SYBASE®
An SAP® Company

系统管理指南，卷 2

Adaptive Server® Enterprise

15.7

文档 ID：DC32964-01-1570-01

最后修订日期：2011 年 9 月

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

目录

第 1 章

限制对服务器资源的访问

本章描述如何使用资源限制来限制关键时期单个登录或应用程序可使用的 I/O 开销、行计数、处理时间或 tempdb 空间。另外，还介绍如何创建规定的时间范围，为资源限制指定连续的时间块。

资源限制

资源限制是系统管理员指定的一组参数，用于防止来自单个登录或应用程序的查询和事务独占服务器资源。

资源限制与时间范围绑定，以允许系统管理员精确定义何时强制实施这些限制。系统管理员修改资源限制时，所有登录的用户（包括系统管理员）都以看到所做的更改。

资源限制的参数集包括一天中强制限制的时间和要采取的操作类型。例如，可以不让大型报告在一天中的关键时期运行，或当某个会话的查询产生不要的**笛卡尔乘积**时，注销该会话。

计划资源限制

计划资源限制时，考虑以下方面：

- 强制实施限制一天中的哪些时段和一周的哪些天。
- 监视哪些用户和应用程序
- 强制实施何种类型的限制：
  - 可能需要进行大量逻辑和物理读取的查询的 I/O 开销（估计的或实际的）
  - 可能返回大结果集的查询的行计数
  - 由于查询自身的复杂性或外部因素（例如服务器负载），造成可能需花费很长时间才能完成的查询所经历时间
- 是对单个查询加以限制还是指定更大的限制作用域（查询批处理或事务）
- 针对启动了连接但在很长时间内使连接保持空闲状态（这个过程中可能使用锁等系统资源）的用户的最长空闲时间
- 是在执行前还是执行时强制限制 I/O 开销
- 超过限制时采取何种措施（发出警告、中止该批查询或事务，还是取消会话）

启用资源限制

若要启用资源限制，请使用：

```
sp_configure "allow resource limits", 1
```

值 1 启用资源限制；值 0 禁用资源限制。allow resource limits 是静态参数，所以必须重新启动服务器才能重新设置所做的更改。

allow resource limits 通知服务器为时间范围、资源限制和服务器内部报警分配内存。它还将适用的范围和限制内部分配给登录会话。

将 allow resource limits 设置为 1 还将更改 showplan 和 statistics i/o 的输出：

- showplan 将优化程序估计用于整个查询的开销显示为没有单位的数字。此开销估计取决于表的统计信息（值的数量和分布）和相关缓冲池的大小。与缓冲池的状态和活动用户的数量之类的因素无关。请参见《性能和调优系列：查询处理和抽象计划》中的第 2 章“使用 showplan”。
- statistics i/o 中包括根据优化程序的开销计算公式计算出来的语句的实际总 I/O 开销。此值代表逻辑 I/O 数乘以逻辑 I/O 的开销所得的值与物理 I/O 数乘以物理 I/O 开销所得的值之和。

定义时间范围

时间范围 是指一周内的一天或连续几天每天中的连续时间块。

Adaptive Server® 有预定义的“所有时间”范围，此范围包括从星期一到星期日，每天 24 小时的时间。可以根据资源限制的需要创建、修改和删除其它时间范围。

指定的时间范围可以重叠。但是，对特定用户/应用程序组合的限制不能与重叠的指定的时间范围相关联。

例如，假定工作时间内，您限制“joe_user”在运行 payroll 应用程序时返回 100 行。稍后，您试图限制高峰期间内此用户的行检索，此高峰时间与工作时间有重叠。新限制失败，因为该限制与现有限制重叠。

可以创建共享相同时间范围的不同限制。例如，可以在与行检索限制相同的时间范围内对“joe_user”进行第二次限制。例如，可以限制该用户的一个询可以运行的时间，使之与其行检索的限定时间范围相同。

创建指定的时间范围后，Adaptive Server 将其存储在 systimeranges 系统表中。每个时间范围都有一个范围 ID 号。“所有时间”范围的范围 ID 是 1。Adaptive Server 消息引用的是特定的时间范围。

确定所需的时间范围

使用与下面类似的图表确定要为每个服务器创建的时间范围。监视整周内的服务器使用情况；然后找出服务器特别繁忙或执行关键任务而不应中断的间段。

星期 \ 时间	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	00:00
星期一																									
星期二																									
星期三																									
星期四																									
星期五																									
星期六																									
星期日																									

创建指定的时间范围

可以使用 sp_add_time_range 执行以下操作：

- 指定时间范围
- 指定一周中开始和结束时间范围的日期
- 指定一天中开始和结束时间范围的时间

请参见《参考手册：过程》中的 sp_add_time_range。

时间范围示例

假定两个关键作业在每周的以下时间运行：

- 作业 1 在星期二和星期三的 07:00 到 10:00 运行。
- 作业 2 在星期六的 08:00 到星期日的 13:00 运行。

下表使用“1”表示作业 1 在运行，“2”表示作业 2 在运行：

星期 \ Time	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	00:00
星期一																									
星期二								1	1	1	1														
星期三								1	1	1	1														
星期四																									
星期五																									
星期六									2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
星期日	2	2	2	2	2	2	2	2	2	2	2	2	2	2											

对于作业 1，用单个时间范围 tu_wed_7_10 就可涵盖其运行时间：

```
sp_add_time_range tu_wed_7_10, tuesday, wednesday, "7:00", "10:00"
```

但是作业 2 则需要两个单独的时间范围，分别在星期六和星期日：

```
sp_add_time_range saturday_night, saturday, saturday, "08:00", "23:59"
sp_add_time_range sunday_morning, sunday, sunday, "00:00", "13:00"
```

修改指定的时间范围

使用 sp_modify_time_range 来：

- 指定修改的时间范围
- 指定对一周内日期的修改
- 指定对一天内时间的修改

请参见《参考手册：过程》中的 sp_modify_time_range。

例如，要将 *business_hours* 时间范围的结束日期变为星期六，保留现有的开始日期、开始时间和结束时间，输入：

```
sp_modify_time_range business_hours, NULL, Saturday, NULL, NULL
```

若要为 *before_hours* 时间范围指定新的结束日期和结束时间，请输入：

```
sp_modify_time_range before_hours, NULL, Saturday, NULL, "08:00"
```

注释　不能修改“所有时间”时间范围。

删除指定的时间范围

使用 sp_drop_time_range 可以删除用户定义的时间范围。

请参见《参考手册：过程》中的 sp_drop_time_range。

例如，要从 master 数据库中的 systimeranges 系统表中删除 *evenings* 时间范围，输入：

```
sp_drop_time_range evenings
```

注释　不能删除“所有时间”时间范围或为其定义了资源限制的任何时间范围。

时间范围更改何时生效？

每次查询批处理开始时，活动的时间范围绑定到登录会话。由于实际时间的更改而导致的服务器的活动时间范围的更改在查询批处理期间不会产生影。换句话说，如果资源限制在给定的时间范围内对查询批处理加以限制，但该批查询是在此时间范围变为活动状态之前开始的，则资源限制对已在行的该批查询不会产生影响。但是，如果您在同一登录会话期间运行第二个查询批处理，则该查询批处理将会受时间更改的影响。

增加、修改和删除时间范围不会影响当前正在进行的登录会话的活动时间范围。

如果资源限制的作用域是某一事务，而在事务运行时，服务器的活动时间范围发生了更改，则新的活动时间范围不会影响当前正在进行的事务。

确定用户和限制

对于每个资源限制，请指定限制对象。

可将资源限制应用于以下任一对象：

- 特定登录使用的所有应用程序
- 使用特定应用程序的所有登录
- 特定登录使用的特定应用程序

其中*应用程序*是运行在 Adaptive Server 基础之上的客户端程序，可通过特定登录名访问。若要在 Adaptive Server 上运行应用程序，请通过 CS_APPNAME 连接属性来指定其名，方法是使用 Open Client DB-Library™ 中的 cs_config（Open Client™ Client-Library™ 应用程序）或 DBSETLAPP 函数。若要列出在服务器上运行的指定应用程序，请从 master..sysprocesses 表中选择 program_name 列。

有关 CS_APPNAME 连接属性的详细信息，请参见《Open Client Client-Library/C 参考手册》。有关 DBSETLAPP 函数的详细信息，请参见《Open Client DB-Library/C 参考手册》。

标识使用大量资源的用户

实施资源限制之前，运行 sp_reportstats。此过程的输出结果可以帮助您了解哪些用户使用大量系统资源。例如：

```
                    sp_reportstats
Name    Since          CPU     Percent CPU  I/O     Percent I/O
------  -----------   -----   ------------ -----   ------------
probe   jun 19 2007   0       0%           0       0%
julie   jun 19 2007   10000   24.9962%     5000    24.325%
jason   jun 19 2007   10002   25.0013%     5321    25.8866%
ken     jun 19 2007   10001   24.9987%     5123    24.9234%
kathy   jun 19 2007   10003   25.0038%     5111    24.865%

                    Total CPU   Total I/O
                    ---------   ---------
                    40006       20555
```

I/O 和 Percent I/O 列表明用户的资源使用情况是均衡的。有关退款核算的详细信息，请参见《系统管理指南，卷 1》中的第 5 章“设置配置参数”。

标识使用大量资源的应用程序

若要确定系统上运行了哪些应用程序以及运行它们的用户，请查询 master 数据库中的 sysprocesses 系统表。

以下的查询决定仅有 isql、payroll、perl 和 acctng 客户程序的名称被传递到 Adaptive Server：

```
                    select spid, cpu, physical_io,
                      substring(user_name(uid),1,10) user_name,
                      hostname, program_name, cmd
                    from sysprocesses
spid  cpu    physical_io  user_name hostname program_name cmd
----  ---    -----------  --------- -------- ------------ ------
  17    4          12748  dbo       sabrina  isql         SELECT
 424    5              0  dbo       HOWELL   isql         UPDATE
 526    0            365  joe       scotty   payroll      UPDATE
 568    1           8160  dbo       smokey   perl         SELECT
 595   10              1  dbo       froth    isql         DELETE
 646    1              0  guest     walker   isql         SELECT
 775    4          48723  joe_user  mohindra acctng       SELECT

(7 rows affected)
```

由于 sysprocesses 是动态建立以报告当前进程的，因此重复的查询会产生不同的结果。全天范围内重复此查询，如此进行一段时间，以确定哪些应用程序在系统上运行。

CPU 和物理 I/O 值定期刷新到 syslogins 系统表，从而增加了 sp_reportstats 所显示的值。

确定系统上运行哪些应用程序后，可使用 showplan 和 statistics io 来评估这些应用程序中查询的资源使用情况。

选择限制类型

确定要限制的用户和应用程序后，请选择资源限制类型。

表 1-1 描述了每种限制类型的功能和作用域，并列出可帮助确定某种类型的限制对特定的查询是否有用的工具。可能要为给定的用户和应用程序指定一种以上的限制。请参见第 10 页的“了解限制类型”。

表 1-1：资源限制类型

限制类型	适用的查询	测量资源使用情况	范围	强制期间
io_cost	要求进行大量的逻辑和物理读取操作。	运行查询之前，请使用 set showplan on 来显示估计的 I/O 开销；使用 set statistics io on 来观察实际 I/O 开销。	查询	执行前或执行时
row_count	返回大结果集。	使用 *@@rowcount* 全局变量可帮助您确定对行计数的适当限制。	查询	执行
elapsed_time	需要花费长时间才能完成的查询，其原因或是由于自身的复杂性，或是由于外部因素（例如服务器负载或等待锁）。	运行查询之前，请使用 set statistics time on 来显示所经历的时间（以毫秒计）。	查询批处理或事务	执行
tempdb_space	当创建工作表或临时表时，使用 tempdb 中的所有空间。	每个会话在 tempdb 中使用的页数。	查询批处理或事务	执行
idle_time	处于非活动状态。	连接处于非活动状态的时间（以秒为单位）。	单个进程	执行前

spt_limit_types 系统表存储关于每种限制类型的信息。

确定强制时间

强制时间是查询处理中的一个阶段，在该阶段中 Adaptive Server 应用给定资源限制。资源限制发生在：

- 执行前 — Adaptive Server 在执行前应用资源限制（基于优化程序估计的 I/O 开销）。使用这种类型的限制可以阻止潜在的消耗资源很多的查询执行。I/O 销是执行前的时间内唯一可加以限制的资源类型。

  评估条件语句的子句内的数据操作语言 (DML) 语句的 I/O 开销时，Adaptive Server 将分别考虑每个 DML 语句。即使实际上只执行一个子句，它也会评估所有语句。

  执行前资源限制仅针对查询；也就是说，仅基于查询逐个计算和监控在编译时限制的资源值。

  Adaptive Server 不会将执行前资源限制语句强行作用于触发器。

- 执行时 — Adaptive Server 在运行时应用资源限制，通常用以阻止查询独占服务器资源和操作系统资源。执行时限制比执行前限制可能要使用更多的资源额外的 CPU 时间以及 I/O）。

确定资源限制的范围

scope 参数（例如 sp_add_resource_limit *scope* 或 sp_help_resource_limit *scope*）在 Transact-SQL 语句中指定限制的持续时间。限制作用域可以是查询、查询批处理和事务：

- 查询 — Adaptive Server 将资源限制应用于访问服务器的任何单个 Transact-SQL 语句，例如 select、insert 和 update。当您在查询批处理中发出这些语句时，Adaptive Server 将分别评估它们。

  Adaptive Server 将一个存储过程看作是一系列 DML 语句。它评估存储过程中每个语句的资源限制。如果一个存储过程执行了另一个存储过程，Adaptive Server 将评估该嵌套的存储过程中内层嵌套级中的每个 DML 语句。

  Adaptive Server 按逐个嵌套级的次序检查作用于查询的执行前资源限制。Adaptive Server 进入每个嵌套级时，先根据估计的每个 DML 语句的资源使用情况来检查活动的资源限制，之后才可以在该嵌套级执行任何语句。如果该嵌套级上任何 DML 查询的估计资源使用超过了活动资源限制的限制值，将发生资源制冲突。Adaptive Server 将采取违反资源限制时所绑定的措施。

Adaptive Server 根据每个 DML 查询的累计资源使用情况来检查针对查询的执行时资源限制。当某个查询的资源使用超过活动的执行时资源限制的限制值时将发生限制冲突。同样，Adaptive Server 将执行绑定到该资源限制的操作。

- 查询批处理 — 由一个或多个 Transact-SQL 语句组成；例如，在 isql 中，当一组查询由单个 go 命令终结符执行时，就成为查询批处理。

  查询批处理从嵌套级 0 开始；每次调用存储过程将使嵌套级增加 1（最多可增至最大嵌套级）。从存储过程的每次返回将使嵌套级减小 1。

  只有执行时的资源限制才可以作用于查询批处理。

  Adaptive Server 根据每个查询批处理中语句的累计资源使用来检查针对查询批处理的执行时资源限制。当查询批处理的资源使用超过活动的执行时资源限制的限制值时，将发生限制冲突。Adaptive Server 将执行绑定到该资源限制的操作。

- 事务 — Adaptive Server 根据事务的累计资源使用情况，在事务执行期间将作用域为事务的限制应用于所有嵌套级。

  当事务的资源使用超过活动的执行时资源限制的限制值时，将发生限制冲突。Adaptive Server 将执行绑定到该资源限制的操作。

  只有执行时的资源限制才可以作用于事务。

  应用资源限制时，Adaptive Server 不能识别嵌套的事务。*@@trancount* 设置为 1 时，事务上的资源限制开始；*@@trancount* 设置为 0 时，事务资源限制结束。

- 会话 — 空闲时间限制适用于应用限制的会话。

了解限制类型

资源限制允许您使用不同方式限制资源使用：

- I/O 开销
- 经历时间
- 行计数
- tempdb 空间使用
- 空闲时间

限制 I/O 开销

I/O 开销以查询处理期间使用的逻辑和物理访问（“读取”）的次数为基础。为了在执行之前确定最有效的处理计划，Adaptive Server 优化程序会同时使逻辑资源和物理资源来计算估计的 I/O 开销。

Adaptive Server 将优化程序的开销计算公式的结果作为一个无单位的数字来使用，即一个不一定基于单个测量单位（例如秒或毫秒）的值。

若要设置资源限制，必须了解如何将这些限制转换为运行时的系统开销。例如，必须了解其开销为 *x* 逻辑 I/O、*y* 物理 I/O 的查询对生产服务器的影响。

限制 io_cost 可以控制 I/O 密集型查询，包括返回大结果集的查询。但是，如果运行一个将返回一个大表中的所有行的简单查询，而没有关于表大小的当前统计信，则优化程序可能不会估计查询将超过 io_cost 资源限制。若要防止查询返回大结果集，请在 row_count 上设置资源限制。

当 Adaptive Server 配置为并行查询处理时，对分区的表跟踪 I/O 开销限制可能没有对未分区的表精确。请参见《性能和调优系列：查询处理和抽象计划》的第 5 章“并行查询处理”。

确定 I/O 开销

若要确定对 I/O 开销的适当限制，请使用 set 命令确定某些典型查询所需的逻辑和物理读取的次数：

- set showplan on 显示优化程序估计的开销。使用这些信息来设置执行前资源限制。优化程序估计的查询 I/O 开销超过限制值时，将发生执行前资源限制冲突。此类限可禁止执行可能耗用很多资源的查询。
- set statistics io on 显示实际所需的逻辑和物理读取的次数。可以使用这些信息来设置执行时资源限制。查询的实际 I/O 开销超过限制值时，将发生执行时资源限制冲突

统计的实际 I/O 开销仅包括查询中涉及的用于用户表和工作表的访问开销。但 Adaptive Server 可能会使用内部的其它表；例如，它访问 sysmessages 来显示统计信息。因此，查询有时可能会超出它的实际 I/O 开销限制，即使在统计信息中看不出来。

计算查询的开销时，优化程序假定所需的每页在第一次访问时都需要物理 I/O，而以后则在高速缓存中重复访问。由于各种原因，实际的 I/O 开销可能优化程序的估计开销不同。

如果某些页已在高速缓存中或统计信息不正确，则估计的开销将高于实际开销。如果优化程序选择 16K 的 I/O，而某些页在 2K 的高速缓冲池中，需要多个 2K 的 I/O，则估计的开销可能低于实际开销。此外，如果一个较大连接强制高速缓存将其页刷新回磁盘，则重复的访问可能要求重复的物理 I/O。

如果分布或密度统计信息已过时或不能使用，优化程序的估计将不准确。

计算游标的 I/O 开销

所有游标（除执行游标外）的处理游标时的开销估计在 declare cursor 时计算，执行游标的开销则是在游标打开时计算的。

对于所有游标类型，I/O 开销的执行前资源限制是在 open *cursorname* 时强制进行的。优化程序在用户每次试图打开游标时重新计算限制值。

执行时资源限制应用于从打开一个游标到关闭一个游标的时间段内游标的累积 I/O 开销。优化程序在每次打开一个游标时重新计算 I/O 限制。

请参见《Transact-SQL 用户指南》的第 18 章“游标：访问数据”。

io_cost 限制类型的作用域

约束 I/O 开销的资源限制仅应用于单个查询。如果您在一批查询中发出几个语句，Adaptive Server 将评估每个查询的 I/O 使用情况。请参见第 9 页的“确定资源限制的范围”。

限制经历时间

经历时间是执行查询批处理或事务所需的秒数。经历时间由诸如查询复杂程度，服务器负载和等待锁等因素决定。

若要确定对所用时间的适当限制，请使用已利用 set statistics time 收集的信息。只能在执行时限制经历时间资源。

当 set statistics time 处于 on 状态时，运行一些典型的查询来确定处理时间（以毫秒计）。当您创建资源限制时，将毫秒转换为秒。

经历时间资源限制可应用于限制作用域（查询批处理或事务）内的所有 SQL 语句，而不仅仅是 DML 语句。当相关作用域的经历时间超过限制值时，会发生资源限制冲突。

单独的经历时间限制不能应用于嵌套的存储过程或事务。换句话说，如果一个事务嵌套在另一个事务之内，则经历时间限制将应用于外部事务，它包了内部事务经历时间。因此，如果您计算事务的运行时间，该运行时间包括所有嵌套事务的运行时间。

elapsed_time 限制类型的作用域

限制经历时间的资源限制的作用范围是查询批处理或事务。请参见第 9 页的“确定资源限制的范围”。

限制结果集的大小

row_count 限制类型限制返回给用户的行数。如果 select 语句返回的行数超过限制值，将发生限制冲突。

如果资源限制的措施是发送一条警告，当查询超过了行限制时，即返回全部的行数，后面跟一条说明超出限制值的警告，例如：

```
Row count exceeded limit of 50.
```

如果资源限制的措施是中止查询批处理或事务，或者取消会话，则当查询超过了行限制时，仅返回限制数目的行数，然后查询批处理、事务或会话中。Adaptive Server 显示与以下内容类似的消息：

```
Row count exceeded limit of 50.
Transaction has been aborted.
```

row_count 限制类型在执行时应用于所有 select 语句。执行前，不能限制估计返回的行数。

使用 *@@rowcount* 全局变量来帮助您确定行计数的适当限制。运行典型查询后，选择此变量可以了解查询返回了多少行。

行计数限制应用于从打开游标到关闭游标的时间范围内通过游标返回的累计行数。优化程序在每次打开游标时重新计算 row_count 限制。

row_count 限制类型的作用域

约束行计数的资源限制仅能应用于单个查询，而不能应用于查询批处理或事务返回的累计行数。请参见第 9 页的“确定资源限制的范围”。

设置 tempdb 空间使用限制

tempdb_space 资源限制将限制 tempdb 数据库在单个会话过程中可拥有的页数。如果用户超过指定限制，则会话将被终止，或者批处理或事务将中止。

对于以并行方式执行的查询，tempdb_space 资源限制在并行线程中等量分布。例如，如果 tempdb_space 资源限制设置为 1500 页，而且用户用三路并行方式执行下列命令，那么每个并行线程最多可在 tempdb 中创建 500 页：

```
select into #temptable from partitioned_table
```

系统管理员或数据库管理员使用 sp_add_resource_limit 设置 tempdb_space 限制，使用 sp_drop_resource_limit 删除 tempdb_space 限制。

限制空闲时间

空闲时间是指连接处于非活动状态并等待用户输入的秒数。即使连接处于非活动状态，它仍使用服务器资源，并且可能还会占用资源（例如锁），这能会阻止其它活动进程在同一服务器上运行。

idle_time 允许您为空闲连接设置时间限制。如果连接的空闲时间超出了设置的限制，Adaptive Server 将停止运行该连接的进程或发出警告。

语法为：

sp_add_resource_limit user, *application*, *time_range*, idle_time , *kill_time*, *enforcement_time*, *action*, *scope*

例如，以下示例为用户“sa”的 isql 连接创建新限制：

```
sp_add_resource_limit sa, isql, 'at all times',
idle_time, 10, 2, 4, 8
```

请参见《参考手册：过程》。

创建资源限制

可以使用 sp_add_resource_limit 创建新的资源限制：

sp_add_resource_limit *name*, *appname*, *rangename*, *limittype*, *limit_value*, *enforced*, *action*, *scope*

请参见《参考手册：过程》中的 sp_add_resource_limit。

资源限制示例

本节包括设置资源限制的示例。

示例

示例 1　此示例创建的资源限制由于其 name 参数设置为 NULL，所以可应用于 payroll 应用程序的所有用户：

```
sp_add_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time, 120, 2, 1, 2
```

该限制在 tu_wed_7_10 时间范围内有效。限制类型 elapsed_time 的值被设置为 120 秒。由于 elapsed_time 仅在执行时强制进行，所以 *enforced* 参数被设置为 2。*action* 参数被设置为 1，即采取的行动是发出警告。最后一个参数将限制的 *scope* 设置为 2，表示查询批处理。因此，如果执行查询批处理的经历时间超过了 120 秒，Adaptive Server 将发出警告。

示例 2　此示例创建的资源限制可应用于 saturday_night 时间范围内，由“joe_user”运行的所有即席查询和应用程序。

```
sp_add_resource_limit joe_user, NULL, saturday_night,
    row_count, 5000, 2, 3, 1
```

如果查询 (*scope* = 1) 返回的行多于 5000 行，Adaptive Server 将中止该事务 (*action* = 3)。此资源限制在执行时强制进行 (*enforced* = 2)。

示例 3　此示例创建的资源限制也可应用于“joe_user”运行的所有即席查询和应用程序：

```
sp_add_resource_limit joe_user, NULL, "at all times",
    io_cost, 650, 1, 3, 1
```

但是，此资源限制指定缺省时间范围“所有时间”。当优化程序估计查询 (scope = 1) 的 *io_cost* 超过指定值 650 时，Adaptive Server 将中止事务 (*action* = 3)。此资源限制在执行前强制进行 (*enforced* = 1)。

注释　虽然在当前事务达到其时间限制时 Adaptive Server 会终止该事务，但在您发出另一个 SQL 命令或批处理之前，不会收到 1105 错误消息；换句话说，只有当您再次尝试使用该连接时才显示该消息。

获得关于现有限制的信息

使用 sp_help_resource_limit 可获得关于现有资源限制的信息。

请参见《参考手册：过程》中的 sp_help_resource_limit 。

列出所有现有资源限制

使用没有任何参数的 sp_help_resource_limit 时，Adaptive Server 列出服务器内的所有资源限制。例如：

```
                sp_help_resource_limit
name      appname rangename rangeid limitid limitvalue enforced  action scope
----      ------- --------- ------- ------- ---------- --------  ------ -----
NULL      acctng  evenings        4       2        120         2       1     2
stein     NULL    weekends        1       3       5000         2       1     1
joe_user  acctng  bus_hours       5       3       2500         2       2     1
joe_user  finance bus_hours       5       2        160         2       1     6
wong      NULL    mornings        2       3       2000         2       1     1
wong      acctng  bus_hours       5       1         75         1       3     1
```

rangeid 列显示 systimeranges.id 中与 rangename 列中的名称相对应的值。limitvalue 列报告由 sp_add_resource_limit 或 sp_modify_resource_limit 设置的值。表 1-2 显示 limitid、enforced、action 和 scope 列中值的含义。

表 1-2：sp_help_resource_limit 的输出值

列	含义	值
limitid	是什么类型的限制？	1 — I/O 开销 2 — 经历时间 3 — 行计数
enforced	何时强制进行限制？	1 — 执行前 2 — 执行中 3 — 上述两种情况
action	超过限制时将采取什么措施？	1 — 发出警告 2 — 中止查询批处理 3 — 中止事务 4 — 取消会话
scope	限制的作用域是什么？	1 — 查询 2 — 查询批处理 4 — 事务 6 — 查询批处理加上事务

如果系统管理员在执行 sp_help_resource_limit 时指定一个登录名，Adaptive Server 将列出该登录的所有资源限制。输出不仅显示指定用户特定的资源限制，而且还显示与特定应用程序相关的所有用的所有资源限制，因为指定的用户也包含在所有用户中。

例如，下面的输出结果显示了应用于“joe_user”的所有资源限制。由于为 acctng 应用程序的所有用户定义了资源限制，因此输出结果中也包括了此限制。

```
                sp_help_resource_limit joe_user
name      appname rangename rangeid limitid limitvalue enforced  action scope
----      ------- --------- ------- ------- ---------- --------  ------ -----
NULL      acctng  evenings        4       2        120         2       1     2
joe_user acctng  bus_hours       5       3       2500         2       2     1
joe_user finance bus_hours       5       2        160         2       1     6
```

修改资源限制

使用 sp_modify_resource_limit 来指定一个新的限制值或超过限制时应采取的新措施，或指定两者。不能更改对其应用了限制的登录名或应用程序，也不能指定新的时间范围、限制类型、强制时间或作用域。

sp_modify_resource_limit 的语法为：

sp_modify_resource_limit *name*, *appname*, *rangename*, *limittype*, *limitvalue*, *enforced*, *action*, *scope*

请参见《参考手册：过程》中的 sp_modify_resource_limit。

删除资源限制

使用 sp_drop_resource_limit 来删除 Adaptive Server 中的资源限制。

语法为：

sp_drop_resource_limit {*name* , *appname* } [, *rangename*, *limittype*, *enforced*, *action*, *scope*]

指定足够的信息以唯一确定限制。必须为 *name* 或 *appname* 指定一个非空值。

请参见《参考手册：过程》中的 sp_drop_resource_limit 。

资源限制的优先级

Adaptive Server 为时间范围和资源限制提供优先规则。

时间范围

对于当前活动的时间范围内的每个登录会话，每种不同的限制类型、强制时间和作用域的组合只能有一个活动的限制。确定活动限制的优先规则如下：

- 如果没有为“所有时间”范围内或当前活动的时间范围内的登录 ID 定义限制，则没有活动的限制。
- 如果为“所有时间”范围和特定时间范围的登录都定义了限制，则用于特定时间范围的限制优先。

资源限制

由于确定资源限制要使用用户的登录名和/或应用程序名，所以 Adaptive Server 在扫描 sysresourcelimits 表以查找适用于登录会话的限制时，遵守预定义的搜索优先顺序。匹配的有序登录名和应用程序名对的优先级为：

级别	登录名	应用程序名
1	joe_user	payroll
2	NULL	payroll
3	joe_user	NULL

如果在给定的优先级别中发现了一个或多个匹配项，则不会搜索其它级别。这样可防止用于不同登录/应用程序组合的类似限制之间出现冲突。

如果在所有级别都没有发现匹配项，则不对会话进行限制。

第 2 章

镜像数据库设备

磁盘镜像

磁盘镜像可以在介质故障时提供不间断的恢复。 disk mirror 命令可以复制 Adaptive Server 数据库设备，即所有写入设备的操作都将被复制到一个单独的物理设备上。如果一台设备失败，另一台设备还会包含所有务的最新副本。

当对镜像设备的读取或写入操作失败时， Adaptive Server 将“取消镜像”出现错误的设备，并显示错误消息。 Adaptive Server 将继续在取消镜像的状态下运。

确定镜像对象

确定镜像设备时，必须衡量如系统停机时间成本、可能出现性能降低和存储介质成本等因素。考虑这些问题有助于确定镜像的对象—只镜像事务日、镜像服务器上的所有设备，还是镜像所选设备。

注释 不能镜像转储设备。

应该镜像所有缺省数据库设备，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。

除了镜像用户数据库设备外，还应该将事务日志放在单独的数据库设备上。若要获得更好的保护，请镜像用于事务日志的数据库设备。

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。也可以使用 alter database 再添加一个设备，然后运行 sp_logdevice 。

在成本和性能之间做出权衡时，应考虑以下问题：

- 快速恢复 — 可在对 master 数据库和用户数据库（包括日志）进行镜像后实现不间断恢复，而且无需重新装载事务日志即可恢复。
- 存储空间 — 立即恢复需要完全冗余（所有数据库和日志均已镜像），这将占用磁盘空间。
- 对性能的影响 — 镜像用户数据库（如第 21 页的图 2-2 和 第 22 页的图 2-3 所示）将增加向两个磁盘写入事务所需时间。

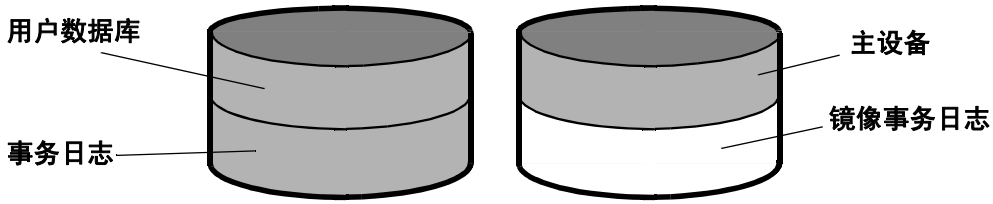
使用最小物理磁盘空间进行镜像

图 2-1 说明了在出现硬件故障的情况下进行数据库恢复的“最低保障配置”。主设备和用户数据库事务日志的镜像分别存储在一个物理磁盘的单独分区中。另一个磁盘在两个单独的磁盘分区中存储用户数据库及其事务日志。

如果带有用户数据库的磁盘出现故障，则可以在新磁盘上从备份和已镜像的事务日志中恢复用户数据库。

如果具有主设备的磁盘发生故障，则可以从 master 数据库的数据库转储中恢复主设备，并重新镜像用户数据库的事务日志。

图 2-1 ： 使用最小物理磁盘空间的磁盘镜像



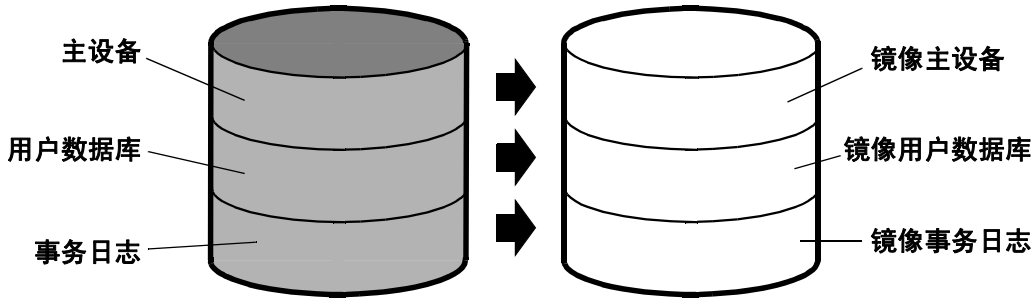
此配置将所需的磁盘存储空间量降到最低，但是事务日志的镜像可以确保完全恢复。然而，此配置不提供不间断恢复，因为 master 和用户数据库没有被镜像并且必须从备份中恢复。

不间断恢复的镜像

图 2-2 表示另一个镜像配置。在这种情况下，主设备、用户数据库和事务日志都存储在同一个物理设备的不同分区中，并且都镜像到另一个物理设备中。

图 2-2 中的配置可在出现硬件故障时提供不间断恢复。对主磁盘中的 master 数据库、用户数据库和日志的工作副本都进行了镜像，因此任何磁盘出现故障都不会中断 Adaptive Server 用户。

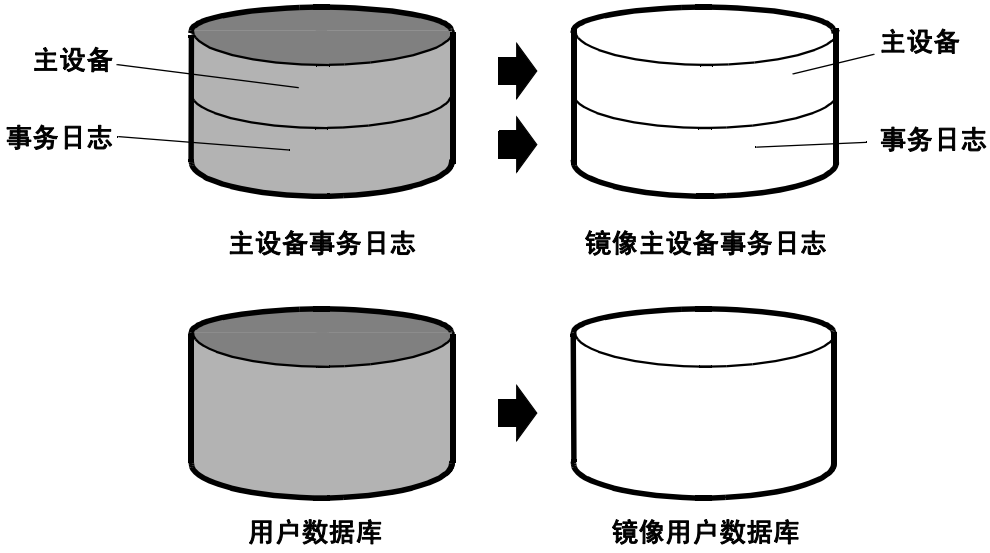
图 2-2：快速恢复的磁盘镜像



利用该配置，所有数据被写入两次，一次写入到主磁盘，另一次写入到镜像磁盘。包含很多写操作的应用程序在使用磁盘镜像时可能会比不使用时速慢。

图 2-3 说明了另一个冗余程度更高的配置。在此配置中，所有三个数据库设备都被镜像，但配置使用四个磁盘而不是两个。此配置在写入事务期间提高性能，因为数据库事务日志与用户数据库没有存储在同一设备中，所以系统访问两个磁盘的磁头移动会更少。

图 2-3 ：磁盘镜像：在单独的磁盘上保存事务日志



不禁用镜像的条件

Adaptive Server 只在镜像设备中遇到 I/O 错误时才禁用镜像。例如，如果 Adaptive Server 尝试在磁盘坏块上写入数据，则产生的错误将禁用设备镜像。然而处理将在未受影响的镜像上继续进行而不中断。

如果出现以下情况，不会禁用镜像：

- 设备上未使用的块损坏。只有到访问该坏块时，Adaptive Server 才会检测到 I/O 错误并禁用镜像。
- 设备上的数据被覆盖。如果已镜像的设备作为 UNIX 文件系统装入，并且 UNIX 覆盖了 Adaptive Server 数据，则可能发生这种情况。这将造成数据库损坏，镜像不会被禁用，因为 Adaptive Server 不会遇到 I/O 错误。

- 错误数据被写入主设备和辅助设备中。
- 活动设备上的文件权限被更改。一些系统管理员可能试图通过更改一个设备上的权限来测试磁盘镜像，希望触发 I/O 故障并取消镜像其它设备。但是 UNIX 操作系统在设备打开后不检查它的权限，所以在下次启动设备前不会发生 I/O 故障。

磁盘镜像不用于检测或防止数据库损坏。上述的某些情况可能会造成损害，所以应该定期在所有数据库上运行如 dbcc checkalloc 和 dbcc checkdb 等一致性检查。请参见第 11 章“检查数据库一致性”。

磁盘镜像命令

disk mirror 、 disk unmirror 和 disk remirror 命令可控制磁盘镜像。所有命令都可在设备使用中发出，这样可以在数据库正在使用时启动或停止数据库设备镜像。

注释　disk mirror 、 disk unmirror 和 disk remirror 变更了 master 数据库中的 sysdevices 表。在发出这些命令中的任何命令后，需转储 master 数据库以确保在 master 损坏的情况下可以恢复。

初始化镜像

disk mirror 将启动磁盘镜像。不要使用 disk init 初始化镜像设备。数据库设备及其镜像组成一个逻辑设备。 disk mirror 命令将镜像名添加到 sysdevices 表的 mirrorname 列中。

disk mirror 的语法为：

```
disk mirror
    name = "device_name" ,
    mirror = "physicalname"
    [ , writes = { serial | noserial }]
```

取消镜像设备

两个物理设备中有一个产生故障后，磁盘镜像将自动失效。如果对已镜像设备的读取或写入操作不成功， Adaptive Server 将显示错误消息。 Adaptive Server 继续运行，但不进行镜像。必须重镜像磁盘才可以重新启动镜像。

使用 disk unmirror 命令可在硬件维护期间停止镜像进程：

```
disk unmirror
    name = "device_name"
    [, side = { "primary" | secondary }]
    [, mode = { retain | remove }]
```

对系统表的影响

mode 选项会更改 sysdevices 中的 status 列，以指示镜像已被禁用（请参见《系统管理指南，卷 1 》中的第 7 章“初始化数据库设备”）。它对 sysdevices 中的 phyname 和 mirrorname 列的影响还取决于 side 参数，如表 2-1 所示。

表 2-1 ： 磁盘镜像命令的 mode 和 side 选项的影响

		side	
		primary	secondary
模式	remove	mirrorname 中的名称已移向 phyname 并且 mirrorname 设置为空值； status 已更改	mirrorname 中的名称被删除； status 已更改
	retain	名称未更改； status 发生更改，指示失效的设备	

本示例挂起主设备操作：

```
disk unmirror
  name = "tranlog",
  side = "primary"
```

重新启动镜像

使用 disk remirror 重新启动由于设备故障或因使用 disk unmirror 而挂起的镜像进程。语法为：

```
disk remirror
    name = "device_name"
```

此命令将数据库设备复制到它的镜像中。

waitfor mirrorexit

因为磁盘故障可以削弱系统安全性，所以在磁盘取消镜像后，可以将 waitfor mirrorexit 命令包含在应用程序中以执行特定任务：

```
begin
    waitfor mirrorexit
   commands to be executed
end
```

这些命令取决于应用程序。如果磁盘已被取消镜像，应在执行更新的应用程序中添加某些警告或使用 sp_dboption 将某些数据库设置为只读。

注释　Adaptive Server 只有在尝试向镜像设备执行 I/O 时才会发现设备已被取消镜像。在已镜像的数据库中，这种情况在检查点或当必须将 Adaptive Server 缓冲区入磁盘时发生。在镜像日志中， I/O 在当进程向日志（包括任何已提交的执行数据修改的事务）、检查点或数据库转储写入时发生。

---

waitfor mirrorexit 、输出到主控台的错误消息和有关镜像故障的错误日志仅由这些事件激活。

镜像主设备

在 UNIX 环境中，如果选择对包含 master 数据库的设备进行镜像，则必须编辑 Adaptive Server 的 runserver 文件，以便在服务器启动时启动镜像设备。

在 UNIX 中，应添加 -r 标志和镜像设备的名称：

```
dataserver -d /dev/rsd1f -r /dev/rs0e -e/sybase/install/errorlog
```

有关在 Windows 上镜像主设备的信息，请参见《实用程序指南》。

获取有关设备和镜像的信息

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 。

磁盘镜像教程

本节说明磁盘镜像命令的使用及其对 master..sysdevices 中所选列的影响。 sysdevices 中各条目的 status 编号及其十六进制的对应编号放在括号中：

1 使用以下命令初始化新的测试设备：

```
disk init name = "test",
physname = "/usr/sybase/test.dat",
size=5120
```

这将在 master..sysdevices 的列中插入以下值：

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   NULL                    16386
```

状态 16386 表示设备是物理设备 (2, 0x00000002)，并且任何写操作都写入 UNIX 文件 (16384, 0x00004000) 。因为 mirrorname 列是空值，所以在此设备上不启用镜像。

2 使用以下命令镜像测试设备：

```
disk mirror name = "test",
mirror = "/usr/sybase/test.mir"
```

这会将 master..sysdevices 列更改为：

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

状态 17122 表示在此设备上镜像当前处于启用状态 (512, 0x00000200) 。读取被镜像 (128, 0x00000080)，写入被镜像到 UNIX 文件设备 (16384, 0x00004000)，设备被镜像 (64, 0x00000040)，并且是串行的 (32, 0x00000020) 。该设备是一个物理磁盘 (2, 0x00000002) 。

3 禁用镜像设备（辅助设备），但保留该镜像：

```
disk unmirror name = "test",
side = secondary, mode = retain
```

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  18658
```

状态 18658 表示设备被镜像 (64, 0x00000040)，并且镜像设备已被保留 (2048, 0x00000800)，但镜像已被禁用（ 512 位关闭）并且仅使用主设备（ 256 位关闭）。读被镜像 (128, 0x00000080)，写入被镜像到 UNIX 文件 (16384, 0x00004000)，并且是串行的 (32, 0x00000020) 。该设备是一个物理磁盘 (2, 0x00000002) 。

4　重新镜像测试设备：

```
disk remirror name = "test"
```

这会将 master..sysdevices 列重新设置为：

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

状态 17122 表示在此设备上镜像当前处于启用状态 (512, 0x00000200)。读取被镜像 (128, 0x00000080)，写入被镜像到 UNIX 文件设备 (16384, 0x00004000)，设备被镜像 (64, 0x00000040)，并且是串行的 (32, 0x00000020)。该设备是一个物理磁盘 (2, 0x00000002)。

5　禁用测试设备（主设备），但保留该镜像：

```
disk unmirror name = "test",
side = "primary", mode = retain
```

这会将 master..sysdevices 列更改为：

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  16866
```

状态 16866 表示设备被镜像 (64, 0x00000040)，但镜像已被禁用（512 位关闭）并且仅使用辅助设备 (256, 0x00000100)。读取被镜像 (128, 0x00000080)，写入被镜像到 UNIX 文件 (16384, 0x00004000)，并且是串行的 (32, 0x00000020)。该设备是一个物理磁盘 (2, 0x00000002)。

6　重新镜像测试设备：

```
disk remirror name = "test"
```

这会将 master..sysdevices 列重新设置为：

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

状态 17122 表示在此设备上镜像当前处于启用状态 (512, 0x00000200)。读取被镜像 (128, 0x00000080)，写入被镜像到 UNIX 文件设备 (16384, 0x00004000)，设备被镜像 (64, 0x00000040)，并且是串行的 (32, 0x00000020)。该设备是一个物理磁盘 (2, 0x00000002)。

7 禁用测试设备（主设备），并删除该镜像：

```
disk unmirror name = "test", side = "primary",
mode = remove
```

这会将 master..sysdevices 列更改为：

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   NULL                    16386
```

状态 16386 表示设备是物理设备 (2, 0x00000002)，并且任何写操作都写入 UNIX 文件 (16384, 0x00004000) 。因为 mirrorname 列是空值，所以在此设备上不启用镜像。

8 删除测试设备，结束教程：

```
sp_dropdevice test
```

这将从 master..sysdevices 中删除测试设备的所有条目。

调整磁盘大小和磁盘镜像

只有在镜像被永久禁用时，才能使用 disk resize。如果尝试在已镜像的设备上运行 disk resize，将显示：

```
disk resize can proceed only when mirroring is
permanently disabled.Unmirror secondary with mode =
'remove' and re-execute disk resize command.
```

当镜像只是被暂时禁用时，将会出现两种情形：

- 辅助设备暂时禁用而主设备处于活动状态时发生的错误与上面的相同。
- 主设备暂时禁用而辅助设备处于活动状态时出现以下错误消息：

```
disk resize can proceed only when mirroring is
permanently disabled.Unmirror primary with mode
= 'remove' and re-execute the command.
```

增加已镜像设备的大小：

1 对该设备永久禁用镜像。

2 增加主设备大小。

3 物理删除镜像设备（文件情形下）。

4 重建镜像。

第 3 章

配置内存

确定可供 Adaptive Server 使用的内存

可用内存越多， Adaptive Server 可用于内部缓冲区和高速缓存的资源就越多。有足够的内存供高速缓存使用可减少 Adaptive Server 必须从磁盘读取数据或过计划的次数。

将 Adaptive Server 配置为使用计算机上最大的可用内存量不会影响性能。不过，应先评估系统上的其它内存需求，然后再将 Adaptive Server 配置为只使用仍然可用的剩余内存。如果 Adaptive Server 不能获取为其配置的内存，可能无法启动。

确定系统中可供 Adaptive Server 使用的最大内存量：

1 确定计算机系统中的物理内存总量。

2 从总的物理内存中减去操作系统所需的内存。

3 减去必须运行在同一台计算机上的与 Adaptive Server 相关的其它软件 （例如， Backup Server）所需的内存。

4 如果计算机并非专供 Adaptive Server 使用，应减去其它系统使用所需的内存。

例如，减去将在 Adaptive Server 计算机上运行的所有客户端应用程序要使用的内存。基于窗口的系统 （例如 X Windows）需占用大量内存，因此，与 Adaptive Server 在同一台计算机上使用这类系统时，会影响 Adaptive Server 的性能。

减去操作系统和其它应用程序的内存需求量后剩余的就是可供 Adaptive Server 使用的总内存。

max memory 配置参数的值指定可为 Adaptive Server 配置的最大内存量。请参见第 39 页的 “影响内存分配的配置参数”。

Adaptive Server 如何分配内存

所有数据库对象页都按**逻辑页大小**来确定大小，逻辑页大小在创建新主设备时指定。所有数据库以及每个数据库中的所有对象都使用相同的逻辑页大小。Adaptive Server 逻辑页的大小（24、8 或 16K）决定服务器的空间分配情况。每个分配页、对象分配映射 (OAM) 页、数据页、索引页、文本页等都建立在逻辑页上。例如，如果 Adaptive Server 的逻辑页大小为 8K，则这些类型页的每一页大小均为 8K。所有这些页都占用由逻辑页大小所指定的整个大小。逻辑页越大，所允许创建的行就越大，这会提高性能，因为 Adaptive Server 在每次读取一页时可访问到更多的数据。例如， 16K 页容纳的数据量是 2K 页的 8 倍， 8K 页容纳的数据量是 2K 页的 4 倍，对于所有逻辑页大小，可以依此类推。

逻辑页大小是一种全服务器范围的设置；在同一个服务器内，数据库的逻辑页大小必须相同。所有表的大小应适当，以使行宽不大于服务器的当前页小。也就是说，一行不能占据多页。

Adaptive Server 按扩充为对象 （表、索引、文本页链）分配空间，每个扩充为八个逻辑页，而与逻辑页大小的配置无关。也就是说，如果服务器的逻辑页被配置为 2K，它便为每个对象分配一个 16K, 的扩充；如果服务器的逻辑页被配置为 16K，它便为每个对象分配一个 128K 的扩充。

对于系统表也是如此。如果服务器有很多小表，当服务器使用较大的逻辑页时，空间消耗就会很大。例如，对于逻辑页配置为 2K 的服务器， systypes （约有 31 个短行，一个聚簇索引和一个非聚簇索引）保留 3 个扩充 （48K 内存）。如果将服务器迁移到使用 8K 页，为 systypes 保留的空间仍为 3 个扩充，但内存为 192K。对于逻辑页配置为 16K, 的服务器， systypes 需要 384K 的磁盘空间。对于小表，如果服务器使用较大逻辑页，则最后一个扩充中的未用空间会非常大。

较大的页大小也会对数据库产生影响。每个数据库包含系统目录及其索引。如果从较小的逻辑页迁移到较大的逻辑页，则必须考虑每个数据库所需的盘空间量。表 3-1 列出了基于每一逻辑页大小的数据库的最小大小。

表 3-1：最小数据库大小

逻辑页大小	最小数据库大小
2K	2MB
4K	4MB
8K	8MB
16K	16MB

磁盘空间分配

逻辑页大小与内存分配页大小不同。无论逻辑页大小多大（可能为 2、 4、 8 或 16K），内存分配页大小始终是 2K。大多数与内存相关的配置参数都使用 2K 作为其内存页大小的单位。这些配置参数包括：

- max memory
- total logical memory
- total physical memory
- procedure cache size
- size of process object heap
- size of shared class heap
- size of global fixed heap

较大逻辑页大小和缓冲区

Adaptive Server 以逻辑页为单位分配缓冲池。例如，在使用 2K 逻辑页的服务器上，为缺省数据高速缓存分配 8MB 空间。这就大约建立了 2048 个缓冲区。如果将相同的 8MB 空间分配给使用 16K 逻辑页大小的服务器的缺省数据高速缓存，那么缺省数据高速缓存大约为 256 个缓冲区。在繁忙系统上，此小数量缓冲区可导致缓冲区始终处于清洗区，减慢了要求干净缓冲区的任务的执行速度。通常，若要在较大页大小上获得与 2K 逻辑页大小相同的缓冲区管特性，请根据较大的页大小来确定高速缓存的大小。因此，如果将逻辑页大小增大为原来的四倍，那么高速缓存和缓冲池大小也应为原来大小的四。

Adaptive Server 通常动态地分配内存并在需要时为行处理分配内存，且为这些缓冲区分配最大大小 （即使大缓冲区不是必要的）。这些内存管理要求可能导致 Adaptive Server 在处理宽字符数据时损失边缘性能。

堆内存

堆内存池是在启动时创建的内部内存池，某些任务用它来根据需要动态分配内存。此内存池由需要堆栈中大量内存的任务 （如使用宽列的任务）使用例如，如果进行宽列或行更改，则此任务所用的临时缓冲区可大至 16K，但这个值太大，无法从堆栈中分配。 Adaptive Server 在该任务的运行期间动态地分配和释放内存。堆内存池显著减少每一任务的预声明堆栈大小，同时提高服务器中的内存使用率。任务结束后，任务所用的堆内存返回堆内存池中

使用 heap memory per user 配置参数可以设置堆内存。

每个用户的堆内存以字节计。缺省情况下，内存量设置为 4096 字节。以下示例将该值设置为每个用户 100 字节：

```
sp_configure 'heap memory per user', 100
```

也可以以字节数指定每一用户的内存量。例如，以下示例指定为每个用户连接分配 4K 字节的堆内存（“0”是指定单位值时 sp_configure 需要的占位符）：

```
sp_configure 'heap memory per user', 0, "4K"
```

在初始配置 Adaptive Server 时，为堆内存留出 1MB 的空间。附加堆内存是为所有的用户连接和工作进程（服务器的配置方式）分配的，因而在服务器启动时以下配置参数将影响可用堆内存量：

- number of user connections
- number of worker processes

全局变量 *@@heapmemsize* 以字节为单位报告堆内存池的大小。

计算堆内存

若要计算 Adaptive Server 留出了多少堆内存，请使用以下公式（Adaptive Server 会为内部结构保留少量内存，因此这些数字在不同的节点上会有所不同）：

((1024 X 1024) + (*heap memory in bytes*)* (number of user connections + number of worker processes))

公式中的第一项 (1024 X 1024) 是堆内存池的初始大小 1MB。Adaptive Server 会为内部结构保留少量内存。

例如，如果服务器配置为：

- heap memory per user — 4K
- number of user connections — 25 （缺省值）
- number of worker processes — 25 （缺省值）

@@heapmemsize 报告 1378304 字节。

用上面的公式估算的值为：

((1024 X 1024) + (4 X 1024 X 50)) = 1253376

现在，如果增大 number of user connections，则堆内存池的大小也相应地增大：

```
sp_configure 'user connections', 100
```

@@heapmemsize 报告 1716224 字节。

此示例中估算值为：

((1024 X 1024) + (4 * 1024 * (100 + 25)) = 1560576

如果应用程序失败并显示以下消息：

```
There is insufficient heap memory to allocate %ld
bytes.Please increase configuration parameter 'heap
memory per user' or try again when there is less
activity on the system.
```

通过增大下列参数之一来增大服务器的可用堆内存：

- heap memory per user
- number of user connections
- number of worker processes

内存池的大小取决于用户连接数。 Sybase® 建议将每一用户的堆内存至少设置为逻辑页大小的三倍。

Sybase 建议，在增大 number of user connections 或 number of worker processes 之前，应首先增大 heap memory per user 配置选项。增大 number of user connections 和 number of worker processes 会消耗其它资源的系统内存，这可能需要增大服务器的最大内存。

请参见 《系统管理指南，卷 1》的第 5 章 “设置配置参数”。

Adaptive Server 如何使用内存

内存在 Adaptive Server 中以总逻辑内存或物理内存的形式存在：

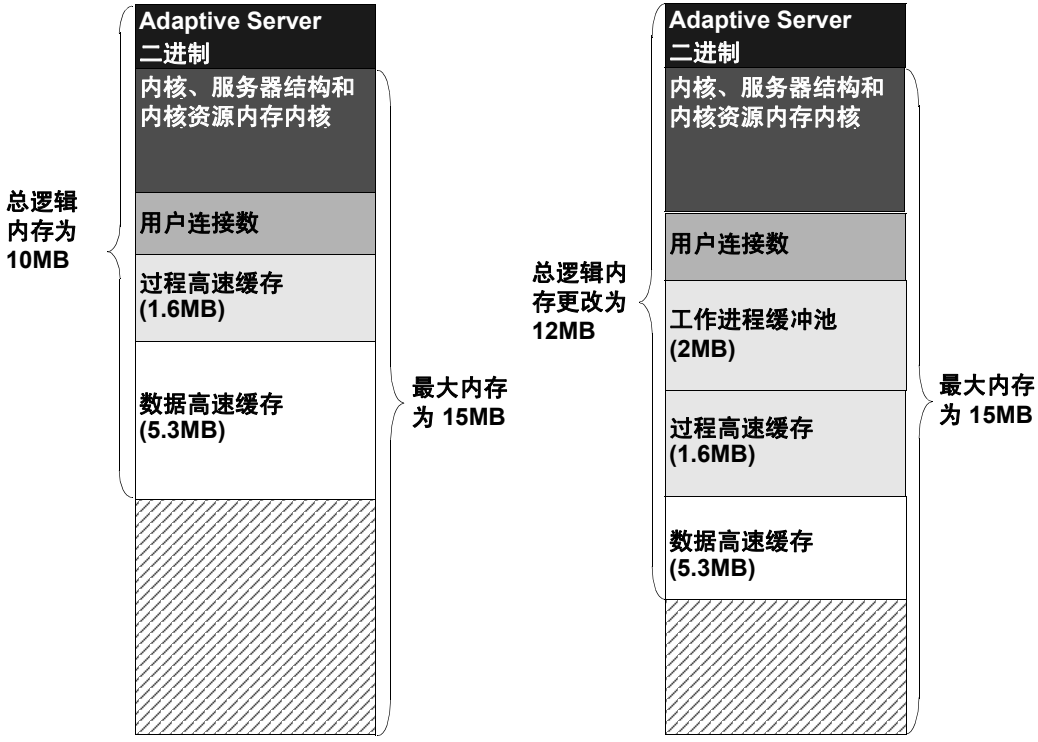
- 总逻辑内存 — 是所有 sp_configure 参数所需的内存总和。总逻辑内存必须始终可供使用，但在某特定时刻可能是空闲的。总逻辑内存值可根据配置参数值的变化而变化。
- 总物理内存 — 是 Adaptive Server 中所有共享内存段的总和。即，总物理内存是 Adaptive Server 在某特定时刻所用的内存量。可使用只读配置参数 total physical memory 检验此值。 total physical memory 的值只能增加，因为 Adaptive Server 在分配内存池后不会再将其减小。可通过更改配置参数并重新启动 Adaptive Server 来减小总物理内存量。

Adaptive Server 启动时，它将分配：

- Adaptive Server 用于不可配置的数据结构的内存。
- 用户可配置的所有参数的内存，包括数据高速缓存、过程高速缓存、内核资源内存和缺省数据高速缓存。

图 3-1 说明 Adaptive Server 在更改了一些内存配置参数后如何分配内存：

图 3-1：内存配置更改



在 2MB 的工作进程缓冲池被添加到 Adaptive Server 内存配置后，过程和数据高速缓存维持其原始的配置大小；分别为 1.6MB 和 5.3MB。因为 max memory 比 total logical memory 大小大 5MB，所以可轻松容纳添加的内存池。如果新工作进程缓冲池使服务器的大小超过了 max memory 的限制，那么发出的任何增加工作进程缓冲池的命令都将失败。如果出现这种情况，将在 sp_configure 故障消息中指出新配置所需的总逻辑内存。将 max memory 的值设置为大于新配置所需的 total logical memory 的值。然后重试 sp_configure 请求。

注释　max memory 和 total logical memory 的值不包括 Adaptive Server 二进制程序。

缺省数据高速缓存和过程高速缓存的大小对总体性能有显著影响。请参见 《性能和调优系列：基础知识》的第 5 章 “内存使用和性能”，了解有关优化过程高速缓存大小的建议。

确定 Adaptive 所需的内存量

Adaptive Server 启动所需的总内存为 *sum of all memory configuration parameters* 加上 *size of the procedure cache* 加上 *size of the buffer cache,* 的总和，其中过程高速缓存的大小和缓冲区高速缓存的大小用整数而不是百分比表示。过程高速缓存大小和缓冲区高速缓存大小不 取决于配置的总存。可独立配置过程高速缓存大小和缓冲区高速缓存大小。使用 sp_cacheconfig 来获取有关信息，如每一高速缓存的总大小、每一高速缓存的缓冲池数、每一缓冲池的大小，等等。

使用 sp_configure 可以确定 Adaptive Server 在给定时刻使用的总内存量：

```
1> sp_configure "total logical memory"

Parameter Name        Default   Memory Used   Config Value   Run Value
   Unit               Type
--------------        --------------- --- ----------- ------------ ---------
-------------------- ----------
total logical memory  33792    127550         63775         63775
   memory pages(2k)   read-only
```

“Memory Used”列的值用千字节表示，“Config Value”列的值用 2K 页表示。

“Config Value”列表示 Adaptive Server 运行时所使用的总逻辑内存。“Run Value”列显示当前 Adaptive Server 配置所消耗的总逻辑内存。运行此命令时，输出会不同，因为没有任何两个 Adaptive Server 的配置完全相同。

请参见 《参考手册：过程》。

确定 Adaptive Server 内存配置

系统启动过程中分配的内存总数是 Adaptive Server 所有配置需求所要求的内存总和。此值可从只读配置参数 total logical memory 获得。此值是由 Adaptive Server 计算得出的。配置参数 max memory 必须大于或等于 total logical memory。 max memory 表示用于满足 Adaptive Server 需要的内存量。

服务器启动过程中，在缺省情况下， Adaptive Server 会根据 total logical memory 的值分配内存。但是，如果已经设定了配置参数 allocate max shared memory，那么将根据 max memory 的值分配内存。配置参数 allocate max shared memory 允许系统管理员在服务器启动过程中，分配允许 Adaptive Server 使用的最大内存。

内存配置的要点是：

- 系统管理员应确定 Adaptive Server 可用的共享内存的大小，并将 max memory 设置为此值。
- 配置参数 allocate max shared memory 可在启动和运行时打开，以便用最少的共享内存段来分配全部共享内存，最多为 max memory。在某些平台上，共享内存段数量过多有导致性能下降的缺点。请检查操作系统文档，确定最佳的共享内存段数量。除非重新启动服务器，否则共享存段在分配后将不能释放。
- max memory 与 total logical memory 之间的差值确定可供过程高速缓存、语句高速缓存、数据高速缓存或其它配置参数使用的内存量。

  启动过程中 Adaptive Server 分配的内存量由 total logical memory 或 max memory 决定。如果将 alloc max shared memory 设置为 1，则 Adaptive Server 使用 max memory 的值。

  如果 total logical memory 或 max memory 太高：

  - 在计算机上的物理资源不足的情况下，Adaptive Server 可能无法启动。
  - 即使 Adaptive Server 确实启动了，操作系统页面错误率可能会显著提高，可能需要重新配置操作系统以解决此问题。

如果正在升级

在 Adaptive Server 12.5 之前的版本中， total logical memory、procedure cache percent 和 min online engines 的配置值自动计算 procedure cache size 和 number of engines at startup 的新值。 Adaptive Server 在升级期间计算缺省数据高速缓存的大小并将该值写入配置文件。如果计算得出的数据高速缓存或过程高速缓存的大小小于缺大小，那么它们被重置为缺省大小。

在升级期间， Adaptive Server：

- 将 max memory 设置为在配置文件中指定的 total logical memory 的值。如有必要，请重新设置 max memory 的值以符合资源要求。
- 将先前配置中的引擎数设置为 syb_default_pool 中的线程数。

使用 sp_configure 的 verify 选项可以检验对配置文件所做的任何更改，而不必重新启动 Adaptive Server：

```
sp_configure “configuration file”, 0, “verify”,
“full_path_to_file”
```

确定 Adaptive Server 可以使用的内存量

表 3-2 列出了 Adaptive Server 版本 12.0 以及更高版本的可寻址共享内存的上限值：

表 3-2：各种平台的可寻址内存极限

平台	32 位 Adaptive Server	64 位 Adaptive Server
HP-UX 11.x （PA-RISC 处理器）	2.75 千兆字节	16 EB[1]
IBM AIX 5.x	2.75 千兆字节	16 EB
Sun Solaris 8 （sparc 处理器）	3.78 千兆字节	16 EB
Sun Solaris 8 （Intel x86 处理器）	3.75 千兆字节	不适用
Red Hat Enterprise Linux（Intel x86 处理器)	2.7 千兆字节	不适用

1 1 艾字节 (EB) 等于 2^{60}，即 1024 PB。16 艾字节是一个理论极限值；实际值受系统上可用的总内存的限制。测试表明， Adaptive Server 的最大共享内存为 256GB。

2 使用 /3G 选项启动 Windows 可以使 Adaptive Server 最多使用 3 千兆字节的共享内存。请参见您的 Windows 文档。

注释　Adaptive Server 12.5 以及更高版本分配内存的方式与早期版本不同。这包含更改了与内存相关的现有配置参数和引入了新的内存参数。

每个操作系统都有缺省的最大共享内存段。确保将操作系统配置为允许分配至少与 max memory 相同大小的共享内存段。请参见针对所用平台的 《安装指南》。

影响内存分配的配置参数

设置 Adaptive Server 的内存配置时，可以通过 sp_configure 用绝对值指定每项内存要求。还可以用绝对值指定过程和缺省数据高速缓存的大小。

影响内存分配方式的配置参数有三个：max memory、 allocate shared memory 和 dynamic allocation on demand。

max memory

max memory 用于建立可分配给 Adaptive Server 的内存量的最大设置。将 max memory 的值设置为稍大于所需内存量的值，这样当 Adaptive Server 的内存需求增加时，就可利用这些附加的内存。

增加 max memory 的值时， sp_configure 将 max memory 设置为您所指定的值。然而，内存分配可能会在稍后进行。 Adaptive Server 分配由 max memory 指定的内存的方式取决于如何配置 allocate max shared memory 和 dynamic allocation on demand。

升级时，如果 max memory 的值不足， Adaptive Server 会自动增加 max memory 的值。Adaptive Server 的较新版本可能需要更多的内存，这是因为内部数据结构的大小已增加。

allocate max shared memory

allocate max shared memory 用于在启动时分配由 max memory 指定的所有内存，或用于在启动期间仅分配总逻辑内存规范所需的内存。

在某些平台上，如果分配给应用程序的共享内存段数量大于平台特定的最佳数量，则性能可能会出现某种程度的下降。如果出现这种情况，应将 max memory 设置为可供 Adaptive Server 使用的最大内存量。将 allocate max shared memory 设置为 1 并重新启动服务器。这将确保 Adaptive Server 在启动时以最小的段数分配 max memory 的所有内存。

例如，如果将 allocate max shared memory 设置为 0 （缺省值），将 max memory 设置为 500MB，但服务器配置在启动时只需要 100MB 内存，则 Adaptive Server 仅在需要附加内存时才分配剩余的 400MB 内存。然而，如果将 allocate max shared memory 设置为 1，则 Adaptive Server 在启动时将分配全部 500MB 内存。

如果将 allocate max shared memory 设置为 0，并且增大 max memory，则根据需要进行实际内存分配。如果将 allocate max shared memory 设置为 1，并且增大 max memory， Adaptive Server 将尝试立即分配内存。如果分配失败， Adaptive Server 会向错误日志中写入消息。

在启动时分配所有内存的好处是服务器在因附加的内存而重新调整时性能不会降低。然而，如果未能正确预知内存增长，且将 max memory 设置为大值，那么就可能会浪费物理内存。因为不能动态降低内存配置参数，所以同时考虑其它内存要求非常重要。

dynamic allocation on demand

dynamic allocation on demand 用于确定是在请求时立即分配内存资源，还是仅在需要时分配。如果将 dynamic allocation on demand 设置为 1，将根据需要更改内存分配；如果设置为 0，则在更改内存配置时分配已配置的内存。

例如，如果将 dynamic allocation on demand 设置为 1，并将 number of user connections 更改为 1024，则 total logical memory 为 1024 乘以每个用户的内存量。如果每一用户的内存量是 112K，则用于用户连接的内存是 112MB (1024 x 112)。

这是允许 number of user connections 配置参数使用的最大内存量。然而，如果只有 500 个用户连接到此服务器，则 number of user connections 参数所用的总物理内存量为 56MB (500 x 112)。

如果 dynamic allocation on demand 设置为 0，那么当您将 number of user connections 更改为 1024 时，会立即配置所有用户连接资源。

最好组织 Adaptive Server 的内存，使 total physical memory 的数量小于总逻辑内存量，而总逻辑内存量小于 max memory。通过将 dynamic allocation on demand 设置为 1，将 allocate max shared memory 设置为 0，可部分实现上述要求。

动态分配内存

Adaptive Server 动态分配物理内存，这意味着您可以更改 Adaptive Server 的内存配置，而不必重新启动服务器。

注释　Adaptive Server 不动态减少内存。准确地评估系统需求非常重要，因为如果降低了内存配置参数并要释放先前使用的物理内存，可能需要重新启动服务器。请参见第 41 页的 “动态降低内存配置参数”。

