**命令** update all statistics, update index statistics, update statistics. **文档** 《性能和调优指南》 .

use

说明

指定要使用的数据库。

语法

use *database_name*

参数

database_name

是要打开的数据库的名称。

示例

```
use pubs2
go
```

当前数据库现在是 pubs2。

用法

- 允许用于存档数据库。
- 必须执行 use 命令，才能引用数据库中的对象。
- 不能在存储过程或触发器中包括 use。
- sp_addalias 添加一个别名，该别名使用户能够以另一名称来使用数据库，从而获得此数据库的访问权。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

对 use 的权限检查因您的细化权限设置而异。

细化权限已启用	在启用细化权限的情况下，您必须是有效、别名或 Guest 用户，或是具有数据库的 own database 或 use database 特权的用户。 如果数据库有 guest 帐户，则所有用户都可以使用该数据库。如果数据库没有 guest 帐户，则您必须是数据库的有效用户、在数据库中具有别名或者具有数据库的 own database 或 use database 特权。
细化权限已禁用	在禁用细化权限的情况下，您必须是有效、别名或 Guest 用户，或者是具有 sa_role 或 sso_role 的用户。 如果数据库有 “guest” 帐户，则所有用户都可以使用该数据库。如果数据库没有 “guest” 帐户，则必须是数据库的有效用户、在数据库中拥有别名、系统管理员或系统安全员才能使用数据库。

另请参见

命令 create database, drop database.

系统过程 sp_addalias, sp_adduser, sp_modifylogin.

waitfor

说明

指定执行语句块、存储过程或事务的特定时间、时间间隔或事件。

语法

waitfor {delay *time* | time *time* | errorexit | processexit | mirrorexit}

参数

delay

指示 Adaptive Server 等待指定的一段时间 （最长 24 小时）。

time

指示 Adaptive Server 等待指定的时间段。

time

以 date/time 数据可接受的格式之一表示的时间或字符类型的变量。但是，不能指定日期 – 不允许指定 date/time 值的日期部分。该信息可以使用 time 数据类型。

errorexit

指示 Adaptive Server 一直等到内核进程或用户进程异常终止。

processexit

指示 Adaptive Server 一直等到内核进程或用户进程因任何原因终止。

mirrorexit

指示 Adaptive Server 等待镜像故障。

示例

示例 1 在下午 2:20，用下一步动作更新 chess 表，然后名为 sendmail 的过程在 Judy 拥有的表中插入一行，通知她现在 chess 表中出现新的一步：

```
begin
    waitfor time "14:20"
    insert chess (next_move)
        values ('Q-KR5')
    execute sendmail 'judy'
end
```

示例 2 10 秒钟后， Adaptive Server 显示指定消息：

```
declare @var char (8)
select @var = "00:00:10"
begin
    waitfor delay @var
    print "Ten seconds have passed.  Your time
        is up."
end
```

示例 3 在任何进程异常退出后， Adaptive Server 显示指定消息：

```
begin
    waitfor errorexit
    print "Process exited abnormally!"
```

```
end
```

用法

- 发出 waitfor 命令后，在到了指定时间或发生指定事件前，不能使用与 Adaptive Server 的连接。
- 可将 waitfor errorexit 用于注销异常终止进程的过程，从而释放系统资源，避免系统资源被受影响的进程占用。
- 若要查看哪个进程终止，请用 sp_who 检查 sysprocesses 表。
- 用 waitfor time 或 waitfor delay 指定的时间可以包括小时、分钟和秒。使用格式 “hh:mi:ss”，详见 《参考手册：构件块》的第 18 页的 “日期和时间数据类型”中的第 1 章 “系统数据类型和用户定义的数据类型”。

  以下命令指示 Adaptive Server 一直等到下午 4:23：

  ```
  waitfor time "16:23"
  ```

  以下语句指示 Adaptive Server 等待 1 小时 30 分钟：

  ```
  waitfor delay "01:30"
  ```

- 系统时间的更改 （例如将时钟设置为夏令时）可以延迟 waitfor 命令。
- 可以在 DB-Library 程序中使用 waitfor mirrorexit 以在发生镜像故障时通知用户。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

使用 waitfor 无需任何权限。

另请参见

命令 begin...end.

数据类型 日期和时间数据类型.

系统过程 sp_who.

where 子句

说明

在 select、 insert、 update 或 delete 语句中设置搜索条件。

语法

在 select、insert、update 或 delete 语句中，搜索条件紧跟在关键字 where 之后。如果要在单一语句中使用一个以上的搜索条件，可使用 and 或者 or 连接这些条件。

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

参数

not

否定任何逻辑表达式或关键字（如 like、null、between、in 和 exists）。

expression

可以是列名、常量、函数、子查询或者任何由算术运算符或逐位运算符连接起来的列名、常量和函数的组合。有关表达式的详细信息，请参见 《参考手册：构件块》的第 339 页的 “表达式”中第 4 章 “表达式、标识符和通配符”。

comparison_operator

是以下运算符之一：

运算符	含义
=	等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于
!=	不等于
<>	不等于
!>	不大于
!<	不小于

比较 char、nchar、unichar、varchar、univarchar 和 nvarchar 数据时，< 意味着更靠近字母表的开始位置，而 > 意味着更靠近字母表的结尾位置。

大小写和特殊字符求值取决于 Adaptive Server 所在计算机上的操作系统的归类序列。例如，小写字母可能比大写字母大，而大写字母可能比数字大。

为了便于比较，将忽略尾随空格。例如，“Dirk”与 “Dirk ” 是相同的。

比较日期时，< 表示较早而 > 表示较晚。应为比较运算符使用的所有字符和日期数据加上引号。例如：

```
= "Bennet"
> "94609"
```

请参见第 42 页的 “用户定义的数据类型”《参考手册第 1 章 “系统数据类型和用户定义的数据类型”：过程》中的。

like

是一个关键字，该关键字表示所跟字符串 （用单引号或双引号引起来）是匹配模式。 like 可用于 char、 varchar、 unichar、 univarchar、 nchar、 nvarchar、 datetime、 date 和 time、 text 以及 unitext 列，但不能搜索秒或毫秒。