在以下情况下，请考虑更改 max_memory 配置参数的值：

- 更改计算机的 RAM 量。
- 更改计算机的使用模式。
- 由于 max_memory 不足而导致配置失败。

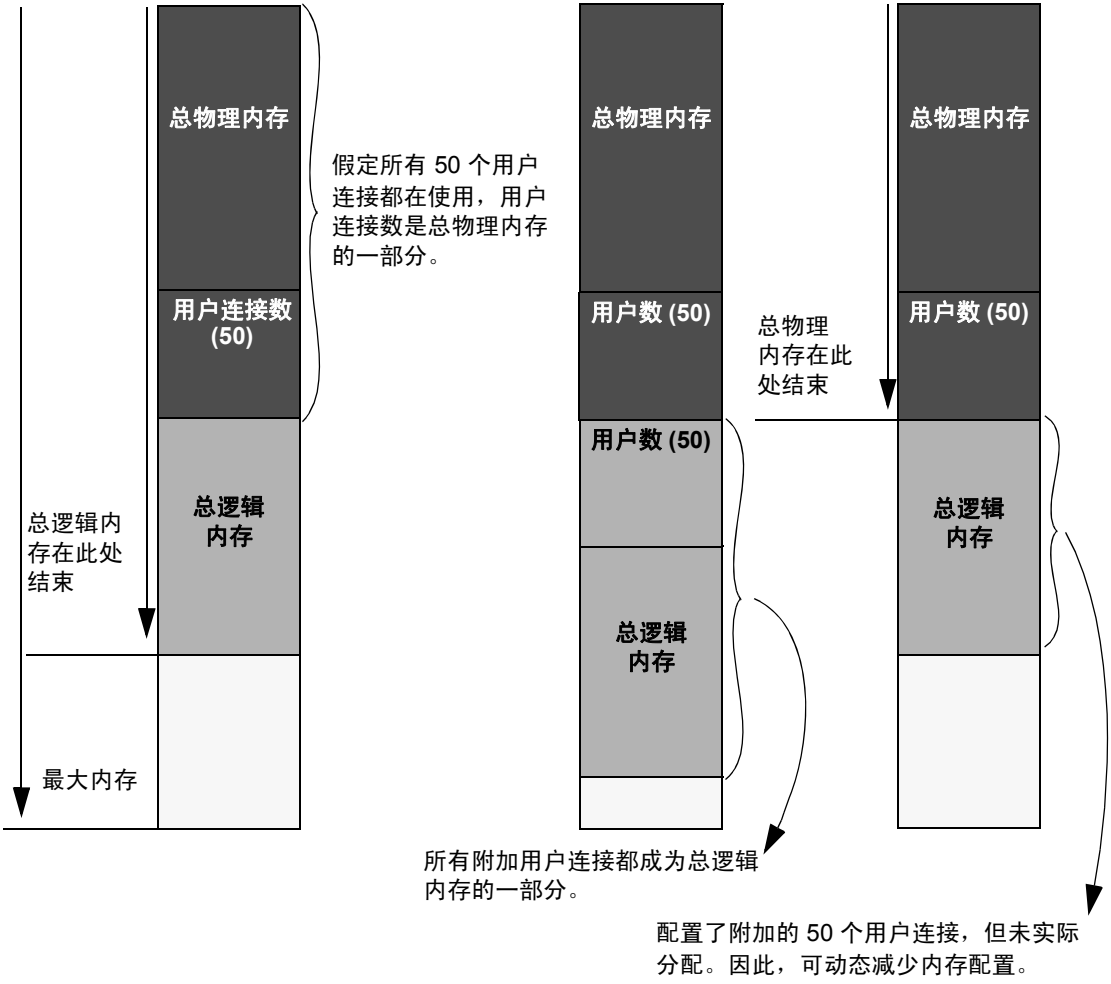
如果 Adaptive Server 不能启动

Adaptive Server 启动时，它必须获取分配给它的全部内存 （由 total logical memory 设置）。如果 Adaptive Server 因无法获得足够内存而不能启动，可通过降低消耗内存的配置参数的值来减少内存要求。可能还需要减少需要大量内存的它配置参数的值。重新启动 Adaptive Server 以使用新值。请参见 《系统管理指南，卷 1》的第 5 章 “设置配置参数”。

动态降低内存配置参数

如果将内存配置参数重新设置为较低值，不会动态释放使用的内存。若要了解如何减少内存配置的更改，请参见图 3-2 和图 3-3。

图 3-2：dynamic allocation on demand 设置为 1，无新用户连接



在图 3-2 中，因为 dynamic allocation on demand 设置为 1，所以仅在有事件触发对额外内存的需求时，才会使用内存。在此例中，这样的事件是当客户端试图登录到 Adaptive Server 中时，对附加用户接的请求。

可将 number of user connections 减小到一个大于或等于实际分配的用户连接数的数目，因为，当 dynamic allocation on demand 设置为 1，且用户连接请求没有实际增长时，不需要向服务器请求额外内存。

图 3-3：dynamic allocation on demand 设置为 1，有新用户连接登录

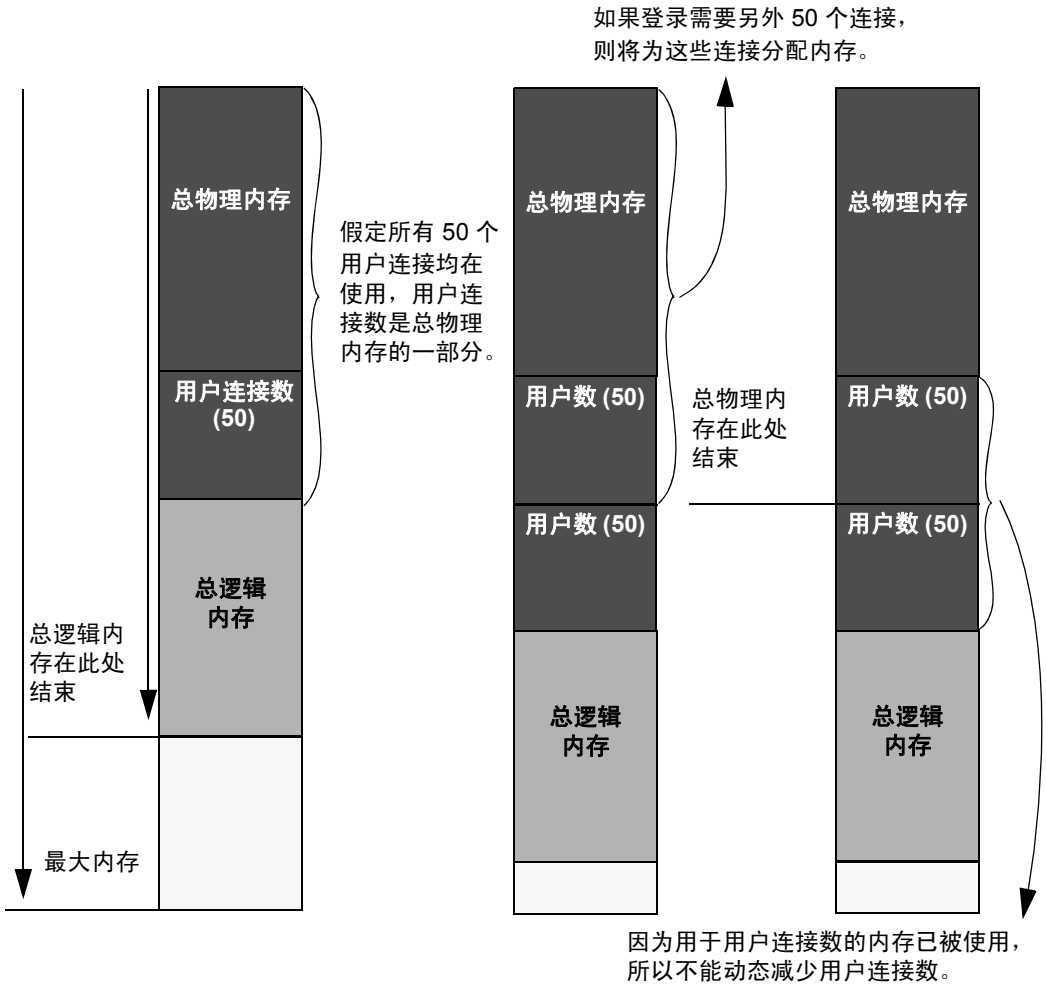
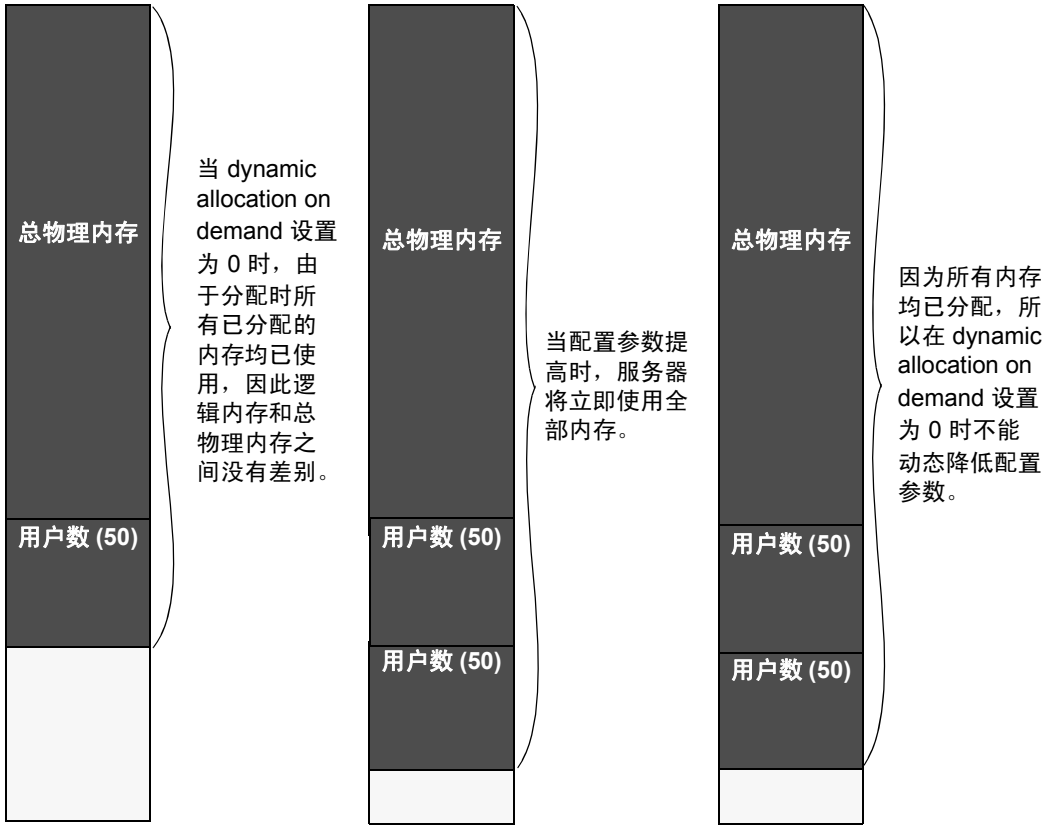


图 3-3 假定额外的 50 个用户连接均被实际使用。由于内存已在使用中，因此不能减少 number of user connections。可用 sp_configure 来指定对内存配置参数的更改，但此更改在服务器重新启动之后才会生效。

图 3-4：dynamic allocation on demand 设置为 0



注释 理论上，当 dynamic allocation on demand 设置为 0 时，总逻辑内存和物理内存之间应无差别。然而， Adaptive Server 在估计内存需求的方式和实际获取内存以供应用的方式上有一些差异。因此两者在运行期间会有一点差别。

当 dynamic allocation on demand 设置为 0 时，会立即分配所有已配置的内存要求。不能动态减少内存配置。

在图 3-3 和图 3-4 中，用户可将内存配置参数值更改为任何更小的有效值。此更改不会动态生效，但是会禁止使用新内存。例如，如果已将 number of user connections 配置为允许 100 个用户连接，然后又将该值更改为 50 个用户连接，在如图 3-3 和图 3-4 所示的情况下，可将 number of user connections 的值减回到 50。服务器重新启动前，此更改不影响 Adaptive Server 使用的内存，但会阻止任何新用户登录服务器。

配置线程池

Adaptive Server 在配置文件中的 `Thead Pool` 标头下记录各个线程池的信息。

缺省情况下， Adaptive Server 包含一组系统线程池，有了这些线程池，它才能正常运行。这些线程池包括 syb_default_pool 引擎池和多个不包含引擎的 run to completion (RTC) 线程池。除了系统池之外，用户还可以创建自己的线程池。用户创建的线程池始终是引擎池。

有关线程池类型的详细信息，请参见第 115 页的 “线程池”。

启动时， Adaptive Server 会列出有关 syb_default_pool 和 syb_blocking_pool 的信息，并将它们的参数设置为缺省值；但是未包含有关 syb_system_pool 的信息，因为 Adpative Server 在运行时根据其它配置要求计算它的线程数。

配置文件会列出所有线程池的以下参数：

- number of threads — 池中配置的线程数。
- description — 池的简要说明。

Adaptive Server 启动时，缺省线程池配置如下：

```
[Thread Pool:syb_blocking_pool]
        number of threads = 4

[Thread Pool:syb_default_pool]
        number of threads = 1
```

在您添加或删除线程池时， Adaptive Server 会更新配置文件，但无需重新启动，更改即可生效。用户创建的线程池 （即使用 create thread pool 创建的线程池）必须是引擎池。

下面的示例包含两个 Sybase 提供的线程池和一个用户创建的线程池 sales_pool：

```
[Thread Pool:sales_pool]
        description = pool for the sales force
        number of threads = 14
        idle timeout = 75

[Thread Pool:syb_blocking_pool]
         number of threads = 20

[Thread Pool:syb_default_pool]
        number of threads = 1
```

可使用文件编辑器编辑配置文件中的线程池信息，也可以使用 create thread pool、 alter thread pool 和 drop thread pool 命令管理线程池。请参见 《参考手册：命令》。

如果编辑配置文件，则 Adaptive Server 会使用新的线程池配置启动，并将所有更改信息输出到日志文件中 （如果使用 create thread pool 添加线程池，则无需重新启动 Adaptive Server）。下面是向 Adaptive Server 添加 smaller_pool 线程池之后，日志文件中的输出：

```
00:0000:00000:00000:2010/06/03 16:09:56.22 kernel  Create Thread Pool 4,
"smaller_pool", type="Engine (Multiplexed)", with 10 threads
```

在 isql 中，可使用 create thread pool 添加线程池。下面的示例添加具有 5 个线程的 sales_pool 线程池：

```
create thread pool sales_pool with thread count = 1
```

可使用 sp_helpthread 确定线程池 （包括 syb_system_pool）的运行时值。下面是上述线程池的 sp_helpthread 输出：

```
sp_helpthread


Name                 Type                    Size    IdleTimeout
             Description
-----------------    --------------------  ----    -----------
          ------------------------------------------
sales_pool           Engine (Multiplexed)     1           100
          NULL
syb_blocking_pool    Run To Completion        4             0
          A pool dedicated to executing blocking calls
syb_default_pool     Engine (Multiplexed)     1           100
          The default pool to run query sessions
syb_system_pool      Run To Completion        4             0
          The I/O and system task pool
```

若要删除 sales_pool 线程池，请使用：

```
drop thread pool sales_pool
```

请参见 《参考手册：命令》。

如果创建线程池时内存 （使用 kernel resource memory 确定）不足，无法创建线程并且可用引擎数不足，那么您可能会看到类似以下内容的消息：

```
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  Setting console to
nonblocking mode.
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  Create thread pool pubs_pool
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  Create Thread Pool 4,
"pubs_pool", type="THREADPOOL_MULTIPLEXED", with 2 threads
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  could not allocate memory
for dynamic engine
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  Thread creation failed to
allocate an engine.
00:0001:00000:00011:2010/06/11 14:46:38.32 server  Configuration file
'/sybase/siena.cfg' has been written and the previous version has been renamed
to '/sybase/siena.009'.
1> 00:0025:00000:00000:2010/06/11 14:46:38.33 kernel  Network and device
connection limit is 1009.
00:0025:00000:00000:2010/06/11 14:46:38.33 kernel  ASE - Dynamic Pluggable
Component Interface is disabled
00:0025:00000:00000:2010/06/11 14:46:38.33 kernel  Encryption provider
initialization succeeded on engine 1.
00:0025:00000:00000:2010/06/11 14:46:38.33 kernel  Thread 25 (LWP 15726) of
Threadpool pubs_pool online as engine 1
```

若要创建线程池，请增大 max online engines 的值，并重新启动 Adaptive Server。

确定线程总数

monThread 监控表包含有关 Adaptive Server 中所有线程的信息。发出 `select count(*) from monThread` 可确定 Adaptive Server 中的线程总数。然而，此查询报告的很多线程来自 syb_blocking_pool，它们需要的 CPU 不多，在进行资源规划时无需考虑它们。

在资源规划期间，要特别注意运行用户查询的查询处理 （或引擎）线程数。通过发出 `select count(*) from monEngine` 和 `select count(*) from sysengines`，或者通过向用户创建的任意线程池的线程计数中添加 syb_default_pool 的线程计数，可确定查询处理线程的数目。

sp_sysmon 显示每个线程消耗的 CPU 资源。请参见 《性能和调优系列：使用 sp_sysmon 监控 Adaptive Server》。

Tuning the syb_blocking_pool

可使用 monWorkQueue 监控表确定 syb_blocking_pool 的适当大小。如果 syb_blocking_pool 包含太多排队请求，需要很长的等待时间，请增大它的大小。如果 WaitCount 和 WaitTime 为零，请减小该池的大小。

sp_sysmon 在其输出的 “Kernel”部分显示阻塞池统计信息。如果在 “Blocking Call Activity”部分，“Queued Requests”相对于 “Serviced Requests”显示较高的百分比或者 “Total Wait Time”较高，可能需要增大该池的大小。

配置内存的系统过程

可以使用以下系统过程配置 Adaptive Server 内存：

- sp_configure
- sp_helpconfig
- sp_monitorconfig

有关 sp_configure 的完整语法和用法以及有关每个参数的详细信息，请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

使用 *sp_configure* 设置配置参数

若要查看与 Adaptive Server 的内存使用相关的参数，请输入：

```
sp_configure "Memory Use"
```

“Memory Used”列中的 “#”表示此参数是另一参数的一部分，其内存使用包含在用于另一部分的内存中。例如，用于 stack size 和 stack guard size 的内存用于满足每个用户连接和工作进程的内存要求，因此该值包含在 number of user connections 和 number of worker processes 的内存要求量中 （如果它们的设置超过 200）。

Memory Use 输出中的某些值是计算值。不能使用 sp_configure 设置这些值，而是报告这些值以显示内存分配给了何处。total data cache size 即是这些计算值中的一个。

可用于动态增长的内存

发出 sp_configure memory 命令将显示所有内存参数，并确定 max memory 和 total logical memory 之间的差值，它是可用于动态增长的内存量。例如：

```
sp_configure memory
Msg 17411, Level 16, State 1:
Procedure 'sp_configure', Line 187:
Configuration option is not unique.
Parameter Name                 Default  Memory Used  Config Value  Run Value
Unit                         Type
------------------------------ ----------- ----------- ------------ ----------
--------------------     ----------
additional network memory                0           0            0          0
                bytes        dynamic
allocate max shared memory               0           0            0          0
               switch        dynamic
compression memory size                  0         152            0          0
     memory pages(2k)        dynamic
engine memory log size                   0           2            0          0
     memory pages(2k)        dynamic
heap memory per user                  4096           0         4096       4096
                bytes        dynamic
kernel resource memory                4096        8344         4096       4096
     memory pages(2k)        dynamic
lock shared memory                       0           0            0          0
               switch         static
max memory                           33792      300000       150000     150000
     memory pages(2k)        dynamic
memory alignment boundary            16384           0        16384      16384
                bytes         static
memory per worker process             1024           4         1024       1024
                bytes        dynamic
messaging memory                       400           0          400        400
     memory pages(2k)        dynamic
pci memory size                      32768           0        32768      32768
     memory pages(2k)        dynamic
shared memory starting address           0           0            0          0
       not applicable         static
total logical memory                 33792      110994        55497      55497
     memory pages(2k)      read-only
```

```
total physical memory                      0       97656            0       48828
     memory pages(2k)      read-only
transfer utility memory size          4096        8194         4096        4096
     memory pages(2k)        dynamic

An additional 30786 K bytes of memory is available for reconfiguration.This is
the difference between 'max memory' and 'total logical memory'.
```

使用 *sp_helpconfig*

sp_helpconfig 估计给定的配置参数和值所需的内存量。它还提供参数的简要描述，并提供该参数的最小值、最大值和缺省值以及运行值和当前运行值使用的内存量的有关信息。如果正在计划对服务器进行重大更改（例如，从其它服务器上装载大型的现有数据库），并且想估计需要多少内存，您可能会发现 sp_helpconfig 特别有用。

若要查看配置参数需要多大内存，请输入参数名称的一部分（足够长以唯一标识该参数）和要配置的值：

```
1> sp_helpconfig "worker processes", "50"

number of worker processes is the maximum number of worker processes that can be
in use Server-wide at any one time.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
Unit     Type
-------------  -------------  -------------  -------------  ----------
-------  ------------
0              2147483647     0              0              0
number   dynamic

Configuration parameter, 'number of worker processes', will consume 7091K of
memory if configured at 50.
Changing the value of 'number of worker processes' to '50' increases the amount
of memory ASE uses by 7178 K.
```

如果知道要为特定资源分配多少内存，还可以使用 sp_helpconfig 来确定 sp_configure 的值：

```
1> sp_helpconfig "user connections", "5M"

number of user connections sets the maximum number of user connections that can
be connected to SQL Server at one time.
Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
Unit     Type
-------------  -------------  -------------  -------------  ----------
-------  ------------
```

```
5               2147483647      25              25              3773
number  dynamic
Configuration parameter, 'number of user connections', can be configured to 33
to fit in 5M of memory.
```

这两个语句的语法的重要区别在于：第二个示例使用了度量单位（表示兆字节的 “M”），以便向 sp_helpconfig 表明该值是一个大小，不是配置值。有效的度量单位是：

- P — 页 （Adaptive Server 2K 页）
- K — 千字节
- M — 兆字节
- G — 千兆字节

有些情况下，该语法对参数类型毫无意义，或者 Adaptive Server 无法计算内存用量。在这些情况下， sp_helpconfig 输出一条错误消息。例如，如果尝试为一个开关参数 （如 allow resource limits）指定内存量，则 sp_helpconfig 为所有不使用内存的配置参数输出描述参数功能的消息。

使用 *sp_monitorconfig*

sp_monitorconfig 显示某些共享服务器资源上的元数据高速缓存使用情况统计信息，包括：

- 可在任何时刻打开的数据库、对象及索引数
- 参照完整性查询使用的辅助扫描描述符的数量
- 可用的描述符和活动的描述符数量
- 可用内存页数
- 活动的描述符的百分比
- 服务器最近启动以来使用的描述符的最大数量
- 过程高速缓存的当前大小和实际使用量
- 在其上运行 sp_monitorconfig 的实例的名称 （如果在集群环境中运行）；或 NULL （如果不在集群环境中运行）。

例如，假设 number of open indexes 配置参数为 500。在高峰期，您可以运行 sp_monitorconfig 获得索引描述符的元数据高速缓存实际使用情况的准确读数：

```
sp_monitorconfig "number of open indexes"
Usage information at date and time:May 28 2010  1:12PM.
Name                        Num_free    Num_active  Pct_act  Max_Used
Reuse_cnt    Instance_Name
------------------------  ----------  ----------  -------  ----------
----------  ------------------------------
number of open indexes           217         283    56.60         300
          0                                NULL
```

尽管 Adaptive Server 配置为 500，但服务器自上次启动以来使用打开索引的最大数量是 300。因此，可以将 number of open indexes 配置参数重新设置为 330，这样可满足使用的索引描述符最大数量 300，另外还有 10% 备用。

也可以使用 sp_monitorconfig “procedure cache size” 来确定过程高速缓存的当前大小。此参数描述为过程高速缓存当前配置的空间量，它也是该过程高速缓存曾实际使用的最大量。在本例中，过程高速缓存配置为 20,000 页：

```
sp_configure "procedure cache size"
Parameter Name                      DefaultMemory         Used
         Config Value          Run Value            Unit
       类型
------------------------------      -------------------   ----------
         -------------------   -------------------- --------------------
          --------------------
 procedure cache size                                7000        43914
                      20000                 20000      memory pages(2k)
                  动态
```

但是，当运行 sp_montorconfig “procedure cache size” 时，您会发现曾使用的最大过程高速缓存是 14241 页，这意味着可以降低过程高速缓存的运行值，从而节省内存：

```
sp_monitorconfig "procedure cache size"
Usage information at date and time:May 28 2010  1:35PM.
Name                        Num_free    Num_active  Pct_act  Max_Used
Reuse_cnt    Instance_Name
------------------------  ----------  ----------  -------  ----------
----------  ------------------------------
procedure cache size           5878        14122    70.61       14241
         384                              NULL
```

查看可用内存页数 (Num_free)，以确定 kernel resource memory 配置是否充足：

```
sp_monitorconfig "kernel resource memory"
Usage information at date and time:Oct 12 2010  1:35PM.
Name                        Num_free     Num_active   Pct_act   Max_Used
Reuse_cnt     Instance_Name
--------------------------  -----------  -----------  -------  -----------
-----------  ------------------------------
kernel resource memory           9512          728     7.11        728
          0                                NULL
```

如果可用页数太少，但活动页百分比 (Pct_act) 很高，可能需要增大 kernel resource memory 的值。

控制 Adaptive Server 内存的配置参数

对于使用大量 Adaptive Server 内存的配置参数，以及在许多 Adaptive Server 安装中通常都需要更改的配置参数，第一次配置 Adaptive Server 的系统管理员应该行检查。升级到 Adaptive Server 的新版本后，在更改系统配置或更改使用内存的其它配置变量时，请检查这些参数。

使用较少内存的配置参数在第 58 页的 “使用内存的其它参数”中讨论。

Adaptive Server 可执行代码大小

total logical memory 或 max memory 的计算值中不包括可执行代码的大小。 total logical memory 报告 Adaptive Server 配置的实际内存要求，但不包括可执行代码的任何内存要求。

可执行代码的内存要求由操作系统管理，而不在 Adaptive Server 的控制范围内。请参见操作系统文档。

数据和过程高速缓存

如第 34 页的 “Adaptive Server 如何使用内存”中所述，您可以指定数据和过程高速缓存的大小。本节介绍这两个高速缓存间的细节，以及如何监控高速缓存的大小。

确定过程高速缓存大小

procedure cache size 以 2K 页为单位指定过程高速缓存的大小，而不管服务器的逻辑页大小为多少。例如：

```
sp_configure "procedure cache size"
Parameter Name         Default  Memory Used  Config Value  Run Value
Unit              Type
--------------------  -------  -----------  ------------  ---------
---------------  ---------
procedure cache size     7000        15254          7000       7000
memory pages(2k)  dynamic
```

用于过程高速缓存的内存量是 30.508KB （即 15254 个 2K 页）。若要将过程高速缓存设置成不同大小，请发出以下命令：

```
sp_configure "procedure cache size", new_size
```

此示例将过程高速缓存的大小重新设置为 10000 个 2K 页 (20MB)：

```
sp_configure "procedure cache size", 10000
```

确定缺省数据高速缓存大小

sp_cacheconfig 和 sp_helpcache 都以兆字节为单位显示当前缺省数据高速缓存。例如，下面显示的是一个缺省数据高速缓存配置为 19.86MB 的 Adaptive Server：

```
sp_cacheconfig

Cache Name                Status     Type       Config Value    Run Value
------------------        --------   --------   ------------    ----------
default data cache        Active     Default    0.00 Mb         19.86Mb
                                                ------------    --------
                         Total                  0.00Mb          19.86 Mb
===========================================================================
Cache:default data cache, Status:Active, Type:Default
       Config Size:0.00 Mb, Run Size:19.86 Mb
       Config Replacement:strict LRU, Run Replacement:strict LRU
       Config Partition:    1,    Run Partition:    1
IO Size       Wash Size     Config Size     Run Size      APF Percent
--------      ---------     ------------    ----------    -----------
2 Kb          4066 Kb        0.00 Mb         19.86 Mb               10
```

若要更改缺省数据高速缓存，请发出 sp_cacheconfig 并指定 "default data cache"。例如，若要将缺省数据高速缓存更改为 25MB，请输入：

```
sp_cacheconfig "default data cache", "25M"
```

此更改是动态的，无需重新启动 Adaptive Server 即可使新值生效。

缺省数据高速缓存的大小是绝对值，高速缓存大小的最小值为逻辑页大小的 256 倍；对于 2KB, 的逻辑页大小，最小值为 512KB：对于 16KB 的逻辑页大小最小值为 4MB。缺省值为 8MB。在升级过程中， Adaptive Server 将缺省数据高速缓存的大小设置为配置文件中缺省数据高速缓存的值。

监控高速缓存空间

可以使用以下命令查看数据高速缓存和过程高速缓存空间：

```
sp_configure "total data cache size"
```

确定 Adaptive Server 如何使用内存的另一种方法是检查 Adaptive Server 启动时写入错误日志的与内存有关的消息。这些消息准确地表明分配了多少数据高速缓存和过程高速缓存、在任一时刻多少个**编译对象**可以驻留在高速缓存中以及缓冲池大小。

这些消息提供了 Adaptive Server 中高速缓存分配情况的最准确信息。分配给过程高速缓存的内存量取决于 procedure cache size 配置参数的运行值。

下面逐一描述了这些错误日志消息。

过程高速缓存消息

以下错误日志消息提供过程高速缓存的有关信息：

```
server:Number of proc buffers allocated:556
server:Number of blocks left for proc headers:629
```

proc 缓冲区

proc 缓冲区 （过程缓冲区）是管理过程高速缓存中编译对象的数据结构。一个 proc 缓冲区用于存储在过程高速缓存中的编译对象的每个副本。 Adaptive Server 在启动时确定所需的 proc 缓冲区数，并将该数量乘以一个 proc 缓冲区的大小 （76 个字节）得到所需的内存总量。

proc 头

在过程高速缓存中，编译对象存储在 proc 头 （过程头）中。根据要存储的对象大小，可能需要一个或多个 proc 头。过程高速缓存中可存储的编译对象总数受到可用的 proc 头数或可用的 proc 缓冲区数中较小的那个数目的限制。

过程高速缓存的总大小等于分配给各个 proc 缓冲区的内存的总和（向上舍入到最接近的页界限）加上分配给 proc 头的内存。

数据高速缓存消息

Adaptive Server 启动时，将每个高速缓存的总大小和高速缓存中每个缓冲池的大小记入错误日志。以下示例显示有两个缓冲池的缺省数据高速缓存和有两个缓冲池的用户定义的高速缓存：

```
Memory allocated for the default data cache cache:8030 Kb
Size of the 2K memory pool:7006 Kb
Size of the 16K memory pool:1024 Kb
Memory allocated for the tuncache cache:1024 Kb
Size of the 2K memory pool:512 Kb
Size of the 16K memory pool:512 Kb
```

Kernel resource memory

kernel resource memory 参数确定可用于内部内核用途 （如线程池、任务和监控计数器）的内存量 （以 2K 页为单位）。 kernel resource memory 的内存来自 max memory，它必须足够大，以便接受较高的 kernel resource memory 值。

用户连接数

每个用户连接所需的内存量因平台不同而有所不同，并且在更改其它配置变量时也会更改。这些配置变量包括：

- default network packet size
- stack size 和 stack guard size
- user log cache size

更改这些参数中的任何一个都会更改每个用户连接所使用的空间量：用大小差值乘以用户连接数。例如，如果您有 300 个用户连接，并想将 stack size 从 34K 增加到 40K，则这个新值需要额外 1800K 内存。

打开的数据库、打开的索引和打开的对象

控制可同时打开的数据库、索引、分区和对象的总数的配置参数由**元数据高速缓存**管理。元数据高速缓存驻留在 Adaptive Server 内存的服务器结构部分。可以使用以下参数配置这些高速缓存的空间：

- number of open databases
- number of open indexes
- number of open objects
- number of open partitions

请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

Adaptive Server 在打开数据库或访问索引、分区或对象时，会从相应的系统表中读取与其有关的信息：数据库的信息位于 sysdatabases 中，索引信息位于 sysindexes 中，分区信息位于 syspartitions 中，等等。

数据库、索引、分区或对象的元数据高速缓存使 Adaptive Server 能够在其内存结构中访问直接在 sysdatabases、sysindexes、syspartitions 或 sysobjects 中描述这些数据库、索引、分区或对象的信息。如果这样做， Adaptive Server 将绕过因需要访问磁盘而占用很多资源的调用，因此可以提高性能。当 Adaptive Server 需要在运行期检索数据库信息、索引信息、分区信息或对象信息时，还可减少同步和螺旋锁争用。

对于包含大量索引、分区和对象并且用户并发性很强的数据库，管理单独的元数据高速缓存很有益处。

锁数目

Adaptive Server 中的所有进程共享一个锁结构缓冲池。可以使用以下方法初步估计配置的锁数：将预计的并发用户连接数加上已配置的 number of worker processes，再乘以 20。

查询所需的锁数变化很大。请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。有关工作进程如何使用内存的信息，请参见第 59 页的 “工作进程”。

Adaptive Server 会在锁用完后发出错误消息 1204，并且只有 “sa” 或具有 sa_role 的用户可以登录到服务器来配置其它锁。这时，其他任何用户尝试登录时会遭到 Adaptive Server 拒绝， Adaptive Server 还会将以下消息输出到错误日志：

```
login:Process with spid <process id> could not
connect to the ASE which has temporarily run out of
locks
```

如果处于这种状态，那么 sa_role 未在登录期间自动激活的任何用户尝试登录时都会遭到 Adaptive Server 拒绝。如果系统安全员执行了下列任何步骤，则自动激活角色：

- 已直接向用户授予 sa_role
- 已通过用户定义的角色间接向用户授予 sa_role，并且已将用户定义的角色指定为用户的缺省角色
- 已向用户的登录配置文件授予 sa_role，并且修改了配置文件，以自动激活 sa_role

登录后，“sa” 用户或具有 sa_role 的用户应该立即执行 sp_configure 以添加锁。在这种状态下，执行其它任何语句都可能导致 Adaptive Server 再次发出错误消息 1024。

如果大量 “sa” 用户或具有 sa_role 的用户在 Adaptive Server 用完锁后立即尝试登录， Adaptive Server 可能会进入不可恢复状态。

数据库设备和磁盘 I/O 结构

number of devices 配置参数控制 Adaptive Server 可以用于存储数据的数据库设备数。请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

当用户进程必须执行物理 I/O 时，该 I/O 在磁盘 I/O 结构中排队。请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

使用内存的其它参数

本节讨论使用中等内存量的配置参数。

并行处理

并行处理比串行处理需要的内存多。影响并行处理的配置参数如下：

- number or worker processes
- memory per worker processes
- number of mailboxes 和 number of messages

工作进程

number of worker processes 设置 Adaptive Server 中同时可用的总工作进程数。每个工作进程需要的内存量与用户连接需要的内存量大体相同。

更改以下任一参数都会更改每个工作进程所需的内存量：

- default network packet size
- stack size 和 stack guard size
- user log cache size
- memory per worker process

memory per worker process 控制缓冲池中供所有工作进程使用的附加内存。该附加内存存储杂类数据结构开销和工作进程间通信缓冲区。

并行查询和过程高速缓存

每个工作进程都在从过程高速缓存借用的空间中生成一份查询计划副本。协调进程在内存中保存查询计划的两份副本。

远程服务器

允许 Adaptive Server 与其它 Sybase 服务器（例如， Backup Server、组件集成服务或 XP Server）通信的一些配置参数使用内存。

影响远程服务器并使用内存的配置参数是：

- number of remote sites
- number of remote sites
- number of remote logins
- remote server pre-read packets

远程节点数

将 number of remote sites 设置为在服务器上必须与之通信的同步节点数。如果只使用 Backup Server，且不使用其它远程服务器，可以将该参数减少为 1 来增加数据高速缓存空间过程高速缓存空间。

从 Adaptive Server 到 XP Server 的连接使用一个远程节点。

用于 RPC 的其它配置参数

这些用于远程通信的配置参数只将少量内存用于各个连接：

- number of remote connections
- number of remote logins

从 Adaptive Server 到 XP Server 的每个用于执行 ESP 的同步连接使用一个远程连接和一个远程登录名。

由于 remote server pre-read packets 参数会增加每个连接 （由 number of remote connections 参数配置）所需的空间，因此增加预读包的数量将显著影响内存使用。

参照完整性

如果数据库中的表使用了大量的参照约束，则在用户连接超出缺省设置时，可能需要调整 number of aux scan descriptors 参数。在大多数情况下，缺省设置是够用的。如果用户连接数超出了当前设置，Adaptive Server 将返回一条错误消息，建议增大 number of aux scan descriptors 参数的设置。

影响内存的其它参数

下面列出了影响内存的其它参数。重新设置这些配置参数后，应检查这些参数使用的内存量，并检查更改这些参数对过程高速缓存和数据高速缓存的响。

• additional network memory	• max online engines
• allow resource limits	• max SQL text monitored
• audit queue size	• number of alarms
• event buffers per engine	• number of large i/o buffers
• max number network listeners	• permission cache entries

语句高速缓存

语句高速缓存用于保存高速缓存的 SQL 语句。 Adaptive Server 将比较传入的 SQL 语句和已缓存的 SQL 语句，如果两者相同，则执行已保存的 SQL 语句的计划通过这种机制，应用程序就可以在多次执行同一语句时分摊查询编译的开销。

设置语句高速缓存

通过语句高速缓存， Adaptive Server 可将新收到的即席 SQL 语句和已高速缓存的 SQL 语句进行比较。如果找到匹配项， Adaptive Server 则使用在最初执行时已高速缓存的计划。这样，由于已有一个现成的计划，因此 Adaptive Server 就不必重新编译 SQL 语句了。

语句高速缓存是服务器范围的资源，可从过程高速缓存内存池分配和使用内存。可以使用 statement cache size 配置参数动态设置语句高速缓存的大小。

注释　如果释放或减少语句高速缓存的内存量，则在 Adaptive Server 重新启动前，分配的原始内存不会被释放。

语法如下所示，其中 *size_of_cache* 是大小，以 2K 页为单位：

```
sp_configure "statement cache size", size_of_cache
```

例如，若要将语句高速缓存设置为 5000 个 2K 页，请输入：

```
sp_configure "statement cache size", 5000
```

请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

配置语句高速缓存使用的内存时，应考虑以下事项：

- 分配给过程高速缓存内存池的内存量是 statement cache size 配置参数和 procedure cache size 配置参数的总和。语句高速缓存内存来自于过程高速缓存内存池。
- statement cache size 限制已高速缓存的 SQL 文本和计划可以使用的过程高速缓存的内存量。 Adaptive Server 用于语句高速缓存的内存不能超过用 statement cache size 配置参数配置的内存。
- *@@nestlevel* 包含用户会话中当前执行的嵌套级别，初始值为 0。每当存储过程或触发器调用另一个存储过程或触发器，嵌套级别就会增加一级。创建高速缓存的句时，嵌套级别也会增加一级。如果超过了最大值 16，事务中止。

- 所有过程高速缓存内存 （包括由 statement cache size 配置参数分配的内存）都可供存储过程使用。存储过程可能会根据最近使用最少的 (LRU) 策略替换高速缓存的语句。
- 增加 max memory 配置参数，增加的量为对语句高速缓存配置的量。例如，如果最开始已将语句高速缓存大小配置为 100 个 2K 页，则应将 max memory 增加同等数量。
- 如果已使用 statement cache size 配置参数，则可在会话级使用 set statement cache 来禁用和启用语句高速缓存。缺省情况下，如果已在服务器级配置了语句高速缓存，则会在会话级予以启用。
- 因为每个高速缓存的语句都要消耗一个对象描述符，所以还必须使用 number of open objects 配置参数相应地增加对象描述符的数量。要估计允许的高速缓存的 SQL 语句数量，请参见第 64 页的 “语句高速缓存大小调整”。

即席查询处理

Adaptive Server 高速缓存语句时，它会将语句从即席查询更改为轻量存储过程。例如，如果不高速缓存调用缺省值或规则的语句，那么在该语句所属的批处理中发出 sp_bindefault 或 sp_bindrule 时，Adaptive Server 会发出错误消息 540。然而，如果高速缓存该语句，Adaptive Server 会将缺省值或规则绑定到该列。

将语句作为存储过程高速缓存并执行时， Adaptive Server 可能会发出运行时错误而不是规范化错误。例如，如果未高速缓存语句，则以下查询引发错误号 241 ；但是如果高速缓存了语句，则引发 `Truncation error` 并中止命令。

```
create table t1(c1 numeric(5,2)
go
insert t1 values(3.123)
go
```

使用语句高速缓存处理即席 SQL 语句：

1 Adaptive Server 分析语句。

如果语句应进行高速缓存 （请参见第 64 页的 “高速缓存条件”）， Adaptive Server 将从语句中计算出一个散列值。Adaptive Server 利用此散列值在语句高速缓存中搜索匹配的语句 （请参见第 63 页的 “语句匹配标准”）。

- 如果在语句高速缓存中找到匹配项，Adaptive Server 将跳至步骤 4。
- 如果未找到匹配项，则 Adaptive Server 继续执行步骤 2。

2　Adaptive Server 高速缓存 SQL 语句文本。

3　Adaptive Server 使用轻量存储过程包装 SQL 语句，并将所有局部变量更改为过程参数。轻量过程的内部表示此时尚未编译到计划中。

4　Adaptive Server 将 SQL 语句转换成相应的轻量过程的 execute 语句。

- 如果高速缓存中没有计划，Adaptive Server 将编译过程并高速缓存计划。Adaptive Server 使用为局部变量指定的运行期值编译计划。
- 如果计划存在但无效， Adaptive Server 将使用高速缓存的 SQL 语句的文本返回步骤 3。

5　Adaptive Server 然后执行编译好的过程。替换轻量过程将使 *@@nestlevel* 全局变量递增。

语句匹配标准

Adaptive Server 按照 SQL 文本、登录名（特别是如果两个用户都有 sa_role 角色时）、用户 ID、数据库 ID 和会话状态设置，将即席 SQL 语句与高速缓存的语句进行匹配。相关的会话状态包括以下 set 命令参数的设置：

- forceplan
- jtc
- parallel_degree
- prefetch
- quoted_identifier
- table count
- transaction isolation level
- chained （事务模式）

这些参数的设置决定了 Adaptive Server 为高速缓存的语句生成的计划的行为。请参见 《参考手册：命令》。

注释　如果启用语句高速缓存，您必须在其自身的批处理中配置 set chained on/off。

高速缓存条件

Adaptive Server 根据以下条件高速缓存语句：

- Adaptive Server 当前高速缓存至少使用了一个表引用的 select、update、 delete 和 insert select 语句。
- 如果启用了 abstract plan dump 或 abstract plan load 参数，则不会高速缓存语句。即，不能同时启用语句高速缓存以及 abstract plan load 和 abstract plan dump at 配置参数
- Adaptive Server 不会高速缓存 select into 语句、游标语句、动态语句、纯文本的 insert （不是 insert select）语句，以及存储过程、视图、触发器内的语句。引用临时表的语句不会被高速缓存，带有以二进制大对象 (BLOB) 数据类型传输的语言参数的语句也会被高速缓存。还有特别大的语句也不会被高速缓存。此外，作为 if exists 或 if not exists 条件子句一部分的 select 语句也不会被高速缓存。

语句高速缓存大小调整

每个高速缓存的语句需要语句高速缓存内大约 1K 的内存，具体数量取决于 SQL 文本的长度。每个高速缓存的计划需要过程高速缓存内至少 2K 的内存。要估计所需的语句高速缓存内存，对于每条要高速缓存的语句应考虑以下事项：

- SQL 语句的长度 （以字节为单位），向上取整为最邻近的 256 的整数倍。
- 大约 100 字节的开销。
- 过程高速缓存中计划的大小。此大小与仅包含高速缓存的语句的存储过程计划的大小相等。对于两个或更多用户正在同时使用的单个高速缓存的语句可能存在重复的计划。

监控语句高速缓存

sp_sysmon 可报告语句高速缓存和存储过程的执行情况。语句高速缓存由以下计数器实施监控：

- Statements Cached — 添加到高速缓存中的 SQL 语句的数目。这个数目通常与 Statements Not Found 数目相同。较小的 statements cached 值意味着语句高速缓存中充满了活动语句。
- Statements Found in Cache — 重复使用查询计划的次数。较低的高速缓存命中次数可能表示语句高速缓存太小。

- Statements Not Found — 表示没有重复的 SQL 语句。 statements found in cache 和 statements not found 之和是已提交的合格 SQL 语句的总数。
- Statements Dropped — 已从高速缓存中删除的语句的数目。较大的值可能表示过程高速缓存内存量不足或者语句高速缓存太小。
- Statements Restored — 利用 SQL 文本重新生成的查询计划的数目。较大的值表示过程高速缓存大小不足。
- Statements Not Cached — 如果启用语句高速缓存，Adaptive Server 本应高速缓存的语句的数目。但是， Statements Not Cached 并不表示本应高速缓存的唯一 SQL 语句的数目。

下面是 sp_sysmon 的输出样本：

```
SQLStatement Cache:
 Statements Cached                0.0            0.0            0          n/a
 Statements Found in Cache        0.7            0.0            2          n/a
 Statements Not Found             0.0            0.0            0          n/a
 Statements Dropped               0.0            0.0            0          n/a
 Statements Recompiled            0.3            0.0            1          n/a
 Statements Not Cached            1.3            0.0            4        n/a
```

清除语句高速缓存

运行 dbcc purgesqlcache 可将所有 SQL 语句从语句高速缓存中删除。任何当前正在运行的语句不会被移除。

必须具有 sa_role 角色才能运行 dbcc purgesqlcache。

输出语句摘要

运行 dbcc prsqlcache 可输出语句高速缓存中的语句摘要。*oid* 选项用于指定要输出的语句的对象 ID； *printopt* 选项用于指定是输出跟踪描述（指定 0）还是输出 showplan 选项（指定 1）。如果不包括 *oid* 或 *printopt* 的任何值，dbcc prsqlcache 将显示语句高速缓存的全部内容。

必须具有 sa_role 角色才能运行 dbcc prsqlcache。

此命令将提供高速缓存中的所有语句的信息：

```
dbcc prsqlcache
Start of SSQL Hash Table at 0xfc67d830
Memory configured:1000 2k pages              Memory used:18 2k pages
Bucket# 625 address 0xfc67ebb8
```

```
SSQL_DESC 0xfc67f9c0
ssql_name *ss1248998166_0290284638ss*
ssql_hashkey 0x114d645e ssql_id 1248998166
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 8
SQL TEXT:select * from sysobjects where name like "sp%"

Bucket# 852 address 0xfc67f2d0
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT:select name from systypes where allownulls = 0

End of SSQL Hash Table

DBCC execution completed.If DBCC printed error messages, contact a user with
```

或者可以获得有关特定对象 ID 的信息：

```
dbcc prsqlcache (1232998109, 0)
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT:select name from systypes where allownulls = 0

DBCC execution completed.If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

下面的示例将 showplan 输出的 printopt 参数指定为 1：

```
dbcc prsqlcache (1232998109, 1)

SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1       ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
```

```
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT:select name from systypes where allownulls = 0

QUERY PLAN FOR STATEMENT 1 (at line 1).

      STEP 1
         The type of query is SELECT.
         FROM TABLE
            systypes
         Nested iteration.
         Table Scan.
         Forward scan.

         Positioning at start of table.
         Using I/O Size 2 Kbytes for data pages.
         With LRU Buffer Replacement Strategy for data pages.
         DBCC execution completed.If DBCC printed error messages,
         contact a user with
          System Administrator (SA) role.
```

显示高速缓存语句的 SQL 计划

可以使用下列函数查看高速缓存语句的计划：

```
show_plan(spid, batch_id, context_id,
statement_number)
```

其中：

- *spid* — 任何用户连接的进程 ID。
- *batch_id* — 批处理的唯一编号。
- *context_id* — 每个过程 （或触发器）的唯一编号。
- *statement_number* — 当前语句在批处理内的编号。

对于未能正确执行的语句，您可以通过变更优化程序设置或指定抽象计划来更改计划。

如果将现有 show_plan 参数中的第一个 int 变量指定为 “-1”，则 show_plan 会将第二个参数视为 SSQLID。

注释　语句高速缓存中的单个条目可以与多个可能不同的 SQL 计划相关联。 show_plan 只显示其中一个计划。

第 4 章

配置数据高速缓存

对数据高速缓存进行管理的最常见原因是为了对它们进行重新配置，从而提高性能。本章主要介绍数据高速缓存的工作机制。《性能和调优系列：基知识》中的第 5 章“内存使用和性能”讨论了与数据高速缓存相关的性能概念。

Adaptive Server 数据高速缓存

数据高速缓存包含当前使用的数据、索引和日志页，以及 Adaptive Server 最近使用的页。完成安装后， Adaptive Server 具有一个用于所有数据、索引和日志动的缺省数据高速缓存。该高速缓存的缺省大小为 8M 。创建其它高速缓存不会减小缺省数据高速缓存的大小。也可以在命名高速缓存和缺省高速缓中创建缓冲池，用以执行大 I/O 。然后可以将数据库、表（包括 syslogs 表）、索引及文本或图像页链绑定到指定的数据高速缓存上。

如果查询优化程序确定预取可以提高性能，则使用较大的 I/O 大小将使 Adaptive Server 能够执行数据预取。例如，在配置 16K 逻辑页的服务器上， 128K 的 I/O 大小意味着 Adaptive Server 可以一次读取整个扩充（ 8 页），而不用执行 8 次单独的 I/O 。

排序操作也可利用为大 I/O 大小配置的缓冲池。

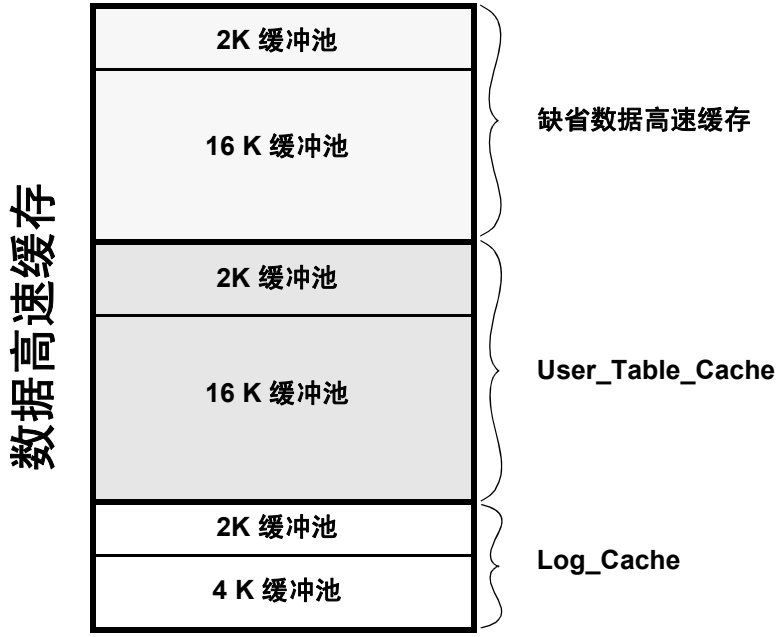
配置命名数据高速缓存并不是将缺省高速缓存划分为若干单独的高速缓存结构。创建的命名数据高速缓存只能由显式地绑定到这些高速缓存的数据库数据库对象使用。未显式绑定到命名数据高速缓存的所有对象均使用缺省数据高速缓存。

Adaptive Server 提供可由用户配置的数据高速缓存以提高性能，尤其对多处理器服务器则更是如此。

图 4-1 显示了具有缺省数据高速缓存和两个命名数据高速缓存的高速缓存。该服务器使用 2K 逻辑页。

缺省高速缓存包含一个 2K 缓冲池和一个 16K 缓冲池。 User_Table_Cache 高速缓存也有一个 2K 缓冲池和一个 16K 缓冲池。 Log_Cache 有一个 2K 缓冲池和一个 4K 缓冲池。

图 4-1 ： 带一个缺省高速缓存和两个命名数据高速缓存的数据高速缓存



高速缓存配置命令和系统过程

表 4-2 列出了用于配置命名数据高速缓存、将对象绑定到高速缓存和解除绑定，以及报告高速缓存绑定情况的命令和系统过程。还列出了可用来检查数据库对象大小的过程，以及在对象级、命令级或会话级控制高速缓存使用情况的命令。

表 4-1：配置命名数据高速缓存的命令和过程

命令或过程	功能
sp_cacheconfig	创建或删除命名高速缓存，更改大小、高速缓存类型、高速缓存策略或高速缓存分区数。
sp_poolconfig	创建和删除 I/O 缓冲池，更改 I/O 缓冲池的大小、清洗大小和异步预取百分比限制。
sp_bindcache	将数据库或数据库对象绑定到高速缓存上。
sp_unbindcache	从高速缓存解除绑定特定的对象或数据库。
sp_unbindcache_all	解除绑定所有绑定到指定高速缓存的对象。
sp_helpcache	报告有关数据高速缓存的摘要信息并列出绑定到高速缓存的数据库和数据库对象。
sp_cachestrategy	报告为表或索引设置的高速缓存策略，禁用或重新启用预取或 MRU 策略。
sp_logiosize	更改日志的缺省 I/O 大小。
sp_spaceused	提供有关表和索引的大小的信息或有关数据库中使用的空间量的信息。
sp_estspace	给定表将要包含的行数后，估计表和索引的大小。
sp_help	报告表绑定到的高速缓存。
sp_helpindex	报告索引绑定到的高速缓存。
sp_helpdb	报告数据库绑定到的高速缓存。
set showplan on	报告查询的 I/O 大小和查询的高速缓存利用策略。
set statistics io on	报告为查询执行的读取数量。
set prefetch [on |off]	为单个会话启用或禁用预取功能。
select... (prefetch...lru | mru)	强制服务器使用指定的 I/O 大小或 MRU 替换策略。

用于配置数据高速缓存的 sp_cacheconfig 的大多数参数都是动态参数，这些参数不需要重新启动服务器即可生效。有关静态和动态操作的说明，请参见第 74 页的表 4-2。

除使用命令交互地配置指定的数据高速缓存外，还可以通过编辑位于 *$SYBASE* 目录中的配置文件进行配置。但是，这样做需要重新启动服务器。请参见第 100 页的“使用配置文件配置数据高速缓存”。

有关数据高速缓存的信息

使用 sp_cacheconfig 可以创建和配置命名数据高速缓存。首次安装 Adaptive Server 时，会有一个名为 default data cache 的高速缓存。若要了解有关高速缓存的信息，请输入：

```
sp_cacheconfig
```

sp_cacheconfig 的结果如下所示：

```
Cache Name                 Status    Type      Config Value  Run Value
-------------------------- --------- --------- ------------ ------------
default data cache         Active    Default         0.00 Mb     59.44 Mb
                                     ------------ ------------
                                     Total           0.00 Mb     59.44 Mb
===============================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 0.00 Mb,   Run Size: 59.44 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:             1,   Run Partition:             1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   12174 Kb      0.00 Mb     59.44 Mb      10
```

报告的开始部分输出每个高速缓存的摘要信息，结尾部分输出所有配置的高速缓存的总计大小。对于每个高速缓存，都有报告高速缓存中的内存池配的信息块。

报告中有如下列：

- Cache Name — 给出高速缓存的名称。
- Status — 指示高速缓存是否处于活动状态。可能的值有：
  - “ Pend/Act ” — 高速缓存刚刚创建，但尚未处于活动状态。
  - Active — 高速缓存当前处于活动状态。
  - “ Pend/Del ” — 高速缓存将被删除。已使用 sp_cacheconfig 和 sp_poolconfig 将高速缓存大小重新设置为 0 。请参见第 73 页的“配置数据高速缓存”。
- Type — 指示高速缓存可以存储数据和日志页（“ Mixed ”），或只能存储日志页（“ Log Only ”）。只有缺省高速缓存的类型为“ Default ”。不能更改缺省据高速缓存的类型，也不能将任何其它高速缓存的类型更改为“ Default ”。

- Config Value — 显示当前配置的值。在示例输出中，未显式配置缺省数据高速缓存，所以其大小为 0 。
- Run Value — 显示 Adaptive Server 当前正在使用的大小。

输出的第二个信息块以说明高速缓存的三行信息开始。前两行重复开头的摘要信息块中的信息。第三行“ Config Replacement ”和“ Run Replacement ”显示高速存策略，该策略或者为“ strict LRU ”，或者为“ relaxed LRU ”。 Run Replacement 描述生效的设置（“ strict LRU ”或“ relaxed LRU ”）。如果在重新启动服务器后改了策略， Config Replacement 设置将不同于 Run Replacement 设置。

然后， sp_cacheconfig 为高速缓存中的每个缓冲池提供一行信息：

- IO Size — 显示缓冲池中缓冲区的大小。缓冲池的缺省大小为服务器逻辑页的大小。首次配置高速缓存时，所有空间都被指派给缓冲池。其有效大小为 2K 、 4K 、 8K 和 16K 。
- Wash Size — 指示缓冲池的清洗大小。请参见第 90 页的“更改内存池的清洗区”。
- Config Size 和 Run Size — 显示配置的大小和当前正在使用的大小。如果已尝试在其它缓冲池之间移动空间，并且某些空间无法释放，则对于这些缓冲池这两个值会有差异。
- Config Partition 和 Run Partition — 显示配置的高速缓存分区数和当前正在使用的分区数。如果在上次重新启动后更改了分区数，这两个值可能会不同。
- APF Percent — 显示可包含由异步预取产生的未用缓冲区的缓冲池百分比。

摘要行输出所显示的一个或多个高速缓存的总计大小。

配置数据高速缓存

可以使用绝对值指定 Adaptive Server 的缺省数据高速缓存和过程高速缓存。计划高速缓存配置和实施高速缓存的第一步是设置 max memory 配置参数。设置了 max memory 后，应确定要为服务器上的数据高速缓存分配的空间量。数据高速缓存的大小仅受可访问的系统内存限制；但是， max memory 应大于 total logical memory 。必须为缺省数据高速缓存的大小和所有其它用户定义高速缓存的大小指定绝对值。有关 Adaptive Server 内存使用情况的概述，请参见第 3 章“配置内存”。

可使用两种方法配置数据高速缓存：

- 使用 sp_cacheconfig 和 sp_poolconfig 交互地操作。这个方法是动态方法，这意味着您不需要重新启动 Adaptive Server 。
- 通过编辑配置文件。这个方法是静态方法，这意味着您必须重新启动 Adaptive Server 。

有关使用配置文件的信息，请参见第 100 页的“使用配置文件配置数据高速缓存”。

每次更改数据高速缓存或者执行 sp_cacheconfig 或 sp_poolconfig 时，Adaptive Server 都会将新的高速缓存或缓冲池信息写入配置文件，并将旧版本的配置文件复制到备份文件中。给出备份文件名的消息将发送给错误志。

使用 sp_cacheconfig 执行的操作中有些是动态的，有些是静态的。

可在一个命令上同时指定静态和动态参数。 Adaptive Server 会立即执行动态更改，并将所有更改都写入配置文件（既包括静态更改，也包括动态更改）。静态更改在下一次启动服务器时生效。

表 4-2 ： 动态和静态 sp_cacheconfig 操作

动态 sp_cacheconfig 操作	静态 sp_cacheconfig 操作
添加新高速缓存	更改高速缓存分区的数目
将内存添加到现有高速缓存	减小高速缓存大小
删除高速缓存	更改替换策略
更改高速缓存类型	

可以通过删除并重新创建高速缓存来重新设置静态参数：

1 解除高速缓存的绑定。

2 删除高速缓存。

3 使用新配置重新创建高速缓存。

4 将对象绑定到高速缓存。

创建新高速缓存

可以使用 sp_cacheconfig 创建新高速缓存。请参见《参考手册：过程》中的 sp_cacheconfig 。

最大的数据高速缓存大小仅受系统中可用内存量的限制。Adaptive Server 的全局内存量决定创建新高速缓存所需的内存。创建高速缓存后：

- 它具有缺省清洗大小。
- 异步预取大小被设置为 global async prefetch limit 的值。
- 它只有缺省缓冲池。

可以使用 sp_poolconfig 重新设置这些值。

若要创建一个名为 pubs_cache 的 10MB 高速缓存，请执行以下命令：

```
sp_cacheconfig pubs_cache, "10M"
```

此命令在系统表中进行更改，并将新值写入配置文件。该高速缓存将立即激活。现在运行 sp_cacheconfig 将显示以下内容：

```
sp_cacheconfig

Cache Name                     Status    Type     Config Value Run Value
------------------------------ --------- -------- ------------ ------------
default data cache             Active    Default       0.00 Mb      8.00 Mb
pubs_cache                     Active    Mixed        10.00 Mb     10.00 Mb
------------ ------------
Total      10.00 Mb     18.00 Mb
================================================================================
Cache:default data cache,   Status:Active,   Type:Default
Config Size:0.00 Mb,   Run Size:8.00 Mb
Config Replacement:strict LRU,   Run Replacement:strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   1636 Kb      0.00 Mb      8.00 Mb     10
================================================================================
Cache:pubs_cache,   Status:Active,   Type:Mixed
Config Size:10.00 Mb,   Run Size:10.00 Mb
Config Replacement:strict LRU,   Run Replacement:strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
```

现在， pubs_cache 处于活动状态，所有空间均指派给最小的缓冲池。

注释 创建新高速缓存时，将根据 max memory 对指定的附加内存进行验证。如果 total logical memory 与所请求的附加内存之和大于 max memory，则 Adaptive Server 将发出一个错误，而不执行更改。

可以创建任意数量的高速缓存，而无需重新启动 Adaptive Server 。

没有足够的空间可供新高速缓存使用

如果 Adaptive Server 不能分配所请求的全部内存，它会分配所有可用内存，并发出以下消息：

```
ASE is unable to get all the memory requested (%d).(%d) kilobytes have been
allocated dynamically.
```

但是，只有在下次重新启动 Adaptive Server 后，才会分配这些附加内存。

Adaptive Server 可能会通知您内存不足，因为资源约束导致一些内存不可用。系统管理员应确保这些资源约束是暂时的。如果仍出现这种情况，之后的重新启动可能会失败。

例如，如果 max memory 为 700MB， pub_cache 为 100MB，服务器的 total logical memory 为 600MB，而您尝试将 100MB 添加到 pub_cache，则附加内存还在 max memory 之内。然而，如果服务器只能分配 90MB，那么它会动态分配这些内存，但配置文件中高速缓存的 `size field` 会被更新为 100MB 。之后重新启动时，由于 Adaptive Server 同时为所有数据高速缓存获取内存，因此 pub_cache 的大小为 100MB 。

向现有命名高速缓存添加内存

可以使用 sp_cacheconfig 向现有高速缓存添加内存。

分配的附加内存将被添加到 Adaptive Server 页大小缓冲池中。例如，在逻辑页大小为 4K 的服务器上，缓冲池的最小大小为 4K 缓冲池。如果高速缓存有分区，会将附加内存平均分配给各个分区。

如果可用内存不足， Adaptive Server 将分配它所能分配的内存，然后在您重新启动服务器时分配足量内存。请参见第 76 页的“没有足够的空间可供新高速缓存使用”。

例如，若要向名为 pub_cache 的高速缓存添加 2MB 内存（当前大小为 10MB ），请输入：

```
sp_cacheconfig pub_cache, "12M"
```

添加完内存后， sp_cacheconfig 输出为：

```
sp_cacheconfig pub_cache
 Cache Name                     Status    Type     Config Value Run Value
------------------------------ --------- -------- ------------ ------------
pub_cache                      Active    Mixed      12.00 Mb       12.00
------------ ------------
Total      12.00 Mb     12.00 Mb
================================================================================
Cache:pub_cache,   Status:Active,   Type:Mixed
Config Size:12.00 Mb,   Run Size:12.00 Mb
Config Replacement:strict LRU,   Run Replacement:strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   2456 Kb      0.00 Mb     12.00 Mb     10
```

这一更改向数据库页大小缓冲池添加内存，并根据需要重新计算清洗大小。如果为清洗大小设置了绝对值， Adaptive Server 将不会重新计算清洗大小。

减小高速缓存的大小

减小高速缓存的大小时，必须重新启动 Adaptive Server 才能使更改生效。

下面是一个有关 pubs_log 高速缓存的报告：

```
sp_cacheconfig pubs_log

Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      7.00 Mb      7.00 Mb
                                      ------------ ------------
                                      Total         7.00 Mb      7.00 Mb
================================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 7.00 Mb,   Run Size: 7.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
```

```
-------- --------- ------------ ------------ -----------
    2 Kb     920 Kb       0.00 Mb      4.50 Mb      10
    4 Kb     512 Kb       2.50 Mb      2.50 Mb      10
```

以下命令将 pubs_log 高速缓存的大小从当前大小 7MB 减小到 6MB：

```
sp_cacheconfig pubs_log, "6M"
```

重新启动 Adaptive Server 后， sp_cacheconfig 将显示以下内容：

```
Cache Name                Status    Type      Config Value Run Value
------------------------ --------- -------- ------------ ------------
pubs_log                  Active    Log Only      6.00 Mb      6.00 Mb
                                    ------------ ------------
                                    Total         6.00 Mb      6.00 Mb
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb     716 Kb       0.00 Mb      3.50 Mb      10
    4 Kb     512 Kb       2.50 Mb      2.50 Mb      10
```

减小数据高速缓存的大小时，所有要删除的空间必须在缺省缓冲池（最小的可用大小）中可用。减小数据高速缓存的大小之前，可能需要将空间从其缓冲池移到缺省缓冲池中。上例中，若要将高速缓存的大小减小为 3MB，请使用 sp_poolconfig 将一些内存从 4K 缓冲池移动到 2K 的缺省缓冲池中。内存移动后成为“可用于指定的高速缓存的内存”。请参见第 94 页的“更改内存池的大小”。

删除高速缓存

若要完全删除数据高速缓存，可将其大小重新设置为 0：

```
sp_cacheconfig pubs_log, "0"
```

该高速缓存将被立即删除。

注释 您不能删除缺省数据高速缓存。

如果删除对象绑定到的数据高速缓存，则该高速缓存将保留在内存中，并且 Adaptive Server 将发出以下消息：

```
Cache cache_name not deleted dynamically.Objects are bound to the cache.Use
sp_unbindcache_all to unbind all objects bound to the cache.
```

配置文件中对应于该高速缓存的条目和 sysconfigures 中对应于该高速缓存的条目都将被删除，而该高速缓存将在下次重新启动 Adaptive Server 时被删除。

如果重新创建该高速缓存并重新启动 Adaptive Server，绑定将重新标记为有效。

可以使用 sp_helpcache 查看所有绑定到该高速缓存的项。可以使用 sp_unbindcache_all 解除对象的绑定。请参见《参考手册：过程》。

显式配置缺省高速缓存

必须显式配置缺省数据高速缓存的大小。可以使用 sp_helpcache 查看可用于高速缓存的剩余内存量。例如：

```
sp_helpcache
Cache Name                Config Size    Run Size        Overhead
------------------------  -------------  ----------      ----------
default data cache        50.00 Mb       50.00 Mb        3.65 Mb
pubs_cache                10.00 Mb       10.00 Mb        0.77 Mb

Memory Available For      Memory Configured
Named Caches              To Named Caches
--------------------      ----------------
91.26 Mb                  91.25 Mb

------------------ Cache Binding Information:-------------------

Cache Name        Entity Name    Type          Index Name     Status
----------          -----------      -----           ---------        ------
```

若要指定缺省数据高速缓存的绝对大小，请执行 sp_cacheconfig，并指定缺省数据高速缓存和一个大小值。例如，若要将缺省数据高速缓存大小设置为 25MB，请输入：

```
sp_cacheconfig "default data cache", "25M"
```

重新运行 sp_helpconfig 时，“ Config Value ”将显示该值。