可以将 like 关键字和通配符用于 datetime 和 date 数据以及 char 和 varchar。将 like 用于 datetime、 date 和 time 值时， Adaptive Server 将日期转换为标准 datetime 格式，然后再转换成 varchar。由于标准存储格式不包括秒或毫秒，所以不能用 like 和某一模式来搜索秒或毫秒。

like 非常适合用来搜索 date/ time 值，因为 date/time 条目可多种日期分量。例如，如果将值 “9:20” 插入名为 arrival_time 的列，则以下子句将找不到它，因为 Adaptive Server 将这个输入内容转换为 “Jan 1, 1900 9:20AM.”：

```
where arrival_time = '9:20'
```

但是，下面的子句可以找到它：

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

match_string

是一个用引号引起来的由字符和通配符组成的字符串。表 1-37 列出了这些通配符。

表 1-37: 通配符

通配符	含义
%	任何包含 0 个或多个字符的字符串
_	任何单个字符
[ ]	指定范围 ([a-f]) 或集合 ([abcdef]) 内的任何单个字符
[^]	不在指定范围 ([^a-f]) 或集合 ([^abcdef]) 内的任何单个字符

escape

指定可以用其搜索通配符的出现的转义字符。

escape_character

是任何单个的字符。请参见 《参考手册：构件块》的第 4 章 “表达式、标识符和通配符”中的第 364 页的 “使用 escape 子句”。

is null

搜索 NULL 值。

between

是表示范围开始值的关键字。对范围结束值使用 and。下面的语句表示介于起始界限之间的范围 （包括界限值）：

```
where @val between x and y
```

下面的语句表示介于起始界限之间的范围 （不包括界限值）：

```
x and @val < y
```

如果指定的第一个值大于第二个值，则使用 between 的查询不返回任何行。

and

连接两个条件并在两个条件都为真时返回结果。

如果在一个语句中使用了多个逻辑运算符，通常先对 and 运算符求值。但是，可使用小括号更改执行顺序。

in

允许选择与值列表中的任一个值相匹配的值。比较元素可以是常量或列名，值列表可以是一组常量或一个子查询 （更常见）。有关将 in 和子查询一起使用的详细信息，请参见 《Transact-SQL 用户指南》。请将值列表用小括号括起来。

value_list

是值的列表。用单引号或双引号将字符值引起来，并用逗号分隔各个值 （请参见示例 7）。列表可以是一组变量，例如：

```
in (@a, @b, @c)
```

但是，值列表不能使用包含列表的变量，如下所示：

```
@a = "'1', '2', '3'"
```

exists

与子查询一起使用可以测试子查询的某些结果是否存在。请参见 《Transact-SQL 用户指南》。

子查询

是 select、insert、delete 或 update 语句或子查询的 where 或 having 子句中受限制的 select 语句 （不允许有 order by 和 compute 子句以及关键字 into）。请参见 《Transact-SQL 用户指南》。

any

与 >、< 或 = 以及子查询一起使用。如果在子查询中检索到的任何值与外层语句的 where 或 having 子句中的值相匹配，就会返回结果。请参见 《Transact-SQL 用户指南》。

all

与 > 或 < 以及子查询一起使用。如果子查询中检索到的所有值都与外层语句的 where 或 having 子句中的值相匹配，则会返回结果。请参见 《Transact-SQL 用户指南》。

column_name

是在比较中使用的列的名称。如果有歧义，请用表名或视图名限定列名。对有 IDENTITY 属性的列，可以指定 syb_identity 关键字 （必要时用表名限定）而不是实际的列名。

column_name 允许采用 text、 unitext 或 image 数据类型。

join_operator

是比较运算符或连接运算符之一 =* 或 *=。请参见 《Transact-SQL 用户指南》。

logical_expression

是一个返回 TRUE 或 FALSE 的表达式。

or

连接两个条件并在两个条件中的任何一个为真时返回结果。

如果在某个语句中使用了多个逻辑运算符，通常先对 and 运算符求值，然后再对 or 运算符求值。但是，可使用小括号更改执行顺序。

示例

示例 1

```
where advance * $2 > total_sales * price
```

示例 2 查找电话号码不是以 415 开头的所有行：

```
where phone not like '415%'
```

示例 3 查找作者名为 Carson、 Carsen、 Karsen 和 Karson 的行：

```
where au_lname like "[CK]ars[eo]n"
```

示例 4 查找 sales_east 表中 IDENTITY 列值为 4 的行：

```
where sales_east.syb_identity = 4
```

示例 5

```
where advance < $5000 or advance is null
```

示例 6

```
where (type = "business" or type = "psychology") and advance > $5500
```

示例 7

```
where total_sales between 4095 and 12000
```

示例 8 查找州为列表中的三个州之一的行：

```
where state in ('CA', 'IN', 'MD')
```

示例 9 根据 text、 unitext 和 image 列的空值选择数据：

```
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
```

示例 10 根据 text 列的非空值更新数据：

```
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
```

示例 11 根据 temp1 中的 text 列的空值将选择的数据添加到表 temp2 中：

```
select c1, c2 into temp2 from temp1 where c2 is null
select * from temp2
go
(1 row affected)
c1      c2
-----  ----------
1       NULL
```

示例 12 根据 temp1 中的 text 列的非空值从 temp1 中选择数据并将其插入表 temp2 中：

```
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)
```

示例 13 根据 text 列的空值选择子查询中的数据：

```
select count(*) from temp2
where c1 in (select c1 from temp1 where c2 is null and c3 is not null)

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

示例 14 根据 unitext 列的空值删除数据：

```
delete from temp1 where c3 is null
go
(2 rows affected)
```

用法

- where 和 having 搜索条件相同，只是 where 子句中不允许有集合函数。例如，以下子句是合法的：

  ```
  having avg (price) > 20
  ```

  下列子句不合法：

  ```
  where avg (price) > 20
  ```

  有关示例，请参见《参考手册：构件块》中的第 2 章 “Transact-SQL 函数”来获得有关集合函数用法的信息，并参见第 434 页的 “group by 和 having 子句”。

- 在搜索条件中指定连接和子查询；有关完整详细信息，请参见《Transact-SQL 用户指南》。

- 可使用关键字 like 来搜索特定模式的 unitext 列。但是，在与 unitext 列一起使用时，like 子句不进行优化。与 unitext 匹配的 like 模式依赖于缺省的 Unicode 排序顺序，该顺序也用于与 unichar 和 univarchar 数据类型匹配的 like 模式。
- 对于变量 *expression* 和 *match_string*， where 子句接受 text 和 unitext LOB 定位符，但不接受 image LOB 定位符。