```
sp_cacheconfig

Cache Name                 Status    Type      Config Value Run Value
-------------------------- --------- --------- ------------ ------------
default data cache         Active    Default       25.00 Mb     50.00 Mb
pubs_cache                 Active    Mixed         10.00 Mb     10.00 Mb
                                       ------------ ------------
                                       Total         10.00 Mb    60.00 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
    Config Size: 25.00 Mb,   Run Size: 50.00 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU
    Config Partition:             1,   Run Partition:             1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb 10110 Kb      00.00 Mb     50.00 Mb       10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
    Config Size: 10.00 Mb,   Run Size: 10.00 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU
    Config Partition:             1,   Run Partition:             1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb     0.00 Mb     10.00 Mb     10
```

可以使用 sp_cacheconfig 更改任何指定的高速缓存的大小。对任何数据高速缓存的大小所做的更改都不会影响任何其它高速缓存的大小。同样，指定了缺省数据高速缓存的大小后，其它用户定义高速缓存的配置也不会改变缺省数据高速缓存的大小。

注释 如果配置了缺省数据高速缓存，然后减小 max memory ，使 total logical memory 值高于 max memory 值，则 Adaptive Server 将无法启动。编辑配置文件以增加其它高速缓存的大小并增大需要内存的配置参数的值，可创建一个 total logical memory 高于 max memory 的环境。请参见第 3 章“配置内存”。

使用绝对值显式配置缺省数据高速缓存和所有用户定义高速缓存。此外，许多配置参数也使用内存。若要充分发挥性能并避免错误，可将 max memory 的值设置得足够高，以容纳使用内存的所有高速缓存和所有配置参数。

如果将 max memory 设置为一个小于 total logical memory 的值， Adaptive Server 将发出警告消息。

更改高速缓存类型

若要保留高速缓存专供事务日志使用，请将高速缓存的类型更改为“ logonly ”。这个更改是动态的。

此示例创建类型为“ logonly ”的高速缓存 pubs_log ：

```
sp_cacheconfig pubs_log, "7M", "logonly"
```

以下内容显示该高速缓存的初始状态：

```
Cache Name           Status    Type     Config Value Run Value
------------------ --------- -------- ------------ ------------

pubs_log             Pend/Act  Log Only      7.00 Mb      0.00 Mb
                               ------------ ------------
                               Total               7.00 Mb      0.00 Mb
```

只要现有的“ mixed ”高速缓存上未绑定非日志对象，就可以更改其类型：

```
sp_cacheconfig pubtune_cache, logonly
```

在高事务环境中，如果缺省值是服务器逻辑页大小的两倍， Adaptive Server 的性能通常最好（对于逻辑页大小为 2K 的服务器为 4K ；对于逻辑页大小为 4K 服务器为 8K ，依此类推）。对于大一些的页大小（ 4 、 8 和 16K ），请使用 sp_sysmon 为节点找到最佳配置。请参见第 85 页的“匹配日志高速缓存的日志 I/O 大小”。

配置高速缓存替换策略

如果高速缓存专用于表或索引，并且在系统到达稳定状态后只有很少或几乎没有缓冲区替换，则可以设置宽松的 LRU （最近使用最少的）替换策略。此策略可以提高只有很少或没有缓冲区替换发生的高速缓存的性能，也能提高大多数日志高速缓存的性能。请参见《性能和调优系列：基础知识》中的 5 章“内存使用和性能”。

若要设置宽松的替换策略，请使用：

```
sp_cacheconfig pubs_log, relaxed
```

缺省值为“ strict ”。高速缓存替换策略和异步预取百分比是可选的，但是如果指定，则必须有正确的参数或为“ DEFAULT ”。

注释 设置高速缓存替换策略是静态的，需要重新启动 Adaptive Server 才能生效。

可以在一个命令中创建高速缓存并指定其类型和替换策略。以下示例创建两个高速缓存， pubs_log 和 pubs_cache ：

```
sp_cacheconfig pubs_log, "3M", logonly, relaxed

sp_cacheconfig pubs_cache, "10M", mixed, strict
```

下面是执行结果：

```
sp_cacheconfig

Cache Name                          Status    Type      Config Value Run Value
----------------------- --------- -------- ----------- ------------
default data cache       Active    Default      25.00 Mb     42.29 Mb
pubs_cache               Active    Mixed        10.00 Mb     10.00 Mb
pubs_log                 Active    Log Only      7.00 Mb      7.00 Mb
                                     ----------- ------------
                                     Total      42.00 Mb     59.29 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ----------- ----------- -----------
    2 Kb   8662 Kb      0.00 Mb     42.29 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ----------- ----------- -----------
    2 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 7.00 Mb,   Run Size: 7.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1
```

```
IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
     2 Kb    1432 Kb       0.00 Mb      7.00 Mb          10
```

将数据高速缓存划分为若干内存池

创建数据高速缓存后，可以将其划分为若干内存池，每个内存池的 I/O 大小可以不同。在任何高速缓存中，每个 I/O 大小只能对应一个内存池。任何高速缓存中缓冲池的总大小不能大于该高速缓存的大小。内存池的最小大小为服务器的逻辑页大小。大于逻辑页大小的内存池必须是二的乘方，其最大小为一个扩充。

Adaptive Server 在执行大 I/O 时，会同时将多页读入高速缓存。这些页始终作为一个单元处理；当其生命周期结束时将作为单元写入磁盘。

缺省情况下，在创建命名数据高速缓存时，其所有空间都被指派给缺省内存池。创建额外内存池时，会将缺省内存池中的某些空间重新指派给其它内池，这将减少缺省内存池的大小。例如，如果创建了一个 50MB 空间的数据高速缓存，所有这些空间都指派给 2K 缓冲池。如果在此高速缓存中配置有 30MB 空间的 4K 缓冲池，则 2K 缓冲池将减少到 20MB 。

创建缓冲池后，可以在缓冲池之间移动空间。例如，在有一个 20MB 的 2K 缓冲池和一个 30MB 的 4K 缓冲池的高速缓存中，可以从 4K 缓冲池中拿出 10MB 空来配置一个 16K 缓冲池。

在高速缓存中的缓冲池之间移动空间的命令不需要重新启动 Adaptive Server ，因此可以重新配置缓冲池以适应应用程序负载的更改，而几乎不会影响服务器的活动。

除了在所配置的高速缓存中创建缓冲池以外，还可以向缺省数据高速缓存添加 I/O 最大为 16K 的内存池。

配置内存池的语法为：

```
sp_poolconfig cache_name, "memsize[P|K|M|G]",
"config_poolK" [, "affected_poolK"]
```

config_pool 设置为该命令中指定的大小。空间在 config_pool 和另一个缓冲池 affected_pool 之间移动。如果不指定 affected_pool ，将从 2K （最小的可用大小）缓冲池中移走空间或给 2K 缓冲池分配空间。缓冲池的最小大小为 512K 。

下例在 pubs_cache 数据高速缓存中创建由 16K 页组成的 7MB 缓冲池：

```
sp_poolconfig pubs_cache, "7M", "16K"
```

若要查看当前配置，请运行 sp_cacheconfig （只需给出高速缓存名称）：

```
sp_cacheconfig pubs_cache
Cache Name                Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
pubs_cache                Active    Mixed        10.00 Mb     10.00 Mb
                                      ------------ ------------
                                      Total        10.00 Mb     10.00 Mb
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:              1,   Run Partition:              1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb      3.00 Mb     10


   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
```

也可以在缺省数据高速缓存中创建内存池。

例如，开始时的高速缓存配置如下：

```
Cache Name                  Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     42.29 Mb
                                                ------------ ------------
                                               Total        25.00 Mb     42.29 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:              1,   Run Partition:              1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     42.29 Mb     10
```

如果然后要在 8MB 的缺省数据高速缓存中创建一个 16K 缓冲池，请输入：

```
sp_poolconfig "default data cache", "8M", "16K"
```

最终配置如下所示，其中包含 2K 缓冲池的“ Run Size ”：

```
Cache Name                 Status    Type     Config Value Run Value
---------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     42.29 Mb
                                              ------------ ------------
                                                Total      25.00 Mb     42.29 Mb
===============================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    8662 Kb      0.00 Mb     34.29 Mb     10
   16 Kb    1632 Kb      8.00 Mb      8.00 Mb     10
```

无需在创建的高速缓存中配置 2K 内存池的大小。其“ Run Size ”表示未显式配置给高速缓存中其它缓冲池的所有内存。

匹配日志高速缓存的日志 I/O 大小

如果要为数据库的事务日志创建高速缓存，则配置该高速缓存的大部分空间以便与日志 I/O 的大小匹配。缺省值是服务器逻辑页大小的两倍（对于逻页大小为 2K 的服务器为 4K ；对于逻辑页大小为 4K 的服务器为 8K ，依此类推）。如果 4K 缓冲池不可用， Adaptive Server 将使用 2K I/O 处理日志。可以使用 sp_logiosize 更改日志 I/O 大小。 Adaptive Server 启动时，错误日志中报告每个数据库的日志 I/O 大小，或者您可以发出不带参数的 sp_logiosize 检查数据库的大小。

以下示例在 pubs_log 高速缓存中创建一个 4K 缓冲池：

```
sp_poolconfig pubs_log, "3M", "4K"
```

也可以在缺省数据高速缓存中创建一个 4K 的内存池，供所有未绑定到其它高速缓存的数据库的事务日志使用：

```
sp_poolconfig "default data cache", "2.5M", "4K"
```

请参见《性能和调优系列：基础知识》的第 5 章“内存使用和性能”中的“选择事务日志的 I/O 大小”。

将对象绑定到高速缓存

sp_bindcache 为高速缓存指派数据库、表、索引、文本对象或图像对象。在将实体绑定到高速缓存之前，必须满足以下条件：

- 命名高速缓存必须存在，并且其状态必须为“ Active ”。
- 数据库或数据库对象必须存在。
- 若要绑定表、索引或对象，必须正在使用存储它们的数据库。
- 要绑定系统表（包括事务日志表 syslogs ），数据库必须处于单用户模式。
- 若要绑定数据库，您必须使用 master 数据库。
- 若要将数据库、用户表、索引、文本对象或图像对象绑定到高速缓存，高速缓存的类型必须为“ Mixed ”。只有 syslogs 表可以绑定到类型为“ Log Only ”的高速缓存。
- 必须拥有该对象，或者是数据库所有者或系统管理员。

将对象绑定到高速缓存是动态的，不需要重新启动服务器。

将对象绑定到高速缓存的语法为：

```
sp_bindcache cache_name, dbname [,[owner.]tablename
[, indexname | "text only" ] ]
```

以下示例将 titles 表绑定到 pubs_cache ：

```
sp_bindcache pubs_cache, pubs2, titles
```

若要绑定 titles 上的索引，可将索引名作为第三个参数添加到命令中：

```
sp_bindcache pubs_cache, pubs2, titles, titleind
```

上例中不需要所有者名称是因为 pubs2 数据库中对象的所有者是“ dbo ”。若要指定所有者是任何其它用户的表，应添加所有者名称。必须用引号将整个参数括起来，因为句点是特殊字符：

```
sp_bindcache pubs_cache, pubs2, "fred.sales_east"
```

以下示例将事务日志 syslogs 绑定到 pubs_log 高速缓存：

```
sp_bindcache pubs_log, pubs2, syslogs
```

若要将任何系统表（包括事务日志 syslogs）绑定到高速缓存，数据库必须处于单用户模式。从 master 使用 sp_dboption，然后使用 use database 命令，之后再运行 checkpoint：

```
sp_dboption pubs2, single, true
```

表的 text 和 image 列使用单独的数据结构存储在数据库中。若要将该对象绑定到高速缓存，请添加“ text-only ”参数：

```
sp_bindcache pubs_cache, pubs2, au_pix, "text only"
```

从 master 执行以下示例，将 tempdb 数据库绑定到高速缓存：

```
sp_bindcache tempdb_cache, tempdb
```

不必删除现有的绑定，即可重新绑定对象。

高速缓存绑定限制

在下列情况下，不能绑定数据库对象或解除其绑定：

- 对象上的脏读处于活动状态
- 该对象上有游标处于打开状态

此外，在进行绑定或解除绑定时， Adaptive Server 必须锁定对象，所以该过程的响应时间可能比较慢，因为需要等待锁定解除。请参见第 99 页的“锁定以执行绑定”。

获得有关高速缓存绑定的信息

如果提供高速缓存名， sp_helpcache 会提供有关该高速缓存和绑定到该高速缓存的实体的信息：

```
sp_helpcache pubs_cache

Cache Name            Config Size      Run Size        Overhead
------------------- ------------   ----------     ----------
pubs_cache             10.00 Mb       10.00 Mb         0.77 Mb

------------------ Cache Binding Information: ------------------

Cache Name      Entity Name            Type    Index Name  Status
---------     -----------          ----   ----------  ------
pubs_cache      pubs2.dbo.titles       index  titleind    V
pubs_cache      pubs2.dbo.au_pix       index  tau_pix     V
pubs_cache      pubs2.dbo.titles       table              V
pubs_cache      pubs2.fred.sales_east table               V
```

如果使用 sp_helpcache 时未指定高速缓存名，该过程将输出有关 Adaptive Server 上配置的所有高速缓存和绑定到这些高速缓存的所有对象的信息。

sp_helpcache 使用 *%cachename%* 对高速缓存名进行字符串匹配。例如，“ pubs ”与“ pubs_cache ”和“ pubs_log ”都匹配。

“ Status ”列报告高速缓存绑定有效（“ V ”）还是无效（“ I ”）。如果数据库或对象绑定到一个高速缓存，而该高速缓存被删除，绑定信息将保留在统表中，但高速缓存绑定被标记为无效。所有具有无效绑定的对象均使用缺省数据高速缓存。如果之后创建了另一个同名的高速缓存，激活该高速缓存后，绑定将变为有效。所有用户定义高速缓存的状态都必须为“ mixed cache ”或“ log only ”。

检查高速缓存开销

sp_helpcache 可以报告管理给定大小的指定的数据高速缓存所需的开销量。创建命名数据高速缓存时，使用 sp_cacheconfig 请求的所有空间均可以用作高速缓存空间。高速缓存管理所需的内存从全局内存块池中分配。

注释 Adaptive Server 版本 12.5.1 和更高版本的高速缓存开销记帐比早期版本更明确。开销量相同，但不再将其视为服务器开销的一部分进行报告，现在将其视高速缓存开销的一部分进行报告。

若要查看高速缓存所需的开销，需要包括预期的大小。可用 P 代表页，K 代表千字节， M 代表兆字节或 G 代表千兆字节。下例检查 20,000 页的开销：

```
sp_helpcache "20000P"

2.96Mb of overhead memory will be needed to manage a
cache of size 20000P
```

配置用户高速缓存并不浪费高速缓存空间。无论是使用一个较大的数据高速缓存还是使用多个较小的高速缓存，存储和跟踪内存中页的结构均需要大 5% 的内存。

开销如何影响总的高速缓存空间

第 72 页的“有关数据高速缓存的信息”中的示例显示了一个缺省数据高速缓存，在创建任何用户定义的高速缓存之前，该高速缓存的可用高速缓存空间为 59.44 MB 。服务器使用 2K 逻辑页。创建了 10MB 的 pubs_cache 后， sp_cacheconfig 的结果显示总的高速缓存大小为 59.44 MB 。

配置数据高速缓存似乎会增加或减小总的可用高速缓存。这是因为管理特定大小高速缓存需要一定的开销； sp_cacheconfig 显示的值不包含这些开销。

使用 sp_helpcache 检查原有的 59.44MB 缺省高速缓存和新的 10MB 高速缓存的开销，结果表明空间的更改源于开销大小的更改。以下示例显示在做出任何更改之前缺省数据高速缓存的开销：

```
sp_helpcache "59.44M"

4.10Mb of overhead memory will be needed to manage a
cache of size 59.44M
```

以下示例显示 pubs_cache 的开销：

```
sp_helpcache "10M"

0.73Mb of overhead memory will be needed to manage a
cache of size 10M
```

以下示例显示大小为 49.44MB 的高速缓存所需的开销：

```
sp_helpcache "49.44M"
3.46Mb of overhead memory will be needed to manage a
cache of size 49.44M
```

4.19MB （等于 0.73MB + 3.46MB ）是维护两个大小分别为 10MB 和 49.44MB 的高速缓存所必需的开销大小，并略大于维护 59.44MB 高速缓存所必需的 4.10MB 的开销。

由 sp_cacheconfig 输出的高速缓存大小被四舍五入到两位，由 sp_helpcache 输出的开销四舍五入到两位，所以输出包含的四舍五入误差很小。

删除高速缓存绑定

若要删除高速缓存绑定，请使用：

- sp_unbindcache 解除高速缓存上单个实体的绑定。
- sp_unbindcache_all 解除高速缓存上所有对象的绑定。

sp_unbindcache 的语法为：

```
sp_unbindcache dbname [,[owner.]tablename
[, indexname | "text only"] ]
```

以下示例解除 pubs2 数据库中 titles 表上 titleidind 索引的绑定：

```
sp_unbindcache pubs2, titles, titleidind
```

若要解除高速缓存上所有对象的绑定，可使用 sp_unbindcache_all ，并给出高速缓存名：

```
sp_unbindcache_all pubs_cache
```

注释 如果在八个数据库中有八个以上的数据库对象绑定到高速缓存，则不能使用 sp_unbindcache_all 。必须对单个数据库或对象使用 sp_unbindcache ，使涉及的数据库数目减少到八个或八个以下。

删除对象的高速缓存绑定时，当前在内存中的所有页均将从高速缓存中清除。

更改内存池的清洗区

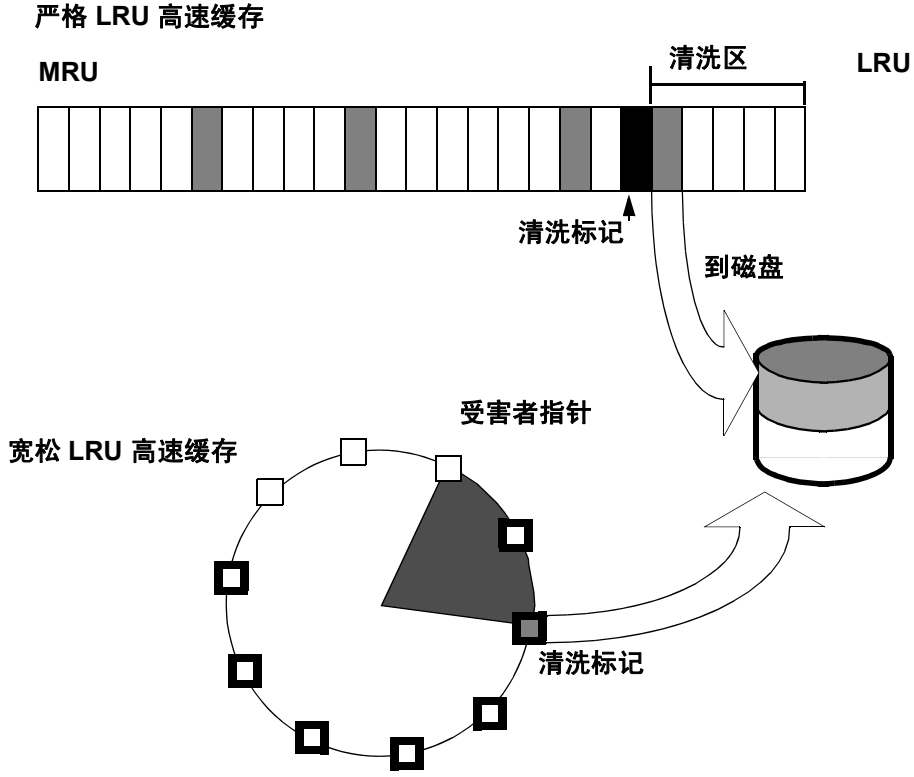
Adaptive Server 必须将缓冲区读入高速缓存时，会将缓冲区放置到下列位置之一：

- 每个内存池的 LRU （最近使用最少的）端（对于使用严格 LRU 策略的高速缓存），或
- 受害者指针处（对于使用宽松 LRU 策略的高速缓存）。如果在受害者标记处设置了缓冲区的最近使用位，受害者指针将移到该缓冲池的下一个缓冲区。

每个缓冲池的一部分被配置为**清洗区**。脏页（高速缓存中已更改的页）通过清洗标记并进入清洗区后， Adaptive Server 将在该页上启动异步 I/O 。写操作完成后，该页将被标记为干净页并且可以在高速缓存中使用。

清洗区中的空间必须足够大，以便缓冲区上的 I/O 可以在页需被替换之前完成。图 4-2 说明缓冲池的清洗区如何使用严格 LRU 高速缓存和宽松 LRU 高速缓存。

图 4-2 ：缓冲池的清洗区



缺省情况下，内存池清洗区的大小配置如下：

- 如果缓冲池小于 300MB ，则缺省清洗大小为该缓冲池中缓冲区的 20% 。
- 如果缓冲池大于 300MB ，则缺省清洗大小为 300MB 中缓冲区数的 20% 。

清洗大小的最小值为 10 个缓冲区。清洗区大小的最大值为缓冲池大小的 80% 。必须为所有大于 2KB 的缓冲池指定缓冲池大小和清洗大小。

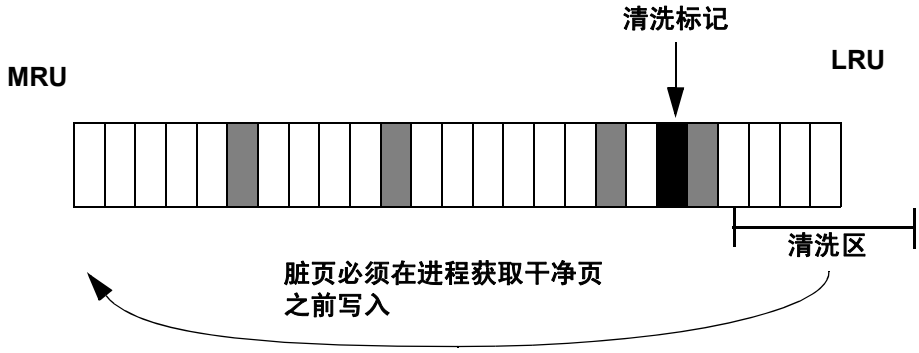
缓冲区是与缓冲池的 I/O 大小匹配的页块。每个缓冲区都被视为一个单元：缓冲区中的所有页均被作为一个单元读入高速缓存，写入磁盘并在高速缓中老化。块大小等于缓冲池大小乘以缓冲区数 — 如果缓冲池大小为 2KB ，缓冲区为 256 个，则块大小等于 512KB ；如果缓冲池大小为 16 KB ，缓冲区为 256 个，则块大小等于 4096KB 。

例如，如果配置了空间为 1MB 的 16K 缓冲池，则该缓冲池有 64 个缓冲区； 64 乘以 20% 等于 12.8 。此结果向下舍入为 12 个缓冲区，或 192K 是分配给清洗的块大小。

如果清洗区过小

如果缓冲池中使用的清洗区太小，需要干净缓冲区的操作只能等待缓冲池的 LRU 端或受害者标记处的脏缓冲区上的 I/O 完成后才能进行。这种情况称为**脏缓冲区争夺**，它会对性能产生严重的影响。图 4-3 显示了严格替换策略高速缓存中的脏缓冲区争夺。

图 4-3 ： 清洗区小会导致脏缓冲区争夺



可以使用 sp_sysmon 确定内存池中是否出现脏缓冲区争夺。应在高速缓存正经历频繁的 I/O 和大量更新操作时运行 sp_sysmon ，因为出现脏缓冲区争夺通常是脏页过多和高速缓存替换率高共同作用的结果。

如果高速缓存摘要部分的“ Buffers Grabbed Dirty ”输出的“ Count ”列显示一个非零值，则应检查每个缓冲池的“ Grabbed Dirty ”行以确定问题所在。应增加受响缓冲池的清洗区大小。以下示例将 8K 内存池的清洗区设置为 720K ：

```
sp_poolconfig pubs_cache, "8K", "wash=720K"
```

如果缓冲池非常小，特别是如果 sp_sysmon 输出显示缓冲池的周转率很高，则可能还需要增加缓冲池的大小。

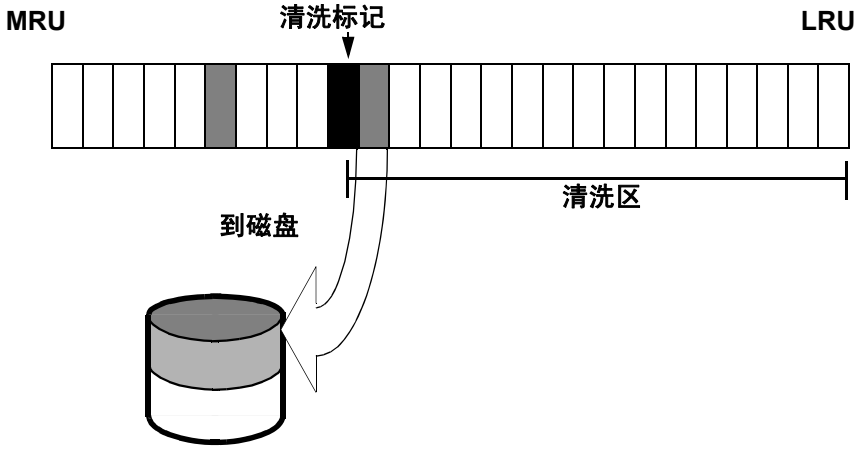
请参见《性能和调优系列：使用 sp_sysmon 监控 Adaptive Server 》 。

如果清洗区过大

如果缓冲池中的清洗区过大，则缓冲区在高速缓存中通过清洗标记的速度会过快，异步写入会在所有脏缓冲区上启动，如图 4-4 所示。缓冲区将被标记为干净，并且在缓冲区到达 LRU 之前将一直保留在 MRU/LRU 链的清洗区中。如果另一个查询更改了缓冲区中的页， Adaptive Server 必须执行额外的 I/O 才能再次将缓冲区写入磁盘。

如果 sp_sysmon 输出显示严格替换策略高速缓存的揾 ound in Wash 缓冲区百分比较高，并且不存在脏缓冲区争夺问题，则应尝试减小清洗区的大小。请参见《性能和调优列：使用 sp_sysmon 监控 Adaptive Server 》。

图 4-4 ： 清洗区过大的结果



设置管家以避免高速缓存的清洗

可以使用 cache status 参数的 HK ignore cache 选项指定不希望管家在特定高速缓存上执行清洗。通过指定不包括在清洗中的高速缓存，可以避免管家和高速缓存管理器螺旋锁之间的争用。可以在配置文件中的每个高速缓存标头下手动设置 HK ignore cache ； 不能使用 sp_cacheconfig 设置 HK ignore cache 。

配置文件中的以下示例显示指定的高速缓存 newcache 的设置：

```
Named Cache:newcache
     cache size = 5M
     cache status = mixed cache
     cache status = HK ignore cache
     cache replacement policy = DEFAULT
```

```
local cache partition number = DEFAULT
```

必须与 default data cache 、 mixed cache 或 log parameters 一起设置 HK ignore cache 。不能将参数 cache status 仅设置为 HK ignore cache 。

更改缓冲池的异步预取限制

异步预取限制指定可用来容纳那些已由异步预取指令取入高速缓存、但尚未由任何查询使用过的页的缓冲池百分比。可以使用 global async prefetch limit 参数设置服务器的缺省值。使用 sp_poolconfig 设置的缓冲池限制将替换单个池的缺省限制。

以下命令将 pubs_cache 中 2K 池的百分比设置为 20 ：

```
sp_poolconfig pubs_cache, "2K", "local async prefetch limit=20"
```

对缓冲池预取限制的更改将立即生效，不需要重新启动 Adaptive Server 。请参见《性能和调优系列：基础知识》中的第 6 章“调优异步预取”。

更改内存池的大小

可以使用 sp_poolconfig 更改内存池的大小，指定高速缓存、新的缓冲池大小、要更改的缓冲池的 I/O 大小和要从其中获得缓冲区的缓冲池的 I/O 大小。如果不指定最后的参，所有空间均将从该缓冲池中取走或均指派给该缓冲池。

从内存池移走空间

可以使用 sp_cacheconfig 检查 pubs_log 高速缓存的当前配置（以下输出基于以前章节中的示例）：

```
sp_cacheconfig pubs_log

Cache Name                Status    Type      Config Value Run Value
------------------------- --------- --------- ------------ ------------
pubs_log                  Active    Log Only       6.00 Mb      6.00 Mb
                                    ----------- ------------
                                    Total          6.00 Mb      6.00 Mb
============================================================================
```

```
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ----------- ----------- -----------
    2 Kb    716 Kb      0.00 Mb      3.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

若要从 2K 缓冲池中移走所需的空间，将 4K 缓冲池的大小增加到 5MB，请输入：

```
sp_poolconfig pubs_log, "5M", "4K"

sp_cacheconfig pubs_log

Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      6.00 Mb      6.00 Mb
                                      ------------ ------------
                                      Total        6.00 Mb      6.00 Mb
===============================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ----------- ----------- -----------
    2 Kb    716 Kb      0.00 Mb      1.00 Mb     10
    4 Kb   1024 Kb      5.00 Mb      5.00 Mb     10
```

从其它内存池移走空间

若要从其它缓冲池移走空间，应指定高速缓存名、“目标” I/O 大小和“源” I/O 大小。以下输出显示了缺省数据高速缓存的当前配置：

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     29.28 Mb
                                      ------------ ------------
                                      Total        25.00 Mb     29.28 Mb
===============================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
```

```
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    3844 Kb      0.00 Mb     18.78 Mb      10
    4 Kb     512 Kb      2.50 Mb      2.50 Mb      10
   16 Kb    1632 Kb      8.00 Mb      8.00 Mb      10
```

若要从 16K 缓冲池中移出空间，将 4K 缓冲池的大小从 2.5MB 增加到 4MB ，请输入：

```
sp_poolconfig "default data cache","4M", "4K","16K"
```

执行此命令后，输出如下：

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     29.28 Mb
                                      ------------ ------------
                                    Total      25.00 Mb     29.28 Mb
================================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    3844 Kb      0.00 Mb     18.78 Mb      10
    4 Kb     512 Kb      4.00 Mb      4.00 Mb      10
   16 Kb    1632 Kb      6.50 Mb      6.50 Mb      10
```

在高速缓存中的各缓冲池间移动缓冲区时， Adaptive Server 。不能移动正被使用的缓冲区或包含尚未写入磁盘的更改的缓冲区。

当 Adaptive Server 不能移动所请求数量的缓冲区时，将显示一条信息性消息，给出所请求的内存池大小和可以移动的内存池大小。

增加高速缓存分区数

在多引擎服务器中，多个任务可以同时尝试访问高速缓存。缺省情况下，每个高速缓存都只有一个螺旋锁，这样一次只能有一个任务可以更改或访问速缓存。如果高速缓存螺旋锁争用超过 10%，则增加高速缓存分区数可以减少螺旋锁争用，从而提高性能。

若要为以下各项配置高速缓存分区数：

- 所有数据高速缓存，请使用 global cache partition number 配置参数
- 单个高速缓存，请使用 sp_cacheconfig

高速缓存中的分区数始终为 1 到 64 之间的 2 的乘方。任何高速缓存分区中的缓冲池均不得小于 512K 。在大多数情况下，因为可以针对个别对象的存储要来调整高速缓存的大小，所以应对存在螺旋锁争用问题的特定高速缓存使用本地设置。

请参见《性能和调优系列：基础知识》的第 5 章“内存使用和性能”中的“使用高速缓存分区减少螺旋锁争用”。

设置高速缓存分区数

若要为服务器中的所有高速缓存设置高速缓存分区数，可以使用 sp_configure 。例如，若要将高速缓存分区数设置为 2，请输入：

```
sp_configure "global cache partition number",2
```

必须重新启动服务器才能使更改生效。

设置本地高速缓存分区数

使用 sp_cacheconfig 或配置文件可以设置本地高速缓存分区数。以下命令将缺省数据高速缓存中的高速缓存分区数设置为 4：

```
sp_cacheconfig "default data cache", "cache_partition=4"
```

必须重新启动服务器才能使更改生效。

优先级

本地高速缓存分区设置总是优先于全局高速缓存分区值。

以下示例将服务器范围的分区数设置为 4 ，并将 pubs_cache 的分区数设置为 2 ：

```
sp_configure "global cache partition number", 4
sp_cacheconfig "pubs_cache", "cache_partition=2"
```

本地高速缓存分区数优先于全局高速缓存分区数，因此 pubs_cache 使用 2 个分区。所有其它已配置的高速缓存均有 4 个分区。

若要删除 pubs_cache 的本地设置，而改用全局值，请输入：

```
sp_cacheconfig "pubs_cache", "cache_partition=default"
```

若要将全局高速缓存分区数重新设置为缺省值，请使用：

```
sp_configure "global cache partition number", 0, "default"
```

删除内存池

若要完全删除缓冲池，可将其大小设置为 0 。以下示例将删除 16K 缓冲池，并将所有空间都放置到缺省缓冲池中：

```
sp_poolconfig "default data cache", "0", "16K"
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     29.28 Mb
                                      ------------ ------------
                                      Total        25.00 Mb     29.28 Mb
===============================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      6.50 Mb     25.28 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
```

如果没有指定受影响的缓冲池大小（上例中为 16K ），所有空间均将放置于在缺省缓冲池中。不能删除任何高速缓存中的缺省缓冲池。

由于页的使用而不能删除缓冲池时

如果尝试删除的缓冲池中包含正在使用的页或尚未写入磁盘的脏读页，Adaptive Server 会将尽可能多的页移动到指定的缓冲池并输出信息性消息，告诉您剩余池的大小。如果缓冲池的大小小于允许的最小缓冲池大小，也将显示警告消息，说明缓冲池已标记为不可用。如果在收到这些警告之一后运行 sp_cacheconfig，则这些缓冲池的缓冲池细节部分将包含额外的“ Status ”列，受影响的缓冲池在“ Status ”列中的状态为“ Unavailable/too small ”或“ Unavailable/deleted ”。

可以稍后重新发出该系统过程以完成删除缓冲池的操作。重新启动 Adaptive Server 时，状态为“ Unavailable/too small ”或“ Unavailable/deleted ”的缓冲池也将被除。

高速缓存绑定对内存和查询计划的影响

绑定和解除绑定对象可能会对性能产生影响。绑定或解除绑定表或索引时：

- 对象的页将从高速缓存中刷新。
- 必须锁定对象才能执行绑定。
- 过程和触发器的所有查询计划必须重新编译。

从高速缓存中刷新页

将对象或数据库绑定到高速缓存时，将从源高速缓存中删除已在内存中的对象页。下一次查询需要这些页时，它们将被读入新的高速缓存。同样，解对象的绑定后，将从用户配置的高速缓存中删除高速缓存中的页，并在下一次查询需要时将这些页读入缺省高速缓存。

锁定以执行绑定

若要绑定或解除绑定用户表、索引或者文本或图像对象，高速缓存绑定命令必须有对象上的排它表锁。如果某个用户持有表上的表锁，而您又对该对发出 sp_bindcache 、 sp_unbindcache 或 sp_unbindcache_all 命令，则系统过程在获得需要的锁之前将保持休眠状态。

对于数据库、系统表和系统表上的索引，数据库必须在单用户模式中，以便没有其他用户持有对象上的锁。

高速缓存绑定对存储过程和触发器的影响

高速缓存绑定和 I/O 大小是存储过程和触发器的查询计划的一部分。更改某一对象的高速缓存绑定后，所有引用该对象的存储过程都将在下一次执行被重新编译。更改数据库的高速缓存绑定后，存储过程如果引用了数据库中任何没有被显式绑定到高速缓存的对象，这些存储过程均将在下一次运行时被重新编译。

使用配置文件配置数据高速缓存

可以通过编辑在启动 Adaptive Server 时使用的配置文件，来增加或删除命名数据高速缓存和重新配置现有的高速缓存及其内存池。

注释 在服务器运行时，不能重新配置其上的高速缓存和缓冲池。如果尝试读取的配置文件包含与已在服务器中配置的高速缓存和缓冲池不同的配置，将会致读取失败。

配置文件中的高速缓存和缓冲池条目

在配置文件中，服务器上每个已配置的数据高速缓存均包含此信息块：

```
[Named Cache:cache_name]
     cache size = {size | DEFAULT}
     cache status = {mixed cache | log only | default data cache}
     cache replacement policy = {DEFAULT |
          relaxed LRU replacement| strict LRU replacement }
```

大小的单位可按如下指定：

- P — 页（ Adaptive Server 页）
- K — 千字节（缺省）
- M — 兆字节
- G — 千兆字节

下例显示了缺省数据高速缓存的配置文件条目：

```
[Named Cache:default data cache]
      cache size = DEFAULT
      cache status = default data cache
      cache replacement policy = strict LRU replacement
```

缺省数据高速缓存条目是启动 Adaptive Server 需要的唯一高速缓存条目。该条目必须包含高速缓存大小和高速缓存状态，并且状态必须是“default data cache”。

如果高速缓存还配置了其它缓冲池，上例中的信息块之后将显示各个缓冲池的信息块：

```
[16K I/O Buffer Pool]
        pool size = size
        wash size = size
        local async prefetch limit = DEFAULT
```

注释　在某些情况下，高速缓存中没有该缓冲池的配置文件条目。如果使用 sp_poolconfig 更改异步预取百分比，则更改只写入系统表，不写入配置文件。

下例显示了 sp_cacheconfig 的输出，后面显示了与此高速缓存和缓冲池配置相匹配的配置文件条目：

```
Cache Name                Status    Type      Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache        Active    Default      29.28 Mb     25.00 Mb
pubs_cache                Active    Mixed        20.00 Mb     20.00 Mb
pubs_log                  Active    Log Only      6.00 Mb      6.00 Mb
tempdb_cache              Active    Mixed         4.00 Mb      4.00 Mb
                                      ------------ ------------
                                      Total        59.28 Mb     55.00 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 29.28 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      6.50 Mb     25.28 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 20.00 Mb,   Run Size: 20.00 Mb
```

```
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ----------- ----------- ----------
    2 Kb   2662 Kb      0.00 Mb     13.00 Mb     10
   16 Kb   1424 Kb      7.00 Mb      7.00 Mb      10
================================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ----------- ----------- ----------
    2 Kb    716 Kb      0.00 Mb      1.00 Mb     10
    4 Kb   1024 Kb      5.00 Mb      5.00 Mb      10
================================================================================
Cache: tempdb_cache,   Status: Active,   Type: Mixed
     Config Size: 4.00 Mb,   Run Size: 4.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ----------- ----------- ----------
    2 Kb    818 Kb      0.00 Mb      4.00 Mb     10
```

以下是匹配的配置文件信息：

```
[Named Cache:default data cache]
         cache size = 29.28M
         cache status = default data cache
        cache replacement policy = DEFAULT
        local cache partition number = DEFAULT

[2K I/O Buffer Pool]
        pool size = 6656.0000k
        wash size = 3844 K
        local async prefetch limit = DEFAULT

[4K I/O Buffer Pool]
        pool size = 4.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT

[Named Cache:pubs_cache]
```

```
        cache size = 20M
        cache status = mixed cache
        cache replacement policy = strict LRU
replacement
        local cache partition number = DEFAULT

[16K I/O Buffer Pool]
        pool size = 7.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT

[Named Cache:pubs_log]
        cache size = 6M
        cache status = log only
        cache replacement policy = relaxed LRU
replacement
        local cache partition number = DEFAULT

[4K I/O Buffer Pool]
        pool size = 5.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT


[Named Cache:tempdb_cache]
        cache size = 4M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = DEFAULT
```

请参见《系统管理指南，卷 1 》中的第 5 章“设置配置参数”。

警告！请检查 max memory 配置参数，为其它 Adaptive Server 需要留出足够的内存。如果在配置文件中为数据高速缓存指派了过多的内存，Adaptive Server 将无法启动。如果发生此情况，可编辑配置文件，减少数据高速缓存中的空间量，或增加分配给 Adaptive Server 的 max memory。请参见《系统管理指南，卷 1》中的第 5 章“设置配置参数”。

高速缓存配置指南

配置用户可定义的高速缓存时，请遵循以下一般准则：

- 应确保缺省数据高速缓存足够大，以适应未绑定的表和索引上的所有高速缓存活动。没有显式绑定到高速缓存上的所有对象均使用缺省高速缓存。其包括用户数据库中所有未绑定的系统表、 master 数据库中的系统表和其它没有显式绑定到高速缓存的任何对象。
- 在恢复期间，只有缺省高速缓存处于活动状态。所有必须回退或前滚的事务必须将数据页读入缺省数据高速缓存。如果缺省数据高速缓存太小，将会长恢复时间。
- 不要“闲置”任何高速缓存中的 2K 缓冲池。许多类型的数据访问都不需要使用大 I/O 。例如，使用索引为用户返回单行的简单查询可能仅使用 4 或 5 2K 的 I/O ，没有必要使用 16K 的 I/O 。
- 某些 dbcc 命令和 drop table 只能执行 2K I/O 。 dbcc checktable 可以执行大 I/O ，而 dbcc checkdb 对表执行大 I/O ，对索引执行 2K I/O 。
- 对于事务日志使用的高速缓存，可配置与缺省日志 I/O 大小匹配的 I/O 缓冲池。可以使用 sp_logiosize 为数据库设置此大小。缺省值为 4K 。
- 尝试管理每个索引和对象及其高速缓存可能会浪费高速缓存空间。如果已经创建了高速缓存或缓冲池，而捆绑到其上的表或索引没有很好地使用它们则这些高速缓存或缓冲池将浪费空间并在其它高速缓存中创建额外的 I/O 。
- 如果应用程序频繁使用 tempdb ，可将它绑定到自己的高速缓存上。只能绑定整个 tempdb 数据库，不能绑定 tempdb 中的单个对象。
- 对于更新率和替换率高的高速缓存，请确保清洗大小足够大。
- 在多 CPU 系统中，应将最忙的表及其索引分布到多个高速缓存中，以避免螺旋锁争用。
- 可考虑重新配置高速缓存或高速缓存中的内存池以适应更改的工作量。重新配置高速缓存需要重新启动服务器，但不需要重新配置内存池。

  例如，如果系统在大多数月份主要执行 OLTP （联机事务处理），而只有几天会进行频繁的 DSS（决策支持系统）活动，可考虑在 DSS 活动频繁时将空间 2K 缓冲池移动到 16K 缓冲池，而在 DSS 负载结束后重新为 OLTP 调整缓冲池的大小。

配置文件错误

如果手动编辑配置文件，需要仔细检查高速缓存、缓冲池和清洗大小。所有高速缓存的总大小不能大于 max memory 量与其它 Adaptive Server 内存需要之差。

大多数情况下，缺失条目的问题将立即作为“未知格式”错误在紧邻缺少大小、状态或其它信息的条目之后的行上报告。其它错误提供了出现错误的速缓存名称和错误的类型。例如，如果未正确指定池的清洗大小，将会显示以下错误：

```
The wash size for the 4k buffer pool in cache pubs_cache
has been incorrectly configured.It must be a minimum of
10 buffers and a maximum of 80 percent of the number of
buffers in the pool.
```

第 5 章

管理多处理器服务器

Adaptive Server 使用 Sybase Virtual Server Architecture™，它使 Adaptive Server 能够利用对称多重处理 (SMP) 系统的并行处理功能。可以将 Adaptive Server 作为单个进、单个多线程进程或多个协同进程运行，这取决于可用的 CPU 数量以及服务器的要求。本章介绍：

- SMP Adaptive Server 的目标计算机体系结构
- SMP 环境的 Adaptive Server 体系结构
- SMP 环境中的 Adaptive Server 任务管理
- 管理多个引擎

有关 SMP 系统的应用程序设计的信息，请参见 《性能和调优系列：基础知识》的第 3 章 “使用引擎和 CPU”。

Adaptive Server 内核

Adaptive Server 15.7 版和更高版本包括两个内核：线程化内核和进程内核。您为 Adaptive Server 配置的内核决定 Adaptive Server 的运行模式：

- 线程模式 — Adaptive Server 作为单个多线程化操作系统进程运行，通过在线程池中的线程上运行的引擎来处理 SQL 查询。线程模式利用没有引擎的线程管理 I/O。管理员可以配置更多线程池来管理负载。

- 进程模式 — Adaptive Server 作为相互合作的多个操作系统进程运行，像单个服务器那样工作。进程模式使用引擎来管理 I/O，管理员配置引擎组来管理负载。

  注释 进程模式在 Windows 中不可用。

对于许多负载，线程模式使用的 CPU 比进程模式少得多，但能提供同样的，甚至更好的性能。线程模式不需要太多任务 - 引擎密切连接，因此，能在混 I/O 密集和 CPU 密集的负载中提供更加一致的性能。

线程化内核能让 Adaptive Server 利用比较低版本内核具有更多处理器、处理器内核以及硬件线程的并行硬件和支持系统。

虽然 15.7 版更改了内核，但查询处理器仍保持不变。要在线程化内核模式中运行，您无需更改大多数针对较低版本 Adaptive Server 编写的脚本，只是有个命令和存储过程发生了更改。应用程序完全与线程模式兼容。

目标体系结构

SMP 产品设计用于具有下列特性的计算机：

- 对称多重处理操作系统
- 通过公用总线共享内存
- 1-1024 个处理器、内核或硬件线程
- 没有主处理器
- 非常高的吞吐量

Adaptive Server 由一个或多个协同进程组成，这些进程由操作系统在物理 CPU 中进行调度。

在进程模式下运行时， Adaptive Server 使用多个协同进程来利用并行硬件；在线程模式下运行时，则使用同一进程的多个线程。进程模式内核中的每个进程都是一个 Adaptive Server 引擎。线模式内核使用一些线程作为引擎，另外还有一些非引擎线程。

进程模式使用多线程进程。但是，因为 Adaptive Server 的大多数工作都是在每个进程的主线程中执行的，所以在搜索和调优 CPU 资源时，应将这些进程视为单线程进程。

Adaptive Server 使用引擎作为处理器来执行 SQL 查询。在进程模式中，引擎是每个进程的主线程。在线程模式中，引擎是一个或多个引擎线程池中的线程多个引擎之间通过共享内存进行通信。进程模式和线程模式都使用共享内存，包括单引擎或单处理器环境。

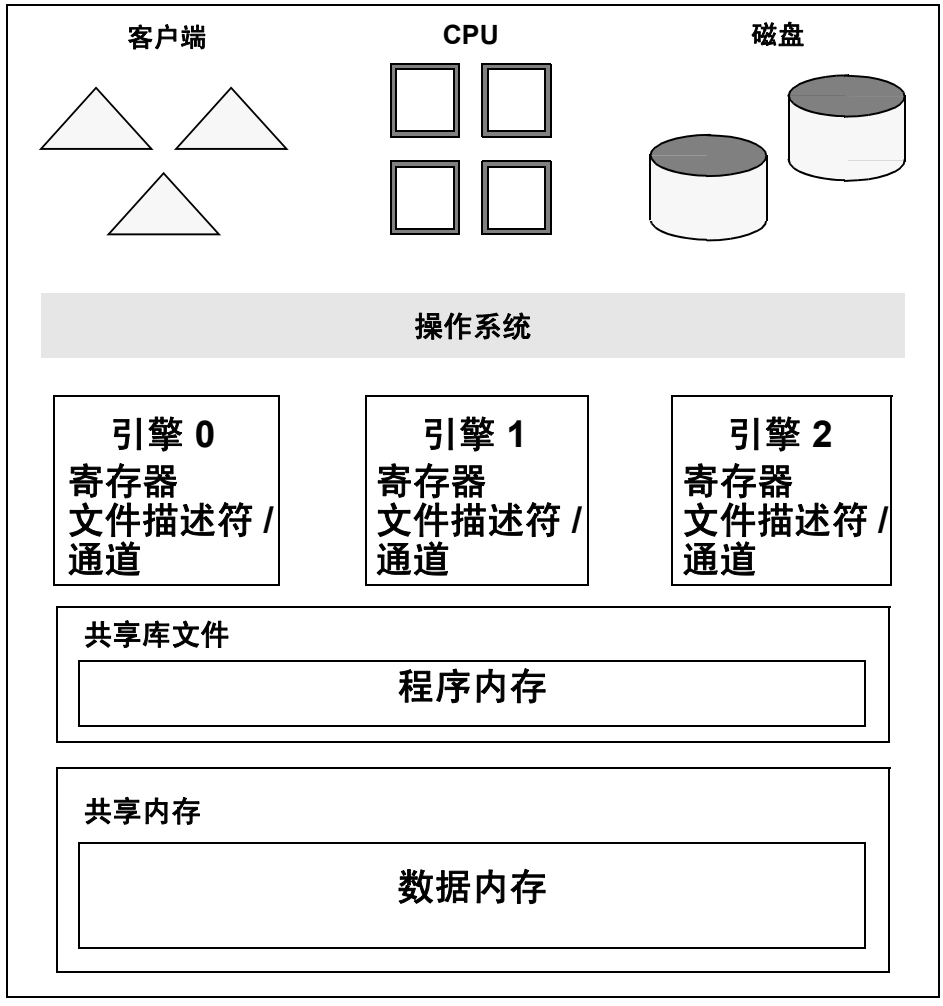
操作系统根据 CPU 资源调度 Adaptive Server 线程。 Adaptive Server 不区分物理处理器、内核或子核线程。

配置为内核模式时， Adaptive Server 在单个操作系统进程的线程内执行引擎。 Adaptive Server 从引擎线程池中获取用于支持引擎的线程。

Adaptive Server 包含其它非引擎线程，它们用于执行特殊任务，不应将它们视为引擎。

图 5-1：线程模式的体系结构

图 5-2：进程模式的体系结构



操作系统根据物理 CPU 资源调度 Adaptive Server 线程（引擎和非引擎），这可以是处理器、内核或子核线程。操作系统（而非 Adaptive Server）将线程指派 CPU 资源。 Adaptive Server 的性能取决于从操作系统接收的 CPU 时间。

Adaptive Server 引擎执行所有数据库功能，包括更新和日志记录。Adaptive Server （而不是操作系统）以动态方式为可用引擎调度客户端任务。任务是 Adaptive Server 中的执行环境。

“关联”是一个过程，在该过程中，特定 Adaptive Server 任务仅在特定引擎上运行 （任务关联）、特定引擎处理特定任务的网络 I/O （网络 I/O 关联）或特定引擎仅在特定 CPU 上运行 （引擎关联）。

在进程模式中，与 Adaptive Server 的连接与接受该连接的引擎之间具有网络 I/O 关联。该引擎必须执行该连接的所有网络 I/O。网络 I/O 关联在线程模式中不存在，因为任何引擎都能执行任何连接的 I/O，这样通常可以减少环境切换，提高性能。

可以使用逻辑进程管理器建立任务关联，以便管理器仅在特定引擎或特定引擎集上运行某一任务或某一组任务。在线程模式中，使用线程池完成任务联。在进程模式中，使用引擎组。

线程池和引擎组具有不同的行为。线程池中的引擎仅执行指派给该线程池的任务。调度程序搜索空间 （调度程序搜索可运行任务的位置）仅限于该线池中的引擎。引擎组中的引擎可运行任何任务，只要未限制该任务只能由其它组中的引擎执行即可。也就是说，将任务包含在引擎组中限制可运行务的位置，但是不会为该任务保留组中的引擎。

请使用 Adaptive Server 逻辑进程管理器配置任务关联 （参见 《性能和调优系列：基础知识》的第 4 章 “分配引擎资源”）。请使用 dbcc tune 或等效的操作系统命令配置引擎或 CPU 关联 （参见 《参考手册：命令》和所用操作系统的文档）。

内核模式

缺省情况下， Adaptive Server 在线程模式下启动，在操作系统中显示为单个进程，并且使用本机操作系统线程实现并行度。缺省情况下， Adaptive Server 使本机线程处理 I/O。在进程模式中， Adaptive Server 在操作系统中显示为多个进程。

进程模式可能不支持较高版本的 Adaptive Server 中提供的功能。Sybase 希望，在线程模式下无法正常运行时再使用进程模式作为后备方案。

注释 出于体系结构考虑，进程模式在 Windows 平台上不可用。早期版本的 Adaptive Server 在 Windows 上也使用多线程、单进程内核。

若要确定 Adaptive Server 的当前模式，请使用以下命令：

```
select @@kernelmode
-------------------------------------
threaded
```

切换内核模式

可使用 sp_configure “kernel mode” 切换内核模式。必须重新启动 Adaptive Server 才能使更改生效。切换模式时，Adaptive Server 将发出类似以下内容的消息：

```
sp_configure "kernel mode", 0, process
Parameter Name                  Default              Memory Used
        Config Value         Run Value            Unit
        Type
------------------------------ -------------------- -----------
        -------------------  -------------------  --------------------
        --------------------
kernel mode                     threaded             0
             process        threaded             not applicable
        static

(1 row affected)
Configuration option changed.Since the option is static, Adaptive Server must
be rebooted in order for the change to take effect.Changing the value of 'kernel
mode' does not increase the amount of memory Adaptive Server uses
```

切换时应考虑以下问题：

- 从线程模式切换到进程模式时， Adaptive Server 在启动时会忽略配置文件中的线程池信息。
- 从进程内核模式切换到线程模式时，所有执行类不再关联到原来的引擎组，它们将与 syb_default_pool 相关联。管理员然后可将执行类与引擎线程池关联起来。从线程模式切换到进程模式时，执行类不再关联到原来的线程池，它们将与现有 ANYENGINE 引组相关联。管理员然后可将执行类与引擎组关联起来。对执行类的重新指派 （重新指派给线程池或引擎组），除了应用于新登录名之外，还应用于该执行类相关联的现有连接。
- 在线程模式下运行时，引擎组会被弃用。在该模式下，可使用 sp_addexeclass 将执行类绑定到用户线程池。线程池在 Adaptive Server 中的应用程序之间分配处理器资源。

任务

任务是 Adaptive Server 内部的可执行环境 （例如，用户连接、管家任务等守护程序和 I/O 处理等内核任务）。Adaptive Server 将任务调度给线程，线程为每任务指派 CPU 时间。

Adaptive Server 任务包括：

- 用户任务 — 表示用户连接，能够执行用户查询。例如， isql 连接就是一个用户任务。
- 服务任务 — 不与特定用户相关联，不发出用户查询。服务任务包括管家任务、检查点、复制代理线程等。
- 系统任务 — 内核级版本的服务任务。系统任务包括 I/O 处理程序、时钟处理程序、集群成员资格服务、 IP 链接监控器等。

Adaptive Server 跨线程对任务进行多路复用。运行完毕 (RTC) 线程池将任务放置在线程池中，下一可用线程选取该任务 （该任务在完成前一直位于该线程），并在任务完成运行时终止。

有关单个任务的任务优先级设置的说明，请参见 《性能和调优系列：基础知识》的第 4 章 “分配引擎资源”中的 “基本优先级”。

对于在 Adaptive Server 上运行的每个任务， monTask 监控表中都有对应的行。请参见 《参考手册：表》。

使用线程运行任务

Adaptive Server 将任务指派给线程池，所有线程池都具有线程。每个线程池都包含一个调度程序，用于将任务指派给线程。调度程序在线程工作时将多路复用任务指派给线程 （或取消指派）。调度程序将 RTC 任务指派给某一线程一次，它将一直位于该线程上，直到工作完成。

Adaptive Server 包含系统创建的线程池和用户创建的线程池。所有系统定义的线程池都以 syb_ 前缀开头 （例如 syb_default_pool）。用户定义的线程池的名称不能以 syb_ 前缀开头，并且必须遵循对象的命名约定 （请参见 《参考手册：块》的第 1 章 “SQL 构件块”）。缺省情况下， Adaptive Server 将用户任务与 syb_default_pool 线程池关联起来。可使用 sp_bindexeclass 将一个或多个任务与用户创建的线程池关联起来。

配置 SMP 环境

配置 SMP 环境与配置单处理器环境相似，尽管 SMP 计算机的功能通常更为强大而且可以处理更多的用户。SMP 环境提供了控制引擎数的其它功能。

线程池

线程池将 CPU 资源组合在一起，它包含用于执行与其关联的 Adaptive Server 任务的线程。线程承载执行用户任务、运行特定作业 （如信号处理）和处理来自工作队列的请求的引擎。 Adaptive Server 包含系统定义的线程池和用户创建的线程池 （如果有）。

注释　只有在 Adaptive Server 配置为线程模式时，线程池才可用。

包含引擎的线程池称为引擎池，它们执行具有内核进程 ID (KPID) 的 Adaptive Server 任务。 Adaptive Server 将引擎指派给引擎池中的每个线程。

Adaptive Server 支持两种类型的线程：

- 引擎 （或多路复用）线程 — 执行数据库查询进程，可在多个数据库进程之间共享。多路复用线程运行与其它任务共享一个或多个线程的任务。缺省况下，用户和服务任务是多路复用型的。多路复用任务在消耗完定义的时间片后或阻塞时必须退让。因为引擎会指派给所有多路复用线程，所以多路复用线程相当于进程模式中的引擎。
- 运行完毕 (RTC) 线程 — 由系统任务使用，不在多个任务之间共享。 RTC 线程运行单个任务，直到任务完成，不受 Adapative Server 调度的影响。

  RTC 线程运行的任务在线程完成之前不能自行从线程的日程表中删除，不因时间片退让，在阻塞时仍保持与线程的连接。如果所有 RTC 线程当前都在运任务，那么 RTC 任务在线程执行任务期间可能需要等待。

创建线程池时，不能指定线程的类型。用户创建的线程池始终是多路复用型的。

Adaptive Server 包含以下系统定义的线程池：

- syb_default_pool — 缺省引擎线程池。 syb_default_pool 中的每个线程都是引擎。所有用户任务和所有多路复用型系统任务 （如管家）都在 syb_default_pool 中运行。然而，您可以通过创建其它线程池，将一些任务从 syb_default_pool 中移出。
- syb_system_pool — 用于系统线程的 RTC 线程池。syb_system_pool 中的每个线程都专用于运行一个特定任务。 syb_system_pool 至少包含一个用于系统时钟和其它异步信号的线程。所有 I/O 处理线程都在 syb_system_pool 中运行。
- syb_blocking_pool — 一个 RTC 池，Adaptive Server 使用它处理多路复用任务的阻塞调用请求，阻塞调用请求通常是指可能导致多路复用 （或引擎）线程阻塞较长时间的操系统调用。syb_blocking_pool 中的线程消耗的 CPU 资源通常很少。

线程池由它们的属性定义，一些属性是由 Adaptive Server 自动指派的，其它属性由用户在创建线程池时确定。线程池属性包括：

- ID — 系统指派给线程池的 ID。 Adaptive Server 在启动期间可能会为线程池指派新的 ID，因此 Sybase 建议您不要对线程池 ID 进行静态引用。
- Name — 线程池名称。只有系统线程池能以 syb_ 前缀开头。
- Description — （可选）线程池说明，最多 255 个字符。
- Type — 它的值只能是 `Engine` 或 `RTC` 中的一个。
- Current number of threads — 池当前包含的线程数 （在池更改大小时可能会有一小段时间不同于池中配置的线程数）。
- Configured number of threads — 为线程池配置的线程数。
- idle timeout — 线程空闲多长时间后进入休眠状态 （以微秒为单位）。

Adaptive Server 在配置文件中记录线程池配置。下面的示例显示了三个缺省线程池 (syb_blocking_pool、 syb_system_pool 和 syb_default_pool) 和一个用户创建的名为 big_pool 的线程池：

```
[Thread Pool:big_pool]
     description = Big thread pool
     number of threads = 15

[Thread Pool:syb_blocking_pool]
     number of threads = 20

[Thread Pool:syb_default_pool]
      number of threads = 1
```

可使用 sp_helpthread 或 monThreadPool 监控表查看当前的线程池配置。在启动 Adaptive Server 之前可在配置文件中编辑线程池信息，也可以使用 alter thread pool 更改线程池配置。

请参见 《系统管理指南，卷 2》中的第 3 章 “配置内存”。

管理引擎

为了获得 SMP 系统的最佳性能，必须保持正确的引擎数。

一个引擎代表一定数量的 CPU 处理能力。它是像内存一样的可配置资源。

注释　Adaptive Server 使用一种负载平衡算法在引擎间均匀地分配负载。在进程模式中，如果服务器连接使用组件集成服务 (CIS)，它们就会关联到单个引擎，并且不能从一个引擎迁移到另一引擎。该限制不适用于线程模式。

在进程模式中配置引擎

第一次安装 Adaptive Server 时，系统配置为单个引擎。若要使用多个引擎，请在第一次重新启动服务器时重新设置引擎数。在其它时间，您可能还要重设置引擎数。例如，您可能需要设置：

- 如果当前性能不能满足应用程序的要求，并且计算机上有足够的 CPU，则需要增加引擎数。
- 如果 Adaptive Server 没有完全利用现有引擎，则需减少引擎数。运行引擎时会涉及到开销， Adaptive Server 运行不需要的额外引擎时会浪费资源。

在线程模式中配置引擎

可在配置文件中编辑线程池信息，也可以使用 create thread pool、alter thread pool 和 drop thread pool 管理线程池。请参见第 45 页的 “配置线程池”。

max online engines 控制 Adaptive Server 可用的引擎总数。

可以使用 sp_configure 重新设置 max online engines。例如，若要将引擎数设置为 3，请发出：

```
sp_configure "max online engines", 3
```

重新启动服务器以重新设置引擎数。

选择正确的引擎数

为 Adaptive Server 选择正确的引擎数很重要：

- 配置的引擎数不能多于 CPU。如果某个 CPU 脱机，可能需要使用 sp_engine （在进程模式中）或 alter thread pool （在线程模式中）减少引擎的数目。
- 在线程模式中， Adaptive Server 可以使用其它操作系统线程处理 I/O 操作。在 I/O 较高的情况下，必须确保 Adaptive Server 具有足够的 CPU，以便这些线程可正常运行。例如，如果 Adaptive Server 在具有 8 个 CPU 的系统上处理高 I/O，则在进程模式下可能需要 8 个引擎才能获得最佳性能，但在线程模式下只需 7 个引擎。如果 I/O 线程与引擎线程争用 CPU 时间，则性能可能会显著降低。
- 引擎数应当与可用的 CPU 的数量一样。如果客户端或其它非 Adaptive Server 进程执行大量处理，那么每个 CPU 一个引擎可能就太多了。操作系统可能会占其中一个 CPU 的一部分。
- 有足够的引擎。开始使用几个引擎，然后在现有 CPU 几乎被完全使用时再添加引擎。如果引擎太少，则会超出现有引擎的处理能力，并且可能会产生颈。

Sybase 建议您仅配置节点所需的引擎数。引擎多不一定能提高性能。通常，对于 Adaptive Server 来说， 4 个引擎中 80% 处于忙碌状态时比 8 个引擎中 40% 处忙碌状态时的性能要好。

启动和停止引擎

可以使用 sp_engine 启动和停止 Adaptive Server 引擎。

注释 Adaptive Server 必须配置为进程模式才能使用 sp_engine。Adaptive Server 配置为内核模式时，可使用 create、 alter 和 drop thread 配置内存池。请参见第 45 页的 “配置线程池”。

监控引擎状态

在使引擎联机或脱机之前，请检查当前正在运行的引擎的状态。sysengines 的 status 列中包含以下一种状态：

- online — 表示引擎处于联机状态。
- in offline — 表示已经运行了 sp_engine offline。引擎仍被分配给服务器，但它的任务正在迁移到其它引擎。
- in destroy — 表示所有任务均已成功迁移出引擎，并且服务器正在等待操作系统级别的任务释放引擎。
- in create — 表示正在使引擎联机。
- dormant — 表示 sp_engine offline 不能迁移该引擎中的所有任务。如果任务自行终止（由于超时），引擎会切换到永久脱机状态。休眠的引擎只处理引起休眠状态的那些任务；这些引擎不能用于处理任何其它任务。

以下命令显示了引擎数、状态、密切连接的任务数量以及引擎被联机的时间：

```
select engine, status, affinitied, starttime
from sysengines
engine status       affinitied  starttime
------ ------------ ----------- --------------------------
     0 online                12        Mar  5 2007 9:40PM
     1 online                 9        Mar  5 2007 9:41PM
     2 online                12        Mar  5 2007 9:41PM
     3 online                14        Mar  5 2007 9:51PM
     4 online                 8        Mar  5 2007 9:51PM
     5 in offline            10        Mar  5 2007 9:51PM
```

使用 *sp_engine* 启动和停止引擎

使用 sp_engine 可以动态停止或启动引擎。利用该命令，系统管理员可以在处理要求随时间不断变化时，重新配置 CPU 资源。

sp_engine 的语法为：

sp_engine { “online” | [offline | can_offline] [, *engine_id*] | [“shutdown” , *engine_id*]

例如，以下命令使引擎 1 联机。消息是平台特定的 （在本例中，使用了 Sun Solaris）：

```
sp_engine "online", 1
02:00000:00000:2001/10/26 08:53:40.61 kernel  Network and device connection
limit is 3042.
02:00000:00000:2001/10/26 08:53:40.61 kernel  SSL Plus security modules loaded
successfully.
02:00000:00000:2001/10/26 08:53:40.67 kernel  engine 1, os pid 8624  online
02:00000:00000:2001/10/26 08:53:40.67 kernel  Enabling Sun Kernel asynchronous
disk I/O strategy
00:00000:00000:2001/10/26 08:53:40.70 kernel  ncheck:Network fc0330c8 online
```

可以使用 can_offline 检查是否可以使特定的引擎脱机。例如，若要检查是否可以使引擎 1 脱机，请使用：

```
sp_engine can_offline, 1
```

如果可以使指定的引擎脱机， sp_engine 指定返回码 0。如果不指定 *engine_id*， sp_engine 将显示 sysengines 中 *engine_id* 最高的引擎的状态。

可以使引擎联机的前提条件是：max online engines 大于当前具有 online 状态的引擎的数目，并且有足够的 CPU 可用来支持额外的引擎。

若要使某个引擎脱机，请输入引擎 ID。例如，若要使引擎 1 脱机，请使用：

```
sp_engine offline, 1
```

Adaptive Server 将等到与该引擎相关联的所有任务都完成后，才会使该引擎脱机，并返回一条类似以下内容的消息：

```
01:00000:00000:2001/11/09 16:11:11.85 kernel  Engine 1 waiting for affinitied
process(es) before going offline
00:00000:00000:2001/11/09 16:16:01.90 kernel  engine 1, os pid 21127  offline
```

不能使引擎 0 脱机。

sp_engine “shutdown” 将强制与指定引擎相关联的所有任务在五秒内完成，然后关闭该引擎。引擎进入休眠状态时或者要使引擎脱机时，可以使用 sp_engine shutdown。 sp_engine 会注销所有阻止引擎正常脱机的任何其余进程。以下命令关闭引擎 1：

```
sp_engine "shutdown", 1
```

请参见 《参考手册：过程》。

网络连接和引擎之间的关系

（仅进程模式）由于操作系统对 UNIX 上每个进程中文件描述符数目的限制，引擎数的减少会使服务器可以具有的网络连接数减少。在 Windows 平台上，络连接数与引擎数无关。

对于为服务器到服务器的远程过程调用创建的网络连接（例如，到 Replication Server 和 XP Server 的连接），没有办法对其进行迁移，因此不能使正在管理这些连接之一的引擎脱机。

逻辑进程管理和 *dbcc engine(offline)*

如果正在使用逻辑进程管理将特定的登录或应用程序绑定到引擎组，则需小心地使用 dbcc engine(offline)。如果将引擎组的所有引擎脱机，则：

- 登录或应用程序可以在任何引擎上运行
- 一条建议消息被发送到登录到服务器的连接

因为当客户端登录时会指派引擎关联，所以如果使用 dbcc engine("online") 将引擎组中的引擎再次联机，则不迁移已登录的用户。

管理用户连接 （仅进程模式）

在进程模式期间，如果 SMP 系统支持网络关联迁移（网络 I/O 从一个引擎移动到另一个引擎的进程。支持此迁移的 SMP 系统允许 Adaptive Server 将网络 I/O 负载分配给所有引擎），每个引擎都处理其自己各个连接的网络 I/O。登录时，Adaptive Server 会将客户端连接任务从引擎 0 迁移到当前服务的连接数最的引擎。客户端任务将在该引擎 （网络密切连接）上运行网络 I/O，直到该连接终止。若要确定您的 SMP 系统是否支持此迁移，请参见所用平台的相应配置文档。

通过将网络 I/O 分配给各个引擎，Adaptive Server 能够处理更多的用户连接。每个进程对打开文件描述符的最大数的限制将不再限制连接的数量。线性添更多的引擎会增加存储在全局变量 *@@max_connections* 中的文件描述符的最大数目。

如果增加了引擎数，重新启动服务器之后 Adaptive Server 会将增大的 *@@max_connections* 值输出到标准输出和错误日志文件。可以使用以下命令来查询该值：

```
select @@max_connections
```

这个数目表示操作系统对于您的进程所允许的最大文件描述符数，减去以下由 Adaptive Server 使用的文件描述符：

- 引擎 0 上的每个主网络监听器使用的一个描述符 （该 Adaptive Server 的 *interfaces* 文件条目中每一 “master” 行使用一个描述符）
- 每个引擎的标准输出使用的一个描述符
- 每个引擎的错误日志文件使用的一个描述符
- 每个引擎的网络密切连接迁移通道使用的两个描述符
- 每个引擎用于配置的一个描述符
- 每个引擎用于 *interfaces* 文件的一个描述符

例如，如果为 Adaptive Server 配置了一个引擎，而且 *@@max_connections* 的值等于 1019，再增加一个引擎会将 *@@max_connections* 的值增大到 2039（如果只有一个主网络监听器）。

可以配置 number of user connections 参数以利用增大的 *@@max_connections* 限制。不过，每次使用 max online engines 减少引擎数时，也必须相应地调整 number of user connections 的值。对 max online engines 或 number of user connections 的重新配置并不是动态的，因此必须重新启动服务器才能更改这些配置值。请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

注释 由于线程模式运行单个进程，因此每进程描述符限制将限制 Adaptive Server 可以支持的用户连接数。可能需要调整启动 Adaptive Server 的 shell 中可用的文描述符数，这可能需要操作系统管理员帮助提高硬限制。

影响 SMP 系统的配置参数

部分配置参数 （例如 spinlock ratio）仅适用于 SMP 系统。请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

配置 spinlock ratio 参数

Spinlock ratio 参数指定内部系统资源的数量，例如，由一个**螺旋锁**保护的内部表或高速缓存中的行数。螺旋锁是一种简单的锁定机制，用于阻止某一进程访问当前正由另一进程使用的系统资源。在释放锁之前，试图问资源的所有进程必须等待。

Spinlock ratio 配置参数只在多重处理系统中有意义。仅配置了一个引擎的 Adaptive Server 只有一个螺旋锁，而不管为 spinlock ratio 配置参数指定的值是多少。

表 5-1 列出了由螺旋锁保护的系统资源以及可用来更改缺省 spinlock ratio 的配置参数。

表 5-1：Spinlock ratio 配置参数

配置参数	被保护的系统资源
lock spinlock ratio	锁定散列桶的数量
open index hash spinlock ratio	索引元数据描述符散列表
open index spinlock ratio	索引元数据描述符
open object spinlock ratio	对象元数据描述符
partition spinlock ratio	内部分区高速缓存中的行
user log cache spinlock ratio	用户日志高速缓存

为 spinlock ratio 参数指定的值定义特定资源与螺旋锁的比率，而不是螺旋锁的数量。例如，如果将 spinlock ratio 的值指定为 100，则 Adaptive Server 为每 100 个资源分配一个螺旋锁。由 Adaptive Server 分配的螺旋锁的数量取决于资源的总数和所指定的比率。为 spinlock ratio 指定的值越低，螺旋锁的数量就越高。

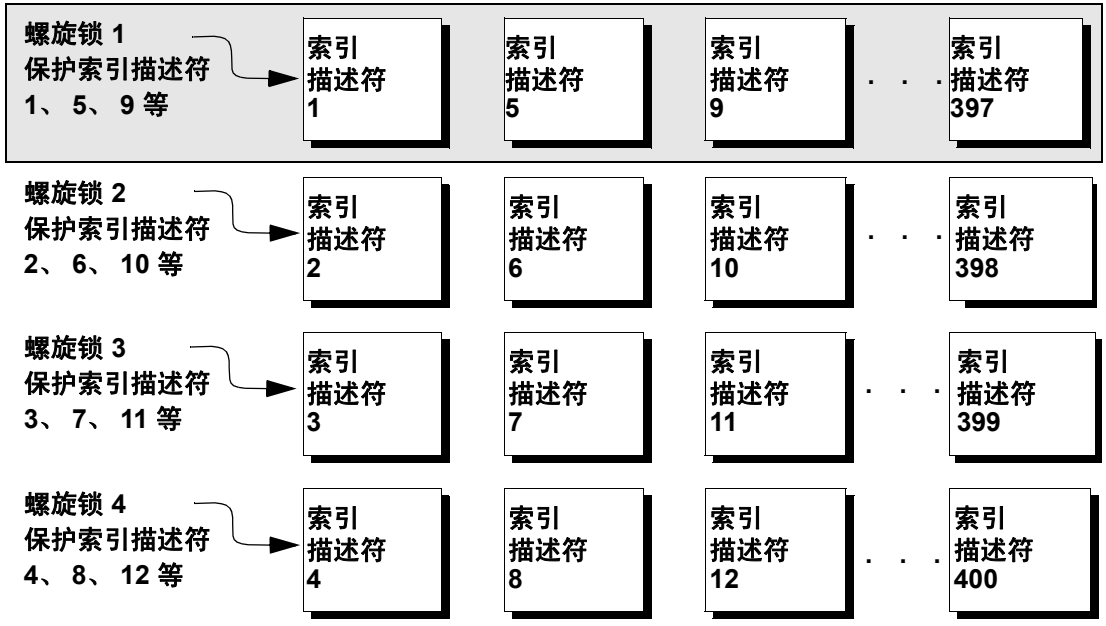
螺旋锁以循环方式或按顺序指派给系统资源：

循环指派

元数据高速缓存螺旋锁（由 open index hash spinlock ratio、open index spinlock ratio 和 open object spinlock ratio 参数配置）使用循环指派方法。

图 5-2 说明了循环指派方法的一个示例，显示了螺旋锁和索引元数据描述符之间的关系。

图 5-3：螺旋锁和索引描述符之间的关系



假设有 400 个索引元数据描述符，即索引描述符内部表中有 400 行。您已将比率设置为 100。这意味着总共有 4 个螺旋锁：螺旋锁 1 保护第 1 行，螺旋 2 保护第 2 行，螺旋锁 3 保护第 3 行，螺旋锁 4 保护第 4 行。之后，螺旋锁 1 保护下一个可用索引描述符，即索引描述符 5，直到每个索引描述符都一个螺旋锁保护。这种循环指派描述符的方法减少了螺旋锁争用的可能性。

顺序指派

表锁螺旋锁 （由 table lock spinlock ratio 参数配置）使用顺序指派方法。 table lock spinlock ratio 的缺省配置为 20，即给每个螺旋锁指派内部散列表中的 20 行。这些行按顺序分为：第一个螺旋锁保护前 20 行，第二个螺旋锁保护第二个 20 行，依此类推。

从理论上讲，使用一个螺旋锁保护一个资源将提供螺旋锁的最小争用，并且将导致最高的并发性。在大多数情况下，这些 spinlock ratio 的缺省值对系统来说可能是最佳的。只有在有螺旋锁争用的情况下才更改该比率。

使用 sp_sysmon 可获取有关螺旋锁争用的报告。请参见 《性能和调优系列：使用 sp_sysmon 监控 Adaptive Server》。

第 6 章

创建和管理用户数据库

创建和管理用户数据库的命令

表 6-1 总结了用来创建、修改和删除用户数据库及其事务日志的命令。

表 6-1：管理用户数据库的命令

命令	任务
create database...on *dev_name* 或 alter database...on *dev_name*	使数据库设备可供某个 Adaptive Server 数据库使用。 在不与 on *dev_name* 子句一起使用时，这些命令从数据库设备的缺省池分配空间。
dbcc checktable(syslogs)	报告日志的大小。
sp_logdevice	指定一个在当前日志设备已满时存储日志的设备。

命令	任务
sp_helpdb	报告关于数据库大小和数据库设备的信息。
sp_spaceused	报告数据库占用存储空间量的概况。

管理用户数据库的权限

缺省情况下，只有系统管理员有 create database 权限，但是系统管理员可以授予他人使用 create database 命令的权限。不过，在许多安装中，为了对数据库位置和数据库设备分配进行集中控制，系统管理员保留对 create database 权限的独占。在这种情况下，系统管理员代表其他用户创建新的数据库，然后将所有权转交给相应的用户。

若要创建数据库并将所有权转交给另一用户，系统管理员需要执行以下步骤：

1 发出 create database 命令。

2 使用 use *database* 命令切换到新的数据库。

3 按第 145 页的“更改数据库所有权”中所述，执行 sp_changedbowner。

当系统管理员授权用户创建数据库时，获得权限的用户必须同时也是 master 数据库的有效用户，因为创建所有数据库时都要使用 master。

系统管理员在保护系统之外操作是一种安全预防措施。例如，如果一个数据库的所有者忘记了其口令或意外地删除了 sysusers 的所有条目，系统管理员可以使用定期进行的备份或转储修复损坏。

alter database 或 drop database 的权限缺省授予数据库所有者，且此权限自动随数据库所有权一起转交。不能使用 grant 或 revoke 来更改 alter database 和 drop database 权限。

使用 *create database* 命令

若要使用 create database，您必须拥有 create database 权限，同时必须是 master 的有效用户。创建新的数据库之前务必键入 use master。请参见《参考手册：命令》。

注释　每次输入 create database 命令时，请转储 master 数据库。这使得以后 master 损坏时可更容易和安全地恢复。请参见第 14 章“恢复系统数据库”。

一次只能创建一个数据库。

如果使用最简单的格式，create database 将在 master..sysdevices 中所列的缺省数据库设备上创建一个数据库：

```
create database newpubs
```

具有所需权限的用户发出 create database 命令后，Adaptive Server 将执行以下操作：

- 检验指定的数据库名称是否唯一并且遵循标识符的规则。
- 确保指定的数据库设备名可用。
- 为新数据库找到一个未使用过的标识号。
- 为指定数据库设备上的数据库指派空间并更新 master..sysusages 来反映这些指派。
- 在 sysdatabases 中插入一行。
- 在新数据库空间中制作 model 数据库的一个副本，从而创建新数据库的系统表。
- 清除数据库设备中的所有剩余页。如果正在创建数据库以装载数据库转储，则 for load 跳过将在装载完成之后执行的页面清除。

新数据库最初包含一组带有描述系统表本身的条目的系统表。它继承了对 model 数据库的所有变动，这些变动包括：

- 用户名的增加。
- 对象的增加。
- 数据库选项设置。最初，选项在 model 中设置为 off。如果想要所有的数据库都继承特定的选项，请使用 sp_dboption 更改 model 数据库中的选项。请参见《系统管理指南，卷 1》中的第 2 章“系统和可选数据库”以及第 8 章“设置数据库选项”。

创建新数据库后，系统管理员或数据库所有者可以使用 sp_adduser 手动向数据库添加用户。如果要添加新 Adaptive Server 登录名，可能还需要系统安全员。请参见《系统管理指南，卷 1》中的第 13 章“Adaptive Server 中的安全性管理快速入门”。

为数据库指派空间和设备

当用户输入 create database 或 alter database 命令时，Adaptive Server 会为数据库分配存储空间。create database 可以指定一个或多个数据库设备，以及每个数据库设备上要分配给新数据库的空间量。

警告！除非是创建很小或不重要的数据库，否则应始终把日志存放到单独的数据库设备上。按照第 130 页的“把事务日志存放在单独的设备上”中的说明创建生产数据库。

如果使用了 default 关键字或省略了 on 子句，则 Adaptive Server 会将数据库放到一个或多个在 master..sysdevices 中指定的缺省数据库设备上。请参见《系统管理指南，卷 1》中的第 7 章“初始化数据库设备”。

若要为将存储到缺省位置的数据库指定空间大小（在下例中为 4MB），请使用：

```
create database newpubs
on default = "4M"
```

若要将数据库存放到特定的数据库设备上，请在命令中包含数据库设备的名称。可以请求将数据库存储到具有不同空间量的多个数据库设备上。在 create database 中指定的所有数据库设备必须在 sysdevices 中列出（即必须已使用 disk init 对它们进行初始化）。请参见《系统管理指南，卷 1》中的第 7 章“初始化数据库设备”。

下面的语句创建了 newdb 数据库并在 mydata 上分配了 3MB 的空间，在 newdata 上分配了 2MB 的空间。数据库和事务日志未分开：

```
create database newdb
on mydata = "3M", newdata = "2M"
```

如果指定的数据库设备上没有所请求的空间量，则 Adaptive Server 会使用每个设备上尽可能多的空间来创建数据库，并显示消息说明已在每个数据库设上分配了多少空间。如果指定数据库设备上的空间量小于数据库所需的最小空间量，则 create database 命令失败。

如果在没有使用 dsync 设置的 UNIX 设备文件中创建（或变更）数据库，Adaptive Server 将在错误日志文件中显示错误消息，例如：

```
Warning:The database 'newdb' is using an unsafe virtual device 'mydata'.The
recovery of this database can not be guaranteed.
```

缺省数据库大小和设备

如果省略 on 子句中的 size 参数，则 Adaptive Server 将使用缺省空间量创建数据库。缺省空间量是 default database size 配置参数和 model 数据库指定的大小中较大的一个。

可以创建的最小数据库的大小是 model 数据库的大小，它由安装的逻辑页大小确定。若要增加数据库的最小大小，请使用 alter database 来扩大 model 数据库。也可以使用 default database size 配置参数来确定缺省数据库大小。请参见《系统管理指南，卷 1》中的第 5 章“设置配置参数”。

如上所述，如果省略 on 子句，则将按照缺省大小创建数据库。空间分配按照数据库设备名的字母顺序，在 master..sysdevices 中指定的缺省数据库设备中进行。

要查看缺省数据库设备的逻辑名，需输入如下内容：

```
select name
    from sysdevices
    where status & 1 = 1
    order by name
```

sp_helpdevice 也会在数据库设备说明中显示“default disk”。

估计所需空间

由于存储空间在指派出去之后很难回收，因此您所做的大小分配决定非常重要。您始终可以增加空间，但不能释放已指派给数据库的空间，除非删除个数据库。

可以通过使用 sp_estspace 或计算数值估计数据库的表和索引的大小。请参见《性能和调优系列：物理数据库调优》中的第 4 章“确定表和索引的大小”。

把事务日志存放在单独的设备上

可以使用 create database 命令的 log on 子句把事务日志（syslogs 表）存放在单独的数据库设备上。除非是创建很小且不重要的数据库，否则应始终把日志存放在单独的数据库设备上。把日志存放在独立的数据库设备上，可以：

- 允许使用 dump transaction 命令而不是 dump database 命令，从而节省时间和磁带。
- 允许建立固定长度的日志，防止它与其它数据库活动争用空间。
- 在日志段上创建缺省的可用空间阈值监控，并允许在数据库的日志和数据部分创建额外的可用空间监控。请参见第 17 章“使用阈值管理可用空间”。
- 提高性能。
- 确保硬盘崩溃后的完全恢复。dump transaction 的特定参数允许转储事务日志，即使数据设备是在损坏的磁盘上。

要为事务日志指定大小和设备，可使用 create database 的 *log on* device = size 子句。大小使用以下单位指示符：“k”或“K”（千字节）、“m”或“M”（兆字节）、“g”或“G”（千兆字节）和“t”或“T”（太字节）。例如以下语句将创建 newdb 数据库，并且在 mydata 上分配 8MB 的空间，在 newdata 上分配 4MB 的空间，在第三个数据库设备上放置了所占空间为 3MB 的事务日志 tranlog：

```
create database newdb
on mydata = "8M", newdata = "4M"
log on tranlog = "3M"
```

估计事务日志大小

事务日志的大小由下列因素决定：

- 相关数据库中更新活动的数量
- 事务日志转储的频繁程度

无论手工进行事务日志转储或使用阈值存储过程自动进行此任务都是这样。作为一般规则，为日志分配的空间是为数据库分配的空间的 10% 到 25%。

插入、删除与更新将增加日志的大小。dump transaction 可通过将提交的事务写入磁盘中并将其从日志中删除来减小日志大小。由于 update 语句需要记录行的“前”映像和“后”映像，因此一次更新许多行的应用程序所占的日志空间至少是要同时更新的行的两倍或最大表的两倍。或者可以把更新操作分为小组进行**批处理**，在批处理之间执行事务转储。

对于有许多插入与更新活动的数据库，日志增长得很快。应定期检查日志以确定日志所需的大小。这也有助于为日志选择阈值以及安排事务日志转储时间。若要检查数据库事务日志所占用的空间，首先要使用此数据库，然后输入：

```
dbcc checktable(syslogs)
```

dbcc 报告日志所用的数据页数量。如果日志是在一个单独的设备上，则 dbcc checktable 还报告已用空间量和可用空间量。下面是一个 2MB 的日志的输出样本：

```
Checking syslogs
The total number of data pages in this table is 199.
*** NOTICE:Space used on the log segment is 0.39 Mbytes, 19.43%.
*** NOTICE:Space free on the log segment is 1.61 Mbytes, 80.57%.
Table has 1661 data rows.
```

若要检查日志的增长，请输入：

```
select count(*) from syslogs
```

定期重复任一命令查看日志的增长速度。

缺省日志大小和设备

如果省略 *log* on 子句中的 size 参数，则 Adaptive Server 会使用允许的最小存储空间量。如果完全省略 log on 子句，则 Adaptive Server 会将事务日志放在与数据表相同的数据库设备上。

将事务日志移动到其它设备

如果没有对 create database 使用 log on 子句，请遵循本节中的说明将事务日志移动到其它数据库设备。

sp_logdevice 会将指定设备上存在的现有数据库的某些部分标记为保留用于事务日志；它不会移动现有数据。如果数据库在该设备上已有数据，Adaptive Server 不会此数据解释为不在其正确的段上。但是，由于 dbcc 会将这种情况报告为错误，因此日志的现有部分不会移动到指定设备；在日志扩展到新设备上且使用 dump transaction 清除日志的该部分以前，当前日志数据会保留在其原来的位置。而且，sp_logdevice 不会为数据库分配新的空间或初始化设备。而是为已属于指定数据库的日志保留指定设备的这些部分。

sp_logdevice 的语法为：

sp_logdevice *database_name*, *devname*

您指定的数据库设备必须使用 disk init 进行初始化，且必须使用 create 或 alter database 命令将其分配给数据库。

要将全部事务日志移动到其它设备，需执行以下步骤：

1 执行 sp_logdevice，命名新的数据库设备。

2 执行足够的事务以填充当前使用的页。更新所需的空间数量取决于逻辑页的大小。可以在开始更新之前或之后执行 dbcc checktable(syslogs) 以确定何时使用新页。

3 等待所有的当前活动事务完成。可能要使用 sp_dboption 将数据库置于单用户模式。

4 运行 dump transaction，删除它写到磁盘上的所有日志页。只要旧设备上那部分日志中没有活动事务，则其所有的页都将被删除。请参见第 12 章“制定备份和恢复计划”。

5 运行 sp_helplog 以确保全部日志都在新的日志设备上。

注释 移动事务日志时，不再由事务日志使用的空间可以由数据使用。但是，不能通过移动事务日志来减少分配到设备的空间量。

第 12 章“制定备份和恢复计划”中对事务日志进行了详细讨论。

缩减日志空间

alter database 命令包括 log off 参数，它用于删除数据库日志的不必要部分，能让您缩减日志空间并释放存储空间而无需重新创建数据库。

语法为：

```
alter database database_name [log off database_device
    [= size | [from logical_page_number] [to logical_page_number]]
    [, database_device
    [= size | [from logical_page_number] [to logical_page_number]]
```

该参数可能在运行数据库操作的完全记录选项（如 select into、alter table 或 reorg rebuild）后，数据库产生不再需要的额外分配空间时特别有用。请参见《参考手册：命令》中的 dump transaction 命令。

缩减日志空间时使用 *dump* 和 *load database*

执行 dump 和 load database 时：

- 要装载的数据库拥有的物理空间至少要与转储时拥有的物理空间相等。
- 要装载的数据库中的物理空间根据转储数据库中的物理片段指定。这意味着要装载的数据库中有“空洞”（先前执行 alter database log off 命令后没有物理存储与之关联的分配单位），装载后无需保留空洞。
- 要装载的数据库中余留空间的指定方式与没有空洞的情况下执行 dump 和 load database 时相同。
- 您可以通过运行 load database with headeronly 命令来确定转储数据库中的物理空间量、它是否有空洞以及有关这些空洞的大小和位置的信息。

❖ 转储和装载数据库之前缩减日志

以下方案缩减即将转储的数据库的日志。使用转储装载之前，可使用 load database with headeronly 命令和 sp_helpdb 系统过程确定目标数据库的大小。完整的命令序列位于第 134 页的“使用 dump 和 load database 的序列示例”中。

1　根据需要使用任意数目的日志设备创建数据库。该示例创建两个日志设备。

2　（可选）运行 select *，确认数据库创建情况并显示组成数据库的设备片段。

3 使用 alter database log off 命令，在不中断数据库转储序列的情况下从数据库中删除不需要的日志部分。如果数据库的转储序列已中断，则 alter database log off 会自动从数据库结尾删除所有缩减的空间。如果删除的空间不在数据库结尾处，则会形成空洞。

   在该示例中，位于数据库中间位置的缩减空间已形成空洞。

4 sysusages 的输出显示了空洞的位置和大小（在该示例中，segmap = 0，location = 4）。sp_helpdb 的输出摘要显示除去空洞之外的数据库大小（在该示例中为 9MB）和数据库中空洞的总大小（在该示例中，仅日志不可用空间为 3072KB，即 3MB）：

5 要装载的数据库的总大小为 12MB，但其中，数据库中实际的物理空间为 9MB，因为存在 3MB 的空洞。dump database with headeronly 命令验证数据库是否包含 12MB 的逻辑页和 9MB 的物理页。要装载此数据库，必须创建大小至少为 9MB 的新数据库。

6 装载数据库并使它联机。

   在新装载数据库的 sysusages 行中，已重新安排 9MB 的物理空间，以与转储数据库的物理空间相匹配，这样数据库现在的总大小为 12MB，其中只有 9MB 的物理页，另外还有 3MB 的空。

使用 dump 和 load database 的序列示例

下面的示例显示按第 133 页的“转储和装载数据库之前缩减日志”中所述，使用 dump 和 load database 时要执行的完整序列：

```
1> create database sales_db on sales_db_dev=3 log on sales_db_log1=3,
    sales_db_log2=3, sales_db_log1=3, sales_db_log2=3
2> go
00:00:00000:00015:2011/01/21 09:38:28.29 server  Timestamp for database
'sales_db' is (0x0000,
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_dev'
(1536 logical pages requested).
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log1'
(1536 logical pages requested).
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log2'
(1536 logical pages requested).
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log1'
(1536 logical pages requested).
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log2'
(1536 logical pages requested).
Warning:The database 'sales_db' is using an unsafe virtual device 'sales_db_dev'.
The recovery of this database can not be guaranteed.
Database 'sales_db' is now online.

1> select * from sysusages where dbid=4
2> go
```

```
dbid segmap lstart size vstart location unreservedpgs  crdate            vdevno
---- ------ ------ ---- ------ -------- -------------  ------             ------
    4      3      0 1536      0        0           670  Jan 21 2011 9:38AM     1
    4      4   1536 1536      0        0          1530  Jan 21 2011 9:38AM     2
    4      4   3072 1536      0        0          1530  Jan 21 2011 9:38AM     3
    4      4   4608 1536   1536        0          1530  Jan 21 2011 9:38AM     2
    4      4   6144 1536   1536        0          1530  Jan 21 2011 9:38AM     3

(5 rows affected)
1> alter database sales_db log off sales_db_log2
2> select * from sysusages where dbid=4
3> go
dbid segmap lstart size vstart location unreservedpgs  crdate            vdevno
---- ------ ------ ---- ------ -------- -------------  ------             ------
    4      3      0 1536      0        0           670  Jan 21 2011 9:38AM     1
    4      4   1536 1536      0        0          1530  Jan 21 2011 9:38AM     2
    4      0   3072 1536   3072        4          1530  Jan 21 2011 9:38AM    -4
    4      4   4608 1536   1536        0          1530  Jan 21 2011 9:38AM     2

(4 rows affected)
1> sp_helpdb sales_db
2> go
 name     db_size       owner dbid  created      durability status
 -------- ------------- ----- ----  ------------ ---------- --------------
sales_db  9.0 MB        sa       4  Jan 21, 2011 full       no options set

(1 row affected)
device_fragments   size    usage      created                free kbytes
------------------ ------  ---------- ------------------     --------------
sales_db_dev       3.0 MB  data only  Jan 21 2011 9:38AM              1340
sales_db_log1      3.0 MB  log only   Jan 21 2011 9:38AM     not applicable
sales_db_log1      3.0 MB  log only   Jan 21 2011 9:38AM     not applicable

------------------------------------------------------------
log only free kbytes = 6082, log only unavailable kbytes = 3072
(return status = 0)
1> dump database sales_db to "c:/temp/sales_db.dmp"
2> go
Backup Server session id is:45. 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 c:/temp/sales_db.dmp.
Backup Server: 6.28.1.1: Dumpfile name 'sales_db11021087C7  ' section number 1
mounted on disk file 'c:/temp/sales_db.dmp'
Backup Server: 4.188.1.1: Database sales_db:848 kilobytes (67%) DUMPED.
Backup Server: 4.188.1.1: Database sales_db:862 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_db:870 kilobytes (100%) DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db).
```

```
1> load database sales_db from "c:/temp/sales_db.dmp" with headeronly
2> go
```

```
Backup Server session id is:48. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'sales_db11021087C7  ' section number 1 mounted
on disk file 'c:/temp/sales_db.dmp'
This is a database dump of database ID 4, name 'sales_db', from Jan 21 2011 9:39AM.ASE
version:lite_642236-1/Adaptive Server Enterprise/15.7/EBF 18567 SMP
Drop#2/B/X64/Windows Server/aseasap/ENG/.Backup Server version: Backup
Server/15.7/B/X64/Windows Server/aseasap/ENG/64-bit/DEBUG/Thu Jan 20 11:12:51 2011.
Database page size is 2048.
Database contains 6144 pages; checkpoint RID=(Rid pageid = 0x604; row num = 0x12); next
object ID=560001995; sort order ID=50, status=0; charset ID=2.
Database log version=7; database upgrade version=35; database durability=UNDEFINED.
segmap:0x00000003 lstart=0 vstart=[vpgdevno=1 vpvpn=0] lsize=1536 unrsvd=670
segmap:0x00000004 lstart=1536 vstart=[vpgdevno=2 vpvpn=0] lsize=1536 unrsvd=1530
Unavailable disk fragment: .lstart=3072 lsize=1536
segmap:0x00000004 lstart=4608 vstart=[vpgdevno=2 vpvpn=1536] lsize=1536 unrsvd=1530
The database contains 6144 logical pages (12 MB) and 4608 physical pages (9 MB).
```

```
1> create database sales_db2 on sales_db_dev=3 log on sales_db_log1=6
2> go
```

```
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_dev'
(1536 logical pages requested).
CREATE DATABASE:allocating 3072 logical pages (6.0 megabytes) on disk 'sales_db_log1'
(3072 logical pages requested).
Warning:The database 'sales_db2' is using an unsafe virtual device 'sales_db_dev'.
The recovery of this database can not be guaranteed.
Database 'sales_db2' is now online.
```

```
1> select * from sysusages where dbid=db_id("sales_db2")
2> go
```

```
dbid segmap lstart size vstart location unreservedpgs crdate             vdevno
---- ------ ------ ---- ------ -------- ------------ ------              ------
   5      3      0 1536   1536        0          670 Jan 26 2011 1:22AM      1
   5      4   1536 3072   3072        0         3060 Jan 26 2011 1:22AM      2
```

```
1> load database sales_db2 from "/tmp/sales_db.dmp"
2> go
```

```
Backup Server session id is:10. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'sales_db1102602564  ' section number 1 mounted
on disk file '/tmp/sales_db.dmp'
Backup Server: 4.188.1.1: Database sales_db2:6148 kilobytes (33%) LOADED.
Backup Server: 4.188.1.1: Database sales_db2:9222 kilobytes (50%) LOADED.
Backup Server: 4.188.1.1: Database sales_db2:9230 kilobytes (100%) LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database sales_db2).
```

```
Started estimating recovery log boundaries for database 'sales_db2'.
Database 'sales_db2', checkpoint=(1544, 22), first=(1544, 22), last=(1544, 22).
Completed estimating recovery log boundaries for database 'sales_db2'.
Started ANALYSIS pass for database 'sales_db2'.
Completed ANALYSIS pass for database 'sales_db2'.
00:00:00000:00011:2011/01/26 05:12:15.86 server  Log contains all committed transactions
until 2011/01/26 01:55:15.71 for database sales_db2.
Started REDO pass for database 'sales_db2'.The total number of log records to process
is 1.
Completed REDO pass for database 'sales_db2'.
00:00:00000:00011:2011/01/26 05:12:15.88 server  Timestamp for database 'sales_db2' is
(0x0000, 0x00001612).
Use the ONLINE DATABASE command to bring this database online; ASE will not bring it
online automatically.

1> online database sales_db2
2> go
Started estimating recovery log boundaries for database 'sales_db2'.
Database 'sales_db2', checkpoint=(1544, 22), first=(1544, 22), last=(1544, 22).
Completed estimating recovery log boundaries for database 'sales_db2'.
Started ANALYSIS pass for database 'sales_db2'.
Completed ANALYSIS pass for database 'sales_db2'.
00:00:00000:00011:2011/01/26 05:12:22.49 server  Log contains all committed transactions
until 2011/01/26 01:55:15.71 for database sales_db2.
Recovery of database 'sales_db2' will undo incomplete nested top actions.
Database 'sales_db2' is now online.

1> select * from sysusages where dbid=db_id("sales_db2")
2> go
dbid segmap lstart size vstart location unreservedpgs crdate             vdevno
---- ------ ------ ---- ------ -------- ------------- ------             ------
   5      3      0 1536   1536        0           670 Jan 26 2011 5:12AM      1
   5      4   1536 1536   3072        0          1530 Jan 26 2011 5:12AM      2
   5      0   3072 1536   3072        4          1530 Jan 26 2011 5:12AM     -5
   5      4   4608 1536   4608        0          1530 Jan 26 2011 5:12AM      2
```

缩减日志空间时使用 *dump* 和 *load transaction*

如果在某一转储序列中首先转储数据库，然后再定期执行事务日志转储，则在该转储序列期间日志的大小可能会发生变化。例如，如果增加日志段来纳由完全记录的 select into 执行的日志记录量，然后在完成该命令后缩减日志使其恢复为以前的大小，则更是如此。装载此类转储序列时请遵循以下准则：

- 在转储序列期间采用最大大小创建要从转储装载的数据库。可通过对要装载的上一事务转储执行 dump tran with headeronly 来确定该值。

- 仅在事务日志序列完成并且使数据库联机后，才将日志缩减到所需的大小。

❖ **装载其日志已得到缩减的数据库的转储序列**

显示这些命令和输出的示例按第 139 页的“使用 dump 和 load transaction 的序列示例”中所述的编号步骤执行。

1 创建数据库。该示例创建 sales_db。

2 使用 sp_dboption 系统过程的 'full logging for all' 数据库选项打开数据库的完全日志记录模式。

3 转储数据库。

4 使用 alter database log on 增加日志段的大小，准备执行完全记录的 select into 命令。

5 运行完全记录的 select into 命令，该命令使用增加的日志段。

6 转储事务日志以截断日志，准备缩减日志段。

7 使用 alter database log off 缩减数据库日志，以删除前面的步骤中增加的日志空间。

8 使用缩减的日志段转储数据库的事务日志。

9 装载转储序列之前，首先根据装载序列中的上一文件获取数据库的逻辑大小。在该示例中，大小为 16MB。

注释 根据装载序列中的上一转储确定的数据库逻辑大小保证至少与整个转储序列中数据库的最大物理大小相等。这样可以方便地确定，要装载序列中的所转储，目标数据库应采用的大小。

使用 load transaction with headeronly 命令确定，为了容纳序列中的所有转储，目标数据库必须采用的大小。

10 根据需要使用任意数目的日志设备创建新数据库。该示例使用两个日志设备创建了 sales_db1 数据库，它的大小为 16MB。

11 装载此数据库。

12 从第一个和第二个事务日志转储将事务日志装载到数据库中。

13 使数据库联机。

14 通过从其日志段中删除空间来减小数据库的大小。在该示例中，日志段的大小减小了 10MB。

15 运行 select * from sysusages 以确认从数据库结尾删除了空间。删除的空间在数据库中形成空洞。

16　使用 dump database 的 with shrink_log 选项删除数据库结尾的空洞。

17　再次运行 select * from sysusages，以确认 Adaptive Server 成功从数据库结尾删除了空洞。

使用 dump 和 load transaction 的序列示例

下面的示例显示按第 138 页的“装载其日志已得到缩减的数据库的转储序列”中所述，使用 dump 和 load transaction 时要执行的完整序列：

```
1> create database sales_db on sales_db_dev=3 log on sales_db_log1=3
2> go
00:00:00000:00018:2011/05/05 12:45:06.36 server  Timestamp for database 'sales_db' is
(0x0000, 0x00002aa9).
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_dev'
(1536 logical pages requested).CREATE DATABASE:allocating 1536 logical pages (3.0
megabytes) on disk 'sales_db_log1' (1536 logical pages requested).
Warning:The database 'sales_db' is using an unsafe virtual device 'sales_db_dev'.The
recovery of this database can not be guaranteed.
Database 'sales_db' is now online.

1> sp_dboption sales_db,'full logging for all',true
2> go
Database option 'full logging for all' turned ON for database 'sales_db'.
Running CHECKPOINT on database 'sales_db' for option 'full logging for all' to take
effect.
(return status = 0)

1> use master
2> go
1> dump database sales_db to "/tmp/sales_db.dmp"
2> go
Backup Server session id is:120. 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_db.dmp.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db11137014BC' section number 1 mounted on
disk file '/tmp/sales_db.dmp'
Backup Server: 4.188.1.1: Database sales_db:852 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: 4.188.1.1: Database sales_db:856 kilobytes (100%) DUMPED.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.188.1.1: Database sales_db:860 kilobytes (100%) DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db).

1> alter database sales_db log on sales_db_log2=10
2> go
Extending database by 5120 pages (10.0 megabytes) on disk sales_db_log2
```

```
Warning:The database 'sales_db' is using an unsafe virtual device 'sales_db_dev'.The
recovery of this database can not be guaranteed.
Warning:Using ALTER DATABASE to extend the log segment will cause user thresholds on the
log segment within 128 pages of the last chance threshold to be disabled.

use sales_db
 go
select * into bigtab2 from bigtab
go
(20000 rows affected)

1> dump tran sales_db to "/tmp/sales_db.trn1"
2> go
Backup Server session id is:9. 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_db.trn1.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D37' section number 1 mounted on
disk file '/tmp/sales_db.trn1'
Backup Server: 4.58.1.1: Database sales_db:250 kilobytes DUMPED.
Backup Server: 4.58.1.1: Database sales_db:254 kilobytes DUMPED.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database sales_db:258 kilobytes DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db).

1> use master
2> go
1> alter database sales_db log off sales_db_log2
2> go
Removing 5120 pages (10.0 MB) from disk 'sales_db_log2' in database 'sales_db'.

1> dump tran sales_db to "/tmp/sales_db.trn2"
2> go
Backup Server session id is:11. 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_db.trn2.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D87' section number 1 mounted on
disk file '/tmp/sales_db.trn2'
Backup Server: 4.58.1.1: Database sales_db:6 kilobytes DUMPED.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database sales_db:10 kilobytes DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db).

1> load tran sales_db from "/tmp/sales_db.trn2" with headeronly
2> go
Backup Server session id is:13. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D87' section number 1 mounted on
disk file '/tmp/sales_db.trn2'
This is a log dump of database ID 5, name 'sales_db', from May 19 2011  4:22AM.
ASE version: lite_670673-1/Adaptive Server Enterprise/15.7.0/EBF 19186 SMP GA
FS3b/B/x86_64/Enterprise Linux/asea.Backup Server version: Backup
Server/15.7/EBF 19186 Drop#3B Prelim/B/Linux AMD Opteron/Enterprise
```

```
Linux/aseasap/3556/64-bi.Database page size is 2048.
Log begins on page 1986; checkpoint RID=Rid pageid = 0x7c2; row num = 0x14;
previous BEGIN XACT RID=(Rid pageid = 0x7c2; row num = 0x4); sequence dates:
(old=May 19 2011  4:21:11:356AM, new=May 19 2011  4:22:31:043AM); truncation
page=1986; 123 pages deallocated; requires database with 8192 pages.
Database log version=7; database upgrade version=35; database
durability=UNDEFINED.
segmap:0x00000003 lstart=0 vstart=[vpgdevno=1 vpvpn=0] lsize=1536 unrsvd=192
segmap:0x00000004 lstart=1536 vstart=[vpgdevno=2 vpvpn=0] lsize=1536
unrsvd=1530
Unavailable disk fragment: lstart=3072 lsize=5120
The database contains 8192 logical pages (16 MB) and 3072 physical pages (6MB).
```

```
1> create database sales_db1 on sales_db_dev=3 log on sales_db_log1=3,
    sales_db_log2=10
2> go
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_dev'
(1536 logical pages requested).
CREATE DATABASE:allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log1'
(1536 logical pages requested).
CREATE DATABASE:allocating 5120 logical pages (10.0 megabytes) on disk 'sales_db_log2'
(5120 logical pages requested).
Warning:The database 'sales_db1' is using an unsafe virtual device 'sales_db_dev'.The
recovery of this database can not be guaranteed.
Database 'sales_db1' is now online.
```

```
1> load database sales_db1 from "/tmp/sales_db.dmp"
2> go
Backup Backup Server session id is:15. Use this value when executing the 'sp_volchanged'
system stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db111390340B' section number 1 mounted on
disk file '/tmp/sales_db.dmp'
Backup Server: 4.188.1.1: Database sales_db1:6148 kilobytes (37%) LOADED.
Backup Server: 4.188.1.1: Database sales_db1:6160 kilobytes (100%) LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database sales_db1).
All dumped pages have been loaded. ASE is now clearing pages above page 3072, which were
not present in the database just loaded.
ASE has finished clearing database pages.
Started estimating recovery log boundaries for database 'sales_db1'.
Database 'sales_db1', checkpoint=(1863, 13), first=(1863, 13), last=(1865, 7).
Completed estimating recovery log boundaries for database 'sales_db1'.
Started ANALYSIS pass for database 'sales_db1'.
Completed ANALYSIS pass for database 'sales_db1'.
Started REDO pass for database 'sales_db1'.The total number of log records to process
is 22.
Redo pass of recovery has processed 2 committed and 0 aborted transactions.
Completed REDO pass for database 'sales_db1'.
Use the ONLINE DATABASE command to bring this database online; ASE will not
bring it online automatically.
```

```
1> load tran sales_db1 from "/tmp/sales_db.trn1"
2> go
```

```
Backup Server session id is:17. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D37' section number 1 mounted on
disk file '/tmp/sales_db.trn1'
Backup Server: 4.58.1.1: Database sales_db1:250 kilobytes LOADED.
Backup Server: 4.58.1.1: Database sales_db1:258 kilobytes LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database sales_db1).
Started estimating recovery log boundaries for database 'sales_db1'.
Database 'sales_db1', checkpoint=(1863, 13), first=(1863, 13), last=(1986, 3).
Completed estimating recovery log boundaries for database 'sales_db1'.
Started ANALYSIS pass for database 'sales_db1'.
Completed ANALYSIS pass for database 'sales_db1'.
Started REDO pass for database 'sales_db1'.The total number of log records to process
is 365.
Redo pass of recovery has processed 8 committed and 0 aborted transactions.
Completed REDO pass for database 'sales_db1'.
Use the ONLINE DATABASE command to bring this database online; ASE will not bring it
online automatically.

1> load tran sales_db1 from "/tmp/sales_db.trn2"
2> go
Backup Server session id is:19. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D87' section number 1 mounted on
disk file '/tmp/sales_db.trn2'
Backup Server: 4.58.1.1: Database sales_db1:10 kilobytes LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database sales_db1).
Started estimating recovery log boundaries for database 'sales_db1'.
Database 'sales_db1', checkpoint=(1986, 3), first=(1986, 3), last=(1986, 20).
Completed estimating recovery log boundaries for database 'sales_db1'.
Started ANALYSIS pass for database 'sales_db1'.
Completed ANALYSIS pass for database 'sales_db1'.
Started REDO pass for database 'sales_db1'.The total number of log records to process
is 16.
Redo pass of recovery has processed 2 committed and 0 aborted transactions.
Completed REDO pass for database 'sales_db1'.
Use the ONLINE DATABASE command to bring this database online; ASE will not bring it
online automatically.

1> online database sales_db1
2> go
Started estimating recovery log boundaries for database 'sales_db1'.
Database 'sales_db1', checkpoint=(1986, 20), first=(1986, 19), last=(1986, 20).
Completed estimating recovery log boundaries for database 'sales_db1'.
Started ANALYSIS pass for database 'sales_db1'.
Completed ANALYSIS pass for database 'sales_db1'.
Recovery of database 'sales_db1' will undo incomplete nested top actions.
Started UNDO pass for database 'sales_db1'.The total number of log records to process
is 2.
Undo pass of recovery has processed 1 incomplete transactions.
Completed UNDO pass for database 'sales_db1'.
```

```
Database 'sales_db1' is now online.

1> alter database sales_db1 log off sales_db_log2
2> go
Removing 5120 pages (10.0 MB) from disk 'sales_db_log2' in database 'sales_db1'.

1> select * from sysusages where dbid=db_id("sales_db1")
2> go
 dbid  segmap  lstart   size  vstart location unreservedpgs  crdate              vdevno
 ----  ------  -------  ----  ------ -------- -------------- ------------------ ------
    6       3        0  1536    1536        0            192  May 19 2011 4:25AM      1
    6       4     1536  1536    1536        0           1536  May 19 2011 4:25AM      2
    6       0     3072  5120    3072        4           5100  May 19 2011 4:25AM     -6

(3 rows affected)

1> dump database sales_db1 to "/tmp/sales_db1.dmp" with shrink_log
2> go
Backup Server session id is:22. 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_db11113903EC3' section number 1 mounted
on disk file '/tmp/sales_db1.dmp'
Backup Server: 4.188.1.1: Database sales_db1:3100 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_db1:3108 kilobytes (100%) DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db1).

1> select * from sysusages where dbid=db_id("sales_db1")
2> go
 dbid  segmap  lstart   size  vstart location unreservedpgs  crdate              vdevno
 ----  ------  -------  ----  ------ -------- -------------- ------------------ ------
    6       3        0  1536    1536        0            192  May 19 2011  4:25AM      3
    6       4     1536  1536    1536        0           1530  May 19 2011  4:25AM      4

(2 rows affected)
```

使用 *for load* 选项进行数据库恢复

创建新数据库时，Adaptive Server 通常会清除数据库设备上所有未使用的页。完成清除页需要几秒钟或几分钟时间，具体取决于数据库的大小和系统的速度。

如果要使用数据库从数据库转储装载（用于介质故障后的恢复，或者用于将数据库从一台计算机移动到另一台），可使用 for load 选项。使用 for load 运行跳过页清除步骤的 create database 精简形式，并创建仅在装载转储时使用的目标数据库。

如果使用 for load 创建数据库，则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

装载数据库转储时，用于数据库的新数据库设备分配必须与转储数据库的使用分配相匹配。有关复制空间分配的讨论，请参见第 13 章“备份和恢复用户数据库”。

把数据库转储装载到新数据库后，您可以使用的命令不再受到限制。

使用 *create database* 的 *with override* 选项

with overide 选项允许空间有限的计算机在与其数据分开的设备段上维护日志。Sybase 建议不要这样做，但它也许是存储空间有限的计算机的唯一可用选项，特别在硬盘出现故障后必须使数据库回到联机状态时。

您仍可以转储事务日志，但是如果遇到介质故障，则不能访问当前日志，因为它与数据在同一设备上。只能恢复到最后一次事务日志转储；此时到故时间之间的所有事务都会丢失。

下例中，日志和数据位于同一逻辑设备的不同段上。

```
create database littledb
    on diskdev1 = "4M"
    log on diskdev1 = "1M"
    with override
```

可创建的最小数据库大小是 model 的大小。

更改数据库所有权

系统管理员可能需要创建用户数据库并在完成一些初始工作之后将其所有权给另一用户。sp_changedbowner 更改数据库的所有权，并且只能由系统管理员在将要更改其所有权的数据库中执行。语法为：

sp_changedbowner *loginame* [, true]

以下示例使用户“albert”成为当前用户的所有者：

```
sp_changedbowner albert
```

新的所有者必须已在 Adaptive Server 中有一个登录名，但不能是该数据库的用户或在该数据库中有别名。在更改数据库所有权之前，可能必须要使用 sp_dropuser 或 sp_dropalias（请参见《参考手册：过程》）。有关更改所有权的详细信息，请参见《系统管理指南，卷 1》中的第 13 章“Adaptive Server 中的安全性管理快速入门”。

注释　不能更改 master 数据库的所有权；该数据库始终由登录名“sa”所有。

变更数据库

当数据库或事务日志增长到填满所有使用 create database 分配的空间时，可以使用 alter database 增加存储空间。可以为数据库对象或事务日志增加空间，或同时为两者增加。也可以使用 alter database 准备从备份装载数据库。

alter database 的权限缺省授予数据库所有者，并自动随数据库所有权一起转交。alter database 权限不能使用 grant 或 revoke 来更改。请参见第 145 页的“更改数据库所有权”。

注释　alter database for proxy update 删除代理数据库中的所有代理表。

alter database 语法

可以使用 alter database 扩展数据库以及指定增加存储空间的位置。

如果使用最简单的形式，alter database 将从缺省数据库设备增加配置的缺省空间量。如果数据库将日志和数据分开，则增加的空间只由数据使用。使用 sp_helpdevice 可以查找缺省列表中的数据库设备名。

请参见《参考手册：命令》。

若要从缺省数据库设备给 newpubs 数据库增加空间，请输入：

```
alter database newpubs
```

on 和 log on 子句的用法与 create database 中相应子句的用法类似。可以指定缺省数据库设备或某些其它数据库设备上的空间，而且可以指定多个数据库设备。如果使用 alter database 扩展 master 数据库，则可以只在主设备上进行扩展。可指定的最小增量为 1MB 或一个分配单位，取二者中较大的。

如果 Adaptive Server 不能分配所请求的大小，则在每一数据库设备上将分配尽可能多的空间，每一设备最少分配 256 个逻辑页。alter database 完成后，将显示消息说明已分配多少空间；例如：

```
Extending database by 1536 pages on disk pubsdata1
```

检查所有消息以确保已增加了所请求的空间量。

以下命令在 pubsdata1 上为 newpubs 的已分配空间增加 2MB 空间，在新设备 pubsdata2 上增加 3MB，在 tranlog 上为日志增加 1MB：

```
alter database newpubs
on pubsdata1 = "2M", pubsdata2 =" 3M"
log on tranlog
```

注释 每次发出 alter database 命令时，都要转储 master 数据库。

使用 with override 在设备上创建包含日志空间的设备段，该设备已包含位于已用于该日志的一个设备上的数据或数据段。只有在没有其它存储选项并且不要求最新最快的恢复时，才使用本选项。

仅在使用 create database for load 之后才使用 for load 命令来重新创建从转储装载到新数据库的数据库的空间分配。有关将转储装载到新数据库时复制空间分配的讨论，请参见第 13 章“备份和恢复用户数据库”。

使用 *drop database* 命令

使用 drop database 可以从 Adaptive Server 删除数据库，从而删除数据库和其中的所有对象，以及：

- 释放为数据库分配的存储空间
- 从 master 数据库中的系统表删除对数据库的引用

只有数据库所有者可以删除数据库。必须在 master 数据库中删除某个数据库。不能删除一个用户正在读取或写入的打开的数据库。

语法为：

drop database *database_name* [, *database_name*]...

可以使用一个语句删除多个数据库，例如：

```
drop database newpubs, newdb
```

在删除数据库本身之前，必须删除该数据库设备上的所有数据库。删除设备的命令为 sp_dropdevice。

删除数据库之后，需转储 master 数据库以确保在 master 损坏之后可以恢复。

管理空间分配的系统表

为了在数据库设备上创建数据库并为其分配一定的空间量，Adaptive Server 首先在 sysdatabases 中为新数据库添加一个条目。然后，它检查 master..sysdevices 以确保 create database 中指定的设备名实际存在而且是数据库设备。如果未指定数据库设备或如果使用了 default 选项，Adaptive Server 将在 master..sysdevices 和 master..sysusages 中查找所有设备上可用于缺省存储的可用空间。它按照设备名的字母顺序执行此检查。

Adaptive Server 从其收集指定存储量的存储空间不必是连续的。数据库存储空间甚至可以从多个数据库设备抽取。数据库被作为逻辑单元对待，即使它存储在多个数据库设备上。

数据库的每一存储必须至少是一个分配单元。每个分配单元的第一页是分配页。它并不象其它页那样包含数据库行，但却包含一个显示剩余页如何使的数组。

sysusages 表

数据库存储信息列于 master..sysusages 中。master..sysusages 中的每一行表示一个指派给数据库的空间分配。这样，每次 create database 或 alter database 指派一个磁盘空间段给数据库时，每个数据库在 sysusages 中有一行。

安装 Adaptive Server 时，sysusages 包含有关这些数据库的行：

- master，dbid 为 1
- 临时数据库 tempdb，dbid 为 2
- model，dbid 为 3
- sybsystemdb，dbid 为 31513
- sybsystemprocs，dbid 为 31514

如果从早期版本升级 Adaptive Server，数据库 sybsystemdb 和 sybsystemprocs 可能具有不同的数据库 ID。

如果安装了审计功能，sybsecurity 数据库将为 dbid 5。

创建新数据库或扩大现有数据库时，新的行增加到 sysusages 中以表示新的数据库分配。

下面是 sysusages 在包含五个系统数据库和一个用户数据库的 Adaptive Server 中的显示情况。用户数据库是使用 `log on` 选项创建的，并使用 alter database 扩展了一次。它的数据库 ID (dbid) 为 4：

```
select dbid, segmap, lstart, size, vdevno, vstart
from sysusages
order by 1
```

```
dbid        segmap      lstart      size        vdevno      vstart
----        ---------   ----------  --------    --------    -----------
1           7           0           6656        0           4
2           7           0           2048        0           8196
3           7           0           1536        0           6660
4           3           0           5120        2           0
4           4           5120        2560        3           0
4           3           7680        5120        2           5120
31513       7           0           1536        0           10244
31514       7           0            63488       1            0
```

在此示例中，lstart 和 size 列说明了逻辑页，其大小可在 2KB - 16KB 之间变化。vstart 列说明了虚拟页（其大小始终是 2KB）。下面的全局变量显示页大小信息：

- *@@maxpagesize* — 逻辑页大小

- *@@pagesize* — 虚拟页大小

以下示例将数据库 ID 与其名称相匹配，显示 size 列所表示的兆字节数，显示列表中每个 vdevno 的逻辑设备名，并计算分配给每个数据库的总兆字节数。本例的输出仅显示 dbid 4 的结果，且结果已被重新设定了格式以提高可读性：

```
select  dbid, db_name(dbid) as 'database name', lstart,
     size / (power(2,20)/@@maxpagesize) as 'MB',
     d.name
from sysusages u, sysdevices d
where u.vdevno = d.vdevno
and d.status & 2 = 2
order by 1
compute sum(size / (power(2,20)/@@maxpagesize)) by dbid
dbid     database name     lstart      MB          device name
------    --------------    --------    --------    --------------------
4         test              0           10          datadev
4         test              5120        5           logdev
4         test              7680        10          datadev

Compute Result:
-----------
25
```

以下示例说明了在添加段时，sysusages 表中的 segmap 值的变化情况。在该示例中，服务器最初包括缺省数据库、一个名为 testdb 的用户数据库（仅数据数据库）和 testlog 设备上的一个日志，如以下 sysusages 表的输出所示：

```
select dbid, segmap from master..sysusages where dbid = 6
 dbid    segmap
------  -----------
  6          3
  6           4
```

如果您向测试数据库添加用户段 newseg，并在 newseg 上创建表 abcd，然后再次选择 sysusages 中的段信息：

```
sp_addsegment newseg, testdb, datadev

create table abcd ( int c1 ) on newseg

select dbid, segmap from sysusages
where dbid=6
dbid     segmap
------   -----------
6           11
6            4
```

注意，用户数据库的段映射从值 3 更改为值 11，这表明用户数据库的段映射会在重新配置数据库时发生变化。

若要确定段的状态，请运行：

```
sp_helpsegment
 segment     name             status
-------    ---------------- ----------
0           system              0
1           default             1
2           logsegment          0
3           newseg              0
```

newseg 段不是缺省池的一部分。

如果向 testdb 数据库再添加一个段 newseg1，并再次选择 sysusages 中的段信息，则 newseg 的段映射即从 11 更改为 27：

```
sp_addsegment newseg1, testdb, datadev

select dbid, segmap from sysusages
 dbid    segmap
------  -----------
6            27
6              4
```

segmap 列

segmap 显示它所表示的数据库片段的允许存储。您应使用用于段管理的存储过程来控制此掩码中的位值。掩码中有效位的编号来自于本地数据库中的 syssegments。（“本地”数据库是您当前正在使用的数据库：或者是登录时使用缺省数据库，或者是通过 use database 最近使用的数据库）

Adaptive Server 提供了三个已命名的段：

- system，即段 0
- default，即段 1
- logsegment，即段 2

使用 sp_addsegment 创建其它段。如果您在 model 数据库中创建段，这些段会存在于随后创建的所有数据库中。如果在任何其它数据库中创建段，这些段只存在于该数据库中。不同数据库中的不同段名称可以具有相同的段号。例如，数据库 testdb 中的 newseg1 和数据库 mydb 中的 mysegment 可以同时具有段号 4。

segmap 列是链接到用户数据库 syssegments 表中 segment 列的位屏蔽。因为每个用户数据库中的 logsegment 为段 2，而且这些用户数据库将其日志放在单独的设备上，所以 segmap 中包含 4 (2^2)，表示在 log on 语句中指定的设备；包含 3，表示包含系统段 ($2^0 = 1$) + 缺省段 ($2^1 = 2$) 的数据段。

包含数据或日志的段的某些可能值为：

值	段
3	仅数据（系统和缺省段）
4	仅日志
7	数据和日志

大于 7 的数值表明这是用户定义的段。第 9 章“创建和使用段”中有关段的教程部分对 segmap 列进行了更加完整的说明。

以下查询说明 master..sysusages 中的段和 segmap 中的 syssegments 之间的联系。查询列出了当前数据库的段映射，显示了 syssegments 中的每个段号都将成为 master..sysusages 中的位号：

```
select dbid, lstart, segmap, name as 'segment name'
from syssegments s, master..sysusages u
where u.segmap & power(2,s.segment) != 0
and dbid = db_id()
order by 1,2
dbid   lstart    segmap       segment name
-----   ------    --------     --------------
4       0         3           system
4       0        3            default
4       5120      4           logsegment
4       7680      3           system
4       7680       3           default
```

本示例显示，lstart 值 0 对应的磁盘片段和 lstart 值 7680 对应的片段使用 system 段（编号 0）和 default 段（编号 1），而 lstart 值 5120 对应的片段使用 logsegment 段（编号 2）。此数据库是同时使用 create database 的 on 和 log on 子句创建的，之后使用 alter database 的 on 子句进行了一次扩展。

因为 sysusages segmap 使用 int 数据类型，它只能包含 32 位，因此数据库无法容纳 32 个以上的段（编号 0 - 31）。由于 segmap 是一个带符号的数量（也就是说，它可以显示为正数也可以显示为负数），段 31 被理解为一个非常大的负数，因此在使用段 31 的数据库中使用 segmap 时，以上查询会发生算术溢出。

lstart、*size* 和 *vstart* 列

- lstart 列 — 此分配单元在数据库中的起始页号。每个数据库起始于逻辑地址 0。如果对数据库进行了其它分配，如 dbid 7 所示的情况，lstart 列会反映出来。
- size 列 — 指派到同一数据库的连续页的数目。数据库该部分的结束逻辑地址可通过在 lstart 和 size 中增加数值来确定。
- vstart 列 — 指派到该数据库的块开始的地址。
- vdevno — 该数据库片段所在的设备。

获取有关数据库存储的信息

本节阐述如何确定当前分配到数据库的数据库设备以及每个数据库使用多少空间。

数据库设备名和选项

要查找特定数据库驻留的数据库设备名，需使用带有数据库名称的 sp_helpdb：

```
sp_helpdb pubs2
```

```
name      db_size    owner     dbid created        status
--------- ---------- --------- ---- -------------- --------------
pubs2     20.0 MB     sa           4 Apr 25, 2005   select
     into/bulkcopy/pllsort, trunc log on chkpt, mixed log and data

device_fragments    size          usage         created      free kbytes
------------------- ------------- ------------- ----------   ------------
master              10.0MB        data and log  Apr 13 2005          1792
pubs_2_dev         10.0MB        data and log  Apr 13 2005         9888

device                  segment
----------------------- -----------------------
master                  default
master                  logsegment
master                  system
pubs_2_dev              default
pubs_2_dev              logsegment
pubs_2_dev              system
```

```
pubs_2_dev              seg1
pubs_2_dev              seg2
```

sp_helpdb 报告指定的数据库使用的设备的大小和使用情况。状态列列出数据库选项。《系统管理指南，卷 1》中的第 8 章“设置数据库选项”描述了这些选项。

如果正在使用指定的数据库，sp_helpdb 还会报告数据库中的段和由段指定的设备。请参见第 9 章“创建和使用段”。

使用不带参数的 sp_helpdb 时，它报告 Adaptive Server 中所有数据库的相关信息：

```
                    sp_helpdb
name            db_size  owner dbid   created            status
-------------   -------- ----- ----   ------------       ------------------
master          48.0 MB   sa      1   Apr 12, 2005       mixed log and data
model            8.0 MB   sa      3   Apr 12, 2005       mixed log and data
pubs2           20.0 MB   sa      6   Apr 12, 2005       select into/
     bulkcopy/pllsort, trunc log on chkpt, mixed log and data
sybsystemdb      8.0 MB   sa      5   Apr 12, 2005       mixed log and data
sybsystemprocs 112.0 MB   sa      4   Apr 12, 2005       trunc log on chkpt,
     mixed log and data
tempdb           8.0 MB   sa      2   Apr 12, 2005       select into/
     bulkcopy/pllsort, trunc log on chkpt, mixed log and data
```

检查使用的空间量

sp_spaceused 提供了空间使用情况的摘要：

- 在数据库中使用的空间
- 由表及其索引以及 text/image 存储使用的空间
- 由表使用的空间，附带有索引以及 text/image 存储的单独信息

检查数据库中使用的空间

要获得数据库使用的存储空间量的总结，需在数据库中执行 sp_spaceused：

```
sp_spaceused
database_name                  database_size
------------------------------ -------------
pubs2                          2.0 MB

reserved      data          index_size      unused
```

```
------------- ------------- --------------- --------
-
1720 KB        536 KB         344 KB           840 KB
```

表 6-2 对报告中的列进行了说明。

表 6-2：sp_spaceused 输出结果中的列

列	说明
database_name	正在检查的数据库的名称。
database_size	通过 create database 或 alter database 命令分配给数据库的空间量。
reserved	分配给所有在数据库中创建的表和索引的空间量。（分配给数据库中的数据库对象的空间增量为一次 1 个扩充，或 8 个页。）
data，index_size	数据和索引使用的空间量。
unused	已保留但尚未被已存在的表和索引使用的空间量。

unused、index_size 和 data 列中数值的和应当与 reserved 列中的数值相等。从 database_size 减去 reserved 以获得未保留的空间量。这些空间可用于新的对象或增长到超出为其保留的空间的已存在的对象。

通过定期运行 sp_spaceused，可以监控可用的数据库空间量。例如，如果 reserved 值与 database_size 值相近，则表明已经没有足够的空间用于新对象。如果 unused 的数值也很小，则表明可用于其它数据的空间也已不足。

检查表的摘要信息

也可以将表的名称作为其参数来使用 sp_spaceused：

```
sp_spaceused titles
name   rowtotal reserved  data    index_size unused
------ -------- --------- ------- ---------- -----
titles 18       48 KB     6 KB    4 KB       38 KB
```

rowtotal 列也许与对表运行 select count(*) 的结果不同。这是因为 sp_spaceused 使用内置函数 rowcnt 计算数值。此函数使用存储在分配页中的数值。不过这些数值并不定期更新，因此对于有很多活动的表，它们可能差别很大。update statistics、dbcc checktable 和 dbcc checkdb 更新每页行数的估计值，因此在运行这些命令中的一个之后，rowtotal 最准确。

在 syslogs 上定期运行 sp_spaceused，因为如果数据库的修改频繁，事务日志会快速增长。特别是当事务日志不在独立的设备上时，这更是个问题，这意味着它与数据库的其它部分争用间。

检查表及其索引的信息

要查看关于单个索引使用的空间的信息，需输入：

```
                    sp_spaceused titles, 1
index_name          size       reserved   unused
------------------- ---------- ---------- ----------
titleidind          2 KB       32 KB      24 KB
titleind            2 KB       16 KB      14 KB

name       rowtotal   reserved   data   index_size unused
---------- --------  ---------  ------- ---------- ----------
titles          18      46 KB     6 KB      4 KB      36 KB
```

对由 text/image 页存储占用的空间与由表占用的空间分别进行报告。text 和 image 存储的对象名始终都是“t”加上表名：

```
                    sp_spaceused blurbs,1
index_name          size       reserved   unused
------------------- ---------- ---------- ----------
blurbs              0 KB       14 KB      12 KB
tblurbs             14 KB      16 KB      2 KB

name       rowtotal reserved    data    index_size unused
---------- -------- ----------- ------- ---------- ----------
blurbs             6       30 KB    2 KB      14 KB      14 KB
```

查询系统表中的空间使用信息

有时需要自己编写某些查询来获取关于物理存储的其它信息。例如，若要确定 Adaptive Server 上存储空间的 2K 块总数，可以查询 sysdevices：

```
select sum(convert(numeric(20,0), high - low + 1))
from sysdevices
where status & 2 = 2

-----------------
             230224
```

在此示例中，status 列（第 3 行）中的 2 指示物理设备。high 表示设备上最高有效的 2KB 块，因此必须加 1 才能通过减法（第一行中的 `high - low`）获得真正的计数并将计数转换为 numeric(20,0)，以避免由于总和中加上整数而导致溢出。

第 7 章

装入和卸下数据库

概述

使用 mount 和 unmount 命令可执行以下操作：

- 更方便地打包专有数据库；例如通过数据文件而不是 SQL 脚本。不需要再执行运行这些 SQL 脚本所必需的相关操作，例如设备和数据库设置。
- 移动数据库 — 将一组数据库从源 Adaptive Server 移动到目标 Adaptive Server 时，实际是在物理移动基础设备。
- 在不关闭 Adaptive Server 的情况下复制数据库。通过命令行复制数据库时，必须在 Adaptive Server 外部执行操作，并使用 UNIX dd 或 ftp 等命令来创建一组数据库（包含一个或多个数据库）中所有页的逐字节副本。

  在 isql 提示符处运行 mount 和 unmount：主 Adaptive Server 是源，辅助 Adaptive Server 是目标。quiesce database 还允许一个辅助 Adaptive Server 充当来自多个主服务器的数据库的备份，因为可将来自多个源的数据库复制到一个目标。

若要卸下并重新装入数据库，请执行以下操作：

1 使用 unmount 从服务器删除数据库及其设备。将在命令子句中指定的位置为数据库创建一个清单文件。该清单文件包含与源 Adaptive Server 上的数据库相关的信息，如，数据库设备、服务器信息和数据库信息，等等。请参见第 158 页的“清单文件”。

2 将数据库复制或移动到目标 Adaptive Server。

3 使用 mount 添加该数据库的设备、属性等。

4 使用 database online 可以在目标 Adaptive Server 上使数据库联机，而无需重新启动服务器。

请参见《参考手册：命令》，查看 mount 和 unmount database 的完整描述。

注释 Cluster Edition 中支持 mount database 和 unmount database。如果在使用这些命令时发生实例故障切换恢复，这些命令可能会中止。在这种情况下，用户必须在实例故障切换恢复完成后重新发出命令。

警告！ 对于每个可以访问原始 Adaptive Server 上的数据库的登录名，目标 Adaptive Server 上必须有 suid 相同的相应登录名。

若要使权限保持不变，目标 Adaptive Server 上的登录名映射必须与源 Adaptive Server 上的登录名映射完全相同。

清单文件

清单文件是一个二进制文件，它包含与数据库相关的信息，例如，数据库设备、服务器信息和数据库信息，等等。仅当占用这些设备的那组数据库处隔离和自我包含状态时才能创建清单文件。清单文件包含：

- 特定于服务器或对所有数据库均相同的源服务器信息，例如，源 Adaptive Server 的版本、任何升级版本、页大小、字符集、排序顺序和创建该清单文件日期，等等。
- 来自 sysdevices 目录的设备信息。清单文件包含有关第一个和最后一个虚拟页的信息、状态和所涉及设备的逻辑和物理名称。

- 来自 sysdatabases 目录的数据库信息。清单文件包含有关以下各项的信息：数据库名称、dbid、数据库管理员 (dba) 的登录名、所有者的 suid、指向事务日志的指针、创建日期、sysdatabases 中状态字段的状态位、上次执行 dump tran 的日期和所涉及数据库的 diskmap 条目等等。

警告！清单文件是一个二进制文件，因此，必须以二进制模式对该文件的内容执行字符转换操作（例如 ftp），否则将损坏该文件。

复制和移动数据库

移动或复制操作是数据库级操作，而且需要 Adaptive Server 外部的活动。若要移动或复制设备和数据库，需要确保它们是使用支持物理传送的单元在源 Adaptive Server 上设置的。

例如，如果任一设备由多个数据库使用，那么所有这些数据库都必须在一个操作中传送。

复制数据库时，实际上是通过物理复制基础设备将一组数据库从源复制到目标，即将一组数据库从源 Adaptive Server 复制到目标 Adaptive Server。

quiesce database 命令允许您包含用于为外部转储创建清单文件的参数。可以使用 Adaptive Server 外部的实用程序或命令（tar、zip 或 UNIX dd 命令）将数据库移动或复制到目标 Adaptive Server。在目标 Adaptive Server 上，数据是使用相同的外部命令或实用程序提取并放置在设备上的。

如果一个设备用于多个数据库，则必须通过一个操作删除该设备上的所有数据库。

在最初配置源 Adaptive Server 时应谨慎。Adaptive Server 无法检验设备是否可作为单元传送。请确保要断开连接的基础磁盘不会导致不进行移动的数据库失其部分存储。Adaptive Server 无法识别一个物理磁盘上的驱动器是否已分区；必须在一个单元中一起移动这些数据库。

警告！对于移动操作，mount 和 unmount 允许标识多个数据库。不过，如果设备用于多个数据库，则所有数据库都必须在一个操作中移动。请指定要传送的数据库组。这些数据库所使用的设备不能与除该命令中指定的数据库以外的任何无关数据库共享。

性能考虑事项

在目标 Adaptive Server 上，传送的数据库的数据库 ID 必须相同，除非您装入数据库只是为了临时使用，在这种情况下必须运行 checkalloc 修复数据库 ID。