  ```
  ...
  where expression like 'match_string'
  ...
  ```

  当 *match_string* 为定位符时， Adaptive Server 仅使用最多 16KB 的对应 LOB。
- 指定空条件会仅选择在指定 LOB 列中具有空值的行。 LOB 值可能因为显式指派了空值或因为 LOB 未初始化而为空。
- where 子句中的 and 和 or 条件的数量仅受到运行查询时可用内存量的限制。
- like 谓词中包括的模式字符串仅受到可以放在 varchar 中的字符串大小的限制。
- 在 char 或 varchar 条目中，有两种方法可以指定文字引号。第一种方法是使用两个引号。例如，如果在某个字符条目开头用了一个单引号，并要将一个单引号作为该条目的一部分，则可使用两个单引号：

  ```
  'I don''t understand.'
  ```

  或使用双引号：

  ```
  "He said, ""It's not really confusing."""
  ```

  第二种方法是给引号加上相反类型的引号。也就是说，给包含双引号的条目加上单引号 （或反之）。下面是一些示例：

  ```
  'George said, "There must be a better way."'
  "Isn't there a better way?"
  'George asked, "Isn"t there a better way?"'
  ```
- 如果输入的字符串超出屏幕宽度，则在转到下一行之前输入反斜杠 (\)。
- 当某列与 where 子句中的常量或变量比较时， Adaptive Server 将把常量或变量转换为该列的数据类型，从而使优化程序可以使用索引进行数据检索。例如，当与 int 列比较时，会将 float 表达式转换成 int。例如：

  ```
  where int_column = 2
  ```

  选择 *int_column* = 2 的行。

- 当 Adaptive Server 优化查询时，它将对 where 和 having 子句中的搜索条件进行评估，以确定哪些条件是可以用于选择最佳索引和查询计划的搜索参数 (SARG)。所有搜索条件都用来限定行。有关搜索参数的详细信息，请参见 《性能和调优指南》。

标准

符合 ANSI SQL 的级别符合初级标准。

另请参见

命令 delete, execute, group by 和 having 子句, insert, select, update.

数据类型 日期和时间数据类型.

系统过程 sp_helpjoins.

while

说明

设置重复执行语句或语句块的条件。只要指定条件为真，语句就会重复执行。

语法

while *logical_expression* [plan "*abstract plan*"] *statement*

参数

logical_expression

是任何返回 TRUE、 FALSE 或 NULL 的表达式。

plan "*abstract plan*"

指定用来优化查询的抽象计划。它可以是用抽象计划语言指定的完整或部分计划。只能为可优化的 SQL 语句 （即访问表的查询）指定计划。有关详细信息，请参见 《性能和调优系列：查询处理和抽象计划》中的 “创建和使用抽象计划”。

statement

可以是单个 SQL 语句，但通常是由 begin 和 end 语句界定的 SQL 语句块。

示例

如果平均价格低于 $30，则将 titles 表中所有书的价格翻倍。只要价格仍低于 $30， while 循环就不断地将价格翻倍。除了确定价格超过 $20 的书籍外， while 循环中的 select 还指示完成了多少次循环 （Adaptive Server 返回的每个平均结果表示一次循环）：

```
while (select avg (price) from titles) < $30
begin
    select title_id, price
        from titles
        where price > $20
    update titles
        set price = price * 2
end
```

用法

- 可以用 break 和 continue 命令从循环内控制 while 循环中语句的执行。
- continue 命令导致 while 循环跳过 continue 之后的所有语句重新开始。break 命令导致从 while 循环退出。接着执行出现在标识循环结束的关键字 end 之后的所有语句。break 和 continue 命令通常由 if 测试激活。

  例如：

```
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
```

只要书的平均价格低于 $30，该批处理就继续将 titles 表中所有书的价格翻倍。但是，如果有任何书的价格超过了 $50， break 命令就会停止 while 循环。如果平均价格超过 $30， continue 命令将阻止执行 print 语句。无论 while 循环如何终止 （正常结束或因为 break 命令结束），最后的查询都显示哪些书的价格超过了 $30。

- 如果嵌套有两个或更多 while 循环，则 break 命令会退出到下一外层循环。将运行内层循环结尾之后的所有语句；然后，重新开始下一外层循环。

警告！ 当 while 循环中出现 或 create view 命令时，Adaptive Server 将在确定条件是否为真之前为表或视图创建模式。如果表或视图已经存在，则会导致出错。

标准　符合 ANSI SQL 的级别Transact-SQL 扩展。

权限　使用 while 无需任何权限。

另请参见　**命令**　begin...end, break, continue, goto label

writetext

说明

允许对现有 text、unitext 或 image 列进行需要日志操作最少的交互式更新。

语法

writetext [[*database*.]*owner*.]*table_name.column_name*
text_pointer [readpast] [with log] *data*

参数

table_name.column_name

是表和要更新的 text、unitext 或 image 列的名称。如果该表位于另一数据库中，请指定数据库名；如果数据库中有多个具有该名称的表，请指定所有者的名称。 *owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。

text_pointer

一个 varbinary (16) 值，该值存储指向 text、 unitext 或 image 数据的指针。使用 textptr 函数确定该值。text、unitext 或 image 数据不与其它表列存储在同一组链接页中。它存储在单独的一组链接页中。指向实际位置的指针和数据存储在一起；textptr 返回该指针。

readpast

指定命令应该只修改未锁定行。如果 writetext 命令发现锁定行，它将跳过这些行，而不是等待锁被释放。

with log

记录插入的 text、unitext 或 image 数据。使用该选项有助于介质恢复，但记录大块的数据将迅速增加事务日志的大小，所以请确保事务日志驻留在单独的数据库设备上。有关详细信息，请参见 create database、 sp_logdevice 和 《系统管理指南》。

data

是要写入 text、unitext 或 image 列的数据。text 和 unitext 数据必须用引号引起来。 image 数据前面必须有 “0x”。请检查有关您正在使用的客户端软件的信息，以确定客户端可以容纳的 text、unitext 或 image 数据的最大长度。

示例

示例 1 将文本指针放在局部变量 *@val* 中。然后， writetext 将文本字符串 “hello world” 放到 *@val* 所指向的文本字段中：