如果改变了 dbid，数据库中的所有存储过程都将被标记为需重新编译。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。

设备检验

目标 Adaptive Server 通过扫描各个数据库的设备分配边界来检验清单文件中的设备。该扫描确保正在装入的设备对应于清单文件中描述的分配。对于每 sysusages 条目的第一个和最后一个分配页，它还根据清单文件中的 dbid 检验分配页中的 dbid。

如果需要进行更严格的设备检查，可以在 mount 命令中使用 with verify 选项，以检验数据库中所有分配页的 dbid。

必须要特别小心，以确保不混淆设备的副本。

例如，如果制作一个由磁盘 dsk1、dsk2 和 dsk3 的副本组成的数据库副本，在八月份，尝试从三月份制作的数据库副本装入 dsk1 和 dsk2 的副本，并从六月份制作的数据库副本装入 dsk3 的副本，则即使在 mount 命令中使用了 with verify 选项，也会通过分配页检查。由于数据库信息对于正在使用的版本无效，因此恢复将失败。

但是，如果不访问 dsk3，恢复可能不会失败。这意味着数据库将联机，但数据可能已损坏。

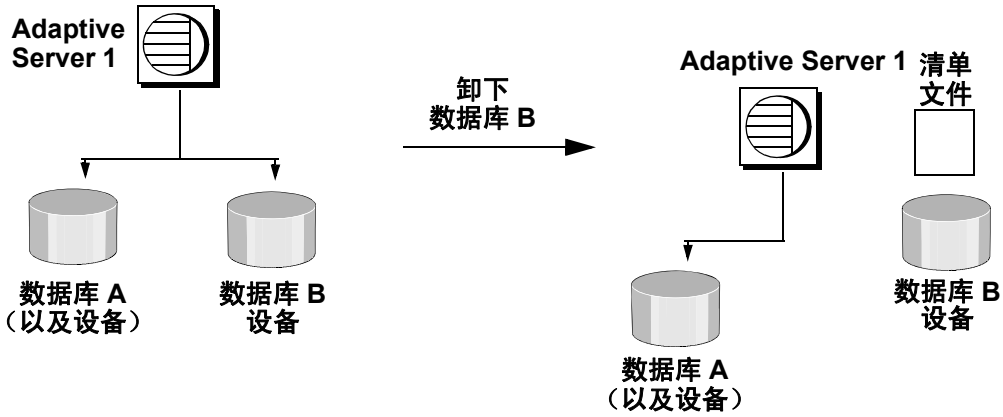
装入和卸载数据库

本节讲解如何使用 mount 和 unmount 命令。quiesce database 命令包含一个可简化 mount 和 unmount 命令的子句。

卸下数据库

卸下数据库时，将从 Adaptive Server 中删除数据库及其设备。unmount 命令将关闭数据库。使用该数据库的所有任务都将被终止。数据库及其页不会更改，仍会保留在操作系统设备上。表 7-1 显示了从系统卸下数据库时发生的情况。

图 7-1：unmount 命令



注释　对于移动操作，您可以通过 unmount 标识多个数据库。不过，如果设备用于多个数据库，则所有数据库都必须在一个操作中移动。指定要传送的数据库组。这些数据库所使用的设备不能与除该命令中指定的数据库以外的任何无关数据库共享。

可在一个 unmount 命令中移动的数据库数限制为八个。

unmount 命令：

- 关闭数据库
- 从 Adaptive Server 中删除数据库
- 使设备失效并删除设备，
- 使用 *manifest_file* 子句创建清单文件。

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）。

```
unmount database <dbname list> to <manifest_file> [with
{override, [waitfor=<delay time]} ]
```

例如：

```
unmount database pubs2 to "/work2/Devices/Mpubs2_file"
```

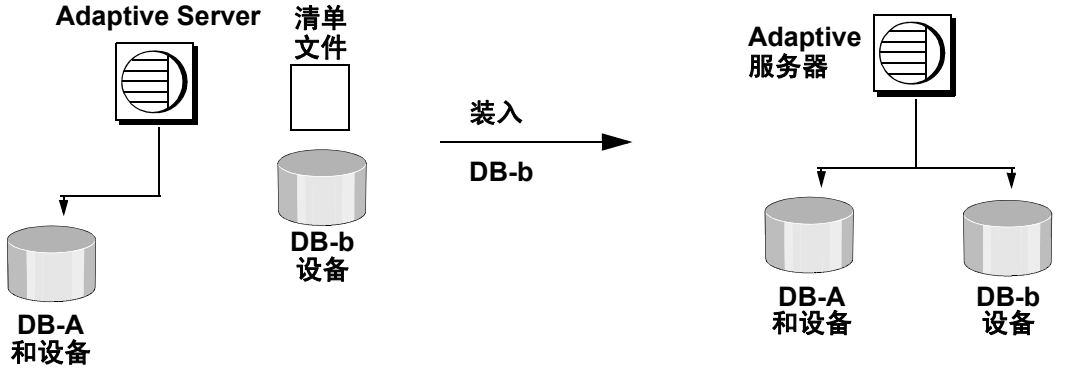
如果现在尝试使用 pubs2 数据库，将显示以下内容：

```
Attempt to locate entry in sysdatabases for database
'pubs2' by name failed - no entry found under that
name.Make sure that name is entered properly.
```

注释 在引用数据库由 unmount 命令以替换方式删除后，将不能删除参照约束（依赖项）或表。

装入数据库

图 7-2：mount 命令



使用 mount 命令将数据库附加到目标或辅助 Adaptive Server。mount 对 manifest 文件中包含的信息进行解码，并使数据库组可供联机。所需的全部支持活动都将执行，包括连接数据库设备（如果需要）并将其激活，为数据库创建目录条目，恢复数据库和使数据库联机。

一个 mount 命令中对数据库数的限制为八个。

请参见《参考手册：命令》中的 mount。

注释　对于移动操作，mount 允许标识多个数据库。不过，如果设备用于多个数据库，则所有数据库都必须在一个操作中移动。指定要传送的数据库组。这些数据库所使用的设备不能与除该命令中指定的数据库以外的任何无关数据库共享。

可以以不同的方式使用 mount：

- 在目标 Adaptive Server 上使用 mount 命令。例如：

  ```
  mount database all from "/data/sybase2/mfile1"
  using "/data/sybase1/d0.dbs" = "1dev1"
  ```

  数据库及其设备将出现在目标 Adaptive Server 上，标记为装入中。系统将由对系统目录的更新和有关数据库的相应信息填充，但不会恢复数据库本身。而，它们能够经受住系统故障。

  然后，目标 Adaptive Server 将恢复数据库（一次恢复一个）。恢复后，数据库将保持脱机状态。

  如果恢复在某个数据库上失败，只会影响该数据库。系统将继续恢复其它数据库。

  使用 database online 命令可以使数据库联机。

  无需重新启动目标服务器。

- 使用带有 listonly 的 mount 命令可显示来自源 Adaptive Server 的清单文件中的路径名，而无需装入数据库。

  装入数据库之前，请使用 listonly 参数在目标 Adaptive Server 上列出设备路径名。例如：

  ```
  mount database all from "/data/sybase2/mfile1" with
  listonly

  /data/sybase1/d0.dbs = ldev1
  ```

  然后使用 mount 实际装入数据库。获得路径名之后，检验并修改它们以使其符合目标 Adaptive Server 上的标准。

在执行 mount 以将数据库装入 Adaptive Server 中时：

- 不能装入清单中描述的数据库的子集。清单中列出的所有数据库和设备都必须一起装入。
- 要装入的数据库必须与以前的 Adaptive Server 具有相同的页大小。
- 为成功添加属于所装入数据库的所有设备，必须在辅助 Adaptive Server 上配置了足够的设备。
- 必须相应设置配置参数 number of devices。
- 与所装入数据库同名的数据库和设备不能已经存在。
- Adaptive Server 必须具有与所装入数据库相同的版本。
- 装入的数据库必须来自与 Adaptive Server 相同的平台。

创建数据库的可装入副本

1 将 quiesce database 命令与 manifest 子句一起使用并抑制数据库。该命令创建描述该数据库的 *manifest* 文件。

2 将 mount 命令与 listonly 一起使用可以显示要复制的设备的列表。

3 使用外部复制实用程序（例如 cp、dd、split mirror 等）将数据库设备复制到其它 Adaptive Server。

这些设备的副本和清单文件是该数据库的可装入副本。

将数据库从一个 Adaptive Server 移到另一个 Adaptive Server

1 使用 unmount 命令可从第一个 Adaptive Server 卸下数据库。该命令创建描述该数据库的清单文件。

2 使数据库设备可用于第二个 Adaptive Server（如果这些设备尚不可用）。如果第二个 Adaptive Server 位于其它计算机上，可能需要您的操作系统管理员来帮您完成上述工作。

3 通过在第 1 步中创建的清单文件对辅助 Adaptive Server 执行 mount 命令。

系统限制

- 不能卸下系统数据库。但可以卸下 sybsystemprocs。
- 不能卸下代理数据库。
- 在事务中不允许执行 mount 和 unmount 数据库命令。
- 在配置了 HA 的服务器中不允许 mount 数据库。

quiesce database 扩展

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 首先会阻塞数据库中的写操作从而影响到 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。

如果被抑制的那组数据库包含对那组数据库之外的数据库的引用，将无法创建清单文件。您可以使用替换选项绕过此限制。

接下来，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。

请参见《参考手册：命令》中的 quiesce database。

第 8 章

分布式事务管理

Adaptive Server 包括以下分布式事务管理功能：

- 改进的事务和线程管理功能。Adaptive Server 将所有事务当作服务器资源进行管理，并提供了在事务中附加和分离线程的能力。这些新增功能提供了一个通用接口，可支持本地服务器事务的客户端，以及支持 X/Open XA 和 MSDTC 环境中的客户端。请参见第 172 页的“配置事务资源”。
- 新增的分布式事务协调服务。对于通过 RPC 和 CIS 来修改远程 Adaptive Server 中数据的事务，Adaptive Server 提供了一致的回退和提交功能。即使没有外部事务管理器，新增的事务协调服务也能确保这种分布式事务的完整性。请参见第 174 页的“使用 Adaptive Server 协调服务”。
- 改进的就绪事务恢复功能。在恢复过程中，Adaptive Server 可识别由 X/Open XA 协议和 Adaptive Server 本机事务协调服务来协调的就绪事务。Adaptive Server 会将这些事务恢复到它们在恢复前的状态，并且使关联数据库联机的速度比早期版本的服务器更快。请参见第 186 页的“分布式事务的崩溃恢复过程”。
- 新增的 dbcc 命令，用于尝试完成分布式事务。请参见第 187 页的“尝试完成事务”。

受影响的事务类型

Adaptive Server 新增的 DTM 功能将影响以下几种事务：

- 由外部事务管理器协调的分布式事务
- 使用 RPC 和 CIS 更新数据的事务

由外部事务管理器协调的分布式事务

分布式事务可发生在以下环境中：外部事务管理器使用特定的协议（例如 X/Open XA）来协调事务的执行。Adaptive Server 支持通过 Adaptive Server 的 DTM XA 接口使用 CICS、Encina、TUXEDO 和 MSDTC 事务管理器的事务。

注释 具有 DTM XA 接口的 Adaptive Server 提供了以前曾是 XA-Server 产品一部分的功能。XA-Server 产品已不是必需的，因而未包括在 Adaptive Server 中。有关 DTM XA 接口的信息，请参见《适用于 CICS、Encina 和 TUXEDO 的 XA 接口集成指南》。

事务管理器协调事务的行为

Adaptive Server 本身所实现的几项功能曾是 XA-Library 和 XA-Server 产品的一部分，它为通过 X/Open XA 协议协调的就绪事务提供了新的恢复过程。有关详细信息，请参见第 172 页的“配置事务资源”和第 186 页的“分布式事务的崩溃恢复过程”。

Adaptive Server 的 XA 接口已经过修改，可支持服务器新增的分布式事务管理功能。对 XA 接口所做的更改对于 X/Open XA 客户端应用程序是透明的。不过，必须将 Adaptive Server DTM XA 接口与 X/Open XA 事务管理器链接起来，这样才能将 Adaptive Server 用作资源管理器。有关所有 XA 接口更改的详细信息，请参见《适用于 CICS、Encina 和 TUXEDO 的 XA 接口集成指南》。

此外，Adaptive Server 还包括对 MSDTC 协调的分布式事务的支持。MSDTC 客户端可以使用本机接口直接与 Adaptive Server 通信。客户端还可以通过 DTM XA 接口与一个或多个在 UNIX 上运行的 Adaptive Server 通信。

注释　使用 DTM XA 接口的 MSDTC 客户端在其所访问的 Adaptive Server 中必须具有 dtm_tm_role。有关 dtm_tm_role 的详细信息，请参见《适用于 CICS、Encina 和 TUXEDO 的 XA 接口集成指南》。

Adaptive Server 版本 15.0.3 或更高版本的增强型事务管理器

在 15.0.3 以前的 Adaptive Server 版本中，当 Adaptive Server 在应用程序不知道的情况下隐式中止外部事务时，通常在该事务内运行的 DML 命令可能改为在显事务外执行。它们将在由 Adaptive Server 启动的隐式事务内执行。此行为可能导致业务数据不一致。要处理这种情况，用户应用程序应始终检查外部事是否仍处于活动状态，并发出相应的命令。

在版本 15.0.3 和更高版本中，如果存在外部事务隐式回退，Adaptive Server 不会允许在附加到外部事务的连接上执行任何 DML 命令，直到事务管理器发出离请求。分离请求表示计划用于外部事务的一批命令的结束。

在版本 15.0.3 和更高版本中，Adaptive Server 会自动阻止原打算在分布式事务内执行的 SQL 命令在该事务外执行。用户应用程序不再必须在发出每个命令检查全局变量 *@@trancount*；当某个事务被隐式中止时，将显示一条错误消息 (3953)：“由于外部事务已回退，无法执行该命令。”发出 detach transaction 命令时，此消息将消失。

要禁止 3953 错误消息并使 Adaptive Server 恢复以前的行为（即，即使 DTM 不处于活动状态，也执行 SQL 命令），请使用跟踪标志 -T3955 启动 Adaptive Server。

RPC 和 CIS 事务

通过使用 Transact-SQL 远程过程调用 (RPC) 和组件集成服务 (CIS)，本地 Adaptive Server 事务可以更新远程服务器中的数据。从本地创建的事务中执行 RPC 即可完成 RPC 更新。例如：

```
sp_addserver westcoastsrv, ASEnterprise, hqsales
begin transaction rpc_tran1
update sales set commission=300 where salesid="12OZ"
exec westcoastsrv.salesdb..recordsalesproc
commit rpc_tran1
```

上述事务不但更新本地 Adaptive Server 中的 sales 表，而且使用 recordsalesproc 这一 RPC 来更新远程服务器上的数据。

通过 CIS，可以将远程表当作本地表来更新其中的数据。通过使用 sp_addobjectdef，用户可以在引用远程数据的 Adaptive Server 中创建本地对象。对本地对象进行更新将修改远程 Adaptive Server 中的数据。例如：

```
sp_addobjectdef salesrec,
"westcoastsrv.salesdb..sales", "table"
begin transaction cis_tran1
update sales set commission=300 where salesid="12OZ"
update salesrec set commission=300 where salesid="12OZ"
commit cis_tran1
```

RPC 和 CIS 事务的新行为

在 Adaptive Server 版本 12.0 以前的版本中，通过 RPC 和 CIS 更新数据的事务不能回退远程服务器的工作，也不能向这些事务保证远程工作已实际提交。因此，Adaptive Server 提供了新的事务协调服务，以确保 RPC 和 CIS 更新随启动事务提交或回退它们的工作。有关详细信息，请参见第 174 页的“使用 Adaptive Server 协调服务”。

如果您的应用程序依赖于以前的 RPC 和 CIS 更新行为，则可禁用事务协调服务。有关信息，请参见第 171 页的“enable xact coordination 参数”。

SYB2PC 事务

SYB2PC 事务使用 Sybase 两阶段提交协议来确保将分布式事务的工作作为一个逻辑单元进行提交或回退。

Adaptive Server 不会修改 SYB2PC 事务的行为。但是，实现 SYB2PC 事务的应用程序开发人员可能需要考虑改用 Adaptive Server 事务协调服务。与 SYB2PC 事务相比，直接由 Adaptive Server 协调的事务将使用较少的网络连接并以更快的速度执行，同时仍可确保分布式事务的完整性。由 Adaptive Server 代替应程序来协调远程事务还可以简化应用程序代码。有关详细信息，请参见第 174 页的“使用 Adaptive Server 协调服务”。

启用 DTM 功能

安装许可密钥

分布式事务管理是单独许可的 Adaptive Server 功能。必须先购买并安装 Adaptive Server 和 DTM 功能的有效许可证，然后才能启用和使用 DTM 功能。

有关安装许可密钥和使用 Sybase 软件资产管理 (*SySAM*) 的信息，请参见安装指南。如果要购买 DTM 或 Adaptive Server 的其它许可功能的许可证，请与 Sybase 销售代表联系。

启用 DTM 功能

购买和安装了 Adaptive Server 和 DTM 功能的有效许可证之后，即可使用带有 enable dtm 和 enable xact coordination 配置参数的 sp_configure 启用 DTM 功能。

enable dtm 参数

enable dtm 参数可启用或禁用 DTM 的基本功能。当 enable dtm 设置为 1（打开）时，Adaptive Server 将支持来自 MSDTC 的外部事务，并通过 DTM XA 接口支持来自 X/Open XA 事务管理器的外部事务。有关详细信息，请参见《适用于 CICS、Encina 和 TUXEDO 的 XA 接口集成指南》。

要启用 DTM 的基本功能，可使用以下命令：

```
sp_configure 'enable dtm', 1
```

必须重新启动 Adaptive Server 才能使该更改生效。

enable xact coordination 参数

enable xact coordination 启用或禁用 Adaptive Server 事务协调服务。当启用该参数时，Adaptive Server 会确保对远程 Adaptive Server 数据的更新将随初始事务一起提交或回退。有关详细信息，请参见第 174 页的“使用 Adaptive Server 协调服务”。

要启用事务协调，可使用以下命令：

```
sp_configure 'enable xact coordination', 1
```

必须重新启动 Adaptive Server 才能使该更改生效。

配置事务资源

Adaptive Server 提供了一个通用接口，可同时支持通过分布式事务协议来协调的本地服务器事务和外部事务。分布式事务协议支持适用于 X/Open XA、MSDTC 和本机 Adaptive Server 事务协调服务。

Adaptive Server 将所有事务当作可配置的服务器资源进行管理，系统管理员可以配置给定服务器中可用资源的总数。在 X/Open XA 环境中访问 Adaptive Server 的客户端任务还可以根据需要将线程挂起和将其连接到事务资源。

本节将介绍如何确定和配置可用于 Adaptive Server 的事务资源的总数。

计算所需的事务描述符

Adaptive Server 使用**事务描述符**资源来管理服务器中的事务。事务描述符是 Adaptive Server 用来表示事务的内部内存结构。

在启动时，Adaptive Server 根据配置参数 txn to pss ratio 的值分配固定数目的事务描述符，并将它们放在一个池中。当新事务需要使用事务描述符时，Adaptive Server 就从池中获取这些事务描述符。当事务完后，描述符将返回到池中。如果没有可用的事务描述符，事务就可能被延迟，因为 Adaptive Server 要等待描述符被释放。

为了正确配置事务描述符的数量，务必要确切理解 Adaptive Server 何时需要从全局池中获取新的描述符。下面列出了需要获取新描述符的情况：

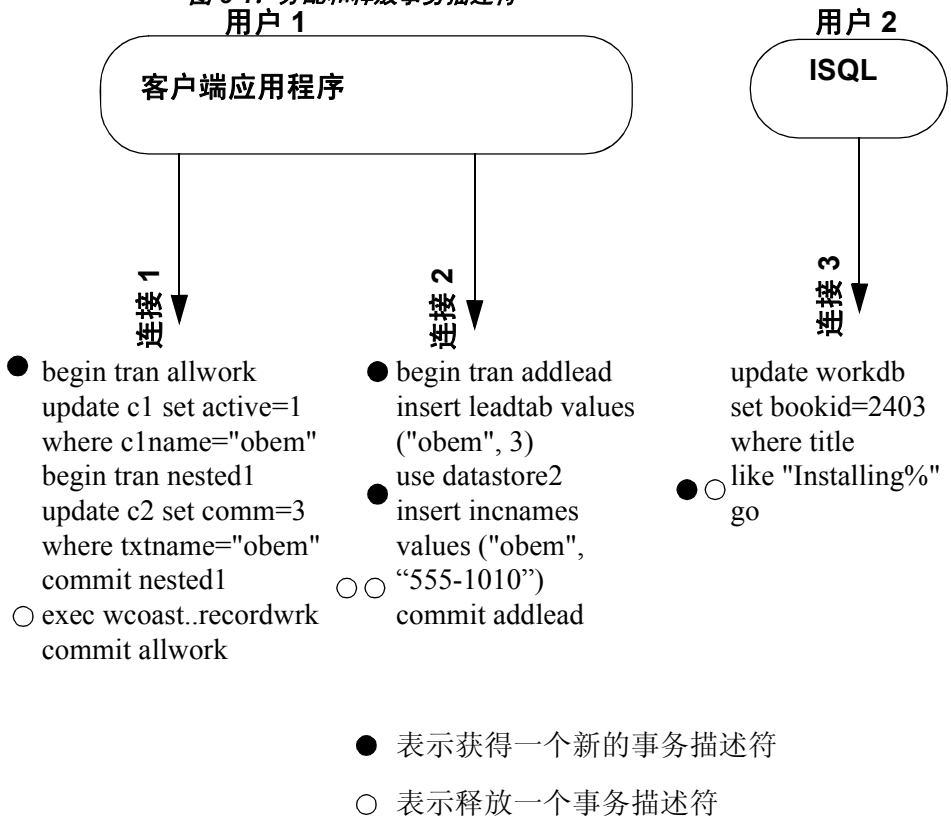
- 客户端连接启动一个新的外层事务。当客户端执行外层 begin transaction 命令时，这种情况将显式发生。当客户端修改数据，而不输入 begin transaction 命令时，这种情况还将隐式发生。

  开始一个外层事务之后，随后的嵌套 begin transaction 命令将不再需要其它事务描述符。事务描述符的分配和重新分配由事务最外层的块来决定。

- 现有事务修改另一个数据库（多数据库事务）。多数据库事务需要为其访问的*每个*数据库分配一个专用的事务描述符。

图 8-1 说明了 Adaptive Server 如何获取和释放不同事务类型的事务描述符。

图 8-1：分配和释放事务描述符



在图 8-1 中，Adaptive Server 为用户 1 使用了总共三个事务描述符，该用户通过一对客户端连接来访问服务器。服务器为事务 allwork 分配了一个描述符，它在该事务提交后被释放。嵌套事务 nested1 不需要专用的事务描述符。

事务 addlead 为多数据库事务，它需要两个事务描述符，一个用于外层事务块，一个用于修改另一个数据库 datastore2。当外层事务块提交后，将释放这两个事务描述符。

用户 2 通过 isql 访问 Adaptive Server，也需要专用的事务描述符。即使用户 2 没有使用 begin transaction 显式地创建外部事务块，Adaptive Server 也会隐式地创建一个事务块以执行 update 命令。与此块关联的事务描述符将在执行 go 命令之后获得，并在完成插入后被释放。

由于事务描述符会占用其它 Adaptive Server 服务可使用的内存，所以务必要尽量少使用描述符，能够满足在给定时间所需的最大事务数量即可。

设置事务描述符的数量

确定了要在系统中使用的事务描述符数量之后，可使用 sp_configure 来设置 txn to pss ratio 的值。txn to pss ratio 确定可用于服务器的事务描述符总数。启动时，系统会用这一比率乘以 number of user connections 参数来创建事务描述符缓冲池：

```
# of transaction descriptors = number of user
connections * txn to pss ratio
```

缺省 txn to pss ratio 值是 16，用于确保与 Adaptive Server 的早期版本兼容。在 12.0 以前的版本中，Adaptive Server 为每个用户连接分配 16 个事务描述符。在 12.0 和以后的版本，同时发生的事务数量仅受到服务器中可用描述符数量的限制。

例如，要为每个用户连接分配 25 个事务描述符，可使用以下命令：

```
sp_configure 'txn to pss ratio', 25
```

必须重新启动 Adaptive Server 才能使该更改生效。

使用 Adaptive Server 协调服务

本地 Adaptive Server 事务工作有时会分配给可修改远程数据的远程服务器。当本地事务通过执行远程过程调用 (RPC) 来更新其它 Adaptive Server 表中的数据时，或者当本地事务使用组件集成服务 (CIS) 来修改远程表中的数据时，就可能会出现这种情况。

事务协调服务概述

在 Adaptive Server 12.0 版本之前，本地事务在执行 RPC 或通过 CIS 更新数据后将无法回退在远程 Adaptive Server 中所做的工作。此外，执行本地事务的客户端无法确保远程工作已实际提交，例如在远程服务器发生系统障时。

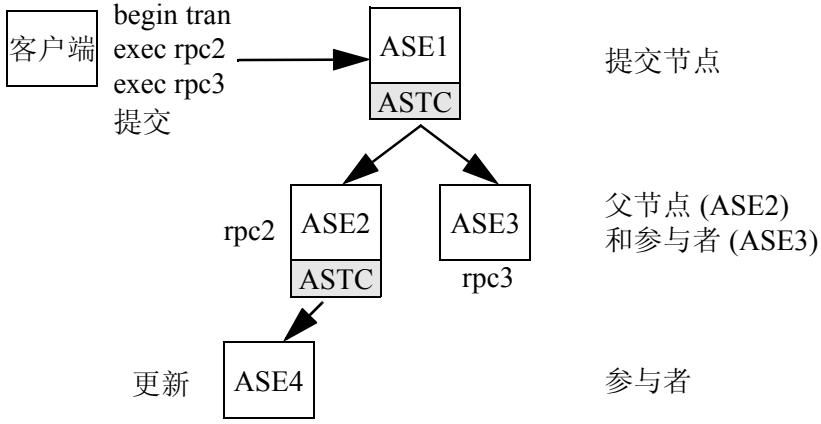
Adaptive Server 提供了相应的服务，可将事务传播给远程服务器并协调所有服务器的工作，以确保所有工作都作为一个逻辑单元来提交或回退。有了这些事务协调服务，Adaptive Server 本身就可以作为分布式事务管理器来管理在多个 Adaptive Server 中进行数据更新的事务。

层次事务协调

由于分布式事务所涉及的其它 Adaptive Server 也可能会协调远程参与者，因此可以进一步将事务以层级的方式传播给其它服务器。例如，在图 8-2 中，连接到 ASE1 的客户端启动了一个分别在 ASE2 和 ASE3 上执行 RPC 的事务。ASE1 的协调服务将该事务传播到 ASE2 和 ASE3。

由于 ASE2 也启用了事务协调服务，因此它可以将该事务传播给其它远程参与者。在该图中，ASE2 将事务传播给使用 CIS 更新数据的 ASE4。

图 8-2：层次事务协调



在图 8-2 中，ASE1 称作分布式事务的提交节点。当 ASE1 上的事务提交时，ASE1 的协调服务将指示 ASE2 和 ASE3 做好向它们传播的事务的准备工作。在 ASE3 的本地工作已准备好用于提交时，它会指明其事务已准备就绪。ASE2 必须完成其本地工作，并指示 ASE3 准备其事务。在 ASE2 和 ASE3 中准备事务时，ASE1 中的协调服务会提交原始事务。然后，提交下级事务的指示将通过与传送准备指示相同的方式，传送到 ASE2、ASE3，并最终传送到 ASE3。

DTP 环境中符合 X/Open XA 的行为

X/Open XA 协议要求资源管理器为传播给远程资源管理器的事务提供协调服务。之所以有这种要求，是因为外部事务管理器（在某些情况下包括启动该事务的客户端）不知道何时将事务传播给远程服务器，因此无法确保远程事务按要求完成或中止。

作为资源管理器，新的事务协调服务使 Adaptive Server 完全符合 X/Open XA 协议。分布式事务可以通过 RPC 和 CIS 隐式传播给远程服务器，而 Adaptive Server 将确保全局事务的提交或回退状态保留在它所协调的远程服务器中。

需求和行为

只要每个远程的 Adaptive Server 版本均为 12.0 或更高版本，Adaptive Server 事务协调服务就可确保远程服务器的工作进行逻辑性提交或回退。

事务协调服务对于执行分布式事务的客户端是透明的。当本地客户端事务执行 RPC 或通过 CIS 更新数据时，协调服务将为远程工作创建新的事务名称，并将该事物传播给下级远程服务器。当本地客户端提交或回退本地事务时，Adaptive Server 会该请求与每个下级服务器进行协调，以确保同时提交或回退远程事务。

Adaptive Server 事务协调服务作为一个或多个名为“ASTC HANDLER”的后台任务运行，并可以使用 sp_who 进行查看。在使用多个 Adaptive Server 引擎的系统中，“ASTC HANDLER”进程的数量（向下舍入为最接近的整数）为：

```
引擎数 * 2/3
```

Adaptive Server 上最多可以运行 4 个“ASTC HANDLER”进程。

下面的 sp_who 输出显示的是单个“ASTC HANDLER”：

```
sp_who
fid spid  status   loginame   origname   hostname           blk_spid   dbname
   tempdbname   cmd              block_xloid    threadpool
--- ----  --------- ---------  ---------  ----------------   --------   ------
-----
   -----------   ----------------- -----------    -----------------
0     1   running       sa        sa          dtmsol1         0       master
        tempdb             SELECT           0    syb_default_pool
0     2  sleeping     NULL      NULL           master         0       master
        tempdb   NETWORK HANDLER          0    syb_default_pool
0     3  sleeping     NULL      NULL                          0       master
        tempdb     DEADLOCK TUNE          0    syb_default_pool
```

```
0      4  sleeping      NULL      NULL                          0       master
          tempdb     MIRROR HANDLER           0    syb_default_pool
0      5  sleeping      NULL      NULL                          0       master
          tempdb        HOUSEKEEPER            0    syb_default_pool
0      6  sleeping      NULL      NULL                          0       master
          tempdb   CHECKPOINT SLEEP           0    syb_default_pool
0      7  sleeping      NULL      NULL       metin1_dtm         0  sybsystemdb
          tempdb       ASTC HANDLER           0    syb_default_pool
```

配置参与者服务器资源

缺省情况下，事务协调服务总是被启用。系统管理员可以使用 enable xact coordination 配置参数来启用或禁用这些服务。有关该参数的完整说明，请参见《系统管理指南》。

同时，系统管理员必须确保 Adaptive Server 具有所需的资源，以协调事务所请求的所有 RPC 和 CIS 更新。每当事务发出 RPC 或 CIS 更新请求时，事务协调器就必须获得可用的 **DTX 参与者**。DTX 参与者或“分布式事务参与者”是一种内部内存结构，Adaptive Server 使用它来协调已经传播给下级 Adaptive Server 的事务。在图 8-2 中，ASE1 需要三个可用的 DTX 参与者，而 ASE2 需要两个可用的 DTX 参与者。（在这两种情况下，都使用单个 DTX 参与者来协调所传播事务的本地工作。）

DTX 参与者资源一直由进行协调的 Adaptive Server 使用，直到关联的远程事务已提交。这通常会在启动事务提交*后*的一段时间内发生，因为只要所有下级事务都成功地准备好它们的工作，启动事务就将立即提交。

如果没有可用的 DTX 参与者，则无法处理 RPC 请求和 CIS 更新请求，该事务也随之被中止。

number of dtx participants 参数

使用 number of dtx participants 配置参数，系统管理员可以配置 Adaptive Server 中可用 DTX 参与者的总数。number of dtx participants 用于设置 Adaptive Server 事务协调服务一次可传播和协调的远程事务的总数。

在缺省情况下，Adaptive Server 可以协调 500 个远程事务。将 number of dtx participants 设置为较小的值会减少服务器能够管理的远程事务的数量。如果没有可用的 DTX 参与者，则新的分布式事务将不能启动。如果没有可用的 DTX 参与者传播新的远程事务，则正在进行的分布式事务可能会中止。

将 number of dtx participants 设置为较大的值会增加 Adaptive Server 能够处理的远程事务分支的数量，但占用的内存也更多。

针对系统优化 *number of dtx participants*

在高峰期间，可使用 sp_monitorconfig 检查 DTX 参与者的使用情况：

```
sp_monitorconfig "number of dtx participants"
Usage information at date and time:Jun 18 1999 9:00AM.
Name                          Num_Free   Num_Active  Pct_act   Max_Used
      Reuse_cnt    Instance_Name
----------------------------   --------   ----------  --------  -----------
      --------  ------------------------
number of dtx participant           480           20      4.00           37
          210                       NULL
participants
```

如果 #Free 的值是零或非常低，则新的分布式事务可能会由于缺少 DTX 参与者而无法启动。可考虑增加 number of dtx participants 的值。

如果 #Max Ever Used 的值太低，则未使用的 DTX 参与者可能会占用可供其它服务器功能使用的内存。可考虑减小 number of dtx participants 的值。

在异构环境中使用事务协调服务

在 Adaptive Server 将事务传播给其它 12.0 版本或更高版本的 Adaptive Server 时，它可确保分布式事务作为整体的完整性。不过，本地 Adaptive Server 事务的工有时会分配给不支持 12.0 版本或更高版本事务协调服务的远程服务器。当事务使用 RPC 来更新以前 Adaptive Server 版本中的数据时，或者使用 CIS 服务更新非 Sybase 数据库中的数据时，就可能出现这种情况。在这些情况下，进行协调的 Adaptive Server 将不能确保远程服务器的工作随原始事务回退提交。

strict dtm enforcement 参数

在 Adaptive Server 中，通过设置 strict dtm enforcement 配置参数，系统管理员可以强制或放松将分布式事务当作一个逻辑单元来提交或回退的要求。

注释 另外，也可以使用具有 strict_dtm_enforcement 选项的会话级别 set 命令来替换 strict dtm enforcement 的值。

---

strict dtm enforcement 决定 Adaptive Server 事务协调服务是否要严格执行分布式事务的 ACID 属性。

如果将 strict dtm enforcement 设置为 1（打开），就可确保只将事务传播给可以参与 Adaptive Server 协调事务的服务器。如果事务试图更新不支持事务协调服务的服务器中的数据，Adaptive Server 将中止该事务。

在异构环境中，可能需要使用不支持事务协调的服务器，包括旧版本的 Adaptive Server 和使用 CIS 配置的非 Sybase 数据库存储。在这些情况下，可以将 strict dtm enforcement 设置为 0（关闭）。这样能使 Adaptive Server 将事务传播到遗留的 Adaptive Server 和其它数据存储，但并不确保这些服务器的远程工作能够随原始事务回退或提交。

监控协调事务和参与者

Adaptive Server 使用新系统表 sybsystemdbdbo.syscoordinations 中的数据，跟踪与下级服务器所执行工作的状态有关的信息。请参见《参考手册：表》，以查看此表的完整定义。

sp_transactions 过程还显示 syscoordinations 表中用于进行中的远程事务的数据。有关详细信息，请参见第 181 页的“获取有关分布式事务的信息”。

DTM 管理和故障排除

事务和控制线程

在 Adaptive Server 版本 12.0 以前的版本中，事务的所有资源都由一个服务器任务独占。服务器只能与启动事务的任务来共享该事务。

Adaptive Server 12.5 和更高版本对由符合 X/Open XA 的事务管理器（如 Encina 和 TUXEDO）使用的“挂起”和“连接”语义提供本机支持。事务可以在不同的执行线程中共享，也可以不与任何线程相关联。

如果一个事务没有与之关联的线程，则称之为“分离的事务”。分配给分离的事务的 spid 值为 0。可以在新的 master.dbo.systransactions 表中查看事务的 spid 值，也可以在新的 sp_transactions 过程的输出中查看该值。有关详细信息，请参见第 181 页的“获取有关分布式事务的信息”。

对于系统管理员的提示

由于客户端应用程序可能需要重新附加初始线程，或者将新线程附加到事务，因此应该在 Adaptive Server 中保留分离的事务。当未将线程附加到该事务，系统管理员无法通过注销与事务相关的 spid 来回退该事务。

处于分离状态的事务也可以阻止用 dump transaction 命令来截断日志。在极为特殊的情况下，可以使用新的 dbcc complete_xact 命令来回退分离的事务，以尝试完成事务。请参见第 187 页的“尝试完成事务”。

dtm detach timeout period 参数

系统管理员也可以指定一个服务器范围的时间间隔，在此时间之后，Adaptive Server 将自动回退处于分离状态的事务。dtm detach timeout period 用于设置分布式事务分支可以保持分离状态的时间长度（单位为分钟）。如果超过此时间，Adaptive Server 将回退分离的事务。

例如，要在 30 分钟后自动回退分离的事务，可使用以下命令：

```
sp_configure 'dtm detach timeout period', 30
```

为支持分离的事务而对锁管理器所做的更改

在 Adaptive Server 版本 12.0 以前的版本中，锁管理器使用事务线程的 spid 值来唯一地标识事务锁。在新的事务管理器中，事务可以从其初始线程中分离出来，从而没有关联的 spid。此外，具有不同 spid 值的多个线程必须能够共享相同的事务锁，以执行分布式事务的工作。

为了便于进行这些更改，Adaptive Server 版本 12.5 和更高版本的锁管理器使用唯一的锁所有者 ID 取代 spid 来标识事务锁。锁所有者 ID 独立于创建事务的 spid，即使事务从线程中分离出来，锁所有者 ID 仍然保留。当事务没有关联的线程，或者新线程被附加到事务时，就可以通过锁所有者 ID 来支持事务锁。

锁所有者 ID 存储在 master.dbo.syslocks 的新 loid 列中。若要确定事务的 loid 值，可以检查 sp_lock 或 sp_transactions 输出。

通过检查 sp_transactions 输出中的 spid 和 loid 列，可了解有关事务及其控制线程的信息。spid 值为 0 表示该事务已从其控制线程中分离。非零的 spid 值表示控制线程当前附加在该事务上。

如果 sp_transactions 输出中的 loid 值为偶数，则表示本地事务拥有该锁。如果 loid 值为奇数，则表示外部事务拥有该锁。

有关 sp_transactions 输出的详细信息，请参见第 181 页的“获取有关分布式事务的信息”。

获取有关分布式事务的信息

Adaptive Server 有一个系统表 master.dbo.systransactions，该表存储关于所有服务器事务的信息。systransactions 可标识每个事务并维护有关事务状态及其关联线程的信息。

新系统过程 sp_transactions 可转换 systransactions 和 syscoordinations 表中的信息，以显示活动事务的状态条件。

systransactions 中的事务标识

Adaptive Server 将事务名称存储在数据类型为 varchar(255) 的列中，以适应不同分布式事务协议所提供的事务名称的长度和格式。在以前的服务器版本中，该列的数据类型为 varchar(64)。例如，在 X/Open XA 协议中，为分布式事务分配的事务名称同时包含全局事务 ID (gtrid) 和分支限定符。在 Adaptive Server 中，该信息已合并在 systransactions 表的 xactname 列中。

systransactions.xactname 同时存储外部创建的分布式事务的名称以及本地服务器事务的名称。外部创建的分布式事务由 X/Open XA 事务管理器或 MSDTC 定义。定义本地事务的客户端可以在 varchar(255) 列的约束范围内用任何名称来命名这些事务。同样地，外部事务管理器可以使用多种不同的格式来命名分布式事务。

事务关键字

事务关键字存储在 systransactions 的 xactkey 列中，可用作服务器事务的唯一内部句柄。对于本地事务，即使事务名称对于服务器来说并不唯一，xactkey 也能确保可以区分各个事务。

从 Adaptive Server 12.0 版开始，所有系统表都引用 systransactions.xactkey 来唯一地标识事务。此规则仅有的例外是 sysprocesses 和 syslogshold 表，它们引用 systransactions.xactname 并将该值截断至 varchar(64) 的长度（对于 sysprocesses）和 varchar(67) 的长度（对于 syslogshold），以向后兼容 Adaptive Server 的以前版本。

使用 *sp_transactions* 查看活动事务

sp_transactions 过程可转换 systransactions 和 syscoordinations 中的信息，以提供有关活动事务的信息。如果不使用关键字，sp_transactions 将显示关于所有活动事务的信息：

```
                sp_transactions
xactkey                         type        coordinator starttime
 state              connection dbid   spid   loid
```

```
 failover                    srvname                          namelen
 xactname
 ----------------------------- ----------- -----------
 --------------------
 ----------------- ---------- ------ ------ -----------
 -------------------------- ------------------------------ -------
 ------------------------------------------------------------------
 0x00000b1700040000dd6821390001 Local       None        Jun 1 1999 3:47PM
 Begun             Attached        1   1      2
 Resident Tx               NULL                                 17
 $user_transaction
 0x00000b1700040000dd6821390001 Remote      ASTC        Jun 1 1999 3:47PM
 Begun             NA              0   8      0
 Resident Tx               caserv2                           108

 00000b1700040000dd6821390001-aa01f04ebb9a-00000b1700040000dd6821390001-
aa01f04ebb9a-caserv1-caserv1-0002
```

标识本地、远程和外部事务

“type”列说明事务是本地事务、远程事务还是外部事务。本地事务在本地服务器（运行 sp_transactions 的服务器）上执行。本地事务在“srvname”列中显示一个空值，因为该事务发生在当前服务器上。

对于远程事务，sp_transactions 将在“srvname”列下面列出执行该事务的服务器的名称。上面的 sp_transactions 输出显示了在名为“caserv2”的服务器上执行的远程事务。

外部事务是指由外部事务协调器（如 CICS、Encina 或其它 Adaptive Server 的“ASTC HANDLER”进程）来协调的事务。外部事务在“srvname”列中也显示一个空值。

标识事务协调器

“coordinator”列指示用来管理事务的方法或协议。在以上输出中，本地事务 $user_transaction 没有外部协调器。在 caserv2 上发生的远程事务有协调器值“ASTC”。这表示使用本机 Adaptive Server 协调服务来协调该事务，如第 174 页的“使用 Adaptive Server 协调服务”所述。

有关可能的协调器值的完整列表和说明，请参见《参考手册》中的 sp_transactions。

查看事务控制线程

spid 列显示附加到事务的进程的进程 ID（如果该事务已从其控制线程中分离，则显示为 0）。对于本地事务，spid 值表示在本地服务器上运行的进程 ID。对于远程事务，spid 表示在指定的远程服务器上运行的任务的进程 ID。以上输出显示了在远程服务器 caserv2 上运行的 spid 值 8。

理解事务状态信息

“state”列显示关于每个事务当前状态的信息。在任意给定时间，本地或外部事务可能处于执行命令、被中止、被提交，或其它状态。另外，分布式务可以处于就绪状态，也可以尝试完成或回退。

“connection”列显示有关事务连接状态的信息。该信息可用来确定事务当前是附加于一个进程，还是从一个进程中分离。为了响应来自事务管理器的请求，X/Open XA 环境中的事务可以从它们的启动进程中分离。

有关可能的协调器值的完整列表和说明，请参见《参考手册：过程》中的 sp_transactions。

将 sp_transactions 输出限制为特定状态

可以使用 sp_transactions 和 state 关键字来将输出限制为指定的事务状态。例如：

```
sp_transactions "state", "Prepared"
```

只显示就绪的分布式事务的信息。

事务故障切换信息

“failover”列显示在高可用性环境中运行的服务器的特殊信息。在高可用性环境中，如果原始服务器发生了严重故障，就可能将就绪事务传送给辅助同服务器。“failover”列可以显示三种可能的故障切换状态，以指示事务的执行方式和位置：

- 在正常操作条件下，不使用 Adaptive Server 高可用性功能的系统上将显示“Resident Tx”。“Resident Tx”表示事务已启动，并且正在一个主 Adaptive Server 上执行。
- 对辅助协同服务器进行故障切换后，将显示“Failed-over Tx”。“Failed-over Tx”表示事务原来在主服务器上启动并达到就绪状态，但被自动迁移到辅助协同服务器上（例如，因为主服务器发生了系统故障）。就绪事务的迁移对外部协调服务来说是透明的。

- 在对辅助协同服务器进行故障切换后，还显示“Tx by Failover-Conn”。“Tx by Failover-Conn”表示应用程序或客户端试图在主服务器上启动事务，但是主服务器因连接故障切换而不可用。在发生这种情况后，事务将在辅助协同务器上自动启动，并将该事务标记为“Tx by Failover-Conn”。

有关 Adaptive Server 故障切换功能的详细信息，请参见第 183 页的“事务故障切换信息”。

使用 sp_*transactions 确定提交节点和 gtrid*

如果将 sp_transactions 和 xid 关键字一起使用，则除了第 181 页的“使用 sp_transactions 查看活动事务”中所述的输出外，还将显示特定事务的提交节点、父节点和 gtrid。这种形式的 sp_transactions 要求指定一个特定的事务名称。例如：

```
sp_transactions "xid", "00000b1700040000dd6821390001-aa01f04ebb9a-
00000b1700040000dd6821390001-aa01f04ebb9a-caserv1-caserv1-0002"
xactkey                          type        coordinator starttime
 state             connection dbid   spid   loid
 failover                   srvname                        namelen
 xactname
 commit_node parent_node
 gtrid
 ------------------------------ ----------- -----------
 --------------------
 ----------------- ---------- ------ ------ -----------
 -------------------------- ------------------------------ -------
 ------------------------------------------------------------------
 -------------
 -------------
 -------------
 0x00000b2500080000dd6821960001 External    ASTC        Jun 1 1999 3:47PM
 Begun             Attached        1      8         139
 Resident Tx               NULL                              108

 00000b1700040000dd6821390001-aa01f04ebb9a-00000b1700040000dd6821390001-
aa01f04ebb9a-caserv1-caserv1-0002

 caserv1 caserv1
 00000b1700040000dd6821390001-aa01f04ebb9a
```

提交节点和父节点

对于由 Adaptive Server 来协调的分布式事务，“commit node”列将列出执行分布式事务的最顶层分支的服务器名称。该事务决定了事务所有分支的提交或回退状态。有关详细信息，请参见第 175 页的“层次事务协调”。

“parent node”列将列出启动事务的服务器的名称。在以上的 sp_transactions 输出中，“commit node”和“parent node”列都列出了相同的服务器 caserv1。这说明分布式事务在 caserv1 上开始，然后 caserv1 将该事务的分支传播到当前服务器。

全局事务 ID

“gtrid”列显示由 Adaptive Server 协调的分布式事务的全局事务 ID。事务分支是同一分布式事务的一部分，它们共享同一个 gtrid。可以将特定的 gtrid 与 sp_transactions gtrid 关键字一起使用，以确定在当前服务器上运行的其它事务分支的状态。这对于系统管理员很有用，因为他们必须确定应尝试提交还是回退分布式事务的一个特定分支。有关将 sp_transactions 和 gtrid 关键字一起使用的示例，请参见第 189 页的“确定 Adaptive Server 事务的提交状态”。

注释　对于由符合 X/Open XA 要求的事务管理器、MSDTC 或 SYB2PC 来协调的事务，gtrid 列将显示外部事务协调器提供的完整事务名称。

执行外部事务的步骤

在所有版本中，Adaptive Server 用来执行外部事务的步骤如下：

1　TM 启动一个 begin transaction。

2　TM 启动一个 attach transaction。

注释　TM 可能将步骤 1 和 2 一起执行。

3　应用程序执行 DML 命令。

4　TM 启动一个 detach transaction。

5　如有必要，重复步骤 2 到 4。

6　如果事务未回退，则 TM 启动一个 prepare transaction。

7　TM 启动一个 commit transaction 或 rollback transaction。

执行步骤 3 可能导致分布式事务回退。

由于在发出每个命令前都检查全局变量非常麻烦，许多用户应用程序根本不对其进行检查。在版本 15.0.3 以前的版本中，如果分布式事务已回退，Adaptive Server 允许用户应用程序继续发出 SQL 命令。这些命令将作为独立的事务在分布式事务外执行。应已包括在回退事务中的 SQL 命令可以独立于该事务提交，从而导致数据在事务上不一致。

在版本 15.0.3 和更高版本中，Adaptive Server 会自动阻止原打算在分布式事务内执行的 SQL 命令在该事务外执行。用户应用程序不再必须在发出每个命令检查全局变量；当某个事务被隐式中止时，将显示一条错误消息 (3953)，指出“由于外部事务已回退，无法执行该命令。”发出 detach transaction 命令时，此消息将消失。

要禁止 3953 错误消息并使 Adaptive Server 恢复以前的行为（即使 DTM 不处于活动状态，也执行 SQL 命令），请使用跟踪标志 -T3955 启动 Adaptive Server。

分布式事务的崩溃恢复过程

在崩溃恢复过程中，Adaptive Server 必须解决它发现的处于就绪状态的分布式事务。解决就绪事务所采用的方法取决于用来管理分布式事务的协调方法或协调协议。

注释 在常规数据库恢复过程中（使用 load database 或 load transaction 命令），将不执行以下崩溃恢复过程。如果 load database 或 load transaction 应用任何已就绪或不确定的事务，Adaptive Server 就会在使关联数据库联机之前中止这些事务。

由 MSDTC 协调的事务

使用 MSDTC 来协调的就绪事务根据主事务的提交状态来前进或回退。在恢复过程中，Adaptive Server 开始与 MSDTC 建立联系，以确定主事务的提交状态，然后相应地提交或回退就绪事务。如果无法与 MSDTC 取得联系，恢复过程就会等待，直到建立联系。在 Adaptive Server 与 MSDTC 建立联系之前，将不会执行进一步的恢复。

由 Adaptive Server 或 X/Open XA 协调的事务

在崩溃恢复过程中，Adaptive Server 还可能会遇到使用 Adaptive Server 事务协调服务或 X/Open XA 协议来协调的就绪事务。当遇到这些事务时，本地服务器必须等待进行协调的 Adaptive Server 或外部事务协调器来启动联系并指示就绪事务是应提交还回退。

为了加快恢复进程，Adaptive Server 将把所有这些事务恢复到故障发生前的状态。事务管理器用初始事务 ID 创建一个新事务，而锁管理将应用锁来保护初始事务所修改的数据。恢复的事务仍处于就绪状态，但是已被分离，没有线程与之关联。

一旦事务协调器与 Adaptive Server 建立联系，事务管理器就可提交或回退事务。

使用这种恢复机制，即使进行协调的 Adaptive Server 或外部事务管理器尚未尝试解决就绪事务，服务器也可以使数据库联机。由于就绪事务持有它在恢之前拥有的锁，所以其它客户端和事务可以恢复对本地数据进行的工作。一旦就绪事务与其协调器建立联系，即可将其提交或回退。

如果控制 Adaptive Server 或外部事务管理器不能完成事务，系统管理员则可以尝试着完成该事务，以释放事务锁和事务资源。有关详细信息，请参见第 187 页的“尝试完成事务”。

由 SYB2PC 协调的事务

使用 SYB2PC 协议来协调的就绪事务将根据主事务的提交状态来前进或回退。在恢复过程中，Adaptive Server 启动与提交服务的联系，以确定主事务的提交状态，然相应地提交或回退就绪事务。如果无法与提交服务联系，Adaptive Server 就不能使数据库联机。但是，Adaptive Server 将继续恢复系统中的其它数据库。

在 Adaptive Server 的早期版本中，这种恢复方法用于 SYB2PC 事务，而 Adaptive Server 12.5 和更高版本仍沿用这种方法。

尝试完成事务

Adaptive Server 包括 dbcc complete_xact 命令，以便于尝试完成事务。dbcc complete_xact 通过提交或回退事务的工作来处理事务，从而释放该事务占用的所有资源。

dbcc complete_xact 用于以下情况：只有系统管理员可以正确处理就绪事务；或系统管理员必须在不等待事务协调器的情况下处理事务。

例如，在第 175 页的图 8-2 中，如果所有远程 Adaptive Server 都已准备好它们的事务，但是与 ASE1 的网络连接已*永远丢失*，就可以考虑尝试完成事务。远程 Adaptive Server 将在 ASE1 的协调服务与之联系之前，使其事务保持就绪状态。在这种情况下，只有 ASE2、ASE3 和 ASE4 的系统管理员才能正确地解决就绪事务。在尝试完成 ASE3 中的就绪事务后，将释放事务和锁资源，并在 systransactions 中记录提交状态，以备将来由事务协调器使用。如果尝试完成 ASE2 中的事务，则同时会完成传播给 ASE4 的事务。

完成就绪事务

警告！尝试完成就绪事务时，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 或事协议的决定相矛盾。

在尝试完成事务之前，系统管理员应该尽量确定进行协调的 Adaptive Server 或事务协议是决定提交还是回退分布式事务（请参见第 189 页的“确定 Adaptive Server 事务的提交状态”）。

通过使用 dbcc complete_xact，系统管理员可强制 Adaptive Server 提交或回退分布式事务的分支。尝试完成就绪事务之后，Adaptive Server 将在 master.dbo.systransactions 中记录事务的提交状态，这样事务协调器（Adaptive Server、MSDTC 或 X/Open XA 事务管理器）就可以知道事务是已提交还是已回退。

Adaptive Server 将把尝试提交或中止事务的命令传播给它为事务分支所协调的任何参与者服务器。例如，如第 175 页的图 8-2 所示，如果您尝试提交 ASE2 上的事务，则 ASE2 会将命令传播到 ASE4，这样 ASE4 上的事务也会提交。

dbcc complete_xact 要求提供活动事务名称和预期的事务结果。例如，以下命令将尝试提交事务：

```
dbcc complete_xact "00000b1700040000dd6821390001-
aa01f04ebb9a-00000b1700040000dd6821390001-
aa01f04ebb9a-caserv1-caserv1-0002", "commit"
```

忘记尝试完成的事务

在系统管理员尝试完成就绪事务后，Adaptive Server 将在 master.dbo.systransactions 中保留事务提交状态的有关信息。保留该信息后，外部事务协调器就可以检测到尝试完成的事务。

如果外部协调器是另一个 Adaptive Server，则服务器将检查提交状态，如果尝试完成与分布式事务的提交状态发生冲突，则会记录警告消息。检查完提交状态后，进行协调的 Adaptive Server 将从 systransactions 中清除提交状态信息。

如果外部协调器是符合 X/Open XA 的事务管理器，那么当尝试完成与分布式事务发生冲突时，该事务管理器将不记录警告消息，但遵循 X/Open XA 的事务管理器将从 systransactions 中清除提交状态信息。

手工清除提交状态

dbcc forget_xact 从 systransactions 中清除尝试完成事务的提交状态。当系统管理员不希望协调服务知道事务已尝试完成时，或者当无法使用外部协调器来清除 systransactions 中的信息时，就可以使用这种方法。

请参见《参考手册：命令》中的 dbcc，了解有关使用 dbcc forget_xact 的详细信息。

完成未就绪的事务

dbcc complete_xact 还可用于回退由 Adaptive Server 协调、尚未达到就绪状态的事务。尝试回退未就绪的事务不会给分布式事务带来风险，因为协调服务器可以确定事务未准备好它的工作。在这些情况下，进行协调的 Adaptive Server 可以回退整个分布式事务以保持一致性。

当尝试回退未就绪的 Adaptive Server 事务时，Adaptive Server 将*不会*在 systransactions 中记录尝试回退，而是在屏幕上输出一条信息性消息并将其记录在服务器的错误日志中。

确定 Adaptive Server 事务的提交状态

如果要提交或回退的分布式事务分支由 Adaptive Server 来协调，则可以使用 sp_transactions 来确定分布式事务的提交状态。为此，请执行以下步骤。

注释　这些步骤不能用于由 X/Open XA 协议、MSDTC 或 SYB2PC 来协调的分布式事务。

---

1 在执行要完成的事务分支的服务器中，将 sp_transactions 和 xid 关键字一起使用来显示有关事务的信息。记录事务的提交节点和 gtrid。例如：

```
sp_transactions "xid", "00000b1700040000dd6821390001-aa01f04ebb9a-
00000b1700040000dd6821390001-aa01f04ebb9a-caserv1-caserv1-0002"
xactkey                          type        coordinator starttime
 state             connection dbid   spid   loid
 failover                  srvname                          namelen
 xactname
 commit_node parent_node
 gtrid
 ------------------------------ ----------- -----------
 --------------------
 ----------------- ---------- ------ ------ -----------
 -------------------------- ------------------------------ -------
 ------------------------------------------------------------------
 -------------
 -------------
 -------------
 0x00000b2500080000dd6821960001 External    ASTC       Jun 1 1999 3:47PM
 Begun             Attached        1      8         139
 Resident Tx               NULL                                108

 00000b1700040000dd6821390001-aa01f04ebb9a-
00000b1700040000dd6821390001-aa01f04ebb9a-caserv1-caserv1-0002

 caserv1 sfserv
 00000b1700040000dd6821390001-aa01f04ebb9a
```

在本例中，分布式事务的提交节点是“caserv1”，gtrid 是“00000b1700040000dd6821390001-aa01f04ebb9a”。

2 登录到提交节点所指定的服务器上。例如：

```
isql -Usa -Psa_password -Scaserv1
```

3 将 sp_transactions 和 gtrid 关键字一起使用来确定在步骤 1 中获得 gtrid 的分布式事务的提交状态：

```
sp_transactions "gtrid", "00000b1700040000dd6821390001-aa01f04ebb9a"
xactkey type coordinator starttime
state connection dbid   spid   loid
failover srvname namelen
xactname
commit_node
parent_node
------------------------------ ----------- -----------
```

```
--------------------
----------------- ---------- ------ ------ -----------
------------------------- ------------------------------ -------
 ------------------------------------------------------------------
 -------------
 -------------
 0x00000b1700040000dd6821390001 Local       None        Jun 1 1999 3:47PM
 Committed Attached        1      1           2
 Resident Tx                NULL                                 17
 $user_transaction

 caserv1
 caserv1
```

在本例中，带有指定 gtrid 的本地事务已提交，如“state”列中所示。系统管理员应该尝试*提交*在步骤 1 中检查的就绪事务。

4　使用具有系统管理员特权的帐号登录到执行要完成的事务分支的服务器：

```
isql -Usa -Psa_password -Ssfserv
```

5　使用 dbcc complete_xact 提交事务。在本例中，系统管理员应使用 commit 关键字来保持与分布式事务的一致性：

```
dbcc complete_xact "00000b1700040000dd6821390001-
aa01f04ebb9a-00000b1700040000dd6821390001-
aa01f04ebb9a-caserv1-caserv1-0002", "commit"
```

编程与配置注意事项

本节介绍在排除故障时应考虑的配置选项。

分布式事务中的 DDL 行为

如果由外部事务管理器通过 X/Open XA 协议或其它 Adaptive Server 的 Adaptive Server 事务协调服务来协调某事务，则在该事务中不允许使用 DDL 命令。即使在启用了数据库选项 ddl in tran 的情况下，这一行为仍然适用。

外部事务中的 Adaptive Server 隐式回退

如果外部事务中出现错误（例如，死锁、更新触发器中止等），Adaptive Server 可能会中止该外部事务。

虽然 Adaptive Server 会发送表示发生故障的错误消息，但应用程序并不总是检查这些消息，特别是对于简单插入（例如，它们可能不知道由 DBA 添加的触发器）。错误消息中可能并不总是明显说明 XA 事务已结束。

如果 Adaptive Server 中止一个外部事务并引发 SQLException，您可以发出 `select @@trancount`。如果 *@@trancount* 的值为零，则表示 DTM 事务已中止。

应用程序应调用事务管理器（通常为应用程序服务器），通知它该事务已中止。如果忽略错误消息，后续更新可能在 DTM 事务上下文外（例如，本地务）进行。您可以记录错误消息并检查 *@@transtate* 或 *@@trancount* 以检验是否进行了更新。

下面介绍了一个导致 Adaptive Server 回退外部事务的触发器。Insert 语句中包含可能失败的触发器。如果 Adaptive Server 无法发出 insert，则更新将运行 ut.commit 函数。

本例（伪代码形式）假设您正在运行一个 JTA/XA UserTransaction：

```
try {

insert into table values (xx....)
update table
ut.commit();

} catch (SQLException sqe) {

if this is a known error then process
else
select @@trancount into count
if count == 0
then ut.rollback() }
```

如果不包括回退函数，则附加更新将在 JTA/XA 事务外进行。

第 9 章

创建和使用段

主题	页码
Adaptive Server 段	193
Adaptive Server 如何使用段	195
创建段	198
更改段的范围	198
向段指派数据库对象	200
删除段	205
获取有关段的信息	205
段和系统表	208
段的教程	209

请参见 《性能和调优系列：物理数据库调优》中的第 1 章 “控制物理数据放置”，了解有关段如何提高性能的信息。

Adaptive Server 段

段是指向一台或多台数据库设备的标签。在 create table 和 create index 命令中使用段名来将表或索引放置到特定数据库设备上。使用段可以提高 Adaptive Server 性能，并增强系统管理员或数据库所有者对数据库对象的位置大小和空间使用情况的控制。

可以在数据库中创建段以描述分配给数据库的数据库设备。每个 Adaptive Server 数据库可以包含多达 32 个段，其中包括系统定义的段 （请参见第 194 页的 “系统定义的段”）。在指派段名之前，必须使用 disk init 初始化数据库设备，然后使用 create database 或 alter database 使数据库可以使用它们。

系统定义的段

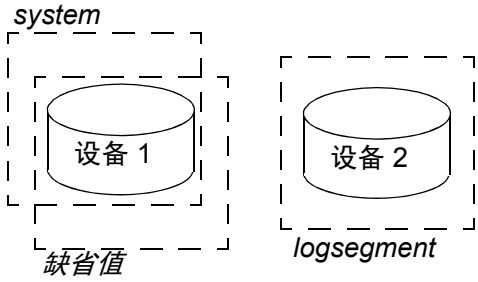
创建数据库时， Adaptive Server 在数据库中创建三个段，如表 9-1 所示。

表 9-1：系统定义的段

段	快捷键
system	存储数据库的系统表
logsegment	存储数据库的事务日志
default	存储所有其它数据库对象 — 除非创建了其它段，并使用 create table...on *segment_name* 或 create index...on *segment_name* 将表或索引存储在新段上

如果在单个数据库设备上创建数据库，将在同一台设备上创建 system、default 和 logsegment 段。如果使用 log on 子句将事务日志放置在单独的设备上，则这些段将如图 9-1 所示。

图 9-1：系统定义的段



尽管可以增加和删除用户定义的段，但不能从数据库中删除缺省段、系统段或日志段。对于以下每种系统定义段类型，数据库必须至少有一个相应类的段：system、 logsegment 和 default。

以下是用于管理段的命令和系统过程：

- sp_addsegment — 在数据库中定义段。
- create table 和 create index — 在段上创建数据库对象。
- sp_dropsegment — 从数据库删除段或从段的范围内删除单个设备。
- sp_extendsegment — 向现有段增加设备。
- sp_placeobject — 向特定段为表或索引分区指派未来空间分配。
- sp_helpsegment — 显示特定段上数据库或数据的段分配情况。
- sp_helpdb — 显示每个数据库设备上的段。有关示例，请参见第 6 章 “创建和管理用户数据库”。

- sp_help — 显示表的信息，包括表所在的段。
- sp_helpindex — 显示表索引的信息，包括索引所在的段。

Adaptive Server 如何使用段

向数据库增加新设备时， Adaptive Server 将新设备放到缺省空间缓冲池 （数据库的 default 和 system 段）中。这样就增加了数据库的可用空间总量，但并不能确定哪些对象将占用新空间。任何表或索引都可能增长到填满整个空间缓冲池，使关键表没有扩展的空间。还可能将几个频繁使用的表和索引放到缺省空间缓冲池中的单个物理设备上，从而导致 I/O 性能下降。

在段上创建一个对象后，该对象可以使用段中可用的所有数据库设备，但不包括其它设备。可以使用段来控制可用于个别对象的空间。

以下章节介绍如何使用段来控制磁盘空间的使用以及如何提高性能。

控制空间使用

如果向段指派不重要的对象，则那些对象的增长不能超出段的设备的可用空间。反之，如果向段指派重要的表，并且段的设备对于其它段是不可用的则没有其它对象争用该表的空间。

当段中的设备已满时，可以根据需要扩展段以包括其它设备或设备片段。段还允许您在特定数据库段上的空间变小时使用阈值提出警告。

如果为数据创建其它段，可以为每个段创建新的阈值过程。请参见第 17 章 “使用阈值管理可用空间”。

改善性能

在大型的、多数据库或多驱动器的 Adaptive Server 环境中，通过仔细地分配数据库的空间和将数据库对象放置在物理设备上来提高系统性能。理想情况，每个数据库都排它使用数据库设备，即它不与其它数据库共享物理磁盘。在大多数情况下，通过将频繁使用的数据库对象放在专用物理磁盘上或跨几个物理磁盘拆分大型表可以提高性能。

将表、索引和日志分开

通常，将表放在一个物理设备上，将其非聚簇索引放在第二个物理设备上，将事务日志放在第三个物理设备上，就可以提高性能。使用单独的物理设（磁盘控制器）可缩短读写磁盘所需的时间。如果您不能以这种方法使用全部设备，至少要将所有非聚簇索引限制到专用的物理设备上。

create database（或 sp_logdevice）的 log on 扩展将事务日志放在单独的物理磁盘上。使用段将表和索引放在特定的物理设备上。请参见第 200 页的 “向段指派数据库对象”。

拆分表

为了提高表的整体读性能，可以跨单独的磁盘控制器上的多个设备拆分大型的、频繁使用的表。当大型表放在多个设备上时，很可能在不同磁盘上同发生小的读操作。

可以使用以下不同的方法跨不同的设备拆分表，每种方法都需要使用段：

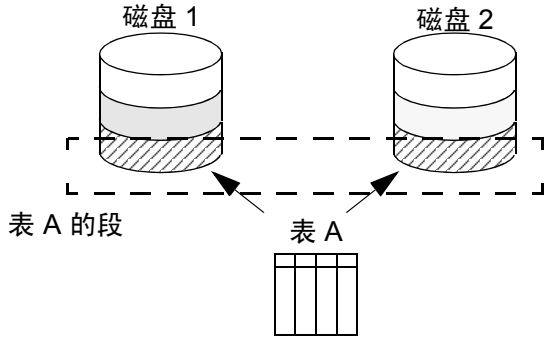
- 使用表分区。
- 如果表有聚簇索引，使用部分装载。
- 如果表包含 text 或 image 数据类型，则将文本链与其它数据分开。

对表进行分区

对表分区将为表创建多个页链，并在表的段中的所有设备上分配那些页链。对表分区将提高插入性能和读性能，因为多个页链可用于插入。

图 9-2 显示了一个表被拆分放在段的两个设备上。

图 9-2：跨物理设备对表分区



在对表分区之前，必须在包含指定数量的设备的段上创建表。请参见《性能和调优系列：物理数据库调优》中的第 1 章“控制物理数据放置”，了解有关使用 alter table 对表分区的信息。

部分装载

要拆分带聚簇索引的表，可使用带有多个 load 命令的 sp_placeobject 来将表的不同部分装载到不同的段上。这种方法执行和维护都很困难，但它实现了跨越物理设备拆分表及其聚簇索引。请参见第 201 页的“在段上放置已存在的对象”。

分隔 text 和 image 列

Adaptive Server 在单独的数据页链上存储 text 和 image 列的数据。缺省情况下，此文本链与表的其它数据一起放置在相同的段上。由于读取文本列需要文本指针在基表中进行读操作，并需要在单独文本链中的文本页上进行其它读操作，因此将文本链和基表数据放置到单独物理设备上可提高性能。请参见第 204 页的“把文本页放在单独的设备上”。

将表移到另一设备

也可通过 create clustered index 命令使用段来将表从一个设备移动到另一设备。聚簇索引（其中索引的底部或叶级包含实际的数据）位于与表相同的段上。因此，通过删除其聚簇索引（如果有），然后在所需的段上创建或重新创建聚簇索引，可完全移动表。请参见第 204 页的“对段创建聚簇索引”。