```
declare @val varbinary(16)
select @val = textptr (copy) from blurbs
    where au_id = "409-56-7008"
writetext blurbs.copy @val with log "hello world"
```

示例 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"
```

示例 3 writetext 包含有关 unitext 数据类型的信息，并将字符串 “Hello world” 放到 *@val* 所指向的 unitext 字段中：

```
declare @val varbinary(16)
select @val = textptr (ut) from unitable
where i = 100
writetext unitable.ut @val with log "Hello world"
```

在更新列之前， varchar 常量将隐式转换为 unitext。

用法

- 对于 text、 unitext 或 image 数据，可以用 writetext 交互地插入的文本的最大长度约是 120K 字节。
- 缺省情况下， writetext 是需要最少日志记录的操作；只记录页分配和取消分配，但在将 text、 unitext 或 image 数据写入数据库时，不进行记录。要在缺省状态 （最小记录状态）下使用 writetext，系统管理员必须使用 sp_dboption 将 select into/bulkcopy/pllsort 设置为 true。
- writetext 更新现有行中的 text 数据。更新完全替换所有现有文本。
- insert 或 update 触发器不会捕获 writetext。
- writetext 需要一个指向 text、unitext 或 image 列的有效文本指针。text 或 unitext 列必须包含实际数据或由 update 显式输入的 NULL 值，才会存在有效文本指针。

  假设表 textnull 有 textid 列和 *x* 列，其中 *x* 是一个允许 NULL 值的 text 列，则此 update 会将所有 text 值设置为 NULL，并在 text 列中赋予一个有效文本指针：

  ```
  update textnull
  set x = null
  ```

  显式 NULL 值的 insert 不会产生文本指针：

  ```
  insert textnull values (2,null)
  ```

  而且，隐式 NULL 值的 insert 也不会产生文本指针：

  ```
  insert textnull (textid)
  values (2)
  ```

- text 列上的 insert 和 update 是记录的操作。
- 不能对视图中的 text 和 image 列使用 writetext。

- 如果在更改为多字节字符集后试图对 text 值使用 writetext，且没有运行 dbcc fix_text，则命令会失败，并会生成一条错误消息指示您要对表运行 dbcc fix_text。
- 缺省情况下 （不记录模式），在进行 dump database 时， writetext 会运行得更慢。
- Client-Library 函数 dbwritetext 和 dbmoretext 比 writetext 运行得更快且使用的动态内存更少。这些函数可以插入多达 2GB 的 text 数据。

使用 *readpast* 选项

- readpast 选项只适用于仅数据锁定表。如果为所有页锁定表指定 readpast，则会忽略该选项。
- 如果会话范围的隔离级别为 3，将不加提示地忽略 readpast 选项。
- 如果会话的事务隔离级别是 0，使用 readpast 的 writetext 命令将不会发出警告消息。如果文本列没有用不兼容锁锁定，则这些在隔离级别 0 执行操作的命令将修改此指定的文本列。

标准

符合 ANSI SQL 的级别Transact-SQL 扩展。

权限

要运行 writetext，您必须是表所有者或具有 update 权限的用户。

另请参见

命令 readtext.

数据类型 转换 text 和 image 数据类型.

第 2 章

Interactive SQL 命令

本章介绍 Interactive SQL 命令。可在 Interactive SQL 显示窗口的顶部窗格中输入这些命令。这些命令仅用于 Interactive SQL，不会发送到 Adaptive Server 来执行。有关 Interactive SQL 的信息，请参见《实用程序指南》的第 9 章 “使用 Interactive SQL”和 Adaptive Server 插件联机帮助。

表 2-1：DBISQL 命令

命令	说明
第 712 页上的 “clear”	清除 Interactive SQL 窗格。
第 713 页上的 “configure”	打开 “Interactive SQL 选项” (Interactive SQL Options) 对话框。
第 714 页上的 “connect”	建立与数据库的连接。
第 717 页上的 “disconnect”	删除与数据库的当前连接。
第 718 页上的 “exit”	退出 Interactive SQL。
第 719 页上的 “input”	将数据从外部文件或通过键盘导入到数据库表中。
第 724 页上的 “output”	将数据从外部文件或通过键盘导入到数据库表中。
第 729 页上的 “parameters”	指定 Interactive SQL 命令文件的参数。
第 730 页上的 “read”	从文件中读取 Interactive SQL 语句。
第 732 页上的 “set connection”	将当前数据库连接更改到另一台服务器。
第 733 页上的 “start logging”	使用此语句可开始将执行的 SQL 语句记录到日志文件中。
第 734 页上的 “stop logging”	使用此语句可停止在当前会话中记录 SQL 语句。
第 735 页上的 “system”	使用此语句可从 Interactive SQL 内启动可执行文件。

clear

说明	清除 Interactive SQL 窗格。
语法	clear
用法	• 使用 clear 语句可清除 “SQL 语句” (SQL Statements) 和 “消息” (Messages) 窗格以及 “结果” (Results) 窗格中的 “结果” (Results)、“消息” (Messages) 和 “计划” (Plan) 选项卡。 • clear 会关闭与要清除的数据相关联的游标。
权限	任何用户都可以执行此命令。

configure

说明　打开 Interactive SQL 的 “选项”对话框。

语法　configure

用法

- configure 语句打开 Interactive SQL 的 “选项” (Options) 对话框，并显示所有 Interactive SQL 选项的当前设置。该语句不会显示数据库选项，也不允许修改这些选项。

- 可以在此对话框中配置 Interactive SQL 设置。如果选择 “设置为永久” (Make Permanent)，系统会保存这些选项以供后续的 Interactive SQL 会话使用。如果不选择 “设置为永久” (Make Permanent)，而是单击 “确定”，则这些选项只是临时设置，仅对当前数据库连接有效。