创建段

在数据库中创建段：

- 使用 disk init 初始化物理设备。
- 对 create database 或 alter database 使用 on 子句，使数据库可使用数据库设备。这样可自动将新设备添加到数据库的 default 和 system 段。

当数据库设备存在并可供数据库使用后，可以使用 sp_addsegment 在数据库中定义段。

请参见 《参考手册：过程》。

此语句在数据库设备 mydisk1 上创建段 seg_mydisk1：

```
sp_addsegment seg_mydisk1, mydata, mydisk1
```

更改段的范围

使用段时，还必须管理它们的范围 — 每个段指向的数据库设备数。利用它您可以：

- 通过使段指向一个附加设备或多个设备来扩展段的范围，或
- 通过使段指向较少设备来减小段的范围。

扩展段的范围

如果指派到段的一个或多个对象用尽了空间，则可能需要扩展段。sp_extendsegment 向现有段添加数据库设备。

扩展段之前：

- 数据库设备必须列在 sysdevices 中，
- 该数据库设备对于要扩展的数据库必须可用，且
- 该段名必须存在于当前数据库中。

下面的示例将数据库设备 pubs_dev2 添加到名为 bigseg 的现有段中：

```
sp_extendsegment bigseg, pubs2, pubs_dev2
```

若要扩展数据库中的 default 段，请将单词 “default” 放在引号中：

```
sp_extendsegment "default", mydata, newdevice
```

请参见 《参考手册：过程》。

自动扩展段的范围

如果使用 alter database 向数据库的新设备上增加空间，将扩展 system 和 default 段，以包括新空间。这样，在每次向数据库添加新设备时，system 和 default 段的范围都将得到扩展。

如果使用 alter database 在现有数据库设备上指派附加空间，则映射到现有设备的所有段都被扩展，以包括新的设备片段。例如，假定初始化了一个名为 newdev 的 4MB 设备，为 mydata 分配了其中的 2MB，为 testseg 段分配了 2MB：

```
alter database mydata on newdev = "2M"
sp_addsegment testseg, mydata, newdev
```

如果以后将 mydata 更改成使用 newdev 上的剩余空间，则剩余空间片段会自动映射到 testseg 段：

```
alter database mydata on newdev = "2M"
```

请参见第 209 页的 “段的教程”。

减小段的范围

如果段中包括您希望为其它段单独保留的数据库设备，则可能需要减小段的范围。例如，如果添加一个表专用的新数据库设备，需要减小 default 和 system 段的范围，以便它们不再指向新设备。

使用 sp_dropsegment 可从段中删除单个数据库设备，以减小段的范围。

sp_dropsegment 只从段所跨的设备范围中删除给定的 device。还可以使用 sp_dropsegment 从数据库中删除整个段，如第 205 页的 “删除段”中所述。

下面的示例是从 bigseg 的范围中删除数据库设备 pubs_dev2：

```
sp_dropsegment bigseg, pubs2, pubs_dev2
```

请参见 《参考手册：过程》。

向段指派数据库对象

通过向用户定义的段指派新的或现有的数据库对象，可以：

- 将新对象限制到一个或多个数据库设备
- 在单独的设备上放置表及其索引以提高性能
- 在多个数据库设备上拆分现有对象

在段上创建新对象

要在段上放置新对象，应首先创建新段。可能还希望更改此段（或其它段）的范围，以便它只指向所需的数据库设备。在向数据库添加新数据库设备，它自动添加到 default 和 system 段的范围中。

请参见第 204 页的“对段创建聚簇索引”。

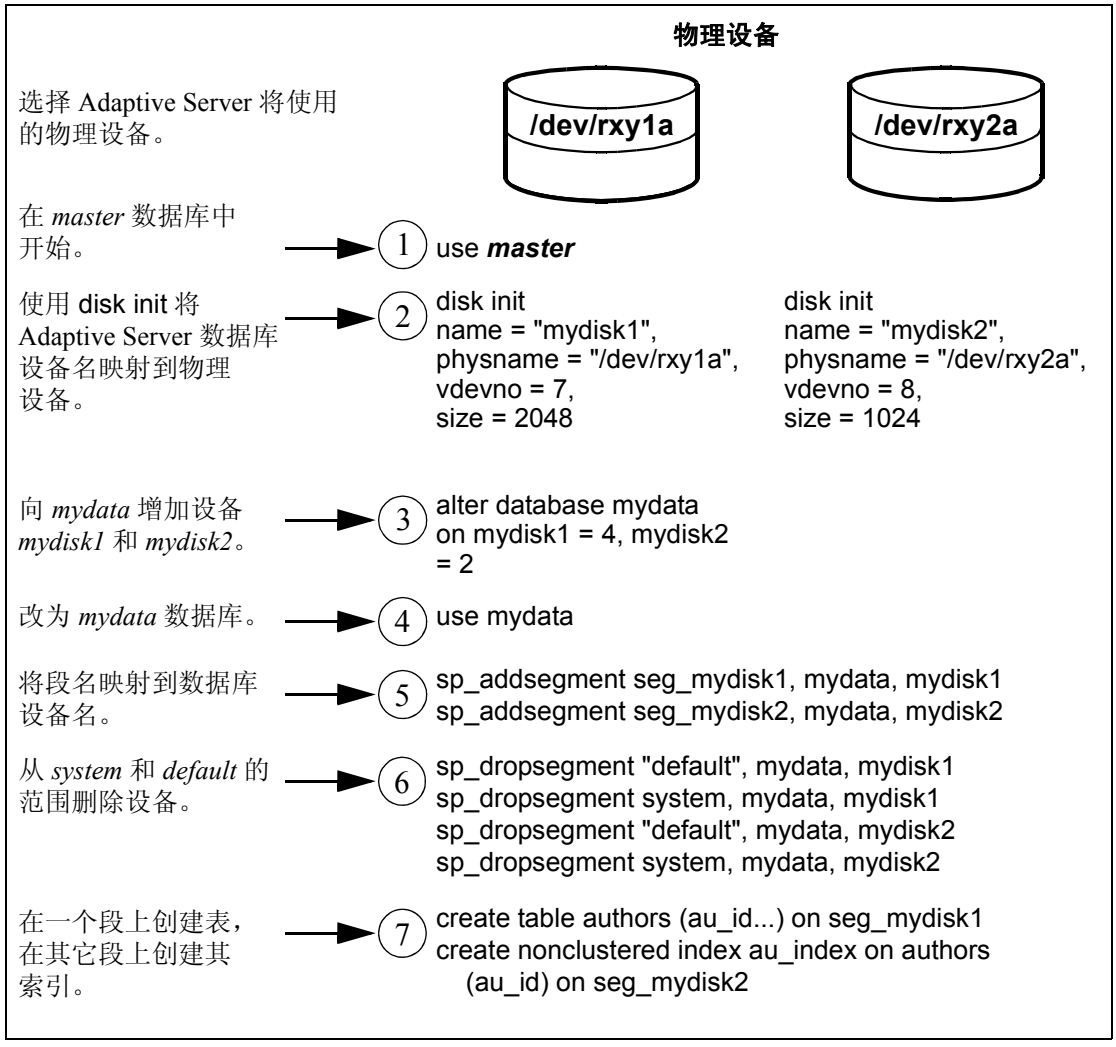
在当前数据库中定义了段之后，使用 create table 或 create index（带有可选的 on *segment_name* 子句）在段上创建对象。

请参见《参考手册：过程》。

示例：在单独的段上创建表和索引

图 9-3 总结了用于在使用 2K 逻辑页大小的服务器上的特定物理磁盘上创建表和索引的 Transact-SQL 命令的次序。

图 9-3：使用段在特定设备上创建对象



在段上放置已存在的对象

sp_placeobject 不从其已分配的段中删除对象。然而，它使该对象的所有未来磁盘分配在它指定的新段上进行。

例如，为了在 bigseg 上对 mytab 表进行所有后续磁盘分配，请使用：

```
sp_placeobject bigseg, mytab
```

sp_placeobject 不将对象从一个数据库设备移动到另一个设备上。在第一台设备上分配的所有页仍保持已分配状态；写入第一台设备的所有数据仍保留在该设备上。 sp_placeobject 只影响后续空间分配。

如果在使用 sp_placeobject 之后执行 dbcc checkalloc，将为跨段拆分的每个对象显示以下消息：

```
Extent not within segment:Object object_name, indid
index_id includes extents on allocation page
page_number which is not in segment segment_name.
```

可以忽略此消息。

请参见 《参考手册：过程》。

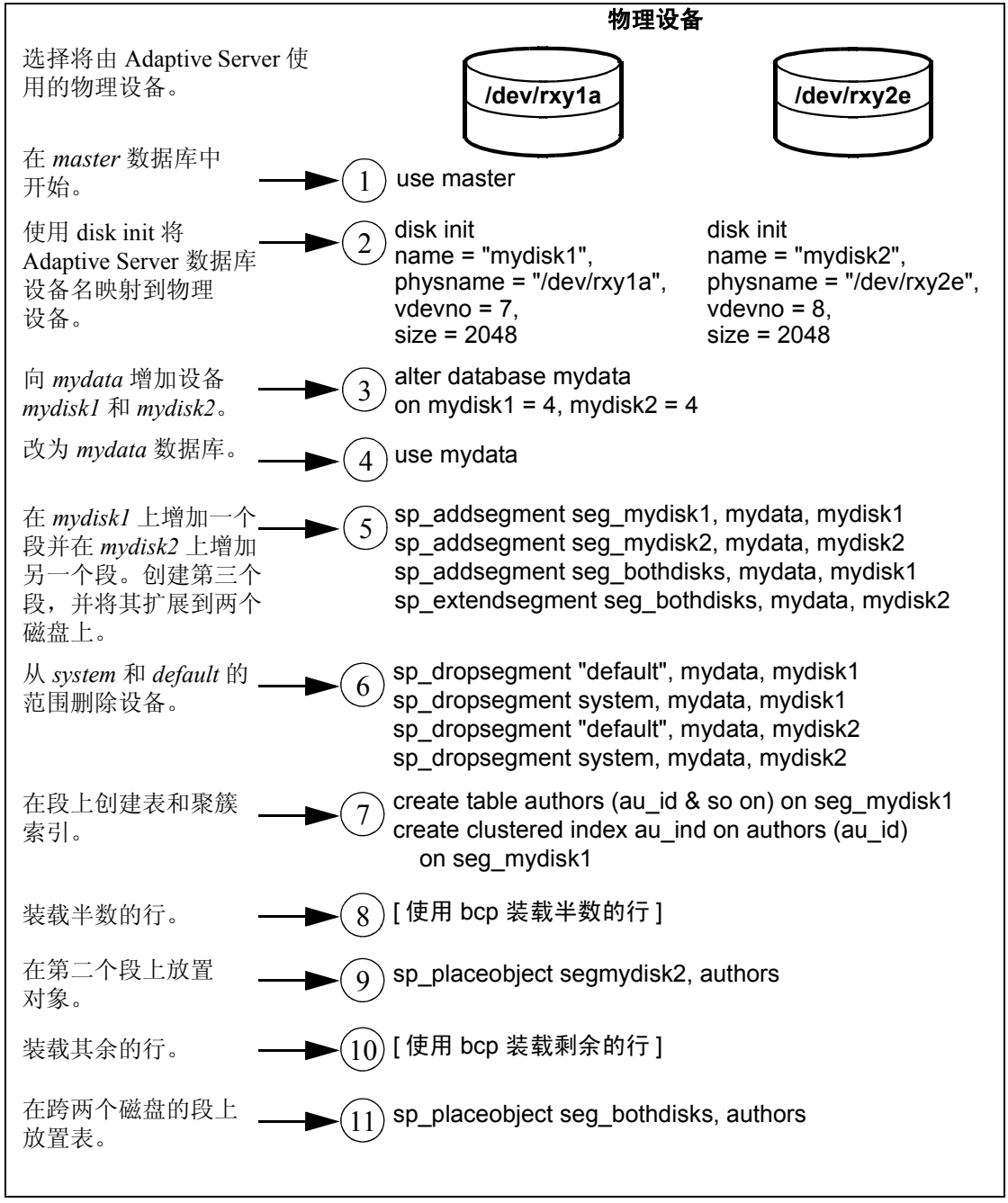
示例：跨多个物理设备拆分表及其聚簇索引

跨多个段 （分别位于单独的磁盘控制器上）拆分大型表可以改进高容量多用户应用程序的性能。

步骤的顺序非常重要；特别是应该在将表放置在第二个段上之前先创建聚簇索引。

图 9-4 总结了如何在使用 2K 逻辑页大小的服务器上跨两个段拆分表：

图 9-4：跨两个段拆分大型表



如果频繁更新表，则磁盘分配的平衡可能会随时间而改变。为确保保持速度优势，可能需要删除并重新创建表。

把文本页放在单独的设备上

创建带有 text 或 image 列的表时，数据存储在单独的文本页链上。具有 text 或 image 列的表在 sysindexes 中有一个额外条目用于文本链，名称列设置为前面带有字母 “t” 的表名称，且 indid 为 255。可以使用 sp_placeobject 在单独的设备上存储文本链 （需要给出表名和 sysindexes 中的文本链的名称）：

```
sp_placeobject textseg, "mytab.tmytab"
```

注释 缺省情况下，文本页链与其表放在同一段上。执行 sp_placeobject 之后，以前写到旧设备上的页仍将保持已分配状态，但将在新段上进行所有新分配。

如果要将文本页放在特定的段上，应首先在该段上创建表 （分配该段上的初始扩充），然后在表上创建聚簇索引，将剩余的数据移动到该段上。

对段创建聚簇索引

聚簇索引的底部 （或叶级）包含数据。因而，表和相应的聚簇索引在同一段上。如果在一个段上创建表，而其聚簇索引在另一个段上，表将迁移到创聚簇索引的段上。这提供了快速和方便的方法来将表移动到数据库中的其它设备上。

请参见 《参考手册：命令》中的 create index。

以下示例使用 new_space 段上的表创建聚簇索引，但没有指定段名 （有关创建此表的步骤，请参见第 209 页的 “段的教程”）：

```
create clustered index mytabl_cix
    on mytabl(c1)
sp_helpsegment new_space


segment     name           status
-------     ------------- ------
      3     new_space           0
```

```
device            size       free_pages
----------------- ------     ----------
newdevice         3.0MB            1523

total_size     total_pages     free_pages  used_pages   reserved_pages
-------------- -----------     ----------- -----------   --------------
3.0MB                 1536            1530           6                0
```

如果已将表放置在段上，并且必须创建聚簇索引，请使用 on *segment_name* 子句，否则表将迁移到 default 段上。

删除段

当使用只带有段名和数据库名的 sp_dropsegment 时，就可从数据库删除指定的段。然而，只要数据库对象仍然被指派到段，就不能删除它。必须将对象指派到另一个段或首先删除对象，然后再删除段。

不能从数据库完全删除缺省段、系统段或日志段。数据库必须至少有一个缺省段、系统段和日志段。但是，可以减小这些段的范围。请参见第 199 页的 “减小段的范围”。

注释　删除段就是将其名称从数据库中段的列表中删除，但不从该数据库的分配中删除数据库设备，也不从设备中删除对象。如果从一台数据库设备中删除了所有段，空间仍分配给该数据库，但不能用于数据库对象。 dbcc checkcatalog 报告 “Missing segment in Sysusages segmap”。若要使设备可供数据库使用，请使用 sp_extendsegment 将设备映射到数据库的缺省段：

```
sp_extendsegment "default", dbname, devname
```

请参见 《参考手册：过程》。

获取有关段的信息

这些系统过程提供了关于段的信息：

- sp_helpsegment — 列出数据库中的段，或显示数据库中特定段的信息。

- sp_helpdb — 显示数据库中设备和段之间的关系的信息。
- sp_help 和 sp_helpindex — 显示表和索引的信息，包括为其指派对象的段。

sp_helpsegment

sp_helpsegment，当不带参数使用时，显示关于在其中执行此命令的数据库中所有段的信息：

```
sp_helpsegment
segment name                          status
------- ----------------------------- ------
      0 system                             0
      1 default                            1
      2 logsegment                         0
      3 seg1                               0
      4 seg2                               0
```

指定段名作为参数，可以获得有关特定段的信息。请求有关 default 段的信息时，请使用引号：

```
sp_helpsegment "default"
```

下例显示有关 seg1 的信息：

```
sp_helpsegment seg1

segment name                           status
------- ------------------------------ ------
      4 seg1                                0

device                 size              free_pages
---------------------- ----------------- -----------
user_data10            15.0MB                   6440
user_data11            15.0MB                   6440
user_data12            15.0MB                   6440

table_name             index_name             indid
---------------------- ---------------------- ------
customer               customer                    0

total_size      total_pages free_pages  used_pages
--------------- ----------- ----------- -----------
45.0MB                23040       19320        3720
```

sp_helpdb

在数据库中执行 sp_helpdb 并指定数据库名称可以查看有关数据库中段的信息。

例如：

```
sp_helpdb pubs2

name      db_size    owner     dbid created        status
--------- ---------- --------- ---- -------------- --------------
pubs2     20.0 MB     sa           4 Apr 25, 2005   select
     into/bulkcopy/pllsort, trunc log on chkpt, mixed log and data

device_fragments    size          usage         created      free kbytes
------------------- ------------- ------------- ----------   ------------
master               10.0MB        data and log  Apr 13 2005          1792
pubs_2_dev          10.0MB        data and log  Apr 13 2005         9888

device                   segment
------------------------ ----------------------
master                   default
master                   logsegment
master                   system
pubs_2_dev               default
pubs_2_dev               logsegment
pubs_2_dev               system
pubs_2_dev               seg1
pubs_2_dev               seg2
```

sp_help 和 *sp_helpindex*

在数据库中执行 sp_help 和 sp_helpindex 并指定表名可以查看有关哪个段存储表或其索引的信息。

例如：

```
    sp_helpindex authors
index_name  index_keys  index_description  index_max_rows_per_page
     index_fillfactor  index_reservepagegap  index_created
     index_local
----------  ----------  -----------------  ------------------------
     ----------------  --------------------  -------------
     -----------
auidind     au_id       clustered,unique                          0
                    0                    0   Apr26 2005  4:04PM
     Global Index
aunwind    au_lname,au_fname  nonclustered,unique                0
                    0                    0   Apr26 2005  4:04PM
     Global Index
(2 rows affected)
index_ptn_name        index_ptn_seg
--------------------  --------------
auidind_400001425     default
aunmind_400001425     default
```

段和系统表

三个系统表存储有关段的信息：master..sysusages 以及用户数据库中的两个系统表 sysindexes 和 syssegments。 sp_helpsegment 使用这些表并在 sysdevices 中查找数据库设备名。

使用 create database 或 alter database 为数据库分配设备时，Adaptive Server 在 master..sysusages 中添加一行。sysusages 中的 segmap 列在数据库中为每个设备提供段的位图。

create database 还在用户数据库中创建 syssegments 表，该表具有以下缺省条目：

```
segment name            status
------- --------------- ------
      0 system                0
      1 default               1
      2 logsegment            0
```

执行 sp_addsegment：

- 向用户数据库中的 syssegments 表增加新行，并且
- 更新 master..sysusages 中的 segmap。

创建表或索引分区时，Adaptive Server 会在 sysindexes 中添加一个新行。该表中的 segment 列存储段号，显示服务器将在什么位置为对象分配新空间。创建对象时如果不指定段名，则它放在 default 段上；否则，它放在指定的段上。

如果创建了包含 text 或 image 列的表，则还为文本页的链接列表向 sysindexes 增加第二行；缺省情况下，文本页链存储在与表相同的段上。第 209 页的 “段的教程”中包含使用 sp_placeobject 将文本链放到其自己的段上的示例。

syssegments 中的 name 用于 create table 和 create index 语句中。status 列表明哪个段是缺省段。

注释　有关 segmap 列和管理存储的系统表的详细信息，请参见第 147 页的 “管理空间分配的系统表”。

段的教程

以下教程说明了如何创建用户段以及如何从设备删除所有其它段映射。本章中的示例假定服务器使用 2K 逻辑页大小。

当使用段和设备时，请记住：

- 如果按片段指派空间，则每个片段都将在 sysusages 中有一个条目。
- 当向数据库指派某一设备的其它片段时，所有映射到现有片段的段都将映射到新的片段上。

- 如果使用 alter database 向数据库的新设备上增加空间，则 system 和 default 段自动映射到新空间。

本教程从创建新数据库开始，为数据库对象创建一个设备，为事务日志创建另一个设备：

```
create database mydata on bigdevice = "5M"
     log on logdev = "4M"
```

现在，如果运行 use mydata，然后运行 sp_helpdb，将显示：

```
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata       9.0 MB sa             5 May 27, 2005  no options set

device_fragments  size    usage      created              free kbytes
----------------  ------  ---------- -----------          -----------
bigdevice         5.0 MB  data only  May 25 2005 3:42PM   3650
logdev            4.0 MB  log only   May 25 2005 3:42PM   not applicable
------------------------------------------------------------------------------
log only free kbytes = 4078
device                 segment
---------------------- ----------------------
bigdevice              default
bigdevice              system
logdev                 logsegment
(return status = 0)
```

就像所有新创建的数据库一样，mydata 具有名为 default、system 和 logsegment 的段。因为 create database 使用了 log on，所以 logsegment 映射到它自己的设备 (logdev) 上， default 和 system 段都映射到 bigdevice 上。

如果为 mydata 增加同一数据库设备上的空间，并再次运行 sp_helpdb，将看到所增加片段的条目：

```
use master
alter database mydata on bigdevice = "2M"
    log on logdev = "1M"
use mydata
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created       status
---------- -------- --------- ------ ------------  ---------------
mydata     12.0 MB  sa             4 May 25, 2005  no options set
```

```
device_fragments  size     usage      created                 free kbytes
----------------- -------- ---------- ----------------------- -----------
bigdevice         5.0 MB   data only  May 25 2005 3:42PM             2048
logdev            4.0 MB   data only  May 25 2005 3:42PM     not applicable
data only         2.0 MB   log only   May 25 2005 3:55PM             2040
log only          1.0 MB   log only   May 25 2005 3:55PM     not applicable
------------------------------------------------------------------------------
log only free kybytes = 5098
device                  segment
----------------------- ----------------------
bigdevice               default
bigdevice               system
logdev                  logsegment
```

应始终为日志空间增加日志空间，为数据空间增加数据空间。如果尝试将已用于数据的段分配给日志（或相反），Adaptive Server 将提示您使用 with override。请记住，段被映射到整个设备上，不是只映射到空间片段上。如果在某一设备上更改了任何段指派，就对所有片段都做了改动。

以下示例分配 mydata 尚未使用的新数据库设备：

```
use master
alter database mydata on newdevice = 3
use mydata
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata      15.0 MB sa             5 May 25, 2005  no options set

device_fragments   size      usage       created                 free kbytes
------------------ --------- ----------- ----------------------- --------------
bigdevice          5.0 MB    data only   May 25 2005 3:42PM            3650
logdev             4.0 MB    log only    May 25 2005 3:42PM  not applicable
bigdevice          2.0 MB    data only   May 25 2005 3:55PM            2040
logdev             1.0 MB    log only    May 25 2005 3:55PM  not applicable
newdevice          3.0 MB    data only   May 26 2005 11:59AM           3060
------------------------------------------------------------------------------
log only free kbytes = 5098
device           segment
---------------- ----------
bigdevice        default
bigdevice        system
logdev           logsegment
newdevice        default
newdevice        system
```

以下示例在 newdevice 上创建一个名为 new_space 的段：

```
sp_addsegment new_space, mydata, newdevice
```

下面是 sp_helpdb 报告的一部分，其中列出了段映射：

```
device                          segment
------------------------------ ------------------
bigdevice                       default
bigdevice                       system
logdev                          logsegment
newdevice                       default
newdevice                       new_space
newdevice                       system
```

default 和 system 段仍然映射到 newdevice。如果计划使用 new_space 存储用户表或索引以便提高性能，并且希望确保缺省情况下其它用户对象没有存储在设备上，则使用 sp_dropsegment 减小 default 和 system 的范围：

```
sp_dropsegment system, mydata, newdevice
sp_dropsegment "default", mydata, newdevice
```

必须在 “default”前后加上引号；它是 Transact-SQL 保留字。

下面是 sp_helpdb 报告的一部分，它显示了段的映射：

```
device                          segment
------------------------------ --------------------
bigdevice                       default
bigdevice                       system
logdev                          logsegment
newdevice                       new_space
```

现在只有 new_space 映射到 newdevice。创建对象的用户可以使用 on new_space 在对应于该段的设备上放置表或索引。因为 default 段没有指向该数据库设备，如果创建表和索引的用户不使用 on 子句，则不会将表或索引放在您特别准备的设备上。

如果再次在 newdevice 上使用 alter database，则新空间片段会获取与该设备的现有段相同的段映射 （即，仅 new_space 段）。

此时，如果使用 create table 并指定 new_space 作为要使用的段，运行 sp_helpsegment 将得到如下结果：

```
create table mytabl (c1 int, c2 datetime)
    on new_space

sp_helpsegment new_space

segment    name                           status
------- ------------------------------    ------
      3    new_space                           0

device           size        free_pages
---------------- ----------- -----------
newdevice        3.0MB              1523

Objects on segment  'new_space' :

table_name        index_name         indid        partition_name
----------------- ------------------ --------     ----------------
mytabl            mytabl                    0     mytabl_400001425

Objects currently bound to segment  'new_space' :

table_name      index_name     indid

total_size    total_pages    free_pages  used_pages   reserved_pages
------------- -----------    ----------- -----------   --------------
3.0MB               1536           1523          13                0
```

第 10 章

使用 *reorg* 命令

对表的更新活动可能会最终导致空间利用不充分和性能降低；可以使用 reorg 命令重组表空间的使用并提高性能。

reorg 命令及其参数

在以下情况下，reorg 命令非常有用：

- 大量转移的行导致读取操作过程中有额外的 I/O。
- 插入和串行化读取较慢，因为它们遇到的页有非连续的可用空间必须回收。
- 大 I/O 操作较慢，因为数据页和索引页的集群比低。
- 已经使用 sp_chgattribute 更改了空间管理设置（reservepagegap、fillfactor 或 exp_row_size），而且更改不仅仅应用于后续更新，还应用于表中所有已存在的行和页。

reorg 命令包含四个参数，它们用于执行不同类型和级别的重组：

- reorg forwarded_rows 撤消行转移操作。
- reorg reclaim_space 回收因删除操作和行缩短更新操作而产生的页上剩余的未用空间。
- reorg compact 既回收空间又撤消行转移。

- reorg rebuild 撤消行转移并回收未使用的页空间，此外还执行以下操作：
  - 重写所有行以便与表的聚簇索引一致（如果有）
  - 重写数据和索引分区的空间。
  - 使用个别分区。
  - 向数据页写入行，以便与通过 sp_chgattribute 对空间管理设置所做的更改保持一致。
  - 删除并重新创建属于该表的所有索引。

运行 reorg rebuild 之前，请考虑以下事项：

- reorg rebuild 在其整个过程中持有排它表锁。对于大表，这可能会是一个相当长的时间。但是，reorg rebuild 可以完成删除并重新创建聚簇索引所需执行的全部操作，而且所需时间更短。此外，reorg rebuild 使用表的所有当前空间管理设置来重新创建表。删除并重新创建索引时不使用 reservepagegap 的空间管理设置。
- 多数情况下，reorg rebuild 要求得到与重新创建的表及其索引大小相等的额外磁盘空间。

有以下限制：

- 用该命令指定的表（如果有）必须使用数据行锁定或数据页锁定方案。
- 只有系统管理员或对象所有者才能发出 reorg 命令。
- 不能在事务内发出 reorg 命令。

使用 *optdiag* 实用程序评估对 *reorg* 的需求

若要评估对运行 reorg 的需求，请使用 systabstats 表和 optdiag 实用程序的统计信息。systabstats 包含有关表空间使用情况的统计信息，而 optdiag 基于 systabstats 和 sysstatistics 表中的统计信息生成报告。

有关 systabstats 表的信息，请参见《性能和调优系列：物理数据库调优》中的第 2 章“统计信息表和使用 optdiag 显示统计信息”。有关 optdiag 的信息，请参见《实用程序指南》。

将转移的行移动到主页

如果更新使一行变得太长以致于当前页不能完全容纳它，则该行被转移到另一页。在该行的原始页（行的主页）上保留对该行的引用，所有对转移行访问都要通过这种引用。所以，总是需要通过两次页访问才能到达一个转移的行。如果扫描需要读取大量转移的页，则由于额外的页访问而引起的 I/O 会使性能降低。

reorg forwarded_rows 命令撤消行转移的方式为：将转移行移回到它自己的主页（如果有足够的空间），或者删除行并将它重新插入一个新的主页。如果表跨跃多个分区，则可以用 *partition_name* 参数指定分区。

可以通过查询 systabstats 和使用 optdiag 命令来显示有关表中转移行数量的统计信息。

reorg forwarded_rows 的语法为：

```
reorg forwarded_rows table_name partition partition_name
    [with {resume, time = no_of_minutes}]
```

有关 resume 和 time 选项的信息，请参见第 222 页的“用于大表重组的 resume 和 time 选项”。

reorg forwarded_rows 不适用于索引，因为索引没有转移的行。

使用 *reorg compact* 撤消行转移

reorg forwarded_rows 使用分配页提示来查找转移的行。因为不必搜索整个表，所以此命令执行得很快，但它可能会遗漏一些转移行。运行 reorg forwarded_rows 之后，可以使用 optdiag 并检查“转移行计数”来评估其效果。如果“转移行计数”很高，可以随后运行 reorg compact，该命令逐页地检查表并撤消所有行转移。

回收删除和更新后留下的未使用空间

当任务执行 delete 操作或将行长度缩短的更新操作时，空闲空间将保留下来，以备万一事务回退之用。如果需要对表频繁执行删除和行缩短更新操作，则未回收的空间可能会积累到一定程度，从而影响性能。

reorg reclaim_space 回收删除和更新后留下的未使用空间。对于因提交删除或行缩短更新操作而产生了未使用空间的每个页，reorg reclaim_space 将连续重写剩余的行，而将所有未使用的空间留在页尾。如果没有剩余的行，reorg reclaim_space 将释放该页。

如果表扩展到整个分区或多个分区，可以通过指定 *partition_name* 来回收分区上的任何可用空间。

使用 optdiag 实用程序可以从 systabstats 表显示有关某个表中未回收的删除行数目的统计信息。然而无法直接测出行缩短更新操作产生了多少未使用的空间。

如果仅指定一个表名，则仅重组表的数据页来回收未使用的空间；换句话说，这样做并不影响索引。如果指定了索引名，则只重组该索引的页。如果定了分区，则只影响该表中位于此分区上的那一部分。

请参见第 222 页的“用于大表重组的 resume 和 time 选项”。

不使用 *reorg* 命令的空间回收

以下活动逐页地回收或重组表中的空间：

- 插入（当遇到一页，而该页如果回收未使用空间就能具有足够空间）
- update statistics 命令（仅用于索引页）
- 重新创建聚簇索引
- 管家碎片收集任务（如果 enable housekeeper GC 设置为 1 或更大）

每一种活动类型都有限制，并且对于大量的页可能不够用。例如，在需要回收空间的时候，插入也许执行的更慢，而且可能不会对需要重新组织空间页产生太大的影响。管家碎片收集任务下的空间回收会压缩未使用的空间，但在用户优先级上运行的单个管家碎片收集任务可能不会到达需要它的一页。

回收未使用的空间并撤消行转移

reorg compact 组合了 reorg reclaim_space 和 reorg forwarded_rows 的功能。在以下情况使用 reorg compact：

- 不需要重建整个表 (reorg rebuild)；但是，行转移和因删除和更新操作产生的未使用空间都可能影响性能。
- 有大量的转移行。请参见第 217 页的“使用 reorg compact 撤消行转移”。

如果指定了分区，则只影响该表中位于此分区上的那一部分。

请参见第 222 页的“用于大表重组的 resume 和 time 选项”。

重新创建表

在以下情况使用 reorg rebuild：

- 查询并没有象通常一样选择使用大 I/O，而且 optdiag 显示数据页、数据行或索引页的集群比很低。
- 使用 sp_chgattribute 更改了 exp_row_size、reservepagegap 或 fillfactor 空间管理设置中的一项或多项，并想让所作的更改不仅应用于未来数据，还要应用于已有的行和页。有关 sp_chgattribute 的信息，请参见《参考手册：过程》。

如果由于集群比低而需要重新创建表，则也可能需要更改其空间管理设置（请参见第 220 页的“使用 reorg rebuild 前更改空间管理设置”）。

如果 reorg rebuild 发现当前表被另一个会话使用，则它会中止整个事务。

reorg rebuild 使用表的当前空间管理设置并根据表的聚簇索引（如果有）重写表中的行。删除表上的所有索引并使用 reservepagegap 和 fillfactor 的当前空间管理值重新创建。运行 rebuild 之后，表没有转移的行，也没有因进行了删除和更新操作而产生的未使用空间。

对表和分区运行 reorg rebuild 时，该命令执行以下操作：

- 获取排它表锁
- 将数据从旧页复制到新页
- 释放旧数据页

- 锁定系统表以进行更新（包括 sysindexes、sysobjects、syspartitions 和 systabstats）
- 针对新的数据页重建聚簇和非聚簇索引
- 提交所有打开的事务
- 释放对系统表的锁定

如果表很大并且有多个索引，则用于更新系统表的锁定可能会持续很长时间，并可能会阻塞进程访问系统表中有关在其上运行 reorg 的用户表的信息。但是，systabstats 不会影响此阻塞，因为它的数据行已锁定。

reorg rebuild 使用 with sorted data 选项建立聚簇索引，因此不必在建立索引的过程中对数据重新排序。

运行 *reorg rebuild* 的前提条件

在表中运行 reorg rebuild 之前：

- 将数据库选项 select into/bulkcopy/pllsort 设置为 true。
- 确定表使用的是数据页锁定还是数据行锁定方案。
- 确保可得到与表及其索引的大小相等的额外磁盘空间。

若要将 select into/bulkcopy/pllsort 设置为 true，请输入：

```
1> use master
2> go
1> sp_dboption pubs2,
    "select into/bulkcopy/pllsort", true
2> go
```

在表上运行 rebuild 之后：

- 在转储事务日志之前，必须转储包含表的数据库。
- 表的分布统计信息已被更新。
- 引用该表的所有存储过程在下次运行时将被重新编译。

使用 *reorg rebuild* 前更改空间管理设置

当 reorg rebuild 重建表时，它根据表当前的 reservepagegap、fillfactor 和 exp_row_size 设置来重写所有表和索引的行。所有这些属性都影响因插入而导致表产生碎片的速度，这可由低集群比来度量。

如果发现表很快地产生碎片且必须频繁地重建该表，您可能需要在运行 reorg rebuild 之前更改表的空间管理设置。

可以使用 sp_chgattribute 更改空间管理设置（请参见《参考手册：过程》）。有关sp_chgattribute 的参考信息，请参见《参考手册：命令》；有关空间管理的其它详细信息，请参见《性能和调优系列：物理数据库调优》中的第 3 章“设置空间管理属性”。

对索引使用 *reorg rebuild* 命令

通过 reorg rebuild 命令，可以在表本身可供访问以进行读取和更新活动的同时重建索引。

使用 *reorg rebuild index_name partition_name* 重建索引

重建单个表或分区索引将把所有的索引行重新写入到新页中。这将通过以下几方面提高性能：

- 提高索引叶级的集群
- 对索引应用 fillfactor 的存储值，这会减少页面拆分
- 应用 reservepagegap 的任何存储值，这有助于保留页以备将来拆分

为减少与那些在查询必须使用索引的用户的争用，reorg rebuild 每次锁定少量页。重建索引是一系列独立的事务，带有某些独立的嵌套事务。大约有 32 页在每个嵌套事务中重建；而大约有 256 页在每个外部事务中重建。在被修改的页中获取地址锁并在第一个操作结束时将其释放。在下一个事务开始前，事务中释放的页不能被重新使用。

如果 reorg rebuild 命令停止运行，则已提交的事务将不会回退。因此，已重组部分的集群已达到所需的空间利用效果，而未重组部分则保持与运行命令前相同。索引保持逻辑一致性。

注释　重建聚簇索引不会影响表的数据页，而只会影响叶页和更高的索引级。不重建级别 1 以上的非叶页。

重建索引的空间要求

如果不指定 fill_factor 或 reservepagegap，则重建索引需要数据段中大约 256 页或稍少一些的额外空间。需要的日志空间量大于删除索引和使用 create index 重新创建该索引所需的空间量，但它应该只是实际索引大小的一小部分。额外的可用空间越多，索引聚簇就越好。

注释 对于已经被很好地集群并且空间利用情况尚可接受的那部分索引，reorg rebuild 不会重新创建它们。

状态消息

对大表运行 reorg rebuild *indexname* 可能需要很长时间。状态消息会定期显示；启动和结束消息被写入错误日志和执行 reorg 的客户端进程。进行中的消息仅在客户端显示。

状态报告间隔以待处理页的 10% 或 10,000 页的 10% 计算，取二者中较大者。处理完该数量的页时，有一条状态消息显示出来。因此，不管索引大小如何输出的消息不超过 10 条。已有 reorg 命令的状态消息输出更频繁。

用于大表重组的 *resume* 和 *time* 选项

如果重组整个表耗时过长并妨碍其它数据库的活动，则使用 reorg 命令的 resume 和 time 选项。time 允许在指定时间段内运行 reorg。resume 允许在表中前一 reorg 完成的位置启动 reorg。组合使用这两个选项可以通过运行一系列的部分重组（例如在非高峰时间）来重组大表。

reorg rebuild 中没有 resume 和 time 选项。

在 *time* 选项中指定 *no_of_minutes*

time 选项中的 no_of_minutes 参数指示已经历的时间，而不是 CPU 时间。例如，要运行 30 分钟的 reorg compact，并从前一 reorg compact 停止的地方开始执行，则可输入：

```
reorg compact tablename with resume, time=30
```

如果 reorg 进程在此 30 分钟内的任何时候转入休眠状态，仍将计入经历时间且不增加 reorg 的持续时间。

如果指定的时间已用完，则 reorg 将关于已处理的表或索引部分的统计信息保存在 systabstats 表中。此信息用来作为带有 resume 选项的 reorg 的重新启动点。带有 resume 和 time 选项的这三个参数中的每个参数的重新启动点都是分别维护的。例如，不能先使用 reorg reclaim_space，然后再使用 reorg compact 重新开始进程。

如果指定了 *no_of_minutes*，而且 reorg 在时间未用完时就到达了表或索引的末尾，则它返回到对象的开头并继续运行直到时间用完。

注释　resume 和 time 允许通过多次运行来重组整个表或索引。然而，如果在两次运行 reorg 之间进行了更新，则一些页可能会被处理两次，而另一些页则根本不会被处理。

第 11 章

检查数据库一致性

什么是数据库一致性检查程序？

数据库一致性检查程序 (dbcc) 提供了用于检查数据库逻辑和物理一致性的命令。dbcc 可执行以下操作：

- 使用 checkstorage 或者 checktable 和 checkdb 在页级和行级检查页链接和数据指针
- 使用 checkstorage、checkalloc、checkverify、tablealloc、textalloc 和 indexalloc 检查页分配
- 使用 checkcatalog 检查数据库中系统表内部和系统表之间的一致性

dbcc checkstorage 将检查结果存储在 dbccdb 数据库中。使用 dbcc 存储过程可输出来自 dbccdb 的报告。

可以在以下情况下使用 dbcc 命令：

- 在进行常规数据库维护期间 — 数据库内部结构的完整性取决于系统管理员或数据库所有者是否定期运行数据库一致性检查。
- 确定发生系统错误后可能造成的危害程度。
- 在备份数据库之前确保备份的完整性。
- 在怀疑数据库已损坏时。例如，如果使用特定的表时出现“Table corrupt”消息，则可以使用 dbcc 确定数据库中是否还有其它表损坏。

使用组件集成服务时，还有其它的 dbcc 命令可用于远程数据库。请参见《组件集成服务用户指南》。

页和对象分配

初始化数据库设备时，disk init 命令将新空间划分为**分配单元**。分配单元的大小取决于服务器使用的逻辑页的大小（2、4、8 或 16K）。每个分配单元的第一页是**分配页**，用于跟踪分配单元中所有页的使用情况。分配页的对象 ID 为 99，它们不是真正的数据库对象，因而并不显示在系统表中，但有关分配页的 dbcc 错误都会报告此值。

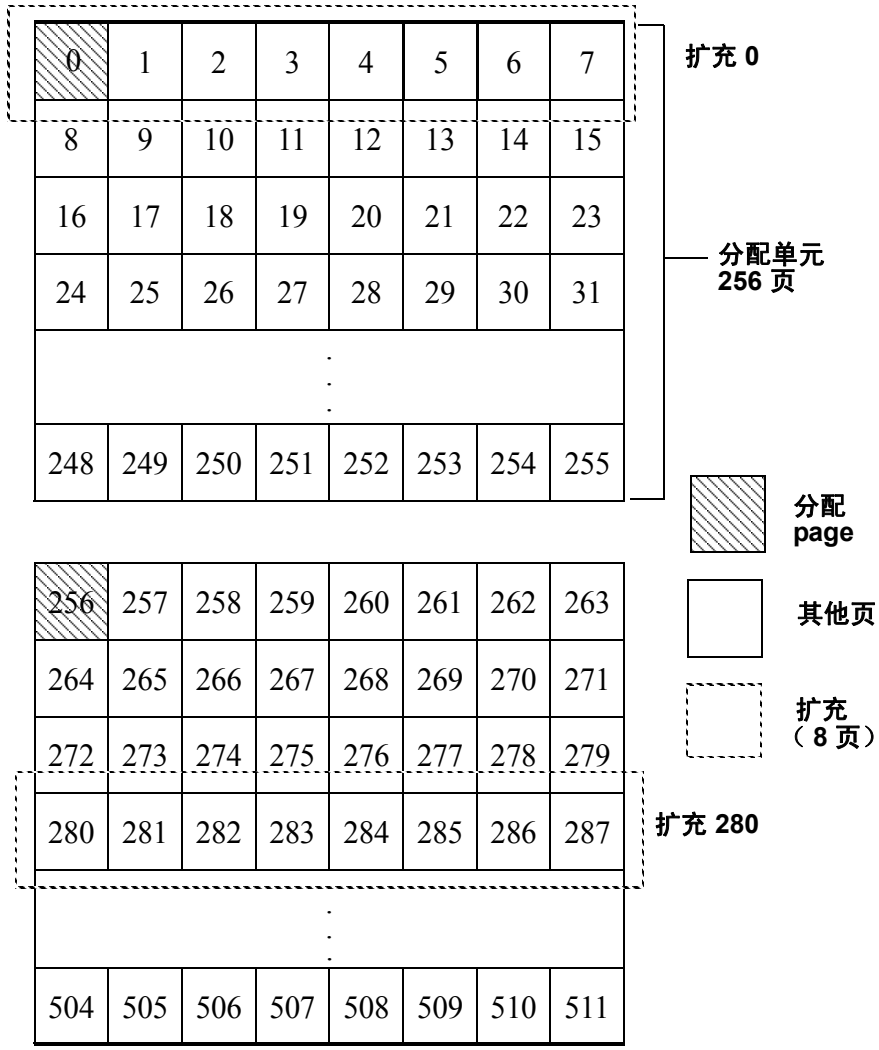
索引分区的表要求空间时，Adaptive Server 将给对象分配一块 8 页的空间。这个 8 页的块称作扩充。每个分配单元包含 32 个扩充。扩充的大小还取决于务器逻辑页的大小。Adaptive Server 使用扩充作为空间管理单元来分配和释放空间，如下所示：

- 创建索引分区的表时，Adaptive Server 将为该对象分配一个扩充。
- 向现有表中增加行而现有页已满时，Adaptive Server 将分配另一页。如果扩充中的所有页都已满，Adaptive Server 将分配另一个扩充。
- 删除索引分区的表时，Adaptive Server 将释放它占用的扩充。
- 删除表中的行从而缩减一页时，Adaptive Server 将释放该页。如果表缩减了一个扩充，Adaptive Server 将释放该扩充。

每次分配或释放扩充中的空间时，Adaptive Server 都会在用于跟踪该对象的扩充的分配页上记录该事件。由于对象可以收缩或增长而不会带来额外开销，这样做可快捷地跟踪数据库中的空间分配。

图 11-1 显示了在 Adaptive Server 数据库中的扩充和分配单元内建立数据页的方法。

图 11-1：使用扩充进行页管理



dbcc checkalloc 用于检查数据库中所有分配页（页 0 和所有能被 256 整除的页），继而报告它所找到的信息。dbcc indexalloc 和 dbcc tablealloc 用于检查特定数据库对象的分配情况。

了解对象分配映射 (OAM)

每个表和表的每个索引都有一个**对象分配映射 (OAM)**。OAM 存储在分配给表或索引的页上。该索引或表需要新页时会检查 OAM。单个 OAM 页可以保存 2,000 到 63,750 个数据页或索引页的分配映射。每个 OAM 页的大小为一个逻辑页的大小。例如，在逻辑页大小为 4K 的服务器上，每个 OAM 页的大小为 4K。

每个 OAM 页的条目数也取决于服务器使用的逻辑页大小。下表列出了对应于每个逻辑页大小的 OAM 条目数：

2K 逻辑页大小	4K 逻辑页大小	8K 逻辑页大小	16K 逻辑页大小
250	506	1018	2042

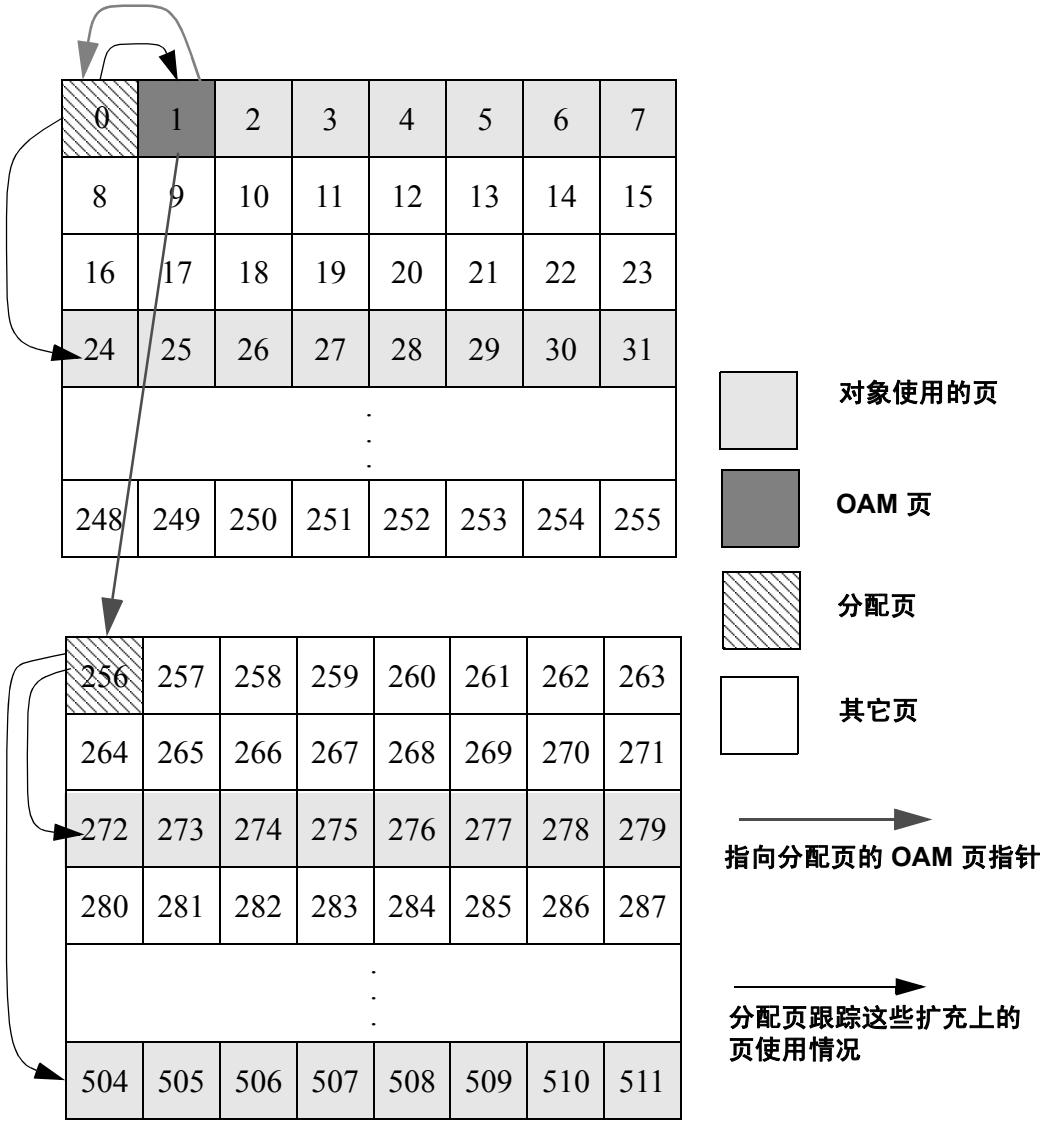
OAM 页指向每个分配单元的分配页，对象使用这些分配单元中的空间。而分配页又跟踪分配单元中有关扩充和页使用情况的信息。换句话说，如果 titles 表存储在扩充 24 和 272 上，则 titles 表的 OAM 页将指向页 0 和 256。

图 11-2 显示了存储在使用 2K 逻辑页的服务器上 4 个扩充（序号为 0、24、272 和 504）中的对象。OAM 存储在第一段的第一页上。这种情况下，由于分配页占用 0，所以 OAM 位于页 1 上。

该 OAM 指向两个分配页：页 0 和页 256。

这些分配页将跟踪所有在分配单元中占据存储空间的对象在每个扩充中所使用的页。对于该示例中的对象，它跟踪扩充 0、24、272 和 504 上页的分配和释放。

图 11-2：OAM 页和分配页指针

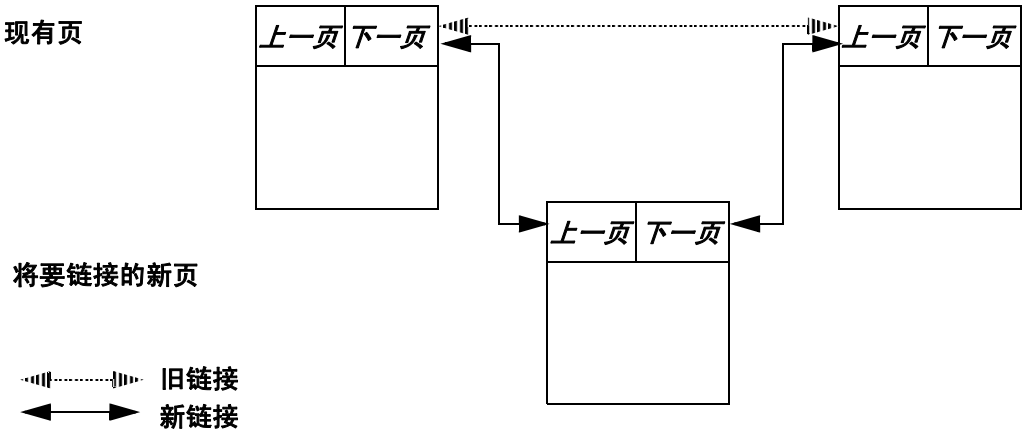


如第 230 页的“了解页链接”中所述，dbcc checkalloc 和 dbcc tablealloc 除检查页链接外，还会检查此 OAM 页信息。

了解页链接

将页分配给索引分区的表后，该页将与其它用于同一对象的页链接在一起。图 11-3 说明了这种链接。每页都包含一个标头，该标头包含它之前一页（“上一页”）的页码和在它之后一页（“下一页”）的页码。分配了新的一页后，其相邻页上的标头信息将改为指向该页。dbcc checktable 和 dbcc checkdb 用于检查页链接。dbcc checkalloc、tablealloc 和 indexalloc 将比较页链接与分配页上的信息。

图 11-3：新分配页与其它页链接的方法



使用 *dbcc* 可执行哪些检查？

表 11-1 总结了 dbcc 命令能够执行的检查。第 244 页的表 11-2 对不同的 dbcc 命令作了比较。

表 11-1：比较各个 dbcc 命令执行的检查

执行的检查	checkstorage	checktable	checkdb	checkalloc	indexalloc	tablealloc	checkcatalog	textalloc
文本值列的分配	X							X[1]
索引一致性		X	X					
索引排序顺序		X	X					

执行的检查	checkstorage	checktable	checkdb	checkalloc	indexalloc	tablealloc	checkcatalog	textalloc
OAM 页条目	X	X	X	X	X	X		X
页分配	X			X	X	X		X
页一致性	X	X	X					
指针一致性	X	X	X					
系统表							X	X[2]
文本列链	X	X	X					X
文本值列	X	X	X					X[3]

[1]textalloc 不检查存储列的数据页的分配状态，而是检查文本页的分配状态。

[2]textalloc 检查与文本或图像列相对应的 syspartition 条目。

[3] textalloc 检查：

- 存储文本值列的数据页
- 保存文本值的文本页。

注释　数据库处于活动状态时，除 dbrepair 和具有 fix 选项的 checkdb 之外，可以运行所有 dbcc 命令。

只有表所有者可以使用 checktable、fix_text 或 reindex 关键字执行 dbcc。只有数据库所有者可以使用 checkstorage、checkdb、checkcatalog、checkalloc、indexalloc、textalloc 和 tablealloc 关键字。只有系统管理员可使用 dbrepair 关键字。

了解各个 *dbcc* 命令的输出

dbcc checkstorage 将结果存储在 dbccdb 数据库中。可从此数据库输出各种报告。请参见第 233 页的“dbcc checkstorage”。

大多数其它 dbcc 命令的输出包含标识正被检查的对象的信息及指明该命令在对象中发现的任何问题的错误消息。运行带有 fix 选项的 dbcc tablealloc 和 dbcc indexalloc 时，输出还将指明命令作出的修复。

以下示例显示了对有分配错误的表执行 dbcc tablealloc 得到的输出结果：

```
dbcc tablealloc(rrtab)
```

```
The default report option of OPTIMIZED is used for this run.
The default fix option of FIX   is used for this run.
************************************************************
TABLE:rrtab            OBJID = 416001482
PARTITION ID=432001539 FIRST=2032 ROOT=2040 SORT=1
Data level:indid 1, partition 432001539. 2 Data pages allocated and 2 Extents
allocated.
Indid :1, partition :432001539. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=448001596 FIRST=2064 ROOT=2072 SORT=1
Data level:indid 1, partition 448001596. 2 Data pages allocated and 2 Extents
allocated.

Indid :1, partition :448001596. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=480001710 FIRST=2080 ROOT=2080 SORT=0
Indid :2, partition :480001710. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=496001767 FIRST=2096 ROOT=2096 SORT=0
Indid :2, partition :496001767. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=512001824 FIRST=2112 ROOT=2112 SORT=0
Indid :3, partition :512001824. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=528001881 FIRST=2128 ROOT=2128 SORT=0
Indid :3, partition :528001881. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=544001938 FIRST=680 ROOT=680 SORT=0
Indid :4, partition :544001938. 1 Index pages allocated and 2 Extents
allocated.
TOTAL # of extents = 18
Alloc page 1792 (# of extent=2 used pages=2 ref pages=2)
Alloc page 1792 (# of extent=2 used pages=3 ref pages=3)
Alloc page 1792 (# of extent=1 used pages=1 ref pages=1)
Alloc page 2048 (# of extent=1 used pages=1 ref pages=1)
Alloc page 1792 (# of extent=1 used pages=1 ref pages=1)
Alloc page 2048 (# of extent=1 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=3 ref pages=3)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 256 (# of extent=1 used pages=1 ref pages=1)
Alloc page 512 (# of extent=1 used pages=1 ref pages=1)
```

```
Total (# of extent=18 used pages=21 ref pages=21) in this database
DBCC execution completed.If DBCC printed error messages, contact a user with
System Administrator (SA) role..
```

检查数据库和表的一致性

检查数据库和表一致性的 dbcc 命令有：

- dbcc checkstorage
- dbcc checktable
- dbcc checkdb

dbcc checkstorage

使用 dbcc checkstorage 可以检查：

- 文本值列的分配
- 页分配和一致性
- OAM 页条目
- 每个分区都存在一个 OAM 页
- 指针一致性
- 文本值列和文本列链

dbcc checkstorage 对磁盘上的数据库运行检查。如果损坏仅在内存中，dbcc checkstorage 可能检测不到损坏。为确保两次 dbcc checkstorage 运行的一致性，请在运行 dbcc checkstorage 之前执行 checkpoint。但是，这样做可能会使瞬时内存损坏成为磁盘损坏。

使用 *dbcc checkstorage* 的优点

dbcc checkstorage：

- 组合了其它 dbcc 命令提供的许多检查
- 不会长时间锁定表或页，这样可在进行并发更新活动的同时使用 dbcc 准确定位错误
- 随总 I/O 吞吐量线性缩放

- 将检查和报告功能分开，允许自定义评估和报告生成
- 提供目标数据库中空间使用情况的详细说明
- 在 dbccdb 数据库中记录 dbcc checkstorage 活动和结果，从而可进行趋势分析并提供正确诊断信息源

比较 *dbcc checkstorage* 和其它 *dbcc* 命令

dbcc checkstorage 与其它 dbcc 命令有所不同，原因在于：

- 它需要 dbccdb 数据库来存储配置信息以及目标数据库上执行的检查的结果。但您可从任意数据库中运行 dbcc checkstorage。
- 执行检查操作时，需使用至少两个工作空间。请参见《参考手册：表》中的“dbccdb 表”。
- 它需要系统和存储过程来帮助准备系统，以便使用 dbcc checkstorage 并生成有关 dbccdb 中存储的数据的报告。

dbcc checkstorage 不修复任何故障。运行 dbcc checkstorage 并生成报告以查看故障后，可运行相应的 dbcc 命令修复这些故障。

了解 *dbcc checkstorage* 操作

dbcc checkstorage 操作包括以下步骤：

1 检查 — dbcc checkstorage 使用正被检查的数据库的设备分配和段定义以及 max worker processing 和 named cache 配置参数，来确定可使用的并行处理的级别。dbcc checkstorage 还使用配置参数 max worker processes 和 dbcc named cache 来限制可使用的并行处理的级别。

2 计划 — dbcc checkstorage 将生成一个计划，以执行利用步骤 1 中所确定的并行度的操作。

3 执行和优化 — dbcc checkstorage 使用 Adaptive Server 工作进程对目标数据库执行并行检查和存储分析。它尝试对每个工作进程执行的工作量平均分配，并汇总未充分利用的工作进程的作。随着检查操作的继续，dbcc checkstorage 将扩展并调整步骤 2 中生成的计划，以利用检查期间收集到的其它信息。

4　报告和控制 — 在检查期间，dbcc checkstorage 在 dbccdb 数据库中记录它在目标数据库中发现的所有故障，以供日后进行报告和评估。它还在 dbccdb 中记录存储分析的结果。dbcc checkstorage 在遇到故障时将试图恢复并继续操作，但如果故障发生后无法恢复，则终止该操作。例如，损坏的磁盘不会导致 dbcc checkstorage 失败；但由于在损坏的磁盘上无法成功执行检查操作，所以不执行这些操作。

如果同时还有另一个会话在执行 drop table，则在初始化阶段 dbcc checkstorage 可能会失败。如果发生了这种情况，则应在删除表进程完成后再次运行 dbcc checkstorage。

注释　有关 *dbcc checkstorage* 错误消息的信息，请参见《故障排除和错误消息指南》。

性能和可伸缩性

dbcc checkstorage 随总 I/O 吞吐量线性缩放，以便大大改善 dbcc checkalloc 的性能。dbcc checkstorage 的缩放属性意味着，如果数据库的大小增长了一倍，硬件的容量（可实现的 I/O 吞吐量）也增长了一倍，dbcc 检查所需的时间将保持不变。使容量加倍通常意味着使盘轴的数目加倍，并提供足够的额外 I/O 通道容量、系统总线容量和 CPU 容量以实现额外的总盘吞吐量。

使用 dbcc checkalloc 和 dbcc checkdb 执行的大多数检查操作（包括文本列链检验）通过 dbcc checkstorage 检查一次即可完成，从而避免了多余的检查操作。

dbcc checkstorage 检查整个数据库（包括未使用的页面），所以执行时间与数据库大小相关。故而使用 dbcc checkstorage 时，检查几乎为空的数据库与检查几乎已满的数据库的区别不大，这与使用其它 dbcc 命令时的情况是一样的。

与其它 dbcc 命令不同的是，数据的放置对 dbcc checkstorage 性能的影响很小。因此，每个会话的性能是一致的，即使不同会话间数据的放置有所不同。

因为 dbcc checkstorage 进行额外的工作来设置并行操作并在 dbccdb 中记录大量数据，所以当目标数据库很小时，其它 dbcc 命令的执行速度会更快。

dbcc checkstorage 使用的工作空间的位置及分配的空间大小将会影响性能和可伸缩性。请参见《参考手册：表》中的“dbccdb 表”。

若要使用单个命令来运行 dbcc checkstorage 和一个用于生成报告的系统过程，请使用 sp_dbcc_runcheck。

dbcc checktable

dbcc checktable 检查指定的表以查看：

- 索引与数据页链接应该正确
- 索引排序应该正确
- 指针应该一致
- 所有索引和数据分区链接应该正确
- 每页上的数据行在行偏移表中应有相应的条目；这些条目与数据行在页中的位置匹配
- 已分区表的分区统计信息应该正确无误

skip_ncindex 选项允许跳过对非聚簇索引上的页链接、指针和排序顺序的检查。聚簇索引和数据页的链接和指针对于表的完整性至关重要。如果 Adaptive Server 报告出页链接或指针存在问题，则可删除并重新创建非聚簇索引。

partition_name 是要检查的分区的名称（由于表可以跨多个分区，因此该分区可能包含也可能不包含整个表）；*partition_id* 是要检查的分区的 ID。

如果指定 *partition_name* 或 *partition_id*，则 dbcc checktable 仅检查位于此分区上的表或表的部分；它无法将检查扩展到其它分区，并受以下限制：

- 如果表由多个分区组成，则仅限于对本地索引进行索引处理。
- 如果指定 *partition_name* 或 *partition_id* 参数，还必须指定第二个参数（skip_ncindex 或 fix_spacebits）或空值。以下示例指定了空值：

  ```
  dbcc checkalloc(titles, null, 560001995)
  ```

- 如果某个表（其中包含用 char 或 varchar 数据类型定义的列）的排序顺序或字符集不正确，dbcc checktable 不会纠正这些值。必须对整个表运行 dbcc checktable 才能纠正这些错误。

- 如果某个索引由于排序顺序发生更改而被标记为“只读”，则 dbcc checktable 不会清除 O_READONLY 位（它位于该表的 sysobjects 条目的 sysstat 字段中）。若要清除此状态位，请对整个表运行 dbcc checktable。
- 如果对 syslogs 运行 dbcc checktable，则 dbcc checktable 不会报告空间使用情况（可用空间与已用空间）。但是，如果不指定 *partition_name* 或 *partition_id* 参数，则 dbcc checktable 会报告空间使用情况。

checkstorage 返回故障代码 100035 而且 checkverify 确认空白位错误是硬故障时，可使用 dbcc checktable 来修复所报告的故障。

下面的命令检查位于 smallsales 分区（其中包含低于 5000 的所有书籍销售额）的部分 titles 表：

```
dbcc checktable(titles, NULL, "smallsales")
Checking partition 'smallsales' (partition ID 1120003990) of table 'titles'.
The logical page size of this table is 8192 bytes.The total number of data
pages in partition 'smallsales' (partition ID 1120003990) is 1.
Partition 'smallsales' (partition ID 1120003990) has 14 data rows.
DBCC execution completed.If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

您可将 dbcc checktable 与表名或表的对象 ID 一同使用。sysobjects 表在 name 和 id 列中存储此信息。

下面的示例显示了一个未损坏表的报告：

```
dbcc checktable(titles)
```

```
Checking table 'titles' (object ID 576002052):Logical page size is 8192 bytes.
The total number of data pages in partition 'titleidind_576002052' (partition ID
576002052) is 1.
The total number of data pages in this table is 1.
Table has 18 data rows.
DBCC execution completed.If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

要检查不在当前数据库中的表，应提供数据库名。若要检查另一对象拥有的表，应提供所有者的名称。必须给限定的表名加上引号。例如：

```
dbcc checktable("pubs2.newuser.testtable")
```

dbcc checktable 可解决以下问题：

- 如果页链接有误，dbcc checktable 将显示一则错误消息。
- 如果其列具有 char 或 varchar 数据类型的表的排序顺序 (sysindexes.soid) 或字符集 (sysindexes.csid) 有误，但该表的排序顺序与 Adaptive Server 的缺省排序顺序兼容，则 dbcc checktable 将改正该表的值。只有二进制的排序顺序可跨字符集兼容。

  注释 如果更改排序顺序，基于字符的用户索引将带有“只读”标记，并且在必要时，必须检查和重建这些用户索引。

- 如果数据行未在对象的第一个 OAM 页中说明，则 dbcc checktable 将更新该页上的行数。这个问题不严重。在 sp_spaceused 等过程中，内置函数 row_count 使用此值快速地对行数进行估计。

通过使用功能增强的页面读取，可提高 dbcc checktable 的性能。

dbcc checkindex

dbcc checkindex 运行的检查与 dbcc checktable 的相同，只不过它仅对特定的索引而不是整个表运行检查。

partition_name 是要检查的分区的名称，*partition_id* 是要检查的分区的 ID。bottom_up 指定 checkindex 从索引的叶开始自下而上进行检查。bottom_up 仅适用于仅数据锁定表。如果对 checkindex 或 checktable 指定此选项，则索引检查会以自下而上的方式进行。

dbcc checkdb

在指定数据库中的每个表上，dbcc checkdb 会运行与 dbcc checktable 相同的检查。如果不指定数据库名，dbcc checkdb 将检查当前数据库。dbcc checkdb 将生成与 dbcc checktable 返回的消息类似的消息，并进行相同类型的改正。

如果指定了可选的 skip_ncindex，则 dbcc checkdb 不会检查数据库中用户表上的任何非聚簇索引。

如果数据库扩展到一系列分区，dbcc checkdb 会在每个分区上执行其检查。

检查页分配

可用来检查页分配的 dbcc 命令为：

- dbcc checkalloc
- dbcc indexalloc
- dbcc tablealloc
- dbcc textalloc

dbcc checkalloc

dbcc checkalloc 可确保：

- 所有页均正确分配
- 分配页上的分区统计信息正确无误
- 分配的每一页都被使用
- 所有页都已正确分配给各个分区
- 只有分配页被使用

如果没有提供数据库名，dbcc checkalloc 将检查当前数据库。

使用 fix 选项，dbcc checkalloc 可修正 dbcc tablealloc 同样能修正的所有分配错误，还可修正保留分配给已从数据库中删除的对象的那些页。在可以将 fix 选项与 dbcc checkalloc 一起使用之前，必须使数据库处于单用户模式。有关使用 fix 和 no fix 选项的详细信息，请参见第 242 页的“使用 fix | nofix 选项更正分配错误”。

dbcc checkalloc 输出中包含每个表（包括系统表）的数据块，以及每个表上的索引。它报告每个表或索引所使用的页数和扩充数。INDID 值包括：

- 对于没有聚簇索引的表，INDID=0。
- 对于具有聚簇索引的所有页锁定表，表数据分区和聚簇索引分区会被合并，数据分区（或聚簇索引分区）的 INDID 为 1。
- 对于具有聚簇索引的仅数据锁定表，表数据分区的 INDID 为 0。聚簇索引和非聚簇索引从 INDID=2 开始连续编号。

分区和页信息列于 PARTITION ID=*partition_number* 之后。

以下有关 pubs2 的报告显示了 titleauthor、titles 和 stores 表的输出结果：

```
**************************************************************
TABLE:titleauthor OBJID = 544001938
PARTITION ID=544001938 FIRST=904 ROOT=920 SORT=1
Data level:indid 1, partition 544001938. 1 Data pages allocated and 2 Extents
allocated.
Indid :1, partition :544001938. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=544001938 FIRST=928 ROOT=928 SORT=0
Indid :2, partition :544001938. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=544001938 FIRST=944 ROOT=944 SORT=0
Indid :3, partition :544001938. 1 Index pages allocated and 2 Extents
allocated.
TOTAL # of extents = 8
**************************************************************
TABLE:titles OBJID = 576002052
PARTITION ID=1120003990 FIRST=1282 ROOT=1282 SORT=1
Data level:indid 0, partition 1120003990. 1 Data pages allocated and 1 Extents
allocated.
PARTITION ID=1136004047 FIRST=1289 ROOT=1289 SORT=1
Data level:indid 0, partition 1136004047. 1 Data pages allocated and 1 Extents
allocated.
TOTAL # of extents = 2
**************************************************************
TABLE:stores OBJID = 608002166
PARTITION ID=608002166 FIRST=745 ROOT=745 SORT=0
Data level:indid 0, partition 608002166. 1 Data pages allocated and 1 Extents
allocated.
TOTAL # of extents = 1
```

dbcc indexalloc

dbcc indexalloc 检查指定的索引以查看：

- 所有的页都分配正确。
- 分配的每一页都被使用
- 只有分配页被使用

dbcc indexalloc 是 dbcc checkalloc 的索引级版本，可对单个索引提供相同的完整性检查，包括索引分区的检查。必须指定对象 ID、对象名称或分区 ID，并且必须指定索引 ID。dbcc checkalloc 和 dbcc indexalloc 输出包括索引 ID。

若要使用 dbcc indexalloc 的 fix 或 nofix 选项，必须指定一个报告选项（full、optimized、fast 或 null）。如果指定分区 ID，则仅检查该分区。请参见第 242 页的“使用 fix | nofix 选项更正分配错误”和第 243 页的“使用 dbcc tablealloc 和 dbcc indexalloc 生成报告”。

dbcc indexalloc 将未分区索引视为具有单个分区的索引。

可运行 sp_indsuspect 来检查索引中排序顺序的一致性，运行 dbcc reindex 修复不一致性。

dbcc tablealloc

dbcc tablealloc 将检查指定的用户表以确保：

- 所有的页都分配正确。
- 分配页上的分区统计信息正确无误。
- 分配的每一页都被使用
- 只有分配页被使用
- 所有页都已正确分配给指定表中的分区

指定表名、数据分区 ID 或表的对象 ID（位于 sysobjects 中的 ID 列）。如果指定数据分区 ID，dbcc tablealloc 将在此分区和所有本地索引分区上执行其检查。如果指定对象名或对象 ID，dbcc tablealloc 将在整个表上执行其检查。如果指定索引分区 ID，dbcc tablealloc 将返回错误 15046。

若要使用 dbcc tablealloc 的 fix 或 nofix 选项，必须指定一个报告选项（full、optimized、fast 或 null）。请参见第 242 页的“使用 fix | nofix 选项更正分配错误”和第 243 页的“使用 dbcc tablealloc 和 dbcc indexalloc 生成报告”。

dbcc textalloc

dbcc textalloc 检查对象的文本和图像数据的分配完整性，报告并修复发现的任何问题。dbcc textalloc 检查指定的对象（或当前数据库中包含文本或图像数据的所有对象）以确保：

- 系统表中的文本和图像条目是正确的。
- OAM 页链以及文本和图像数据的 OAM 页分配是正确的。
- 所有文本和图像页均分配正确。

- 没有已分配但未使用的文本页。
- 没有已使用但未分配的文本页。

可以发出不带任何参数的 dbcc textalloc。缺省情况下，dbcc textalloc 以缺省模式运行。如果不指定参数，则 dbcc textalloc 将检查当前数据库中包含文本或图像列的每个表。

可以对不相互干扰的不同对象同时运行 dbcc textalloc。

使用 *fix | nofix* 选项更正分配错误

dbcc checkalloc、dbcc tablealloc、textalloc 和 dbcc indexalloc 的 fix | nofix 选项指定命令是否修复表中的分配错误。用户表的缺省值为 fix。系统表的缺省值为 nofix。

在系统表上使用 fix 选项之前，必须将数据库置于单用户模式：

```
sp_dboption dbname, "single user", true
```

只有在没有人使用数据库时才能发出此命令。

带有 fix 选项的 dbcc tablealloc 的输出会显示分配错误和作出的所有改正。下面的示例是无论是否使用 fix 选项都会显示的一条错误消息：

```
Msg 7939, Level 22, State 1:
Line 2:
Table Corrupt:The entry is missing from the OAM for
object id 144003544 indid 0 for allocation page 2560.
```

使用 fix 选项后，以下消息指示缺失的条目已恢复：

```
The missing OAM entry has been inserted.
```

fix|nofix 选项在 dbcc indexalloc 中的作用与在 dbcc tablealloc 中一样。

使用 *dbcc tablealloc* 和 *dbcc indexalloc* 生成报告

使用 dbcc tablealloc 或 dbcc indexalloc 可生成三种类型的报告：

- full — 生成的报告包含所有类型的分配错误。使用 dbcc tablealloc 的 full 选项与在表级上使用 dbcc checkalloc 所得到的结果相同。
- optimized — 根据表的 OAM 页中列出的分配页生成报告。使用 optimized 选项时，dbcc tablealloc 不报告而且不能修正 OAM 页中未列出的分配页中的未引用扩充。如果不指定报告类型，或者指定了 null，则缺省值为 optimized。
- fast — 为已引用但未在扩充中分配的页生成例外报告（2521 级别错误）；不生成分配报告。

检查系统表的一致性

dbcc checkcatalog 检查数据库中系统表内部和系统表之间的一致性。例如，它检验：

- syscolumns 中的每个类型在 systypes 中都有一个相匹配的条目。
- sysobjects 中的每个表和视图在 syscolumns. 中都至少有一列。
- sysindexes 一致并修复所有错误
- 对于 syspartitions 中的每一行，sysegments 中均存在一个与之匹配的行。
- syslogs 中的最后一个检查点有效。

它还列出已定义的供数据库使用的段并修复它找到的任何错误。

如果没有指定数据库名，dbcc checkcatalog 将检查当前数据库。

```
dbcc checkcatalog (testdb)
Checking testdb
The following segments have been defined for database 5 (database name testdb).
virtual start addr      size      segments
--------------------    ------    --------------------------
33554432                4096       0
                                   1
16777216                102        2
DBCC execution completed.If DBCC printed error messages, see your System
Administrator.
```

使用一致性检查命令的策略

以下各部分比较了各个 dbcc 命令的性能，提出了合理安排的建议和避免遭受严重性能影响的策略，并提供了有关 dbcc 输出的信息。

表 11-2 比较了 dbcc 命令。请注意：dbcc checkdb、dbcc checktable 和 dbcc checkcatalog 执行的完整性检查的类型与 dbcc checkalloc、dbcc tablealloc 和 dbcc indexalloc 不同。dbcc checkstorage 综合执行其它命令执行的某些检查。第 230 页的表 11-1 中显示了各种命令执行的检查。

表 11-2：dbcc 各种命令的性能比较

命令和选项	级别	锁定和性能	速度	完全性
checkstorage	分配页的页链和数据行、索引的页链接、OAM 页、设备和分区统计信息	不锁定；执行大量的 I/O，可能会使系统 I/O 饱和；可使用专用高速缓存，使得对其它高速缓存的影响最小	快速	高
checktable checkdb	所有索引的页链、排序顺序、数据行和分区统计信息	共享表锁；dbcc checkdb 一次锁定一个表，并在结束对该表的检查后释放该锁	慢速	高
checktable checkdb（使用 skip_ncindex 选项）	表和聚簇索引的页链、排序顺序和数据行	共享表锁；dbcc checkdb 一次锁定一个表，并在结束对该表的检查后释放该锁	比不使用 skip_ncindex 选项时快，最多达 40%。	中
checkalloc	页链和分区统计信息	无锁定；执行大量的 I/O 可能会使 I/O 调用饱和；只有分配页被高速缓存	慢速	高
具有 full 选项的 tablealloc、indexalloc 和 textalloc	页链	共享表锁；执行大量的 I/O；只有分配页被高速缓存	慢速	高
具有 optimized 选项的 tablealloc、indexalloc 和 textalloc	分配页	共享表锁；执行大量的 I/O；只有分配页被高速缓存	中速	中
具有 fast 选项的 tablealloc、indexalloc 和 textalloc	OAM 页	共享表锁	快速	低
checkcatalog	系统表中的行	系统目录中的共享页锁；检查每页后释放锁；很少有页被高速缓存	中速	中

使用大 I/O 和异步预取

如果为要检查的数据库或对象使用的高速缓存配置了大 I/O 和异步预取，则某些 dbcc 命令可以使用它们。

当表使用配置了大 I/O 的高速缓存时，dbcc checkdb 和 dbcc checktable 将在对表进行页链检查时使用大 I/O 缓冲池。将使用最大的可用 I/O 大小。检查索引时，dbcc 仅使用 2K 缓冲区。

如果使用中的缓冲池可使用异步预取，则 dbcc checkdb、dbcc checktable 和 dbcc 分配检查命令 checkalloc、tablealloc 和 indexalloc 将使用异步预取。请参见《性能和调优系列：查询处理和抽象计划》第 6 章“调优异步预取”中的“为 dbcc 设置限制”。

高速缓存绑定命令和更改缓冲池的大小和异步预取百分比的命令都是动态命令。在非高峰期间使用这些 dbcc 命令时，如果对用户应用程序影响甚微，则可以更改这些设置来加速 dbcc 的执行，并在 dbcc 检查完毕时恢复正常设置。请参见第 4 章“配置数据高速缓存”。

在您的节点安排数据库维护

有多个因素可确定应该运行 dbcc 命令的频率及要运行哪些命令。

数据库使用

如果主要在星期一到星期五的上午 8:00 到下午 5:00 之间使用 Adaptive Server，则可以在晚上和周末运行 dbcc 检查，这样检查就不会对用户造成太大影响。如果表不是非常大，则可以经常运行一整套的 dbcc 命令。

dbcc checkstorage 和 dbcc checkcatalog 能够以最低的成本在最大范围内执行检查，并确保从备份中恢复。不必很频繁地运行用于检查索引排序顺序和一致性的 dbcc checkdb 或 dbcc checktable。不必在此检查与任何其它的数据库维护活动之间进行协调。保留对象级的 dbcc 检查和使用 fix 选项的那些检查，以对 dbcc checkstorage 发现的故障进行进一步诊断和修复。

如果每周 7 天，每天 24 小时地使用 Adaptive Server，则可能需要通过限制工作进程数或通过使用应用程序队列来限制 dbcc checkstorage 对资源的使用。如果决定不使用 dbcc checkstorage，则可能要调度使用 dbcc checktable、dbcc tablealloc 和 dbcc indexalloc 对单个表和索引执行的一整套检查。整套检查结束时，如果所有表都已检查完毕，则可运行 dbcc checkcatalog，并备份数据库。请参见《性能和调优系列：基础知识》中的第 5 章“分配引擎资源”。

某些 24 小时都需运行而且要求高性能的节点通过以下方式运行 dbcc 检查：

- 将数据库转储到磁带
- 将数据库转储装载到单独的 Adaptive Server 中以创建数据库副本
- 在数据库副本上运行 dbcc 命令
- 如果检测到可以使用 fix 选项修复的错误，则在原始数据库中的相应对象上运行带有 fix 选项的 dbcc 命令。

转储是数据库页的逻辑副本；因此，原始数据库中发现的问题也会出现在数据库副本中。使用转储创建用于报告或其它目的的数据库副本时，此策略常有用。

合理安排锁定对象的 dbcc 命令的运行时间，以避免与业务活动冲突。例如，dbcc checkdb 获取执行数据库检查的每个表的锁，然后在完成时释放该锁并继续处理下一个表。dbcc checkdb 持有锁期间，这些表不可访问。不要安排 dbcc checkdb（或其它具有类似副作用的 dbcc 命令）在其它业务活动需要锁定的表时运行。

备份日程表

备份数据库和转储事务日志的次数越频繁，发生故障时可恢复的数据就越多。必须决定在灾难发生时允许丢失的数据量，并制定一个转储日程表来支此决定。

调度转储后，应决定如何将 dbcc 命令并入该日程表中。每次转储之前，不必执行 dbcc 检查；但是，如果转储过程中出现损坏情况，则可能会丢失其它数据。

转储数据库的理想时间是在使用 dbcc checkstorage 和 dbcc checkcatalog 对该数据库进行彻底检查之后。如果这些命令在数据库中没有发现错误，即表明备份的是一个清洁的数据库。可以重建索引来改正装载转储后发生的问题。在单个表和索引上使用 dbcc tablealloc 或 indexalloc 改正 dbcc checkalloc 报告的分配错误。

表大小和数据重要性

回答以下有关数据的问题：

- 有多少个表包含至关重要的数据？
- 数据更改的频率如何？
- 这些表有多大？

dbcc checkstorage 是数据库级别的操作。如果仅有少数几个表包含关键数据或经常更改的数据，则您可能希望在这些表上更频繁地运行表和索引级别的 dbcc 命令比在整个数据库上运行 dbcc checkstorage 更有效。

数据库一致性问题导致的错误

执行 dbcc checkstorage 时遇到的数据库一致性问题所生成的错误记录在 dbcc_types 表中。大多数错误号介于 5010 - 5024 和 100,000 - 100,038 之间。有关特定错误的信息，请参见《参考手册：表》中的第 2 章“dbccdb 表”。

执行除 dbcc checkstorage 之外的 dbcc 命令时，对于因数据库一致性问题而生成的错误，其错误号一般在 2500 - 2599 或 7900 - 7999 之间。这些消息和其它因数据库一致性问题而产生的消息（如错误 605）可能包含像“Table Corrupt”或“Extent not within segment”这样的短语。

某些消息指明存在严重的数据库一致性问题；其它消息可能不是如此急迫。少数故障可能需要 Sybase 技术支持部门的帮助，但大多数可通过以下方式决：

- 运行使用 fix 选项的 dbcc 命令
- 按照《错误消息和故障排除指南》中的说明操作，其中包含处理很多 dbcc 数据库错误的分步说明。

无论解决问题需要何种技术，如果在损坏或不一致性发生后很快发现了问题，则解决方案会简单得多。一致性问题可能存在于不经常使用的数据页上如每月仅更新一次的表。dbcc 可以发现这些问题，往往也能修正这些问题。

报告被中止的 checkstorage 和 checkverify 操作

当 checkstorage 或 checkverify 操作中止时，就会输出一条消息，其中包括操作 ID (opid) 和操作中止时正在检查的数据库的名称。已中止的 checkverify 操作还会在消息中提供一个序列号，指示用户与提供的 dbname、*opid* 以及该序列号 seq（如果是 checkverify 操作）一起运行 *sp_dbcc_patch_finishtime*。在执行 sp_dbcc_patch_finishtime 后，可以创建有关已中止的操作的故障报告。

因错误 100032 中止

checkstorage 在遇到页链接错误 (100032) 时可能会中止对象检查。

如果页的更新版本消除了页链接错误，或者页链接错误数少于配置的最大链接错误值，checkstorage 会继续检验对象。

可以使用 sp_dbcc_updateconfig 配置最大链接错误值。下例将 great_big_db 配置为值 8：

```
sp_dbcc_updateconfig great_big_db, "linkage error abort", "8"
```

请参见《参考手册：构件块》中的第 4 章“dbcc 存储过程”。

如果出现以下情况，则 checkstorage 可能会在达到此页链接错误值之前中止检查：

- 对使用 APL 索引的对象的并发更新会破坏页链接，因为 checkstorage 可能无法访问页链的剩余部分
- 在页链接检查期间删除了索引

在无提示状态下运行 checkstorage 时，应减少（或消除）导致索引检查中止的暂时性错误。为了消除暂时性故障，请在运行 dbcc checkstorage 后立即运行 checkverify。

软故障和硬故障的比较

dbcc checkstorage 发现目标数据库中的故障时，将把故障作为一个软故障或**硬故障**记录在 **dbcc_faults** 表中。

软故障

软故障是 Adaptive Server 中通常不会持久存在的不一致性问题。大多数软故障均来源于目标数据库中的暂时不一致性，这是在运行 dbcc checkstorage 期间或 dbcc checkstorage 遇到数据定义语言 (DDL) 命令时，由于用户对数据库进行更新所导致的。第二次运行命令时，这些故障不会重复出现。通过比较两次执行 dbcc checkstorage 的结果，或在 dbcc checkstorage 发现软故障后，通过运行 dbcc tablealloc 和 dbcc checktable，可以重新分类软故障。

如果在几次连续执行 dbcc checkstorage 时都出现了相同的软故障，则它们是“持久的”软故障，而且可能表明有损坏。如果以单用户模式执行 dbcc checkstorage，则报告的软故障是持久性的。通过使用 sp_dbcc_differentialreport 或通过运行 dbcc tablealloc 和 dbcc checktable，可以解决这些故障。如果使用后面的两个命令，则只需检查出现软故障的表或索引。

硬故障

硬故障是 Adaptive Server 的持久性损坏，这种损坏无法通过重新启动 Adaptive Server 得到修复。并非所有硬故障的严重程度都相同。例如，下面的每种情况都导致了硬故障，但结果并不同：

- 分配给不存在的表的页面最低限度地减少了可用磁盘存储空间。
- 具有不能扫描的行的表可能会返回错误结果。
- 链接到另一表的表导致查询停止。

某些硬故障可通过简单操作（例如截断受影响的表）加以改正。其它的只能通过从备份恢复数据库才能得到改正。

使用 *dbcc checkverify* 检验故障

dbcc checkverify 检查最近一次 checkstorage 操作的结果，并将每个软故障重新分类为硬故障或无关紧要的故障。checkverify 用作第二个过滤器，以从 checkstorage 结果中删除假故障。

dbcc checkverify 的工作方式

checkverify 从 dbcc_faults 中读取记录的故障，并通过一个与 checkstorage 操作使用的过程相似的过程解决每个软故障。

注释　checkverify 锁定表以防止并发更新，这将确保正确地对软故障进行重新分类。checkverify 不能查找自上次运行 checkstorage 以来所发生的错误。

---

checkverify 按与 checkstorage 同样的方式将信息记录在 dbcc_operation_log 和 dbcc_operation_results 表中。记录的 opid 的值与上一次 checkstorage 操作的 opid 值相同。checkverify 更新 dbcc_faults 表中的 status 列，并在 dbcc_fault_params 表中为它处理的故障插入一行。

checkverify 不使用 scan 或 text 工作空间。

checkverify 会按以下形式之一对 checkstorage 查出的每个故障进行检验：

- 硬故障，与 checkstorage 所确定的分类一样。
- 因从原因中排除并发活动而被 checkverify 重新分类为硬故障的软故障。
- 由 checkverify 确认的软故障。在数据库中没有并发活动时出现的一些软故障并不表示有重大危险，因而没有被重新分类为硬故障。如果软故障只是信息性的，不表示有损坏，则不会将它重新分类。
- 重新分类为不重要故障的软故障，这类软故障可能是由并发活动造成的，或是因为后续活动掩盖了最初的不一致性而造成的。

如果文本列存在硬故障，由 checkstorage 指定代码为 100011（文本指针故障）的故障将被检验为硬故障。如果文本列没有硬故障，该故障被重新分类为软故障。

对于由 checkstorage 指定代码为 100016（分配了页但未链接）的故障，如果相同的故障出现在两个连续的 checkstorage 操作中，该故障将被检验为硬故障。否则，它将被重新分类为软故障。

对于由 checkstorage 指派代码为 100035（空白位不匹配）的故障，如果被检验为硬故障，可使用 dbcc checktable 修复此故障。

checkverify 确认数据库中存在硬故障时，请按照适当的过程操作以修复这些故障。

checkverify 将以下故障代码划分为软故障：

- 100020 — 检查已中止。
- 100025 — 行计数故障。
- 100028 — 页分配脱离了当前段。

何时使用 *dbcc checkverify*

通过在运行 checkstorage 后的任何时候（甚至可长至几小时或几天后）运行 checkverify，可以检验持久故障。然而，决定日程表时，应记住数据库状态随时间而更改，并且这种更改可以屏蔽软故障和硬故障。

例如，如果页被链接到表但未被分配，这就是一个硬故障。如果该表被删除，故障将不存在从而被屏蔽掉。如果此页被分配给另一个表，故障将持续其特征将更改。此页现在好象是被链接到两个不同的表。如果此页被重新分配到同一个表，故障将表现为有损坏的页链。

因随后数据库的更改而修复的持久故障，通常不会引起操作性问题。然而，通过检测和快速检验这些故障，可以在遇到更严重的问题之前或在原故障特征更改之前找到损坏源。因此，Sybase 建议在运行 dbcc checkstorage 之后尽快运行 checkverify。

注释　如果以单用户模式对目标数据库执行 checkstorage，则没有瞬时软故障，因此不需要执行 checkverify。

每次执行 checkstorage，checkverify 只需运行一次。然而，如果 checkverify 被中断而没有完成，可以再次运行它。操作将从它被中断的位置继续。

处理很大的数据库时，checkverify 可能需要很长时间才能完成。在这段时间内，checkverify 不提供任何有关其将于何时结束的信息。若要在 checkverify 期间查看进度状态报告，请使用 command_status_reporting 命令

```
set command_status_reporting on
```

现在，运行 checkverify 时，生成的结果如下所示：

```
dbcc checkverify (pubs2)
Verifying faults for ’ pubs2’ .
Verifying faults for table 't1'.The total number of tables to verify is 5. This
```

```
is table number 1.
Verifying faults for table 't2'.The total number of tables to verify is 5. This
is table number 2.
Verifying faults for table 't3'.The total number of tables to verify is 5. This
is table number 3.
Verifying faults for table 't4'.The total number of tables to verify is 5. This
is table number 4.
Verifying faults for table 't5'.The total number of tables to verify is 5. This
is table number 5.
DBCC CHECKVERIFY for database ’pubs2’ sequence 4 completed at Apr  9 2003
2:40PM.72 suspect conditions were resolved as faults, and 11 suspect conditions
were resolved as harmless.0 objects could not be checked.
```

如何使用 *dbcc checkverify*

checkverify 的操作对象是上次仅针对指定数据库完成的 checkstorage 操作的结果。

checkverify 操作完成后，Adaptive Server 返回：

```
DBCC checkverify for database name, sequence
n completed at date time. n suspect conditions
resolved as faults and n resolved as innocuous.
n checks were aborted.
```

通过使用 sp_dbcc_runcheck，可在运行 checkstorage 之后自动运行 checkverify。

可以使 checkverify 不处理表、故障以及故障或表的组合。可以使用 sp_dbcc_exclusions 指示要从 checkverify 中排除的项目。sp_dbcc_exclusions 是动态的；也就是说，checkverify 会立即排除您在 sp_dbcc_exclusions 中指定的项目。一旦从 checkverify 中排除这些对象后，下次运行该命令时将不会报告有关这些项目的信息。

使用 *dbcc checkstorage* 之前的准备工作

使用 dbcc checkstorage 之前，请配置 Adaptive Server 并设置 dbccdb 数据库。表 11-3 按使用的先后顺序总结了各个步骤和命令。

以下各节详细描述了每个操作。本节中的示例假设服务器使用 2K 逻辑页。

表 11-3：使用 dbcc checkstorage 之前的准备工作

对于此操作	请参见	使用此命令
1. 为目标数据库获取有关数据库大小、设备（如果 dbccdb 不存在）、工作空间大小、高速缓存大小和工作进程数的建议。	第 253 页的“计划资源” 第 255 页的“计划工作空间大小”	use master sp_plan_dbccdb
2. 如有必要，调整 Adaptive Server 使用的工作进程数。	第 257 页的“配置工作进程”	sp_configure number of worker processes memory per worker processes
3.（可选）为 dbcc 创建一个命名高速缓存。	第 258 页的“为 dbcc 设置命名高速缓存”	sp_cacheconfig
4. 配置缓冲池。	第 259 页的“配置一个 8 页的 I/O 缓冲池”	sp_poolconfig
5. 如果 dbccdb 已存在，则应在创建新的 dbccdb 数据库之前将它和所有与其关联的设备全部删除。		drop database
6. 初始化用于存储 dbccdb 数据和日志的磁盘设备。	第 260 页的“为 dbccdb 分配磁盘空间”	disk init
7.（可选）在数据磁盘设备上创建 dbccdb。		create database
8.（可选）添加磁盘段。	第 260 页的“用于工作空间的段”	use dbccdb
9. 填充 dbccdb 数据库并安装 dbcc 存储过程。		`isql -Usa -P -i $SYBASE/$SYBASE_ASE/scripts/installdbccdb`

注释　dbcc checkstorage 对磁盘上的数据库运行检查。如果损坏仅在内存中，dbcc 可能检测不到损坏。为确保连续两个 dbcc checkstorage 命令之间的一致性，应首先运行 checkpoint。请注意，运行 checkpoint 可能会使瞬时内存损坏成为磁盘损坏。

计划资源

为 dbccdb 选择适当的设备和大小对于执行 dbcc checkstorage 操作很重要。sp_plan_dbccdb 根据 dbccdb 是否存在为指定的目标数据库提供配置建议。可以使用这些信息来配置 Adaptive Server 和设置 dbccdb 数据库。

sp_plan_dbccdb 输出的示例

如果 dbccdb 不存在，sp_plan_dbccdb 将返回：

- dbccdb 的最小大小
- 适用于 dbccdb 的设备
- scan 和 text 工作空间的最小大小
- 高速缓存的最小大小
- 工作进程数

建议的高速缓存大小值是近似值，因为 dbccdb 的最佳高速缓存大小取决于目标数据库中页分配的模式。以下示例来自使用 2K 逻辑页的服务器，它显示了当 dbccdb 不存在时，对 pubs2 数据库运行 sp_plan_dbccdb 所产生的输出：

```
sp_plan_dbccdb pubs2
Recommended size for dbccdb is 4MB.

Recommended devices for dbccdb are:

Logical Device Name   Device Size   Physical Device Name

sprocdev              28672         /remote/SERV/sprocs_dat
tun_dat               8192          /remote/SERV/tun_dat
tun_log               4096          /remote/SERV/tun_log

Recommended values for workspace size, cache size and process count are:

dbname         scan ws    text ws    cache    process count

pubs2         64K         64K        640K    1

(return status = 0)
```

如果 dbccdb 已存在，sp_plan_dbccdb 将返回：

- dbccdb 的最小大小
- 现有 dbccdb 数据库的大小
- scan 和 text 工作空间的最小大小
- 高速缓存的最小大小
- 工作进程数

以下示例显示当 dbccdb 已存在时，对 pubs2 数据库运行 sp_plan_dbccdb 所产生的输出：

```
sp_plan_dbccdb pubs2
```

```
Recommended size for dbccdb database is 23MB (data = 21MB, log = 2MB).
dbccdb database already exists with size 8MB.
Recommended values for workspace size, cache size and process count are:
dbname                          scan ws    text ws    cache    process count
pubs2                          64K        48K        640K    1
(return status = 0)
```

如果计划检查多个数据库，应为目标数据库使用最大的数据库的名称。如果没有提供目标数据库名，sp_plan_dbccdb 将为 master..sysdatabases 中列出的所有数据库返回配置值：

```
sp_plan_dbccdb
```

```
Recommended size for dbccdb is 4MB.

dbccdb database already exists with size 8MB.

Recommended values for workspace size, cache size and process count are:

dbname           scan ws    text ws    cache    process count

master            64K        64K        640K     1
tempdb            64K        64K        640K     1
model             64K        64K        640K     1
sybsystemprocs   384K       112K       1280K     2
pubs2             64K        64K        640K     1
pubs3             64K        64K        640K     1
pubtune          160K        96K       1280K     2
sybsecurity       96K        96K       1280K     2
dbccdb           112K        96K       1280K     2
```

请参见《参考手册：过程》中的“sp_plan_dbccdb”。

计划工作空间大小

dbccdb 需要两个工作空间：scan 和 text。这些工作空间的空间需求取决于要检查的最大数据库的大小。若要运行并发 dbcc checkstorage 操作，需设置额外的工作空间。

确定要检查的最大数据库的大小

不同的数据库可以使用相同的工作空间。因此，工作空间必须足够大，以能够适合要使用的最大的数据库。

注释 sp_plan_dbccdb 建议工作空间大小 — 下面关于确定工作空间大小的细节仅供参考。

目标数据库中的每一页由 scan 工作空间中的一个 18 字节的行表示。此工作空间应该大约是目标数据库大小的 1.1%。

目标数据库中的每个非空 text 列在 text 工作空间中插入一个 22 字节的行。如果目标数据库中有 *n* 个非空 text 列，则 text 工作空间的大小必须至少是 (22 * *n*) 字节。非空 text 列的数目是动态的，所以需要为 text 工作空间分配足够的空间以适应将来的需求。text 工作空间所需的最小空间是 24 页。

可以同时使用的工作空间的数目

可以配置 dbccdb 以在多个数据库上同时运行 dbcc checkstorage。只有在第二个和随后的 dbcc checkstorage 操作有自己专用的资源时，此方法才可行。若要执行并发 dbcc checkstorage 操作，每项操作必须具有自己的 scan 和 text 工作空间、工作进程和保留的高速缓存。

工作空间所需的全部空间取决于对用户数据库是串行检查还是并发检查。如果 dbcc checkstorage 操作以串行方式运行，则最大的 scan 和 text 工作空间可用于所有用户数据库。如果 dbcc checkstorage 操作以并发方式运行，则 dbccdb 应设置为可以适合将并发使用的最大工作空间。通过增加将并发检查的最大数据库的大小，可以确定工作空间大小。

请参见《参考手册：表》中的“dbccdb 表”。

工作空间自动扩展

sp_dbcc_updateconfig.. automatic workspace expansion 选项指示 checkstorage 是否在必要时自动调整工作空间的大小。

开始运行 checkstorage 时，会验证工作空间的大小。如果需要更多空间，并且启用了 automatic workspace expansion，当相应的段上有足够的可用空间时，checkstorage 会自动扩展工作空间。如果需要更多空间，并且未启用 automatic workspace expansion，checkstorage 将退出并显示信息性消息。

值 1（缺省值）将启用 automatic workspace expansion。值 0 将禁用 automatic workspace expansion。请参见《参考手册：过程》。

缺省的 scan 工作空间和 text 工作空间

运行 installdbccdb 脚本时，会创建缺省的 scan 工作空间和 text 工作空间。缺省 scan 工作空间的名称为 def$scan$ws，其大小为 256K。缺省 text 工作空间的名称为 def$text$ws，其大小为 128K。如果尚未配置缺省工作空间并且目标数据库尚未配置工作空间值，将使用这些缺省工作空间。

配置工作进程

以下参数可影响 dbcc checkstorage：

- max worker processes — 使用 sp_dbcc_updateconfig 设置此参数。它为每个目标数据库更新 dbcc_config 表中 max worker processes 的值。
- number of worker processes — 使用 sp_configure 设置此配置参数。它更新 *server_name.cfg* 文件。
- memory per worker process — 使用 sp_configure 设置此配置参数。它更新 *server_name.cfg* 文件。

max worker processes 为每个目标数据库指定 dbcc checkstorage 使用的工作进程最大数，而 number of worker processes 指定 Adaptive Server 支持的工作进程总数。工作进程不是专用来运行 dbcc checkstorage 操作的。

将 number of worker processes 的值设置得足够高，使之支持 max worker processes 指定的进程数。工作进程数低会降低性能并减少 dbcc checkstorage 的资源消耗。dbcc checkstorage 使用的进程数不能多于数据库使用的数据库设备数。高速缓存大小、CPU 性能和设备大小表明的工作进程数可能较低。如果没有为 Adaptive Server 配置足够的工作进程数，dbcc checkstorage 将不能运行。

maximum parallel degree 和 maximum scan parallel degree 对于 dbcc checkstorage 的并行功能没有影响。将 maximum parallel degree 设置为 1 时，不会禁用 dbcc checkstorage 中的并行度。

dbcc checkstorage 需要多个工作进程，所以 number of worker processes 必须至少设置为 1 以支持父进程和工作进程。

sp_plan_dbccdb 根据数据库大小、设备数和其它因素推荐工作进程数的值。可使用较小的值来限制系统上的负载。dbcc checkstorage 使用的工作进程可能会少于 sp_plan_dbccdb 推荐的或您自己配置的工作进程。

使用更多的工作进程并不能保证更好的性能。以下情况说明了两种不同配置的效果：

一个 8GB 的数据库在磁盘 A 上有 4GB 数据，在磁盘 B、C、D、E、F、G、H 和 I 上各有 0.5GB 数据。

如果有 9 个工作进程是活动的，则运行 dbcc checkstorage 所需时间为 2 小时，这也是用来检查磁盘 A 的时间。其它 8 个工作进程分别在 15 分钟内完成检查，并等待磁盘 A 工作进程完成检查。

如果有两个工作进程是活动的，运行 dbcc checkstorage 仍然需要 2 小时。第一个工作进程处理磁盘 A，另一个工作进程处理磁盘 B、C、D、E、F、G、H 和 I。这种情况下无需等待，可以更有效地利用资源。

memory per worker process 为 Adaptive Server 中支持的工作进程指定总的内存分配量。缺省值足够用于 dbcc checkstorage。

为 *dbcc* 设置命名高速缓存

如果为 dbcc checkstorage 使用命名高速缓存，可能需要调整 Adaptive Server 配置参数。

dbcc checkstorage 运行期间，工作空间将临时绑定到也读取目标数据库的高速缓存。使用专用于 dbcc 的命名高速缓存将会尽可能减少数据库检查对其他用户的影响，并可提高性能。可为同时运行的每个 dbcc checkstorage 操作创建一个单独的高速缓存，也可以创建一个足够大的高速缓存以适应并发操作的全部需求。最佳性能所需的容量大小取决于目标数据库的大小及该数据库中数据的分布。dbcc checkstorage 要求命名高速缓存中的每个工作进程都至少具有 640K 的缓冲区（每个缓冲区是一个扩充）。

为获得最佳性能，应将专用高速缓存的大部分指派给缓冲池，而且不要对高速缓存分区。建议的高速缓存大小是缓冲池的最小大小。对此值增加一页缓冲池空间。

如果将一个高速缓存专用于 dbcc checkstorage，则此命令要求的大小不会大于一页缓冲池的最小值。如果高速缓存是共享的，则可以通过在运行操作之前增加缓冲池的大小，并在操作完成之后减其大小来提高 dbcc checkstorage 的性能。缓冲池的要求与共享高速缓存的要求一样。然而，虽然共享高速缓存可以满足大小要求，但高速缓存中的其它要求可能会限制可供 dbcc checkstorage 使用的缓冲区，并大大影响 checkstorage 和 Adaptive Server 的整体性能。

注释　由于为 dbcc checkstorage 指定专用的高速缓存分区时 Adaptive Server 可能会发出错误消息 9946 或 9947，因此 Sybase 建议您为 dbcc checkstorage 指定专用的命名高速缓存。

若要为 Adaptive Server 配置一个用于 dbcc checkstorage 操作的命名高速缓存，请使用 sp_cacheconfig 和 sp_poolconfig。请参见第 4 章“配置数据高速缓存”

配置一个 8 页的 I/O 缓冲池

dbcc checkstorage 要求一个其大小为一个扩充的 I/O 缓冲池。使用 sp_poolconfig 配置缓冲池大小并检验缓冲池是否配置正确。缓冲池大小存储在 dbcc_config 表中。

dbcc checkstorage 缓冲池的大小等于页大小乘以 16。不同页大小的 dbcc checkstorage 缓冲池要求分别为：

- （2KB 页服务器）*（8 个扩充）= 16k 缓冲池
- （4KB 页服务器）*（8 个扩充）= 32k 缓冲池
- （8KB 页服务器）*（8 个扩充）= 64k 缓冲池
- （16KB 页服务器）*（8 个扩充）= 128k 缓冲池

下例显示了如何使用 sp_poolconfig 在配置为 2K 逻辑页的服务器上为“master_cache”设置 16K 缓冲池。命名高速缓存是为 master 数据库创建的。

```
1> sp_poolconfig "master_cache", "1024K", "16K"
2> go

(return status = 0)
```

下例显示已为专用高速缓存“master_cache”设置了缓冲池：

```
1> sp_poolconfig "master_cache"
2> go
```

```
Cache Name     Status    Type     Config Value Run Value
-------------  --------- -------- ------------ ------------
master_cache   Active    Mixed         2.00 Mb      2.00 Mb
                                    ------------ ------------
                          Total        2.00 Mb      2.00 Mb
===========================================================
Cache: master_cache,   Status: Active,   Type: Mixed
   Config Size: 2.00 Mb,   Run Size: 2.00 Mb
   Config Replacement: strict LRU,   Run Replacement: strict LRU
IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
2 Kb     512 Kb    0.00 Mb      1.00 Mb      10
16 Kb    192 Kb    1.00 Mb      1.00 Mb      10
(return status = 0)
```

请参见《参考手册：过程》。

为 *dbccdb* 分配磁盘空间

dbccdb 数据库要求额外的磁盘存储空间。由于 dbcc checkstorage 将大量使用 dbccdb，所以应该将 dbccdb 置于与其它数据库设备不同的设备上。

注释 不要在主设备上创建 dbccdb。确保用于 dbccdb 的日志设备和数据设备是分开的。

用于工作空间的段

通过将段专用于工作空间，可以控制工作空间的位置并提高 dbcc checkstorage 性能。将新段专门供工作空间使用时，应使用 sp_dropsegment 从缺省段解除这些新段附带的设备的映射。

创建 *dbccdb* 数据库

❖ **创建 *dbccdb* 数据库**

1 在 master 数据库中运行 sp_plan_dbccdb，获取有关目标数据库的数据库大小、设备、工作空间大小、高速缓存大小和工作进程数的建议。

有关 sp_plan_dbccdb 提供的信息的细节，请参见第 253 页的“计划资源”。

2　如果 dbccdb 已存在，则应在创建新的 dbccdb 数据库之前将它和所有与其关联的设备全部删除：

```
use master
 go
if exists (select * from master.dbo.sysdatabases
     where name = "dbccdb")
begin
     print "+++ Dropping the dbccdb database"
     drop database dbccdb
end
go
```

3　初始化用于 dbccdb 数据和日志的磁盘设备：

```
use master
 go
disk init
   name = "dbccdb_dat",
   physname = "/remote/disks/masters/",
   size = "4096"
 go
disk init
   name = "dbccdb_log",
   physname = "/remote/disks/masters/",
size = "1024"
go
```

4　在已在步骤 3 中初始化的数据磁盘设备上创建 dbccdb：

```
use master
go

create database dbccdb
   on dbccdb_dat = 6
   log on dbccdb_log = 2
go
```

5　（可选）将用于 scan 和 text 工作空间的段添加到 dbccdb 数据设备上：

```
use dbccdb
 go
sp_addsegment scanseg, dbccdb, dbccdb_dat
 go
sp_addsegment textseg, dbccdb, dbccdb_dat
go
```

6 为 dbccdb 创建表并初始化 dbcc_types 表：

```
isql -Ujms -P***** -iinstalldbccdb
```

installdbccdb 在试图创建表之前检查数据库是否存在。它只创建 dbccdb 中尚未存在的那些表。如果任何一个 dbccdb 表被损坏，请使用 drop table 将其删除，然后使用 installdbccdb 重新创建。

7 创建并初始化 scan 和 text 工作空间：

```
use dbccdb|
go|
sp_dbcc_createws dbccdb, scanseg, scan_pubs2, scan, "64K"|
sp_dbccvreatews dbccdb, textseg, text_pubs2, text, "64K"
```

安装完 dbccdb 之后，必须更新 dbcc_config 表。

更新 *dbcc_config* 表

dbcc_config 表说明当前正在执行的或者上次完成的 dbcc checkstorage 操作，包括：

- dbcc checkstorage 操作的专用资源的位置
- dbcc checkstorage 操作的资源使用限制

本节说明如何更新此表中的值。

使用 *sp_dbcc_updateconfig* 添加缺省配置值

sp_dbcc_updateconfig 允许为 dbcc 配置参数提供缺省配置值。可以为 checkstorage 分析的每个数据库提供单独的配置值，或者您可以使所设置的缺省值适用于没有相应配置值（即配置值不会发生冲突）的所有数据库。

使用 *sp_dbcc_updateconfig* 删除配置值

sp_dbcc_updateconfig 接受 delete 作为 *str1* 参数的值，从而允许删除已在数据库上设置的配置值。

例如，若要删除 max worker processes 配置参数的缺省值，请输入：

```
sp_dbcc_updateconfig null, 'max worker processes', 'delete'
```

若要删除数据库 my_db 的 max worker processes 配置参数的值，请输入：

```
sp_dbcc_updateconfig my_db, 'max worker processes', 'delete'
```

查看当前配置值

sp_dbcc_configreport *defaults* 参数允许查看配置的缺省值。*defaults* 参数是可选的，并会在 *dbname* 为空时被忽略。*defaults* 参数的有效值是 true 或 false（缺省值）。值为 true 指示只显示配置值的缺省值。值为 false 指示查看所有配置的值。

例如，若要只查看配置的缺省值，请为 缺省 参数输入 *true*：

```
sp_dbcc_configreport null, 'true'
```

若要查看所有配置的值，不要为 defaults 参数提供值或使用“false”：

```
sp_dbcc_configreport
```

或，

```
sp_dbcc_configreport null, 'false'
```

维护 *dbccdb*

有时可能需要对 dbccdb 执行维护任务。

- 使用以下方式重新评估和更新配置：
  - sp_dbcc_evaluatedb — 使用上一次 dbcc checkstorage 操作的结果建议配置参数的值。
  - sp_dbcc_updateconfig — 为指定的数据库更新配置参数。

- 清除陈旧的数据：

  sp_dbcc_deletedb — 从 dbccdb 中删除有关指定数据库的所有信息。

  - sp_dbcc_deletehistory — 从 dbccdb 中删除在指定数据库上执行的 dbcc checkstorage 操作的结果。
- 删除不必要的工作空间。
- 对 dbccdb 自身执行一致性检查。

重新评估并更新 *dbccdb* 配置

如果用户数据库的特性发生了更改，请使用 sp_dbcc_evaluatedb 重新评估当前的 dbccdb 配置并建议更合适的值。

以下对用户数据库的更改可能会影响 dbccdb 配置，如下所示：

- 创建、删除或变更用户数据库时，dbcc_config 表中存储的工作空间和命名高速缓存的大小或工作线程数可能会受到影响。
- dbcc_checkstorage 的命名高速缓存大小或工作进程计数的更改可能要求重新配置缓冲区高速缓存和工作进程。

如果 dbcc checkstorage 操作的结果可用于目标数据库，则使用 sp_dbcc_evaluatedb 确定新的配置值。sp_dbcc_configreport 还报告指定数据库的配置参数。

使用 sp_dbcc_updateconfig 将新数据库添加到 dbcc_config 表中，并更改 dbcc_config 中的配置值以反映 sp_dbcc_evaluatedb 建议的值。

清除 *dbccdb*

Adaptive Server 将 dbcc checkstorage 生成的数据存储在 dbccdb 中。应该使用 sp_dbcc_deletehistory 定期清理 dbccdb 以从数据库删除数据。必须在指定的日期之前创建数据库。

删除数据库时，还应该从 dbccdb 中删除与该数据库相关的所有配置信息以及 dbcc checkstorage 结果。使用 sp_dbcc_deletedb 从 dbccdb 中删除所有数据库信息。

删除工作空间

若要删除不必要的工作空间，请在 dbccdb 中发出：

```
drop table workspace_name
```

对 *dbccdb* 执行一致性检查

在 dbccdb 表中进行有限的更新活动应该能够降低损坏的发生频率。dbccdb 中表明损坏的两种信号是：

- 初始化阶段评估需要执行的工作时或完成阶段记录结果时，dbcc checkstorage 失败
- 丢失 dbcc checkstorage 发现的、已记录故障中损坏导致的故障的相关信息

dbccdb 中的严重损坏可能会导致 dbcc checkstorage 失败。为便于 dbcc checkstorage 找到 dbccdb 中的严重损坏，可以创建一个只用于检查 dbccdb 的替代数据库 dbccalt。使用与用于创建 dbccdb 相同的过程创建 dbccalt，如第 252 页的“使用 dbcc checkstorage 之前的准备工作”中所述。

如果没有设备可用于 dbccalt，则可使用 master 数据库或 dbccdb 不使用的任何设备。

dbcc checkstorage 和 dbcc 系统过程对 dbccalt 的作用与对 dbccdb 的作用相同。目标数据库为 dbccdb 时，dbcc checkstorage 将使用 dbccalt（如果它存在）。如果 dbccalt 不存在，则可将 dbccdb 自身用作管理数据库来检查。如果目标数据库是 dbccdb，而且 dbccalt 存在，则 dbcc checkstorage 对 dbccdb 操作的结果将存储在 dbccalt 中。如果 dbccalt 不存在，结果将存储在 dbccdb 自身中。

或者，可以使用 dbcc checkalloc 和 dbcc checktable 检查 dbccdb。

如果 dbccdb 已损坏，则可将其删除并重新创建，或从备份装载旧版本。如果将其删除，某些诊断历史记录将会丢失。

从 *dbccdb* 生成报告

有几个 dbcc 存储过程随 dbccdb 提供，以便根据 dbccdb 中的数据生成报告。

报告 *dbcc checkstorage* 操作的摘要

sp_dbcc_summaryreport 报告在指定的日期或该日期之前为指定的数据库完成的所有 dbcc checkstorage 操作。下例显示了此命令的输出结果：

```
    sp_dbcc_summaryreport

DBCC Operation : checkstorage
Database Name      Start time           End Time   Operation ID
         Hard Faults Soft Faults Text Columns Abort Count
         User Name
------------------ -------------------- ---------- ------------
         ---------- ---------- ------------ -----------
         ------------------------------
sybsystemprocs     05/12/1997 10:54:45  10:54:53              1
         0           0            0           0
     sa
sybsystemprocs     05/12/1997 11:14:10  11:14:19              2
         0           0            0           0
     sa
```

请参见《参考手册：过程》中的第 4 章“dbcc 存储过程”。

报告配置、统计和故障信息

以下 dbcc 系统过程报告配置和统计信息：

- sp_dbcc_summaryreport
- sp_dbcc_configreport
- sp_dbcc_statisticsreport
- sp_dbcc_faultreport short

sp_dbcc_fullreport 对所有这些 dbcc 系统过程运行完整报告。

请参见《命令参考手册：过程》。

使用 *dbcc upgrade_object* 升级编译对象

Adaptive Server 基于源文本升级编译对象。编译对象有：

- 检查约束
- 缺省值
- 规则
- 存储过程（包括扩展存储过程）
- 触发器
- 视图

如果每个编译对象的源文本未被手动删除，它们应存储在 syscomments 表中。在升级服务器时，将在升级过程中检验源文本是否存在于 syscomments 中。但是，编译对象在被调用之前实际上并不会升级。

例如，如果有一个名为 list_proc 的用户定义的存储过程，在升级到 Adaptive Server 的最新版本时将检验是否存在 list_proc 的源文本。在升级后首次调用 list_proc 时，Adaptive Server 检测到 list_proc 编译对象还未升级。于是，Adaptive Server 将根据 syscomments 中的源文本，重新编译 list_proc，然后执行新编译的对象。

升级后的对象将保留升级前对象所用的对象 ID 和权限。

对于使用 sp_hidetext 隐藏了其源文本的编译对象，其升级方式与未隐藏源文本的对象相同。有关 sp_hidetext 的信息，请参见《参考手册：过程》。

注释　如果从 32 位安装升级为使用 64 位 Adaptive Server，则当升级对象时，每个数据库的 sysprocedures 表中的每个 64 位编译对象的大小将增加大约 55%。预升级进程会计算出准确的大小，并相应增加升级后的数据库大小。

为了确保编译对象在被调用之前已成功地升级，可以使用 dbcc upgrade_object 命令手动对其进行升级。请参见第 268 页的“在生产之前查找编译对象错误”。

在生产之前查找编译对象错误

对 Adaptive Server 12.5.x 之前的版本所做的更改可能会导致编译对象在升级到更高版本后的工作结果不同。可以使用 dbcc upgrade_object 查找下列错误和潜在的问题区域，然后可能需要手工改变它们来获得正确的行为：

- 保留字错误
- 丢失、截断或破坏的源文本
- 带引号标识符错误
- 临时表引用
- select * 潜在问题区域

在检查完错误和潜在问题区域并解决了那些需要更改的错误后，可以使用 dbcc upgrade_object 手动升级编译对象，而不必等待服务器来自动升级这些对象。请参见《参考手册：命令》。

保留字错误

如果 dbcc upgrade_object 在编译对象中发现将某个保留字用作对象名，它将返回一个错误，并且不升级该对象。要修复此错误，可以手动更改对象名，或者用引号将对象名括起来，然后发出 set quoted identifiers on 命令。然后，删除并重新创建编译对象。

例如，假设要将数据库转储从早期版本的 Adaptive Server 装载到最新版本，并且该转储中包含一个使用“XYZ”一词的存储过程，“XYZ”是最新版本的保字。对该存储过程运行 dbcc upgrade_object 时，该命令返回一个错误，因为尽管“XYZ”在早期版本中不是保留字，但是在最新版本中是保留字。事先知道了这个情况，就可以对存储过程和任何相关表进行适当更改，之后再用于生产环境。

丢失、截断或破坏的源文本

如果 syscomments 中的源文本被删除、截断或破坏，dbcc upgrade_object 可能报告语法错误。如果源文本不是隐藏的，可以使用 sp_helptext 检验源文本的完整性。如果出现截断或其它破坏，可删除并重新创建编译对象。

带引号标识符错误

在下列情况下，dbcc upgrade_object 返回一个带引号标识符错误：

- 编译对象是在低于 11.9.2 的版本中创建的，且带引号标识符有效 (set quoted identifiers on)。
- 在当前会话中，带引号标识符无效 (set quoted identifiers off)。

为避免此错误，在运行 dbcc upgrade_object 之前应激活带引号标识符。当带引号标识符有效时，必须使用单引号代替双引号来括住所引用的 dbcc upgrade_object 关键字。

如果出现带引号的标识符错误，可使用 set 命令激活 quoted identifiers，然后运行 dbcc upgrade_object 来升级对象。

注释　带引号的标识符与用双引号括住的文字不相同，后者不要求在升级前执行任何特殊操作。

临时表引用

如果一个编译对象（如存储过程或触发器）引用一个在该对象体外创建的临时表 (#temp *table_name*)，升级将失败，且 dbcc upgrade_object 返回一个错误。要纠正此错误，可完全按编译对象的要求创建临时表，然后再次执行 dbcc upgrade_object。如果编译对象在调用时被自动升级，则不必执行此操作。

select * 潜在问题区域

如果 dbcc upgrade_object 在某个存储过程的最外层查询块中发现 select * 子句，它将返回一个错误，并且不升级该对象。

例如，考虑下列存储过程：

```
create procedure myproc as
  select * from employees
go
create procedure yourproc as
  if exists (select * from employees)
    print "Found one!"
go
```

dbcc upgrade_object 会在 myproc 上返回一个错误，因为 myproc 在最外层查询块中包含一个带有 select * 子句的语句。此过程不会升级。

dbcc upgrade_object 在 yourproc 上不会返回错误，因为 select * 子句出现在一个子查询内。此过程将升级。

确定是否应在视图中更改 *select **

如果 dbcc upgrade_object 报告在一个视图中存在 select *，则应比较原始视图的 syscolumns 输出和表的输出，以确定自创建视图以来是否在表中添加或删除了一些列。

例如，假设有以下语句：

```
create view all_emps as select * from employees
```

在升级 all_emps 视图之前，应使用下列查询来确定原视图中的列数和更新后的表中的列数：

```
select name from syscolumns
  where id = object_id("all_emps")
select name from syscolumns
  where id = object_id("employees")
```

比较两个查询的输出。如果表包含的列比视图的列多，并且有必要保留 select * 语句的预升级结果，可将 select * 语句更改为带有具体列名的 select 语句。如果视图是从多个表创建的，则应检查组成该视图的所有表中的列，必要时重写 select 语句。

警告！ 不要从视图执行 select * 语句。这样做会升级视图，并覆盖关于 syscolumns 中的原始列信息的信息。

还有一种方法可以确定视图中的列与新表中的列的差异，就是在视图上和组成该视图的表上各自运行 sp_help。

这种比较只对视图起作用，对其它编译对象不起作用。若要确定其它编译对象中的 select * 语句是否需要修正，可查看每个编译对象的源文本。

在升级中使用数据库转储

用转储和装载进行升级

可以装载 12.5 版之前的数据库转储和事务日志，并升级数据库。

您需要了解以下问题：

- 在升级期间，需要有用于复制数据和记录系统表变化的空间。如果转储中的源数据库接近充满，则升级过程可能会因空间不足而失败。然而这种情况太常见，在出现空间不足错误时可以使用 alter database 扩展可用空间。
- 在重新装载早期转储后，应从新安装中对装载的数据库运行 sp_checkreswords 以检查保留字。

升级数据库转储中的编译对象

当装载在低于当前版本的 Adaptive Server 中创建的数据库转储时，并不要求在装载前执行预升级任务。因此，如果数据库转储中的编译对象缺失其源文，将不会收到任何通知信息。在装载数据库转储后，应运行 sp_checksource 检验数据库中所有编译对象的源文本是否存在。然后，可以允许编译对象在执行时进行升级，或者运行 dbcc upgrade_object 查找潜在问题并手动升级对象。

请参见《参考手册：过程》。

确定编译对象是否已升级

若要确定一个编译对象是否已升级，请执行下列操作之一：

- 查看 sysprocedures.version 列。如果该对象已升级，则此列包含数字 12500。
- 如果是在同一版中升级到 64 位指针大小，可查看 sysprocedures.status 列。如果该对象已升级，并且使用的是 64 位指针，则此列包含一个十六进制位设置 0x2，表示该对象使用 64 位指针。如果未设置此位，则表示该对象仍是一个 32 位对象，还没有进行升级。

第 12 章

制定备份和恢复计划

Adaptive Server 具有**自动恢复**过程，可以使用户避免由于断电和计算机故障所造成的损失。若要避免介质故障带来损失，请定期经常对数据库进行备份。

本章提供有关信息，帮助您制定备份和恢复计划。本章的第一部分概述了 Adaptive Server 备份和恢复过程。本章的第二部分论述在您开始将系统用于生产之前应解决的备份和恢复问题。

如果您的节点使用 IBM Tivoli Storage Manager 提供的存储管理服务，则另请参见 《将 Backup Server 与 IBM Tivoli Storage Manager 配合使用》。

跟踪数据库的变化

Adaptive Server 使用事务来跟踪数据库的所有变化。事务是 Adaptive Server 的工作单元。一个事务包括一个或多个作为一个单元成功或失败的 Transact-SQL 语句。

每条修改数据的 SQL 语句都被视为一个**事务**。通过将一系列语句放在 begin transaction...end transaction 块中，用户也可以定义事务。请参见《Transact-SQL 用户指南》中的第 18 章“事务：维护数据一致性和恢复”。

每个数据库都拥有自己的**事务日志**，即系统表 syslogs。事务日志自动记录每个数据库用户发出的每个事务。不能关闭事务记录。

事务日志是**前写式日志**。当用户发出要修改数据库的语句时，Adaptive Server 将这些更改写入日志中。在这条语句要做的所有更改都已记录在日志中后，这些更改将被写入到数据页的高速缓存副本中。此数据页将一直保留在高速缓存中，直到另一数据库页需要内存为止。那时，已更改的数据页才写入磁盘中。

如果事务中任何语句未能完成执行， Adaptive Server 将撤消由该事务所引起的所有更改。 Adaptive Server 在每个事务结束时将一条“end transaction”记录写入日志，记录该事务的状态 （成功或失败）。

获取有关事务日志的信息

事务日志包含有关每个事务的足够信息，以确保其可以恢复。使用 dump transaction 命令将事务日志所包含的信息复制到磁带或磁盘。使用 sp_spaceused syslogs 检查日志的大小，或使用 sp_helpsegment logsegment 检查可用于日志增长的空间。

警告！ 不要使用 insert、 update 或 delete 命令修改 syslogs。

使用 *delayed_commit* 确定提交日志记录的时间

必须有一个关系数据库才能确保多个事务属性，包括原子性、一致性、完整性和持久性 （也称作 ACID 属性）。若要确保这一点，Adaptive Server 应符合以下规则，即：

- 将所有操作写入事务日志。
- 在修改数据或索引页之前写入日志记录。

- 在发出事务的 commit 时将日志页写至磁盘中。
- 仅当 Adaptive Server 收到通知并且通知指示已成功地从基本操作系统和 I/O 子系统写入到磁盘后，才通知客户端应用程序已经成功地进行了 commit。

set delayed_commit 是一个仅适用于特定应用程序的性能选项。它提高 Adaptive Server 执行数据操作语言 (DML) 操作 （例如， insert、update、 delete）的性能，但也增加了在系统出现故障期间丢失数据的风险。性能的改善取决于所使用的应用程序。

可从 set delayed_commit 受益的应用程序类型通常包括以串行方式快速发送到 Adaptive Server 的短事务。例如发出多个 insert 语句的批处理应用程序，其中每个 insert 都是一个独立的事务。

使用 set 命令为会话启用 delayed_commit 或为数据库将 set 命令与 sp_dboption 配合使用。

启用 set delayed_commit 后，在相应的日志记录写入磁盘之前，会通知客户端应用程序已成功地执行了 commit。这之所以会提高性能，是因为日志中除最后一页之外的所有页都会被写至磁盘中，从而缓解对最后一个活动日志页的争用。

在启用 set delayed_commit 之前，请考虑：

- 发出 shudown with nowait 可导致数据持久性问题，但如果发出在服务器关闭之前完成的 checkpoint 则例外。
- 为会话启用 set delayed_commit 仅会影响该会话。所有其它会话的事务均强制实施其所有属性，包括 ACID 属性。这还意味着，其它会话的物理日志会写入最后一个日志页以及与启用 set delayed_commit 的会话相对应的日志记录。
- set delayed_commit 在临时数据库上是冗余的，不会提供性能改善。
- 只有在认真考虑应用程序和操作要求以及环境后，才能使用 set delayed_commit。虽然降低数据持久性的危险性非常低，但如果您的数据库很大，而且对丢失数据的容错能力很低，则使用恢复选项可能会很费时间。

日志行为的更改

下面是启用 delayed_commit 时会出现的日志行为的更改。

会话隐式或显式提交事务时：

- 用户日志高速缓存 (ULC) 刷新到内存中的事务日志。
- 任务对所有非写入日志页发出写操作，但最后一个日志页 （其中包含 commit）除外。

- 该任务会通知客户端应用程序已成功地执行了 commit，而不必等待 I/O 完成。

  注释 将由以下项写入此事务的 “最后一个日志页”：

  - 由另一个事务在它不再是 “最后一个日志页”时写入。
  - 在完成时由另一个非延迟事务写入。
  - 通过 checkpoint 或管家缓冲区清洗机制写入。
  - 由隐式检查点原因 （例如， shutdown、 dump database、 dump tran 和 sp_dboption truncate log on checkpoint）写入。

- 任务随时可以继续进行下一个事务。

使用 delayed_commit 的危险性

当 set delayed_commit 处于启用状态时， Adaptive Server 会在实际物理磁盘写入完成之前通知客户端应用程序。因此，该应用程序会认为事务已经完成，无论物理磁盘写入是否成功都是如此。如果系统出现故障 （磁盘错误、系统崩溃，等等），则在恢复之后，未写至磁盘的事务 （其 commit 记录位于最后一个日志页上的事务）将会消失，无论是否已通知应用程序已成功提交这些事务都是如此。

要求紧密系统相关性的系统 （如通过一个使用实时数据服务 (RTDS) 的消息传送系统）会使 set delayed_commit 的使用结果进一步复杂化。

在以下两种情况下，应用程序可以管理风险：

- 应用程序维护它自己的跟踪或日志，并且在系统出现故障之后，确保数据库状态与它自己的跟踪或日志相对应。
- 您可以将数据库恢复到运行应用程序之前所处的状态。它假定您在运行批处理作业类型的应用程序之前进行了完整的数据库备份。如果出现故障，则会装载数据库备份并重新启动批处理作业。

启用 *set delayed_commit*

您可以为数据库或会话启用 set delayed_commit，并用会话设置覆盖数据库设置。这意味着不管数据库设置情况如何，启用该选项的会话必然会启用 delayed_commit。

指定备份的职责

许多组织都有一位执行所有备份和恢复操作的操作员。只有系统管理员、数据库所有者或操作员才可以执行 dump 和 load 命令。数据库所有者只能转储自己的数据库。操作员和系统管理员可以转储和装载任何数据库。

请参见第 315 页的 “使用 Backup Server 执行备份和恢复”和

同步数据库及其日志：检查点

检查点操作将所有脏页写入数据库设备中；所谓脏页，是指自最后一个检查点以来，已在内存中修改、但未在磁盘上修改的页。Adaptive Server 的自动**检查点**机制确保定期将那些被已完成的事务所更改的数据页从内存高速缓存写入数据库设备。同步数据库及其事务日志会缩短在系统出现故障后恢复数据库所需的时间。

设置恢复间隔

通常情况下，每个数据库的自动恢复所需的时间介于几秒钟和几分钟之间。时间长度取决于数据库的大小、事务日志的大小，以及必须提交或回滚的事务的数量和大小。

将 sp_configure 与 recovery interval in minutes 参数配合使用可指定允许的最长恢复时间。 Adaptive Server 会以足够高的频率运行自动检查点以在指定的时间段内恢复数据库。

缺省值为 5，允许每个数据库在 5 分钟内完成恢复。若要将恢复间隔更改为 3 分钟，请使用：

```
sp_configure "recovery interval in minutes", 3
```

注释　恢复间隔对以下事务没有影响：在 Adaptive Server 发生故障时处于活动状态、长期运行且记入日志内容很少的事务 （如 create index）。恢复这些事务所花费的时间可能与运行它们所花费的时间相等。为避免长时间的延迟，需在数据库的一个表上创建索引后立即转储每个数据库。

自动检查点过程

检查点任务大约每分钟检查一次服务器上的每个数据库，以便了解自上次进行检查点操作以来已添加到事务日志中的记录数。如果服务器估计恢复这些事务所需的时间大于数据库的恢复间隔，Adaptive Server 会发出一个检查点。

已修改的页从高速缓存写入数据库设备，并且检查点事件被记录在事务日志中。然后，检查点任务 “休眠”一分钟。

若要查看检查点任务，请执行 sp_who。检查点任务通常会在 “cmd” 列中显示为 “CHECKPOINT SLEEP”：

```
fid  spid  status      loginame  orignam e  hostname         blk_spid  dbname
        tempdbname    cmd               block_xloid   threadpool
---  ----  ----------  --------  --------  ---------------  --------  ------
        ----------    ----------------  -----------   -------------------

. . .
0       2    sleeping      NULL      NULL                NULL       0  master
            tempdb     DEADLOCK TUNE              0    syb_default_pool
0       3    sleeping      NULL      NULL                NULL       0  master
            tempdb     ASTC HANDLER               0    syb_default_pool
0       4    sleeping      NULL      NULL                NULL       0  master
            tempdb     CHECKPOINT SLEEP           0    syb_default_pool
. . .
```

用户数据库升级后进行检查点操作

Adaptive Server 在升级用户数据库后立即插入一条检查点记录。Adaptive Server 使用此记录确保 dump database 先于 dump transaction 在已升级的数据库上执行。

进行自动检查点操作后截断日志

系统管理员可在 Adaptive Server 执行自动检查点操作后截断事务日志。

若要设置 trunc log on chkpt 数据库选项（该选项在出现自动检查点时截断事务日志），请从 master 数据库执行以下命令：

```
sp_dboption database_name, "trunc log on chkpt", true
```

此选项不宜用于生产环境，因为它在截断事务日志之前不会保留一个副本。仅将 trunc log on chkpt 用于：

- 其事务日志由于不在单独的段上而无法备份的数据库
- 当前备份不重要的测试数据库

注释　如果将 trunc log on chkpt 选项保留为缺省条件 off，则事务日志将不断增长，直到您用 dump transaction 命令将其截断。

若要避免用尽日志空间，请设计最后机会阈值过程以转储事务日志。请参见第 17 章 “使用阈值管理可用空间”。

可用检查点

在 Adaptive Server 没有要处理的用户任务时，管家清洗任务会自动开始将脏缓冲区中的数据写入磁盘。如果管家任务可以刷新所有已配置高速缓存中的所有活动缓冲池，它将唤醒检查点任务。检查点任务确定是否必须对数据库执行检查点操作。

作为管家清洗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清洗任务完成。自由检查点可能会缩短数据库恢复时间。

请参见 （《性能和调优系列：基础知识》中的第 3 章 “使用引擎和 CPU”。

手动请求检查点

数据库所有者可以执行 checkpoint 命令来强制将内存中的所有已修改页写至磁盘中。手动检查点不会截断日志，即使 sp_dboption 的 trunc log on chkpt 选项处于启用状态也是如此。

使用 checkpoint 命令：

- 作为特殊情况 （例如，在计划的 shutdown with nowait 之前）下的预防措施，以便 Adaptive Server 恢复机制在恢复间隔内执行。（普通的 shutdown 执行检查点操作。）
- 在执行 sp_dboption 后，使数据库选项中所做的更改生效。（运行 sp_dboption 后，会有信息性消息提醒您运行 checkpoint。）

您可以使用 checkpoint 标识一个或多个数据库或使用 all 子句 （该子句对所有数据库执行检查点检查）。请参见 《参考手册：命令》。

在系统出现故障或关机后自动恢复

每次重新启动 Adaptive Server 后 （例如，在电源故障、操作系统故障或使用 shutdown 命令后），它都将在每个数据库上自动执行一系列恢复过程。

恢复机制将每个数据库与其事务日志进行比较。如果经某种具体更改的日志记录比数据页新，恢复机制会根据事务日志重新应用此更改。如果某一事务在故障发生时仍在进行，恢复机制将撤消由该事务进行的所有更改。

启动 Adaptive Server 时，它将按以下顺序执行数据库恢复：

1 恢复 master。

2 恢复 sybsystemprocs。

3 恢复 model。

4 创建 tempdb （通过复制 model 来实现）。

5 恢复 sybsystemdb。

6 恢复 sybsecurity。

7 按照 sysdatabases.dbid 顺序或按照 sp_dbrecovery_order 指定的顺序恢复用户数据库。请参见 “恢复顺序”。

一旦系统数据库得以恢复，用户就可以登录到 Adaptive Server，但在其它数据库恢复之前，用户不能对其进行访问。

配置变量 print recovery information 确定在恢复期间， Adaptive Server 是否将每个事务的详细消息显示在主控台屏幕上。缺省情况下，这些消息不会显示。若要显示消息，请使用：

```
sp_configure "print recovery information", 1
```

快速恢复

在计划内或计划外关机之后重新启动服务器期间，或者在高可用性故障切换期间，要花费一大部分时间来执行数据库恢复。更快地恢复可以尽量减少数据库停止工作的时间。快速恢复的目标是：