权限　任何用户都可以运行 configure。

另请参见　set

connect

说明

建立与数据库的连接。

语法

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*}

参数

engine_name
是要连接到的引擎的名称。

database_name
是要连接到的数据库的名称。它必须符合标识符规则，并且不能是变量。

as
通过指定 as 子句可以对连接进行命名 （可选）。这允许建立到同一个数据库的多个连接，或者建立到同一个或不同的多个数据库服务器的多个连接，所有连接都是同时发生的。每个连接均有自己的关联事务。如果通过两个不同连接修改同一数据库中的相同记录，则这些事务之间可能发生锁定冲突。

connection_name
是用来建立连接的登录名。

user
表示正以用户身份连接到 Adaptive Server。

user_id
是正在连接的用户的 ID。

identified by password
表示用户在连接时需要提供口令。

password
是指连接到 Adaptive Server 的用户的口令。

identifier
是要用于连接信息的标识符。

string
是要用于连接信息的字符串。

hostvar
是代表主机名和端口的变量信息。

connect_string
是格式为 keyword = value 的参数设置列表，各参数设置用分号分隔，并且必须用单引号引起来。

示例

示例 1 通过交互式 SQL 连接到数据库。Interactive SQL 提示输入用户 ID 和口令：

```
connect
```

示例 2 通过交互式 SQL 作为 DBA 连接到缺省数据库。Interactive SQL 提示输入口令：

```
connect user "DBA"
```

示例 3 以用户 dba 身份（口令为 sql）连接到在主机“tribble”（端口号为 5000）上运行的 Adaptive Server 的 pubs2 数据库：

```
connect to "tribble:5000"
database pubs2
user dba
identified by sql
```

示例 4 以用户 dba 身份（口令为 sql）连接到名为“tribble”（在 *interfaces* 文件中定义）的 Adaptive Server：

```
connect to tribble
user dba
identified by sql
```

用法

- connect 建立与服务器（标识为 engine_name）上运行的数据库（标识为 database_name）的连接。
- 在成功执行完 connect 语句后，才允许执行其它语句。
- Interactive SQL 行为 如果不在 connect 语句中指定数据库或服务器，则 Interactive SQL 仍连接到当前数据库，而不是连接到缺省服务器和数据库。如果确实指定数据库名，但未指定服务器名，则 Interactive SQL 将尝试连接到当前服务器上的指定数据库。如果指定服务器名，但未指定数据库名，则 Interactive SQL 将连接到指定服务器上的缺省数据库。
- 在用户界面中，如果不指定口令或者用户 ID 和口令，系统会提示用户输入缺失信息。

- 当 Interactive SQL 的运行模式为命令提示符模式（从命令指示符启动 Interactive SQL 时指定了 -nogui）或批处理模式时，或如果未使用 as 语句执行 connect，系统将打开一个未命名的连接。如果已经打开另一个未命名连接，则原来的连接会自动关闭。否则，当运行 connect 时，现有连接不会关闭。
- 多个连接通过当前连接的概念进行管理。成功执行 connect 语句后，新连接将变成当前连接。要切换至其它连接，请使用 set connection 语句。使用 disconnect 语句可删除连接。
- 在 Interactive SQL 中，连接信息（包括数据库名、用户 ID 和数据库服务器）显示在“SQL 语句”窗格上方的标题栏中。如果未连接到数据库，标题栏中将显示“未连接” (Not Connected)。

权限

任何用户都可以执行此命令。

另请参见

disconnect, set connection

disconnect

说明

删除与数据库的当前连接。

语法

disconnect [{*identifier* | *string* | *hostvar*} | current | all]

参数

{identifier | string | hostvar}

是用来建立连接的登录名。

- identifier – 是要用于连接信息的标识符。
- string – 是要用于连接信息的字符串。
- hostvar – 是代表主机名和端口的变量信息。

current

表示正在断开当前连接。

all

表示正在断开所有连接。

示例

断开所有连接：

```
disconnect all
```

用法

- disconnect 删除与数据库服务器的连接，同时释放该连接使用的所有资源。如果要删除的连接已在 connect 语句中命名，则可以指定该名称。指定 all 将删除应用程序与所有数据库环境的所有连接。current 是缺省设置，它删除当前连接。
- 系统将对删除的连接执行隐式 rollback。

权限

任何用户都可以执行此命令。

另请参见

connect, set connection

exit

说明

退出 Interactive SQL。

语法

{exit | quit | bye} [{*number* | *connection_variable*}]

参数

exit | quit | bye

关闭与数据库的连接，然后关闭 Interactive SQL 环境。

{number | connection_variable}

可用于批处理文件中，以指示 Interactive SQL 命令文件中的命令是执行成功还是执行失败。缺省返回码为 0。

- number – 是返回码的编号。
- connection_varible – 是指定特定连接的变量。

用法

在关闭数据库连接之前，如果 commit_on_exit 选项设置为 on，则 Interactive SQL 会自动执行 commit 语句。如果此选项设置为 off，则 Interactive SQL 执行隐式 rollback。缺省情况下，commit_on_exit 选项设置为 on。

权限

任何用户都可以执行此命令。

input

说明

将数据从外部文件或键盘导入数据库表中。

语法

```
input into [ owner.]table_name
    [ from filename | prompt]
    [ format { ascii | dbase | dbasell | dbaselll | 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}]
```

参数

from 子句

是作为加引号的字符串传递给服务器的文件名。因此，该字符串遵循的格式要求与其它 SQL 字符串相同。需要特别指出的是：

- 要指明目录路径，必须用两个反斜杠表示反斜杠字符 (\)。要将数据从 *c: emp\input.dat* 文件装载到 employee 表中：

  ```
  input into employee
  from 'c:nn temp nn input.dat'
  ```

- 路径名是相对于运行 Interactive SQL 的计算机的。

prompt

允许用户为一行中的每一列输入值。在窗口模式下运行时，会显示一个对话框，用户可以在其中为新行输入值。如果用户是在命令行上运行 Interactive SQL，则 Interactive SQL 会提示用户在命令行上输入每一列的值。

format

每组值都必须遵循 format 子句指定的格式；或如果未指定 format 子句，则必须遵循 set option input_format 语句设置的格式。用户输入命令后，会显示一个对话框，供用户以输入格式在每行输入一个数据行。

某些文件格式包含有关列名和列类型的信息。

使用此信息，input 语句会在数据库表尚不存在时进行创建。这是一种非常简单的将数据装载到数据库中的方法。以下格式包含创建表所需的足够信息：dbasell、 dbaselll、 foxpro 和 lotus。

从命令文件输入的操作由包含 end 的行终止。从文件输入的操作在文件末尾被终止。

允许的 input 格式包括：

- ascii – 假定输入行是 ASCII 字符，每个输入行占一行，值用逗号分隔。字母字符串可以用撇号 （单引号）引起来，也可以用引号（双引号）引起来。包含逗号的字符串必须用单引号或双引号引起来。如果字符串本身包含单引号或双引号，则应使用双重引号字符，才能在字符串中使用该字符。也可以选择使用 delimited by 子句指定其它分隔字符串，而不是使用缺省分隔符 （逗号）。

  同时，系统还识别其它三种特殊序列。两个字符表示一个换行符，“\”表示单个 \，序列 \xDD 表示具有十六进制代码 DD 的字符。

- dbase – 文件为 DBASEll 或 DBASElll 格式。Interactive SQL 将尝试基于文件中的信息来确定是哪种格式。如果表不存在，则创建表。
- dbasell – 文件为 DBASEll 格式。如果表不存在，则创建表。
- dbaselll – 文件格式为 DBASElll。如果表不存在，则创建表。
- excel – 输入文件格式为 Microsoft Excel 2.1。如果表不存在，则创建表。
- fixed – 输入行使用固定格式。请使用 column widths 子句指定列宽。如果不指定列宽，则文件中的列宽必须与相应数据库列类型的任何值所要求的最大字符数相同。

  不能将 fixed 格式用于包含嵌入式换行符和文件尾字符序列的二进制列。

- foxpro – 文件为 FoxPro 格式。如果表不存在，则创建表。
- lotus – 文件为 Lotus WKS 格式工作表。 input 假定 Lotus WKS 格式工作表中的第一行为列名。如果表不存在，则创建表。在这种

情况下，由于文件中的信息适用于单元格而不是列，因此所创建的列类型和列大小可能不正确。

escape character

十六进制代码和符号的缺省转义字符是反斜杠 (\)，例如， \x0A 表示换行字符。

可以使用 escape character 子句更改转义字符。例如，要将感叹号用作转义字符，请输入：

```
... escape character '|'
```

只能使用一个单字节字符作为转义字符。

escapes

启用 escapes （缺省设置）后，数据库服务器将反斜杠字符后面的字符识别并解释为特殊字符。换行符可以作为组合 \n 包含在数据中，其它字符可以作为十六进制 ASCII 码包含在数据中，例如使用 \x09 表示制表符。连续两个反斜杠字符 (\) 解释为一个反斜杠。反斜杠后跟除 n、 x、 X 或 \ 外的其它任何字符都解释为两个单独的字符。例如， \q 将插入一个反斜杠和字母 q。

by

允许用户指定输入文件中的列是基于它们在列表中的序号位置（order，缺省值）还是按它们的列名 (name) 与表列进行匹配。并非所有输入格式都在文件中包含列名信息。 name 只能用于具有列名的输入格式。 dbasell、 dbaselll、 foxpro 和 lotus。

delimited

允许指定要用作 ASCII 输入格式中的分隔符的字符串。

column widths

只能为 fixed 格式指定；它指定输入文件中列的宽度。如果未指定 column widths，则宽度由数据库列类型确定。如果在 fixed 格式中插入 long varchar 或 binary 数据，请不要使用此子句。

nostrip

通常，对于 ASCII 输入格式，在插入值之前会从不带引号的字符串中去除尾随空白。 nostrip 可用于禁止去除尾随空白。无论是否使用该选项，都不会从带引号的字符串中去除尾随空白。而无论是否使用 nostrip 选项设置，都会从无引号的字符串中去除前导空白。

如果 ASCII 文件包含列显示为空值的条目，则将其视为 NULL。如果该位置的列不能为 NULL，则会在数值列中插入零，在字符列中插入空字符串。

encoding

允许指定用于读取文件的编码。 encoding 只能用于 ASCII 格式。

如果未指定 encoding， Interactive SQL 将按如下方式确定用于读取文件的代码页，列表中位置靠前的代码页值优先于位置靠后的值：

- 使用 default_isql_encoding 选项指定的代码页 （如果设置此选项）
- 启动 Interactive SQL 时使用 -codepage 选项指定的代码页
- 运行 Interactive SQL 的计算机的缺省代码页

示例

ASCII 文本文件中的一个 input 语句：

```
input into employee
from new_emp.inp
format ASCII
```

用法

- input 语句允许向指定数据库表中进行高效的大量插入。输入行可通过输入窗口从用户那里读取 （如果指定 prompt），也可从文件中读取 （如果指定 from file_name）。如果两者均未指定，则从包含 input 语句的命令文件中读取。在 Interactive SQL 中，甚至可以直接从“SQL 语句”(SQL Statements) 窗格中读取输入。在这种情况下，输入以只包含字符串 end 的行结束。
- 如果为任何输入格式指定了列列表，则数据将插入到指定表的指定列中。缺省情况下， input 语句假定输入文件中的列值显示顺序与它们在数据库表定义中显示的顺序相同。如果输入文件的列顺序不同，则必须在 input 语句的末尾列出输入文件的实际列顺序。

  在本例中，创建一个名为 inventory 的表。若要从先包含名称值然后是数量值的输入文件中导入 ASCII 数据，则必须在 input 语句的末尾列出输入文件的实际列顺序，以便正确插入数据：

  ```
  create table inventory (
  quantity int,
  item varchar(60)
  )
  ```

  顺序包含名称值和数量值的输入文件 *stock.txt* 中的 ASCII 数据：

  ```
  'Shirts', 100
  'Shorts', 60
  ```

  input 语句末尾用于正确插入数据的输入文件实际列顺序：

  ```
  input into inventory
  from stock.txt
  FORMAT ASCII
  (item, quantity)
  ```

- 缺省情况下，在尝试插入导致错误的行时， input 将会停止。通过设置 on_error 和 conversion_error 选项，可以按不同的方式处理错误。如果在 input 上截断任何字符串值，则 Interactive SQL 会在 “消息” (Messages) 窗格中输出警告。 NOT NULL 列中缺少的值将被设置为零 （对于数值型列）或者设置为空字符串 （对于非数值型列）。如果 input 尝试插入 NULL 行，则输入文件包含空行。

权限

必须对表或视图具有 insert 权限。

output

说明

将数据从外部文件或通过键盘导入到数据库表中。

语法

```
output to filename
    [ append ]
    [ verbose ]
    [ format {ascii | dbase | dbaseII| dbaseIII
        | excel | fixed | foxpro | lotus | sql | xml}]
    [ escape character character ]
    [ escapes { on | off}
    [ delimited by string ]
    [ quote string [ all ] ]
    [ column widths (integer , .. . ) ]
    [ hexidecimal { on | off | asis } ]
    [ encoding {string | identifier}]
```

参数

append

将查询结果附加到现有输出文件的末尾，但不覆盖该文件的原有内容。如果未使用 append 子句，则缺省情况下 output 语句会覆盖输出文件的内容。如果输出格式为 ASCII、 fixed 或 SQL，则 append 关键字有效。

verbose

将有关查询的错误消息、用于选择数据的 SQL 语句以及数据本身写入输出文件。不包含数据的行带有两个连字符作为前缀。如果省略 verbose （缺省设置），则仅将数据写入到文件中。如果输出格式为 ASCII、 fixed 或 SQL，则 verbose 有效。允许的输出格式包括：

- ascii – 输出为 ASCII 格式文件，每个输出行在该文件中占一行。所有值都用逗号分隔，字符串用撇号 （单引号）引起来。可以使用 delimited by 和 quote 子句更改分隔符和引号字符串。如果在 quote 子句中指定 all，则所有值 （不止是字符串）都用引号引起来。

  还可使用其它三种特殊序列。两个字符表示一个换行符，“\”表示单个 \，序列 \xDD 表示具有十六进制代码 DD 的字符。这是缺省输出格式。

- dbasell – 输出为 DBASEll 文件，该文件包含列定义。最多可以输出 32 列。列名被截断至 11 个字符，每行数据都单独占一行。如果表不存在，则创建表。

- dbaselll – 输出为 dBASE III 格式文件，该文件包含列定义。最多可以输出 128 列。列名被截断至 11 个字符，每一列中的每行数据都被截断至 255 个字符。

- excel – 输出为 Excel 2.1 工作表。工作表的第一行包含列标签（如果未定义任何标签，则为列名）。后续的工作表行包含实际的表数据。

- fixed – 输出为固定格式，每一列都具有固定宽度。可使用 column widths 为每一列指定宽度。此格式不输出任何列标题。

  如果省略 column widths 子句，则每一列的宽度通过该列的数据类型来计算，计算出的宽度足以存放该数据类型的任何值。例外情况是 long varchar 和 long binary 数据，它们缺省为 32K。

- foxpro – 输出为 FoxPro 格式文件，该文件包含列定义。最多可以输出 128 列。列名被截断至 11 个字符。列名被截断至 11 个字符，每一列中的每行数据都被截断至 255 个字符。

- html – 输出为超文本标记语言格式。

- lotus – 输出为 Lotus WKS 格式工作表。列名作为第一行放置在工作表中。其它软件 （例如 Lotus 1-2-3）可装载的 Lotus WKS 格式工作表有对最大大小的特定限制。 Interactive SQL 生成的文件没有大小限制。

- SQL – 输出为重新创建表中信息所需的 Interactive SQL input 语句。

- XML – 输出为以 UTF-8 格式编码的 XML 文件，其中包含嵌入式

DTD。二进制值以 CDATA 块进行编码，二进制数据显示为两位数十六进制字符串。 input 语句不接受将 XML 作为文件格式。

escape character

存储为十六进制代码和符号的字符的缺省转义字符是反斜杠 (\)，例如， \x0A 表示换行字符。

可以使用 escape character 更改缺省转义字符。例如，要将感叹号用作转义字符，请输入：

```
... escape character '|'
```

escapes

如果已启用 （缺省设置），则数据库服务器将反斜杠字符后面的字符识别并解释为特殊字符。换行符可作为组合 \n 包括在数据中，其它字符可作为十六进制 ASCII 码包括在数据中 （如 \x09 表示制表符）。连续两个反斜杠字符 (\) 解释为一个反斜杠。反斜杠后跟除 n、 x、 X 或 \ 外的其它任何字符都解释为两个单独的字符。例如， \q 将插入一个反斜杠和字母 q。

delimited by

仅用于 ASCII 输出格式。分隔符字符串放置在列之间 （缺省为逗号）。

quote

仅用于 ASCII 输出格式。引号字符串放置在字符串值的两边。缺省值为单引号字符。如果在 quote 子句中指定 all，则所有值两边都放置引号字符串，并非仅字符串两边。

column width

为 fixed 格式输出指定列宽

hexidecimal

仅用于 ASCII 格式，指定如何卸载二进制数据。设置为 on 时，以 0xabcd 格式卸载二进制数据。设置为 off 时，在卸载时转义二进制数据 (\xab\xcd)。设置为 asis 时，值将按原样写入，即，不进行任何转义，即使值包含控制字符也是如此。 asis 对于包含格式设置字符 （如制表符或回车）的文本非常有用。

encoding

允许指定用于写入文件的编码。 encoding 只能用于 ASCII 格式。

如果未指定 encoding， Interactive SQL 将按如下方式确定用于写入文件的代码页，列表中位置靠前的代码页值优先于位置靠后的值：

- 使用 default_isql_encoding 指定的代码页 （如果设置此选项）
- 启动 Interactive SQL 时使用 -codepage 选项指定的代码页
- 运行 Interactive SQL 的计算机的缺省代码页

示例

示例 1 将 employee 表的内容放置在 ASCII 格式的文件中：

```
select *
    from employee
go
output to employee.txt
    format ASCII
```

示例 2 将 employee 表的内容放置在现有文件的末尾，同时还在此文件中包含有关查询的所有消息：

```
select *
    from employee
go
output to employee.txt append verbose
```

示例 3 在本例中，需要导出包含嵌入式换行符的值。换行符的数值为 10，在 SQL 语句中，该值可表示为字符串 “\x0a”。如果在 hexidecimal 设置为 on 的情况下执行以下语句：

```
select 'line1 n x0aline2'
go
output to file.txt hexidecimal on
```

您将得到一个包含下面一行文本的文件：

```
line10x0aline2
```

但是，如果在 hexidecimal 设置为 off 的情况下执行同一语句，您将得到下列文本：

```
line1 n x0aline2
```

最后，如果将 hexidecimal 设置为 asis，您将得到一个包含两行文本的文件：

```
line1
line2
```

由于已导出的嵌入式换行符没有转换成两位数的十六进制形式，也没有使用任何前缀，因此在使用 asis 时获得两行。

用法

- output 语句会将通过当前查询检索到的信息复制到文件中。
- 可以使用可选的 format 子句指定输出格式。如果不指定 format 子句，则使用 Interactive SQL output_format 选项设置。
- 当前查询是 select 或 input 语句，生成的信息显示在 “结果” (Results) 窗格的 “结果” (Results) 选项卡上。如果当前查询不存在，则 output 语句将报告错误。

- 在 Interactive SQL 中，“结果”选项卡仅显示当前查询的结果。所有先前的查询结果都替换为当前查询结果。

权限

任何用户都可以执行此命令。

parameters

说明

指定 Interactive SQL 命令文件的参数。

语法

parameters parameter1, parameter2, .. .

示例

此交互式 SQL 命令文件采用两个参数：

```
parameters department_id, file;
select emp_lname
    from employee
    where dept_id = {department_id}
    >#{file}.dat
```

如果将此脚本保存在名为 *test.sql* 的文件中，则可以使用以下命令从 Interactive SQL 中运行此脚本：

```
read test.SQL [100] [data]
```

用法

- parameters 语句为命令文件指定参数，以便将来在该命令文件中引用这些参数。
- 引用参数的方法是，将 `{parameter1}` 放置在需要替代指定参数的文件中。大括号与参数名之间不能包含任何空格。
- 如果调用命令文件的参数少于所需的参数个数，则 Interactive SQL 会提示输入缺失参数的值。

权限

任何用户都可以执行此命令。

另请参见

read

read

说明

从文件中读取 Interactive SQL 语句。

语法

read [encoding {*identifier* | *string*}] *file_name* [*parameters*]

参数

encoding {identifier | string}

允许指定用于写入文件的编码。 encoding 只能用于 ASCII 格式。

- identifier – 是用于表示正在读取的文件的标识符。
- string – 是用于表示正在读取的文件的字符串。

file_name

是正在读取的文件的名称。

parameters

对应于语句文件中列出的参数。

示例

下面是 read 语句的示例：

```
READ status.rpt '160'

READ birthday.SQL [>= '1988-1-1'] [<= '1988-1-30']
```

用法

- read 语句从指定文件中读取一系列 Interactive SQL 语句。此文件可包含任何有效的 Interactive SQL 语句，包括其它 read 语句。 read 语句可以嵌套任意多层。如果文件名不包含绝对路径，则 Interactive SQL 会搜索该文件。 Interactive SQL 首先会搜索当前目录，接着搜索在环境变量 SQLPATH 中指定的目录，然后搜索在环境变量 PATH 中指定的目录。如果指定文件没有文件扩展名， Interactive SQL 将搜索每个目录以查找具有 *.SQL* 扩展名的相同文件名。
- encoding 参数允许指定用于读取文件的编码。 read 语句在读取文件时不处理转义字符。它假定整个文件都采用指定的编码。如果未指定编码，则 Interactive SQL 将按如下方式确定用于读取文件的代码页，列表中位置靠前的代码页值优先于位置靠后的值：
  - 使用 default_isql_encoding 选项指定的代码页 （如果设置此选项）
  - 启动 Interactive SQL 时使用 -codepage 选项指定的代码页
  - 运行 Interactive SQL 的计算机的缺省代码页
- 可以在命令文件的名称后面列出参数。这些参数对应于在语句文件开头的 parameters 语句中指定的参数。Interactive SQL 在源文件中包含 {parameter_name} （其中 parameter_name 为适当参数的名称）的所有位置改为使用该对应参数。

- 传递给命令文件的参数可以是标识符、数字、带引号的标识符或字符串。使用引号将参数引起来时，在替换时引号也放置到文本中。如果参数不是标识符、数字或字符串 （包含空格或制表符），则必须用方括号 ([ ]) 括起来。这允许在命令文件中执行任意文本替换。
- 如果传递给命令文件的参数不够，则 Interactive SQL 会提示输入缺失参数的值。

权限

任何用户都可以执行此命令。

set connection

说明

将当前数据库连接更改到另一台服务器。

语法

set connection {*identifier* | *string* | *hostvar*}

参数

identifier
是要用于连接信息的登录名标识符。

string
是要用于连接信息的字符串。

hostvar
是代表主机名和端口的变量信息。

用法

set connection 语句用于将活动数据库连接更换到另一台服务器。当前连接状态将被保存，当该连接再次变为活动连接时，将重新恢复到原来的状态。如果省略 connection_name，并且存在未命名的连接，则该连接将成为活动连接。

权限

任何用户都可以执行此命令。

另请参见

connect, disconnect

start logging

说明　开始将执行的 SQL 语句记录到日志文件中。

语法　start logging *file_name*

参数　file_name
是要将会话记录到的文件。

示例　开始记录到名为 *filename.sql* 的文件中，该文件位于 c: 目录中：

```
start logging 'c:n filename.sql'
```

用法　start logging 语句开始将以后执行的所有 SQL 语句复制到指定的日志文件中。如果该文件不存在， Interactive SQL 会进行创建。除非使用 stop logging 语句显式停止记录进程，或结束当前 Interactive SQL 会话，否则记录将一直进行。您还可以通过选择 “SQL” | “开始记录” (Start Logging) 来开始记录，并通过选择 “SQL” | “停止记录” (Stop Logging) 来停止记录。

权限　任何用户都可以执行此命令。

另请参见　stop logging

stop logging

说明

停止记录当前会话中已执行的 SQL 语句。

语法

stop logging

示例

停止当前记录会话：

```
stop logging
```

用法

stop logging 语句阻止 Interactive SQL 将执行的每条 SQL 语句写入日志文件中。可以使用 start logging 语句开始记录。您还可以通过选择 “SQL” | “开始记录” (Start Logging) 来开始记录，并通过选择 “SQL” | “停止记录” (Stop Logging) 来停止记录。

权限

任何用户都可以执行此命令。

另请参见

start logging

system

说明　从 Interactive SQL 内启动可执行文件。

语法

```
system '[path] file_name'
```

参数

path
　是记事本程序的路径

file_name
　是正在启动的程序的文件名。

示例　启动记事本程序，假定记事本可执行文件位于您的路径中。

```
system 'notepad.exe'
```

用法　启动指定的可执行文件。

- system 语句必须完全包含在一行内。
- 不允许在 system 语句末尾添加注释。
- 将路径和文件名用单引号引起来。

权限　任何用户都可以执行此命令。

另请参见　connect

索引

符号

英文

A

B

C

D

E

F

G

H

J

K

L

M

N

P

Q

R

S

T

W

X

Y

Z