- 增强数据库恢复的性能
- 通过利用可用服务器资源并以智能化的方式对它们进行调优，并行恢复多个数据库

- 在运行期提供多个检查点任务，这些检查点任务可同时运行以尽量减少恢复时的工作量

Adaptive Server 启动序列

下面是 Adaptive Server 启动时的事件序列：

1　系统数据库在引擎 0 上恢复。

2　Adaptive Server 接受用户连接。

3　启动期间配置为处于联机状态的所有引擎都进入联机状态。

4　使用为提供最佳恢复性能而调优的缺省数据高速缓存，通过“自调优”数目的恢复任务并行恢复用户数据库。

请参见第 282 页的 “数据库恢复”。有关恢复任务数量的信息，请参见第 281 页的 “并行恢复”。

在 HA 故障切换期间，进行故障切换的用户数据库被并行恢复并进入联机状态。

尽早使引擎进入联机状态

在恢复系统数据库后，但在恢复用户数据库前，引擎进入联机状态。这样，用户数据库将可被并行恢复，并且使引擎可用于联机活动。

引擎仅在启动期间以这种方式联机。在所有其它情况下 （如故障切换），引擎已经在辅助服务器上联机。

并行恢复

使用 Adaptive Server 12.5.1 及更高版本，在启动和故障切换期间，将通过多项恢复任务并行恢复数据库。数据库恢复是 I/O 密集型进程。通过并行恢复方式恢复 Adaptive Server 所用的时间取决于基础 I/O 子系统的带宽。 I/O 子系统应能够处理 Adaptive Server 的并发 I/O 请求。

通过并行恢复，多个任务可以同时恢复多个用户数据库。恢复任务的数量取决于配置参数 max concurrently recovered db。该配置参数的缺省值是 0，指示 Adaptive Server 采用自调优方法。在这个方法中，并不进行任何有关基础存储体系结构的假设。统计 I/O 采样方法根据基础 I/O 子系统的性能确定恢复任务的最佳数目。提供对最佳恢复任务数目的建议。如果配置值不是零，则 Adaptive Server 会生成由配置参数以及 number of open databases 参数指示的数量的任务。

在并行恢复期间，系统管理员会通过将 max concurrently recovered db 设置为 1 来强制执行串行恢复。活动恢复任务会在恢复完正在处理的数据库之后完成。其余的数据库以串行方式恢复。

请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

数据库恢复

Adaptive Server 数据库恢复包括：

- 日志 I/O 大小 — Adaptive Server 将缺省数据高速缓存中可用的最大缓冲池用于日志 I/O。如果具有最大缓冲区大小的缓冲池不可用，则服务器将动态创建该缓冲池，并将该缓冲池用于日志 I/O。此缓冲池的缓冲区来自缺省缓冲池。为提供最佳恢复性能，恢复将调优大缓冲池的大小。如果大缓冲池可用，但其大小并非最佳，则 Adaptive Server 会以动态方式调整其大小，并使用缺省缓冲池来优化恢复性能。缓冲池配置在恢复过程结束后恢复。

  请参见 《性能和调优系列：基础知识》中的第 5 章 “内存使用和性能”。

- async prefetch limit — 在恢复期间，服务器自动将恢复时使用的缺省数据高速缓存中缓冲池的本地 async prefetch 限制设置为最佳值。这将替换恢复期间用户指定的任何设置。

  在恢复完成后，将恢复原始配置值。

恢复顺序

用户可以指定为全部或部分用户数据库恢复数据库的顺序。您可以使用 sp_dbrecovery_order 将较重要的数据库配置为较早恢复。

若要使用 sp_dbrecovery_order 输入或修改用户定义的恢复顺序，您必须使用 master 数据库并且拥有系统管理员特权。任何数据库中的任何用户都可以使用 sp_dbrecovery_order 更改为用户定义的数据库恢复顺序。请参见 《参考手册：过程》。

sp_dbrecovery_order 拥有表示联机顺序的附加参数。

```
sp_dbrecovery_order [database_name [, rec_order [,
force [ relax | strict ]]]]
```

- relax — 数据库随着恢复的进行而生成 （缺省行为）。
- strict — 数据库按恢复顺序指定。

缺省值是 relax，这意味着在完成恢复后数据库立即进入联机状态。

恢复顺序必须是连续的，并从 1 开始。不能由于打算以后将恢复顺序 3 指派给另一数据库，而将恢复顺序指定为 1， 2， 4。

若要在用户定义的恢复序列中插入数据库，又不想将该数据库放在最后，请输入 *rec_order* 并指定 force。例如，如果数据库 A、B 和 C 采用用户定义的恢复顺序 1， 2， 3，并且您想要插入 pubs2 数据库作为第二个要恢复的用户数据库，请输入：

```
sp_dbrecovery_order pubs2, 2, force
```

此命令指定数据库 B 的恢复顺序为 3，并指定数据库 C 的恢复顺序为 4。

Adaptive Server 12.5.1 及更高版本根据用户指定的顺序使用并行恢复任务确定下一个要恢复的数据库。其余数据库按其数据库 ID 的顺序进行恢复。恢复数据库的时间取决于许多因素，包括可恢复日志的大小。因此，虽然您使用 sp_dbrecovery_order 确定了恢复顺序，但 Adaptive Server 可以按数据库启动顺序以外的顺序完成数据库恢复。有些应用程序必须强制数据库按照与恢复顺序相同的顺序进入联机状态，对于这样的应用程序， Adaptive Server 在 sp_dbrecovery_order 中提供了 strict 选项。

更改或删除数据库的恢复位置

若要更改数据库在用户定义的恢复序列中的位置，请从恢复序列中删除数据库，然后在您希望它出现的位置插入它。如果新位置不在恢复顺序的末尾，则使用 force 选项。

若要从恢复序列删除一个数据库，请将其恢复顺序指定为 -1。

例如，若要将 pubs2 数据库从恢复位置 2 移至恢复位置 1，请从恢复序列中删除该数据库，然后重新指定它的恢复顺序，如下所示：

```
sp_dbrecovery_order pubs2, -1
sp_dbrecovery_order pubs2, 1, "force"
```

列出用户指定的数据库恢复顺序

若要列出指派了恢复顺序的所有数据库的恢复顺序，请使用：

```
sp_dbrecovery_order
```

这将生成如下输出内容：

```
The following databases have user specified recovery order:
Recovery Order Database Name        Database Id
-------------- ------------------- -----------
    1          dbccdb                    8
    2          pubs2                     5
    3          pubs3                     6
    4          pubtune                   7
The rest of the databases will be recovered in default database id order.
```

若要显示特定数据库的恢复顺序，请输入数据库名：

```
1> sp_dbrecovery_order pubs2
2> go

Database Name Database id Recovery Order
 ------------ ---------- --------------
 pubs2              5           2
```

并行检查点

检查点任务的缓冲池并行处理活动数据库的列表。该缓冲池由配置参数 number of checkpoint tasks 控制。如果出现检查点瓶颈情况，则更多的检查点任务转换为短一些的可恢复日志，并且发生故障时恢复需要处理的工作较少，因而提高了可用性。

number of checkpoint tasks 的缺省值为 1，表示串行检查点。number of engines 和 number of open databases 限制此参数的值。若要促进并行恢复，请配置要在启动时联机的最大引擎数量。如果减小此参数的值，则检查点会结束，而当您增大该值时，会生成其它任务。

检查点是 I/O 密集型，因此，并行检查点的效用取决于数据库的布局以及基本 I/O 子系统的性能。根据活动数据库的数目和 I/O 子系统处理写入操作的能力，调优 number of checkpoint tasks。

请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

恢复状态

全局变量 @@ *recovery_state* 确定 Adaptive Server 是否处于恢复状态。 @@ *recovery_state* 可以具有以下值：

- NOT_IN_RECOVERY — Adaptive Server 不处于启动恢复或故障切换恢复中。恢复已完成，且所有可以联机的数据库都已处于联机状态。
- RECOVERY_TUNING — Adaptive Server 处于恢复状态 （启动或故障切换）中并且正调优恢复任务的最佳数目。
- BOOTIME_RECOVERY — Adaptive Server 处于启动恢复状态中并且已完成最佳任务数目的调优工作。但并非所有数据库都已恢复。
- FAILOVER_RECOVERY — Adaptive Server 在 HA 故障切换期间处于恢复状态并且已完成恢复任务的最佳数目的调优工作。所有数据库都还没有联机。

@@*recovery_state* 可由应用程序使用，以确定何时所有数据库都被恢复并进入联机状态。

为进行快速恢复调优

本节论述对 Adaptive Server 进行调优以减少恢复时间的一些准则。

数据库布局

- 数据库应在其自己的物理设备中具有日志和数据。日志和数据的访问模式不同并且应该单独保存。
- 将基础 I/O 子系统配置为可以处理来自 Adaptive Server 中多个数据库的并发 I/O 请求。

运行期配置建议

- 配置由 housekeeper free write percent 控制的最佳管家清洗百分比，以便在可用周期中写尽脏页。缺省值通常是最佳的。
- 确保保留尽量少的长期运行的事务。长期运行的事务占有资源并且也会导致恢复时间延长。
- 若要避免恢复时间延长，请使用 polite shutdown 关闭服务器。

设置空间计算

如果对于某一数据库而言数据空间计数不是必需的，则使用 sp_dboption 设置数据库选项以禁用可用空间计数。这样可禁用对数据段的阈值操作。

恢复期间的故障隔离

恢复过程 （简称为 “恢复”）可根据事务日志重建服务器的数据库。以下情况将导致恢复过程运行：

- Adaptive Server 启动
- 使用 load database 命令
- 使用 load transaction 命令

通过对恢复故障隔离模式进行设置，可以控制在数据库中撤消或重新应用事务时检测到损坏的数据情况下的恢复行为。

如果某索引被标记为可疑索引，则系统管理员可通过删除并重新创建该索引来修复该索引。

恢复故障隔离可以：

- 配置是整个数据库还是只有可疑页会在恢复检测到损坏时变得无法访问
- 配置整个含可疑页的数据库是否在 read_only 模式下联机，或者是否仅能访问联机页来进行修改
- 列出有可疑页的数据库。
- 按页 ID、索引 ID 和对象名列出指定数据库中的可疑页。
- 在修复可疑页时为系统管理员将可疑页联机
- 在修复完可疑页后为所有数据库用户将可疑页联机

该功能将可疑页隔离出来而令其余的数据库处于联机状态，从而可以更灵活地处理数据损坏。您可以诊断问题 （有时可以纠正它们），同时用户仍可访问数据库的大部分。您可以评估损坏的程度并安排紧急修复，或在方便时重新装载。

恢复故障隔离仅应用于用户数据库。如果系统数据库有任何损坏页，恢复过程总是使系统数据库完全脱机。在修复或删除系统数据库中的所有损坏页之前，不能恢复系统数据库。

脱机页的持续性

已脱机的可疑页在重新启动服务器后仍然处于脱机状态。有关脱机页的信息存储在 master.dbo.sysattributes 中。

使用 drop database 和 load database 命令从 master.dbo.sysattributes 中清除可疑页的条目。

配置恢复故障隔离

如果安装了 Adaptive Server，则缺省恢复故障隔离模式为 databases，该模式将数据库标记为可疑，并在检测到任何损坏页时使整个数据库脱机。

隔离可疑页

若要隔离可疑页以便只有它们处于脱机状态，同时使用户可以访问数据库的其余部分，请使用 sp_setsuspect_granularity 将恢复故障隔离模式设置为 page。下次在数据库中执行恢复时，此模式会生效。请参见 《参考手册：过程》。

如果没有 database 或 page 参数， sp_setsuspect_granularity 会显示指定数据库的当前的和已配置的恢复故障隔离模式设置。如果没有任何参数，它将显示当前数据库的这些设置。

如果无法将损坏隔离到特定页，则恢复会将整个数据库标记为可疑，即使恢复故障隔离模式设置为 page 也是如此。例如，损坏的事务日志或不可用的全局资源就会导致这种情况发生。

恢复过程将特定页标记为可疑后，缺省行为是使可疑页脱机从而不可访问，而对数据库可进行读写操作。但是，如果将 read_only 选项指定给 sp_setsuspect_granularity，并且恢复过程将任何页标记为可疑，则整个数据库将以 read_only 模式进入联机状态且不可修改。如果您首选 read_only 选项，但在某些情况下，感觉允许用户修改非可疑页会比较好，则可以使用 sp_dboption 使数据库的联机部分变得可写：

```
sp_dboption pubs2, "read only", false
```

在这种情况下，可疑页仍处于脱机状态，直到您将其修复或强制其联机，如第 289 页的 “使脱机页联机”中所述。

增加所允许的可疑页数目

可疑性增加阈值是恢复将整个数据库标记为可疑时的可疑页数量（甚至是在恢复故障隔离模式为 page 时）。缺省情况下，此值在单个数据库中设为 20 页。使用 sp_setsuspect_threshold 可更改可疑性增加阈值。请参见 《参考手册：过程》

可配置数据库级的恢复故障隔离和可疑性增加阈值。

此示例显示， pubs2 数据库的恢复故障隔离模式为 page 并且上次针对此数据库运行恢复时的增加阈值为 20 （当前可疑性增加阈值）。下次针对此数据库运行恢复时，恢复故障隔离模式为 page 并且增加阈值为 30 （已配置的值）。

```
sp_setsuspect_granularity pubs2

DB Name  Cur. Suspect Gran.  Cfg. Suspect Gran.  Online mode
-------  -----------------  -----------------  -----------
pubs2    page               page               read/write


sp_setsuspect_threshold pubs2

DB Name       Cur. Suspect threshold   Cfg. Suspect threshold
------------- ------------------------ ----------------------
pubs2         20                       30
```

如果没有参数， sp_setsuspect_granularity 和 sp_setsuspect_threshold 将显示当前数据库的当前设置和所配置的设置 （如果该数据库是用户数据库）。

获取有关脱机数据库和页的信息

使用 sp_listsuspect_db 可查看哪些数据库包含脱机页。

以下示例显示有关可疑页的常规信息：

```
sp_listsuspect_db
The database 'dbt1' has 3 suspect pages belonging to 2 objects.
```

使用 sp_listsuspect_page 可显示有关各个脱机页的详细信息。

如果未指定 dbname，则缺省值为当前数据库。以下示例显示 dbt1 数据库中 sp_listsuspect_page 的详细页级输出。

```
sp_listsuspect_page dbt1

DBName   Pageid       Object    Index   Access
```

```
-------- ------------ --------- ------- ------------------
dbt1     384          tab1      0       BLOCK_ALL
dbt1     390          tab1      0       BLOCK_ALL
dbt1     416          tab1      1       SA_ONLY

(3 rows affected, return status = 0)
```

如果 Access 列中的值是 SA_ONLY，并且可疑页为 1，则只有具有 sa_role 角色的用户才可以访问该可疑页。如果该值为 BLOCK_ALL，则任何人都不能访问该页。

任何用户都可以从任何数据库运行 sp_listsuspect_db 和 sp_listsuspect_page。

使脱机页联机

使用 sp_forceonline_db 可使数据库中的所有脱机页可以访问，使用 sp_forceonline_page 可使单个脱机页可以访问。请参见 《参考手册：过程》。

使用这两个过程指定访问类型。

- “sa_on” 使可疑页或数据库只能由具有 sa_role 角色的用户访问。这样，就可在数据库已启用并在运行的情况下修复可疑页和测试修复情况，同时禁止一般用户访问可疑页。您也可以使用它针对含可疑页的数据库执行 dump database 或 dump transaction with no_log，如果页脱机，则将禁止该操作。
- “sa_off” 阻止所有用户的访问，包括系统管理员。这将使用 “sa_on” 撤消以前的 sp_forceonline_db 或 sp_forceonline_page 。
- “all_users” 在修复页之后为所有用户使脱机页联机。

  不同于先使用 “sa_on” 使可疑页联机，然后再使用 “sa_off” 使它们重新脱机，当您针对 “所有用户” 使用 sp_forceonline_page 或 sp_forceonline_db 使页联机时，此操作将无法撤消。没有方法可使联机页再脱机。

警告！ Adaptive Server 在被联机的页上不执行任何检查。确保已经修复每个将转至联机的页。

您无法在事务中执行 sp_forceonline_db 或 sp_forceonline_page。

您必须具有 sa_role 角色且在 master 数据库中，才能执行 sp_forceonline_db 和 sp_forceonline_page。

DOL 锁定表的索引级故障隔离

如果 DOL 锁定表的索引页在恢复期间被标记为可疑，则整个索引都将被脱机。两个系统过程管理脱机索引：

- sp_listsuspect_object
- sp_forceonline_object

大多数情况下，系统管理员使用 sp_forceonline_object 将可疑索引标记为仅可供具有 sa_role 角色的人员使用。如果该可疑索引在用户表中，可通过删除并重新创建该索引来修复它。

请参见 《参考手册：过程》。

脱机页的副作用

以下限制适用于具有脱机页的数据库：

- 需要脱机数据的事务会直接或间接 （例如，因为参照完整性约束）出现故障并生成消息。
- 数据库的任何部分脱机时，不能使用 dump database 命令。

  系统管理员可以通过配合使用 sp_forceonline_db 和 “sa_on” 来强制脱机页联机，接着转储数据库，然后在转储完成后将 sp_forceonline_db 与 “sa_off” 配合使用。
- 如果数据库的任何部分脱机，则不能使用 dump transaction with no_log 或 dump transaction with truncate_only。

  系统管理员可以通过配合使用 sp_forceonline_db 和 “sa_on” 来强制脱机页联机，接着使用 with no_log 转储事务日志，然后在转储完成后将 sp_forceonline_db 与 “sa_off” 配合使用。
- 要删除包含脱机页的表或索引，必须使用 master 数据库中的事务。否则，删除操作将失败，因为它必须从 master.dbo.sysattributes 中删除可疑页条目。下例将从 master.dbo.sysattributes 中删除对象并删除与其脱机页有关的信息。

  若要删除名为 authors_au_id_ind 的索引 （该索引包含可疑页），请从 pubs2 数据库中删除 master 数据库事务中的索引：

```
use master
 go
sp_dboption pubs2, "ddl in tran", true
 go
checkpoint pubs2
```

```
 go
begin transaction
drop index authors.au_id_ind
commit
 go
use master
 go
sp_dboption pubs2, "ddl in tran", false
 go
checkpoint pubs2
go
```

使用恢复故障隔离的恢复策略

有两种主要的策略，可以在保证用户能够访问数据库的同时，将带有可疑页的数据库恢复到一致的状态：重装和修复。

这两种策略都要求：

- 一个干净的数据库转储
- 一系列可靠的事务日志转储（直到带有可疑页的数据库恢复时）
- 在数据库被恢复，能够捕获对脱机页的更改后，事务日志立即转储到设备
- 连续的事务日志向设备转储，此时用户使用的数据库部分脱机

重装策略

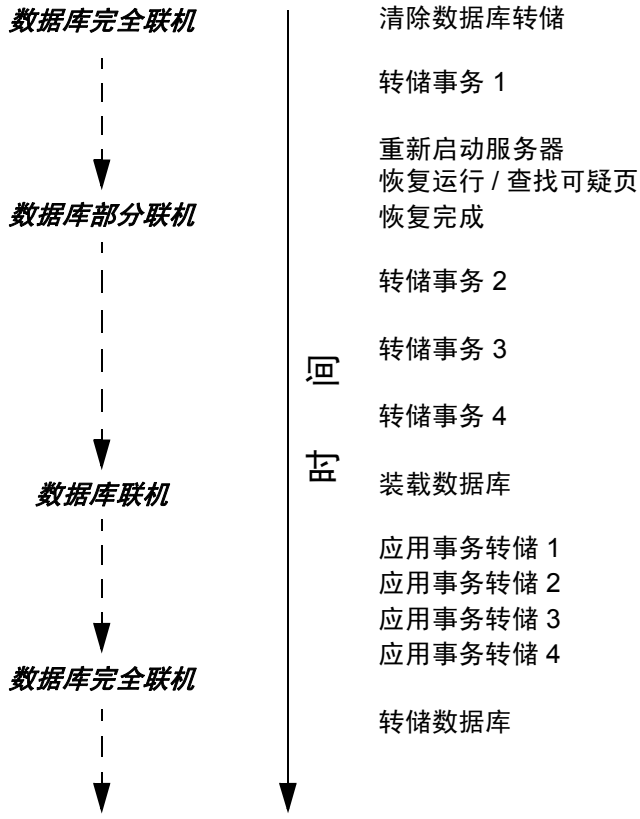
重装需要从备份中恢复一个干净的数据库。方便时，装载最近的干净数据库转储，并应用事务日志以恢复数据库。

load database 从 master.dbo.sysdatabases 和 master.dbo.sysattributes 系统表中清除可疑页信息。

当已恢复的数据库进入联机状态时，立即转储该数据库。

图 12-1 说明了用于重装数据库的策略。

图 12-1：重装策略



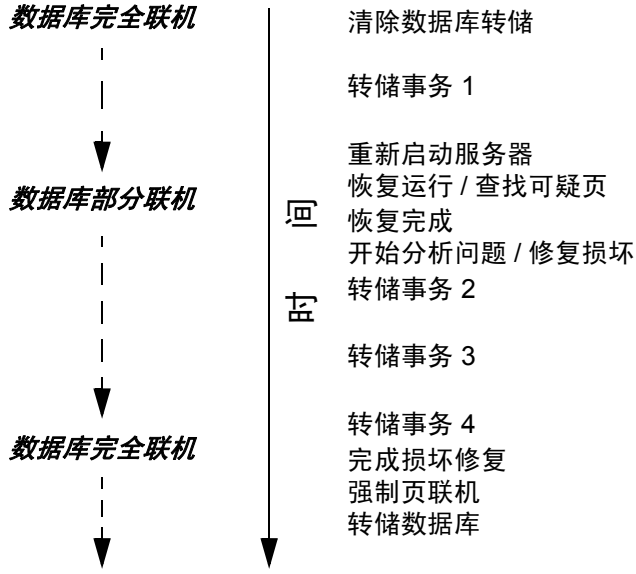
修复策略

修复策略涉及在数据库部分联机时修复损坏的页。使用已知方法诊断和修复问题，包括 dbcc 命令、针对可疑页查询已知结果，以及根据需要致电 Sybase 技术支持部门。修复损坏还包括删除并重新创建包含可疑页的对象。

您可以使用 sp_forceonline_page 在修复部分脱机页后将这些脱机页分别联机，或者待所有脱机页都被修复后使用 sp_forceonline_db 一次性使它们联机。

修复策略不要求整个数据库都脱机。图 12-2 说明了用于修复损坏页的策略。

图 12-2：修复策略



评估损坏的程度

有时您可使用恢复故障隔离来评估损坏的程度，方法是强制运行恢复过程和检查被标记为可疑的页的数目以及这些页所属的对象。

例如，如果用户报告某个数据库有问题，可以将恢复故障隔离模式设置为 “页”，并通过重新启动 Adaptive Server 强制恢复。恢复完成后，使用 sp_listsuspect_db 或 sp_listsuspect_page 确定有多少个可疑页以及哪些数据库对象受影响。

如果整个数据库都被标记为可疑且收到以下消息：

```
Reached suspect threshold '%d' for database '%.*s'
.Increase suspect threshold using
sp_setsuspect_threshold.
```

使用 sp_setsuspect_threshold 可增加可疑增加阈值并强制恢复再次运行。每次收到此消息时，都可以提高该阈值并运行恢复过程，直到数据库进入联机状态。如果未收到此消息，则表明损坏内容未被隔离到特定页，在这种情况下，用于确定可疑页的数目的这一策略将不起作用。

使用 *dump* 和 *load* 命令

遇到介质故障 （如磁盘崩溃）时，仅当您有数据库的定期备份及其事务日志的情况下才可恢复这些数据库。完整恢复依赖于定期使用 dump database 和 dump transaction 命令备份数据库以及使用 load database 和 load transaction 命令恢复它们。下面将对这些命令进行简要描述，有关详细信息，请参见第 13 章 “备份和恢复用户数据库”和第 14 章 “恢复系统数据库”。

警告！切勿使用操作系统复制命令复制操作数据库设备。对复制的设备运行 Adaptive Server 可能导致数据库损坏。关闭 Adaptive Server，或者使用 quiesce database 命令保护复制操作。

转储命令可成功完成执行，即使数据库已被损坏。在备份数据库之前，请使用 dbcc 命令检查它的一致性。请参见第 11 章 “检查数据库一致性”。

警告！如果直接转储到磁带，则不要在该磁带上存储任何其它类型的文件 （UNIX 备份、目标文件，等等）。如果不这样，将使 Sybase 转储文件失效。但是，如果转储到 UNIX 文件系统，则可以将结果文件存档到磁带中。

进行例行数据库转储：*dump database*

dump database 命令可复制整个数据库，包括数据和事务日志。dump database 不会将日志截断。

dump database 允许**动态转储**，这意味着，用户可以在执行转储时继续更改数据库。因此，可以方便地定期备份数据库。

进行例行事务日志转储：*dump transaction*

使用 dump transaction 命令可定期备份事务日志。 dump transaction 类似于许多操作系统提供的增量备份。它复制事务日志，并提供自上一次事务日志转储以来对数据库所进行的所有更改的记录。 dump transaction 复制完日志后，会截断其中不活动的部分。

dump transaction 比完全数据库备份所花费的时间和存储空间要少，一般更为经常使用。用户可以在转储数据库时继续对该数据库进行更改。只有在数据库将其日志存储在单独的段上时，才可运行 dump transaction。

遇到介质故障后，使用 dump transaction 的 with no_truncate 选项备份事务日志。这样，可提供一直到发生故障时的事务日志的记录。

设备出现故障后复制日志：*dump tran with no_truncate*

如果数据设备出现故障，并且数据库不可访问，则使用 dump transaction 的 with no_truncate 选项获取日志的当前副本。此选项不会截断日志。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。

恢复整个数据库：*load database*

使用 load database 命令可装载使用 dump database 创建的备份。可以将转储装载到先前存在的数据库中，或使用 for load 选项创建一个新数据库。创建新数据库时，所分配的空间至少要与分配给原始数据库的空间相等。

load database 命令将数据库状态设置为 “脱机”。这意味着在装载数据库之前不必使用 sp_dboption 的 no chkpt on recovery、dbo use only 和 read only 选项。但是，在装载数据库和随后装载事务日志期间，任何人都不能使用数据库。若要使用户可以访问该数据库，请发出 online database 命令。

装载数据库后， Adaptive Server 可能需要：

- 将所有未用的页 “置零”（如果正在装入的数据库比被转储的数据库大）。
- 完成恢复，将事务日志的更改应用到数据。

根据未分配页或长事务的数量，这可能需要几秒钟时间；对于非常大型的数据库，可能需要数个小时。 Adaptive Server 会发出消息，说明它正在对页 “置零”，或已开始恢复。这些消息通常存储于缓冲区中，若要查看它们，请使用：

```
set flushmessage on
```

将更改应用到数据库：*load transaction*

装载数据库之后，使用 load transaction 命令按各个事务日志转储的创建顺序装载它们。此进程通过重新执行事务日志中记录的更改来重建数据库。如有必要，可以通过使用 load transaction 的 until_time 选项将数据库向前滚动到其事务日志中某个特定时间，开始恢复数据库。

在 load database 和 load transaction 命令之间，用户不能对数据库进行更改，因为 load database 已将状态设置为 “脱机”。

您只能装载与相关数据库位于相同释放级别的事务日志转储。

装载完事务日志转储的整个序列后，数据库会显示上一次事务日志转储时提交的所有事务。

使用户可以使用数据库：*online database*

当装载序列完成后，将数据库的状态改为 “联机”，以便用户可以使用数据库。除非发出 online database 命令，否则仍旧无法访问由 load database 装载的数据库。

在发出 online database 之前，一定要装载全部所需的事务日志。

跨平台转储和装载数据库

Adaptive Server 允许您跨体系结构规模不同的平台转储和装载数据库。这意味着您可以从大型平台到小型平台，或从小型平台到大型平台执行 dump database 和 load database。

在大型系统中，存储类型 （如整型或长整型）的最重要字节在较低地址空间。反之，对于小型系统，存储类型的最重要字节在较高地址空间。

注释 在跨体系结构规模不同的平台执行 dump database 和 load database 时，用户和系统数据不需要转换。转储和装载数据库时，对操作没有任何限制。

Adaptive Server 在执行 load database 的过程中，自动检测数据库转储文件所在起始系统的体系结构类型，然后执行必要的转换。而且它还支持从较早版本（如 11.9、12.0 和 12.5 版）进行装载。转储和装载过程可以从 32 位平台向 64 位平台执行，反之亦然。

支持的平台：

大型平台	Sun Solaris	IBM AIX	HPPA、HPIA 上的 HP-UX
小型平台	Linux IA	Windows	Sun Solaris x86

对于某些平台组合，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。

转储数据库

对于跨平台的转储和装载，在运行 dump database 之前，请使用以下过程将数据库的状态更改为事务性抑制状态：

1　执行 dbcc checkdb 和 dbcc checkalloc 以检验数据库是否在顺利运行。

2　使用 sp_dboption 将数据库置于单用户模式。

3　使用 sp_flushstats 将统计数据刷新为 systabstats。

4　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。

5　对该数据库运行 checkpoint 以刷新已更新的页。

6　运行 dump database。

装载数据库

装载数据库后， Adaptive Server 会自动标识转储文件上的规模类型，并在 load database 和 online database 命令执行期间执行全部所需的转换。

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 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

在执行跨平台转储和装载操作期间，将按如下方式处理可疑分区：

- 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。
- 循环分区 （已使用 unichar 或 varchar 分区键在内部生成分区条件）上的任何全局聚簇索引都会被标记为可疑索引。
- 数据库联机后，使用 sp_post_xpload 可修复可疑分区和索引。

关于转储和装载数据库和事务的限制

- dump transaction 和 load transaction 不允许跨平台使用。
- 不支持跨平台向 / 从远程 Backup Server 执行 dump database 和 load database。
- 不能跨平台装载有口令保护的转储文件。
- 如果针对已分析的 XML 对象执行 dump database 和 load database，则必须在 load database 完成后再次分析文本。
- 只能将转储装载到与从中转储它们的服务器具有相同排序顺序的服务器。例如，您无法将转储从使用词典顺序、区分大小写、区分重音排序顺序的服务器装载到使用词典顺序、不区分大小写、不区分重音排序顺序的服务器。
- 对于版本早于 11.9 的 Adaptive Server，不能跨平台执行 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 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 系统表中的登录记录从 Solaris 迁移到 Linux，则可以将 bcp 与字符格式配合使用。Solaris 平台上的登录口令与此版本中没有跟踪标志的 Linux 平台上的登录口令兼容。对于所有其它组合和平台，由于口令不兼容，因此需要重新创建登录记录。

性能注释

为了快速访问表的数据行，对索引行进行了排序。包含行标识符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。

一般而言，需要进行计划以在大型表上重新创建索引，并且这将是一个冗长的过程。

sp_post_xpload 验证索引，删除无效索引，然后通过一条命令在数据库上重新创建已删除的索引。因为 sp_post_xpload 执行许多操作，所以删除并重新创建索引所需的时间会较长。将 sp_post_xpload 用于小于 10G 的数据库。对于大于 10G 的数据库，Sybase 建议您删除并重新创建索引。

将数据库移到另一 Adaptive Server

您可以使用 dump database 和 load database 将数据库从一个 Adaptive Server 移至另一个 Adaptive Server。但是，必须要确保目标 Adaptive Server 上的设备分配情况与原始 Adaptive Server 上的设备分配情况相匹配。否则，新数据库中的系统段和用户定义的段将与原始数据库中的那些段不匹配。

若要在将数据库转储装载到新的 Adaptive Server 时保留设备分配状态，使用的指令应与由于设备故障而恢复用户数据库时所用的指令相同。请参见第 365 页的 “检查空间使用情况”。

此外，将系统数据库移到不同设备上时，应遵循以下一般准则：

- 移动 master 数据库之前，务必取消主设备的镜像。如果不这样做，在您用新设备启动 Adaptive Server 时，Adaptive Server 会尝试使用旧的镜像设备文件。
- 移动 master 数据库时，应使用与原设备的大小相同的新设备，以避免 sysdevices 中的分配错误。
- 若要移动 sybsecurity 数据库，请先将新数据库置于单用户模式，然后再将旧数据装载到该数据库中。

升级用户数据库

可以从版本 11.9 或任何更高版本的 Adaptive Server 将转储装载到当前版本的 Adaptive Server。在发出 online database 命令之前，不会升级所装载的数据库。

升级用户数据库的步骤与升级系统数据库的步骤相同：

1 使用 load database 装载 11.9 版或更高版本的 Adaptive Server 的数据库转储。 load database 将数据库状态设置为 “脱机”。

2 使用 load transaction 命令来*按顺序*装载上次数据库转储后生成的所有事务日志。在执行第 3 步之前，装载所有事务日志。

3 使用 online database 升级数据库。 online database 命令会升级数据库，因为其当前状态与 Adaptive Server 的当前版本不兼容。升级完成后，数据库状态被设置为 “联机”，这样该数据库便可供使用。

4 对已升级的数据库进行转储。必须在 dump database 命令发生后，才允许执行 dump transaction 命令。

请参见 《参考手册：命令》。

使用特殊 *dump transaction* 选项

在某些情况下，以上所述的简单模型不适用。表 12-1 说明何时使用特殊的 with no_log 和 with truncate_only 选项，而不用标准的 dump transaction 命令。

警告！ 仅按表 12-1 中的说明使用特殊的 *dump transaction* 命令。需要特别指出的是，应将 dump transaction with no_log 作为最后一种手段来使用，并仅在 dump transaction with no_truncate 失败后使用此命令一次。 dump transaction with no_log 命令仅释放事务日志中非常少的空间。如果在输入 dump transaction with no_log 后继续装载数据，日志可能会被完全填满，从而导致随后的 dump transaction 命令失败。应使用 alter database 命令为数据库分配额外的空间。

表 12-1：何时使用 dump transaction with truncate_only 或 dump transaction with no_log

情形	Use
日志与数据在同一段时。	dump transaction with truncate_only 截断日志 dump database 复制包括日志在内的整个数据库
在不用关心当前事务的恢复时（例如，在早期开发环境中）。	dump transaction with truncate_only 截断日志 dump database 复制整个数据库
由于日志空间不足而导致转储事务日志的常规方法（标准的 dump transaction 命令或 dump transaction with truncate_only）失败时。	dump transaction with no_log 截断日志，而不记录事件之后立即使用 dump database 来复制包括日志在内的整个数据库

使用特殊装载选项标识转储文件

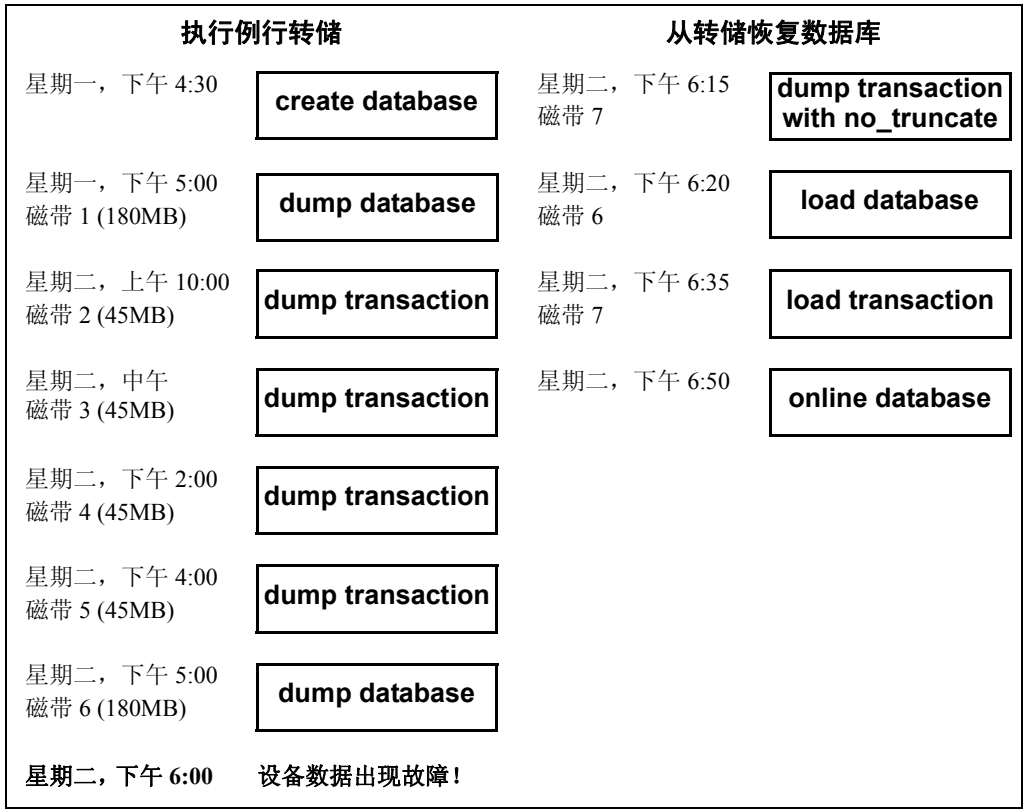
使用 with headeronly 选项提供磁带上的指定文件或第一个文件的标头信息。使用 with listonly 选项返回有关磁带上所有文件的信息。这些选项实际不装载磁带上的数据库或事务日志。

注释　这些选项是互斥的。如果同时指定两者，则 with listonly 优先。

从备份恢复数据库

图 12-3 说明恢复星期一下午 4:30 创建的数据库，之后立即进行转储操作的进程。完全数据库转储在每天下午 5:00 进行。事务日志转储在每天上午 10:00、中午 12:00、下午 2:00 和下午 4:00 进行：

图 12-3：恢复数据库，方案一



如果存储数据的磁盘在星期二下午 6:00 出现故障，应遵循以下步骤来恢复数据库：

1 使用 dump transaction with no_truncate 获取当前事务日志转储。

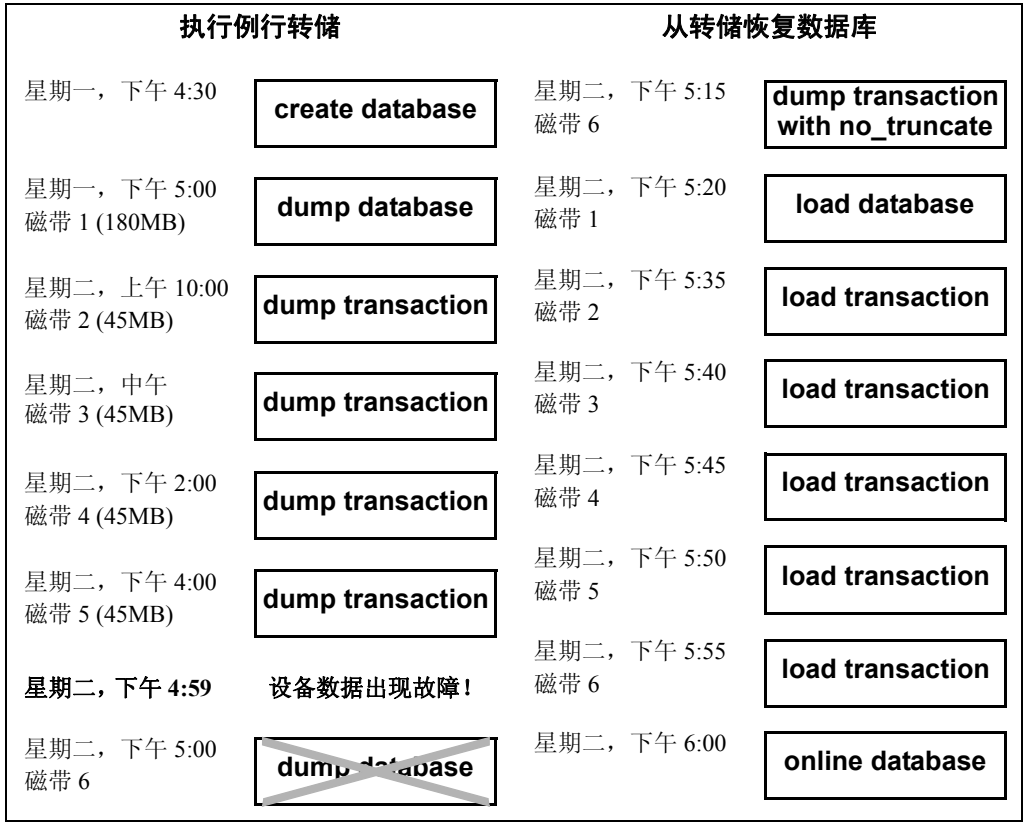
2 使用 load database 装载最新的数据库转储，即磁带 6。 load database 将数据库的状态设置为 “脱机”。

3 使用 load transaction 应用最新的事务日志转储，即磁带 7。

4 使用 online database 将数据库状态设置为 “联机”。

图 12-4 说明数据设备在星期二下午 4:59 出现故障后 （在操作员按预定计划进行每晚的数据库转储之前），如何恢复数据库：

图 12-4：恢复数据库，方案二



若要恢复数据库，请执行以下操作：

1　使用 dump transaction with no_truncate 在磁带 6（将用于常规数据库转储的磁带）上获取当前的事务日志转储。

2　使用 load database 装载最新的数据库转储，即磁带 1。 load database 将数据库状态设置为 “脱机”。

3　使用 load transaction 装载磁带 2、3、4 和 5，以及最新的事务日志转储，即磁带 6。

4　使用 online database 将数据库状态设置为 “联机”。

挂起和恢复对数据库的更新

quiesce database hold 用于在对每个数据库设备执行磁盘取消镜像或外部复制时，阻塞对一个或多个数据库的更新。因为在此期间未执行任何写入，所以数据库的外部副本 （辅助映像）与主映像一致。当数据库处于抑制状态时，允许对数据库上的操作进行只读查询。若要恢复对数据库的更新，请在完成外部复制操作后，发出 quiesce database release 命令。可将数据库的外部副本装载到辅助服务器上，确保拥有与主映像一致的事务性副本。可从一个 isql 连接发出 quiesce database hold，然后用另一 isql 连接登录并发出 quiesce database release。请参见 《参考手册：命令》。

注释 *tag_name* 必须遵循标识符规则。必须为 *quiesce database...hold* 和 quiesce database...release 使用相同的 tag_name。

例如，若要挂起对 pubs2 数据库的更新，请输入：

```
quiesce database pubs_tag hold pubs2
```

Adaptive Server 将如下消息写入错误日志：

```
QUIESCE DATABASE command with tag pubs_tag is being executed by process 9.
Process 9 successfully executed QUIESCE DATABASE with HOLD option for tag
pubs_tag.Processes trying to issue IO operation on the quiesced database(s) will
be suspended until user executes Quiesce Database command with RELEASE option.
```

对 pubs2 数据库的任何更新都将延迟到数据库释放，届时更新将会完成。若要释放 pubs2 数据库，请输入：

```
quiesce database pubs_tag release
```

释放数据库之后，可以使用 -q 参数弹出辅助服务器，但前提是使用了 for external dump 子句。恢复使数据库达成事务性一致，或者可以等到数据库进入联机状态，再应用事务日志。

使用 quiesce database 的准则

使用 quiesce database 的最简单方式是完全复制整个安装，这可确保系统映射保持一致。当将包含这些映射的系统数据库作为 quiesce database hold 的数据库集的一部分实际进行复制时，这些映射将转至辅助安装中。当源安装中的所有用户数据库作为同一组的一部分被复制时，实现这些映射。 quiesce database 允许在单个操作中存在八个数据库名称。如果源安装具有八个以上的数据库，则可以发出 quiesce database hold 的多个实例以便为多组数据库创建多个并发抑制状态。

要创建新的源安装，可以使用几乎相同的脚本创建主安装和辅助安装。辅助安装的脚本可能在传递到 disk init 命令的物理设备名称中有所不同。此方法要求，对主服务器上系统设备的更新要通过同样的更改反映到辅助服务器上。例如，如果在主服务器上执行 alter database 命令，那么在辅助服务器上也必须用相同的参数执行相同的命令。此方法要求，数据库设备由卷管理器支持，卷管理器可为主服务器和辅助服务器显示设备的相同物理设备名称 （物理上不同且单独的）。

您的节点可能为制作数据库设备的外部副本开发自己的过程。然而， Sybase 建议：

- 将 master 数据库包括在 quiesce database 的数据库列表中。
- 任何无法向磁盘中的抑制数据库进行写入的进程都可能拥有阻止其它进程执行的资源。例如，如果某进程修改事务中的数据库页，但无法在 commit 期间刷新日志页，则该进程可能拥有排它页锁，并且可能会阻止读者尝试在 quiesce database 操作执行期间获取同一页上的共享页锁。

  虽然当您抑制系统数据库 （sybsystemprocs、 sybsystemdb 或 sybsecurity，前提是审计处于启用状态）时可能会发生此问题，但当您由于 master 数据库包含许多常使用的系统表而 quiesce master 数据库时，此问题最严重。例如，如果某进程使用 create login 修改 syslogins，但无法在抑制 master 数据库期间提交事务，则为修改 syslogins 而获取的排它锁会阻止任何登录，这是因为这些登录必须在 syslogins 上获取共享页锁。

  注释　抑制 master 数据库或任何系统数据库可能会大大影响服务器性能，这是因为，这样做会阻止任何尝试更新已抑制数据库的进程。

- 在主服务器上和辅助服务器上使用相同字符串命名设备。

- 使主安装和辅助安装中的 master、 model 和 sybsystemprocs 系统数据库的环境相同。需特别注意的是，复制的数据库的 sysusages 映射和数据库 ID 在主服务器和辅助服务器上必须相同，这两种服务器的数据库 ID 在 sysdatabases 中也必须相同。
- 保持 syslogins.suid 和 sysusers.suid 之间的映射在辅助服务器中一致。
- 如果主服务器和辅助服务器共享一个 master 的副本，且每一复制设备的 sysdevices 条目都使用相同的字符串，那么这两个服务器中的 *physname* 值必须在物理上不同且是单独的。
- 使用下列限制生成数据库的外部副本：
  - 复制进程只能在 quiesce database hold 完成之后开始。
  - quiesce database 的数据库列表中每个数据库的每一设备都必须被复制。
  - 外部复制操作必须首先结束，然后才能调用 quiesce database release。
- 在 quiesce database 为外部复制操作提供的间隔期间，对于属于 quiesce database 的数据库列表中任何数据库的任何磁盘空间的更新操作都将被阻止。此空间在 sysusages 中定义。然而，如果某设备上的空间被 quiesce database 的数据库列表中的数据库和不在该列表中的数据库共享，那么在制作外部副本时，可能会更新该共享设备。当您决定使用系统中的哪些数据库来计划制作外部副本时，可：
  - 隔离数据库，使它们不共享要使用 quiesce database 的环境中的设备，或
  - 计划复制设备上的所有数据库 （这样做符合上述制作整个安装的副本的建议）。
- 仅当数据库中具有少量更新活动时才应使用 quiesce database （首选在执行只读活动期间）。在空闲时间抑制数据库时，不但很少有用户觉得不便，而且根据要执行外部复制的第三方 I/O 子系统的不同，复制操作所涉及的同步设备所需的时间也可能会减少。
- 使用 mount 和 unmount 命令，可以更加轻松地移动或复制数据库。您可以在不重新启动服务器的情况下将数据库从一个 Adaptive Server 移动或复制到另一个 Adaptive Server，也可以一次移动或复制多个数据库。

  也可以使用这些命令物理地移动设备，然后重新激活数据库。

当您 unmount 数据库时，会将数据库及其设备从 Adaptive Server 中删除。 unmount 会关闭数据库并将它从 Adaptive Server 中删除；此外，也将停用并删除设备。在卸下时不会对数据库或数据库的页进行任何更改。

维护在主服务器和辅助服务器关系中的服务器角色

如果您的节点由两个 Adaptive Server 组成，一个作为主服务器使用，另一个作为接收主服务器数据库的外部副本的辅助服务器，一定不要混淆它们各自的角色。即，每台服务器所担任的角色可以更改 （主服务器和辅助服务器的角色可互换），但这些角色不能同时由同一服务器担当。

使用 *-q* 选项启动辅助服务器

dataserver -q 选项标识辅助服务器。不要使用 -q 选项来启动主服务器。用 -q 选项，在 quiesce database for external dump 期间复制的用户数据库保持脱机状态，直到：

- 用 standby access 转储主服务器上数据库的事务日志 （即，dump tran with standby_access），随后对辅助服务器上此数据库的副本执行 load tran，再对此数据库执行 online database for standby access。
- 通过发出 online database 命令强制数据库进入联机状态以进行读写访问。然而，如果这样做，数据库恢复过程将写入补偿日志记录，并且在未装载数据库或未使用 quiesce database 制作主设备的新副本的情况下，无法装载事务日志。

系统数据库会联机 （无论 -q 选项如何），并为全部回滚的事务编写补偿日志记录。

更新的 “in quiesce” 数据库日志记录值

如果使用 dataserver 的 -q 选项启动辅助服务器，则对于每个在内部标记为 “处于抑制状态” 的用户数据库， Adaptive Server 会在启动时发出消息，指定数据库处于 " 用于使用 "。

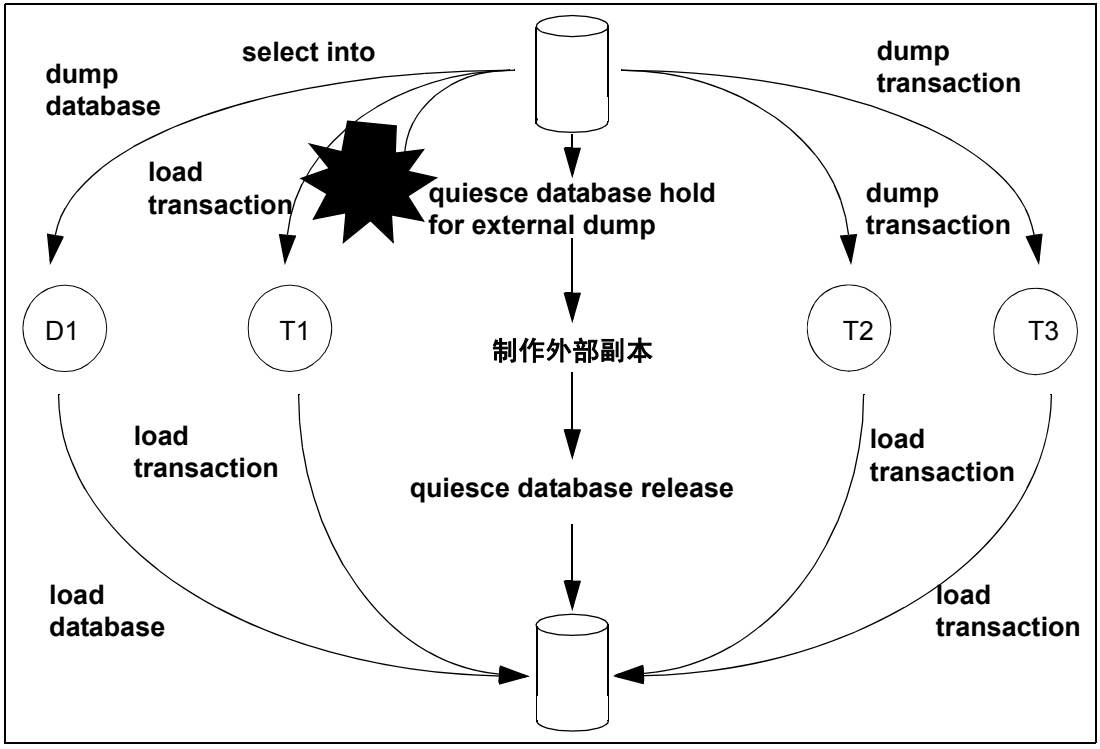
quiesce database for external dump 复制的数据库的 -q 恢复与用于 load database 命令的恢复大体相同。像 load database 的恢复一样，dataserver -q 会在内部记录上一个当前日志记录的地址，以便后续 load transaction 可以将此地址与上一个当前日志记录的地址进行比较。如果这两个值不一致，则辅助数据库中一直有活动，并且 Adaptive Server 会引发错误号 4306。

更新转储序列号

与 dump database 一样，如果有未记录的写入操作，quiesce database 会更新转储序列号。这样就可防止您错误地将较早的数据库转储或外部副本作为转储序列的基础。

例如，在使用图 12-5 描述的热备份方法中，存档文件由 dump database (D1)、 dump transaction (T1)、 quiesce database、 dump transaction (T2) 和 dump transaction (T3) 生成：

图 12-5：热备份转储序列



通常情况下，在含有已记录的更新并且没有 dump tran with truncate_only 的环境中，可以依次装载 D1、 T1、 T2 和 T3，忽略任何 quiesce database hold。此方法用于热备份情况，其中在主服务器上继续数据库转储可简化介质故障恢复方案。在用于决策支持系统的辅助 （即备份）服务器上，可以首选继续增量应用 load transaction 而是不中断外部复制操作。

不过，如果在生成 T1 的 dump transaction 操作后发生未记录的操作（如图 12-5 中发生的 *select into*），则不允许以后的 dump transaction to archive，并且必须要么创建数据库的另一转储，要么发出 quiesce database for external copy，然后制作数据库的新外部副本。发出这些命令中的任何一个将更新转储序列号，并清除阻塞 dump transaction to archive 的标记。

是否使用 for external dump 子句取决于您希望恢复如何处理将标记为 `in quiesce` 的抑制数据库。

quiesce database hold

如果发出 quiesce database 并且不使用 for external dump 子句，则在用于创建辅助数据库集的外部复制操作期间，辅助服务器没有运行，并且使用 -q 执行的恢复不会将任何已复制的数据库视为处于 “抑制状态”。在启动恢复期间以正常方式恢复每台服务器，但不会按照前面所述将它们恢复为 for load database。随后，对任何这些数据库执行 load tran 的尝试都被禁止，并发出错误 4306 `"There was activity on database since last load ..."` 或错误 4305 `"Specified file '%.*s' is out of sequence ..."`

无论主数据库中是否有未记录的活动，转储序号都不会递增 quiesce database hold，并且 quiesce database release 不会清除未记录的写入位。

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。

quiesce database hold for external dump

当您发出 quiesce database for external dump 命令时，数据库的外部副本 “记起”它是在抑制间隔期间生成的，以便 -q 恢复能像 load database 一样将其恢复。 quiesce database release 从主数据库中清除此信息。如果未记录的写入操作已在主服务器上阻止 dump tran *to archive*，现在即会启用 dump tran *to archive*。

对于任何位于 quiesce database 列表中的数据库，如果自上次执行 dump database 或 quiesce database hold for external dump 以来发生了未记录的写入，则转储序号由 quiesce database hold for external dump 更新，并且 quiesce database release 会清除未记录的写入信息。如果更新的序列号应用到由更新它的 quiesce database 创建的外部副本以外的目标，会导致 load tran 失败。这类似于带有未记录的写入状态的数据库的 dump database 的行为。

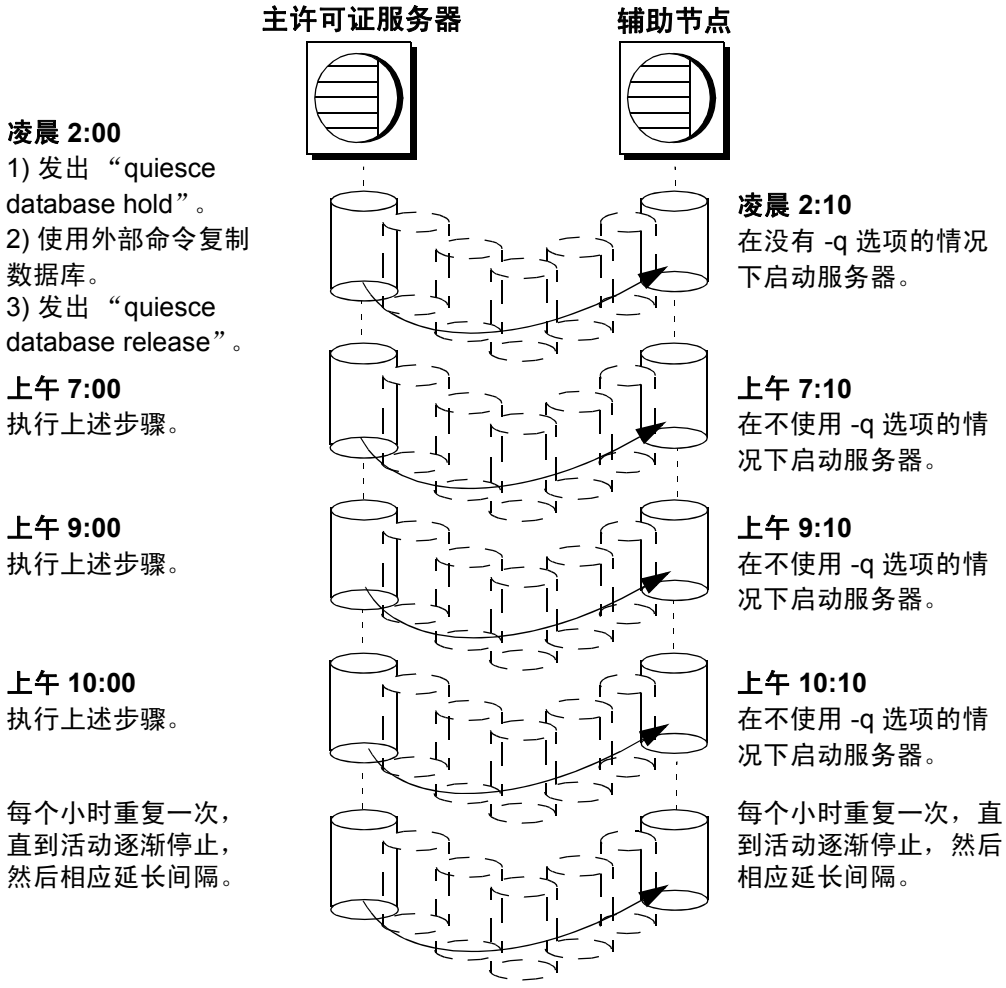
警告！ quiesce database for external dump 清除阻止执行 dump transaction to *archive_device* 的内部标志，而不管您实际生成了外部副本还是执行了数据库转储。 quiesce database 无法知道您是否生成了外部副本。履行此义务是您的职责所在。如果使用 quiesce database hold for external dump 来影响瞬时写入保护，而不是实际执行作为新转储序列基础的副本，且您的应用程序包括偶然的未记录的写入，那么 Adaptive Server 可能允许创建不能使用的事务日志转储。在此情况下， dump transaction to *archive_device* 最初可以成功完成，但以后的装载事务命令可能由于顺序混乱而拒绝这些存档。

使用 *quiesce database* 备份主设备

通常，用户通过 quiesce database 命令使用下列方法之一备份其数据库。这两种方法都允许您在正常操作过程中从联机事务处理 (OLTP) 服务器上卸载决策支持应用程序：

- 主设备的迭代刷新 — 在刷新间隔，将主设备复制到辅助设备。在每次刷新前抑制数据库。在图 12-6 中显示了使用此系统提供每周备份的系统：

图 12-6：迭代刷新方法的备份安排

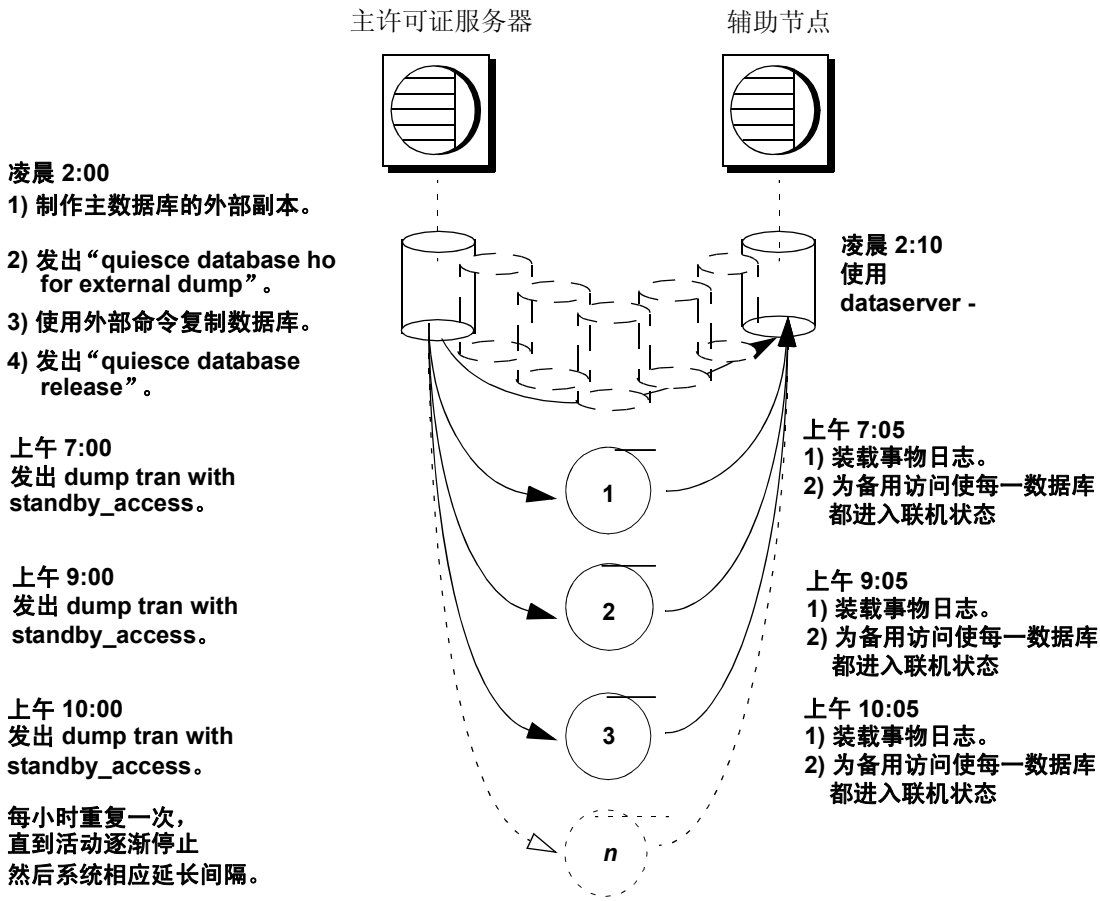


如果您使用的是迭代刷新方法，则不必使用 -q 选项重新启动辅助服务器 （在崩溃或系统维护之后）。任何不完全的事务都会生成补偿日志记录，并且受影响的数据库以正常方式进入联机状态。

- 热备份方法 — 允许 OLTP 服务器完全并行操作，因为它不阻塞写入。

使用 for external dump 子句制作主数据库设备的外部（辅助）副本之后，通过使用来自主服务器的转储定期应用事务日志来刷新辅助数据库。对此方法，抑制数据库一次以制作该组数据库的外部副本，然后使用 dump tran with standby_access 定期刷新每一数据库。在图 12-7 中显示了一个系统，该系统使用主设备进行每日更新、然后每小时备份事务日志。

图 12-7：热备份方法的备份安排



热备份方法的数据库恢复

如果使用热备份方法， Adaptive Server 必须知道是启动主服务器还是启动辅助服务器。使用 dataserver 命令的 -q 选项来指定启动辅助服务器。如果不用 -q 选项启动服务器：

- 数据库会正常恢复，而不是像对于 load database 一样。
- 当发出 quiesce database 时，所有未提交的事务都被回退。

请参见第 307 页的 “使用 -q 选项启动辅助服务器”。

根据数据库是否标记了 `in quiesce`，恢复序列的进展情况也有所不同。

未标记 “in quiesce” 的数据库的恢复

使用 -q 选项时，如果未将数据库标记为 `in quiesce`，那么它像在主服务器中一样进行恢复。即，如果数据库当前不在先前操作的装载序列中，那么它将被完全恢复并进入联机状态。任何未完成的事务都将回滚，并且会在恢复期间写入补偿日志记录。

标记为 “in quiesce” 的数据库的恢复

- 用户数据库 — 标记了 `in quiesce` 的用户数据库以使用 load database 命令时相同的方式恢复。这将使 load tran 能够检测到服务器脱机以后在主数据库中发生的任何活动。用 -q 选项启动了辅助服务器后，恢复进程遇到 `in quiesce` 标记。 Adaptive Server 发出一条消息，指出该数据库处于装载序列中，并且处于脱机状态。如果使用热备份方法，则在装载了由 dump tran with standby_access 生成的第一个事务转储前，不要将数据库置于联机状态以用作决策支持系统。然后使用 online database for standby_access。
- 系统数据库 — 系统数据库立即完全联机。该 `in quiesce` 标记被清除并忽略。

在抑制状态下制作存档副本

quiesce database hold for external dump 表示您想要在抑制状态下制作数据库的外部副本。因为这些外部副本是在您发出 quiesce database hold 之后创建的，数据库在事务上保持一致，这是因为您可确保在 quiesce database hold 和 quiesce database release 之间的时间段内不会进行写入，并且恢复可与启动恢复以相同方式运行。在第 308 页的图 12-5 中对此过程进行了说明。

如果环境没有未记录的更新并且不包括 dump tran with truncate_only，可以依次装载 D1、 T1、 T2 和 T3，忽略全部 quiesce database...hold 命令。但是，如果在生成 T1 的转储事务以后发生未记录的操作（如图 12-5 中所示的 select into），则不再允许将事务转储到档案。

通过清除阻止下一个 dump transaction to archive 的状态位以及更改外部副本的序列号以创建装载序列的基础，使用 quiesce database hold for external dump 子句可解决此问题。但是，如果没有未记录的写入，序列号则不会增加。

无论是否使用 for external 转储子句，您都可以制作数据库的外部副本。但是，若要将主服务器中的后续事务转储应用到辅助服务器，请包括 for external dump 子句：

```
quiesce database tag_name hold db_name [, db_name]
...[for external dump]
```

例如：

```
quiesce database pubs_tag hold pubs2 for external
dump
```

假设自启动数据库的主实例以来数据库映射尚未更改，则可以通过下列步骤为单个数据库创建外部转储：

1 发出：

```
quiesce database pubs_tag hold pubs2 for external
dump
```

2 使用适合您的节点的方法制作数据库的外部副本。

3 发出：

```
quiesce database pubs_tag release
```

警告！ 清除状态位并更新序列号使您可执行转储事务，而不管在发出 quiesce database 以后是否实际制作了外部副本。 Adaptive Server 无法知道您在 quiesce database... hold for external dump 和 quiesce database... release 之间是否制作了外部副本。如果使用 quiesce database hold for external dump 命令影响瞬时写入保护，而非实际执行作为新转储序列基础的副本，且您的应用程序包括偶然的未记录写入，那么 Adaptive Server 允许创建不能使用的事务日志转储。dump transaction to *archive_device* 成功完成，但 load transaction 因为顺序混乱而拒绝这些存档。

使用 *mount* 和 *unmount* 命令

使用 mount 和 unmount 命令，可以更加轻松地移动或复制数据库。您可以在不重新启动服务器的情况下将数据库从一个 Adaptive Server 移动或复制到另一个 Adaptive Server（不同于使用 dump 和 load database，这两条命令将数据库复制到磁带或磁盘）。您可以使用 mount 和 unmount 命令一次移动或复制多个数据库。

也可以使用这些命令物理地移动设备，然后重新激活数据库。

请参见第 7 章 “装入和卸下数据库”。

警告！ 在 Adaptive Server 中的数据库中不会维护到登录名的直接映射。这意味着，对于允许对原始 Adaptive Server 上的数据库进行访问的每一登录，在目标 Adaptive Server 上必须存在同一 suid 的相应登录。

为使权限和保护保持不变，辅助 Adaptive Server 上的登录映射必须与第一个 Adaptive Server 上的文件相同。

使用 Backup Server 执行备份和恢复

转储和装载由 Backup Server 执行， Backup Server 是与 Adaptive Server 运行在同一计算机上的一种 Open Server 程序。可以通过网络执行备份操作，在远程计算机上使用一个 Backup Server，在本地计算机上使用另一个 Backup Server。

注释　Backup Server 无法转储到多磁盘卷。

Backup Server：

- 从 “分条的转储”中创建和装载。**转储分条**允许并行使用最多 32 个备份设备。这会将数据库分成大体相等的几部分，并将每个部分备份到一个单独的设备上。
- 创建和装载跨多个磁带的单个转储。
- 通过网络转储并装载到运行在另一计算机上的 Backup Server。
- 将多个数据库或事务日志转储到单个磁带上。
- 从包含许多数据库或日志转储的磁带装载单个文件。

- 支持平台特定的磁带处理选项。
- 将批量处理请求定向到发出 dump 或 load 命令的会话或其操作员主控台中。
- 检测转储设备的物理特性，以便确定协议、块大小和其它特性。

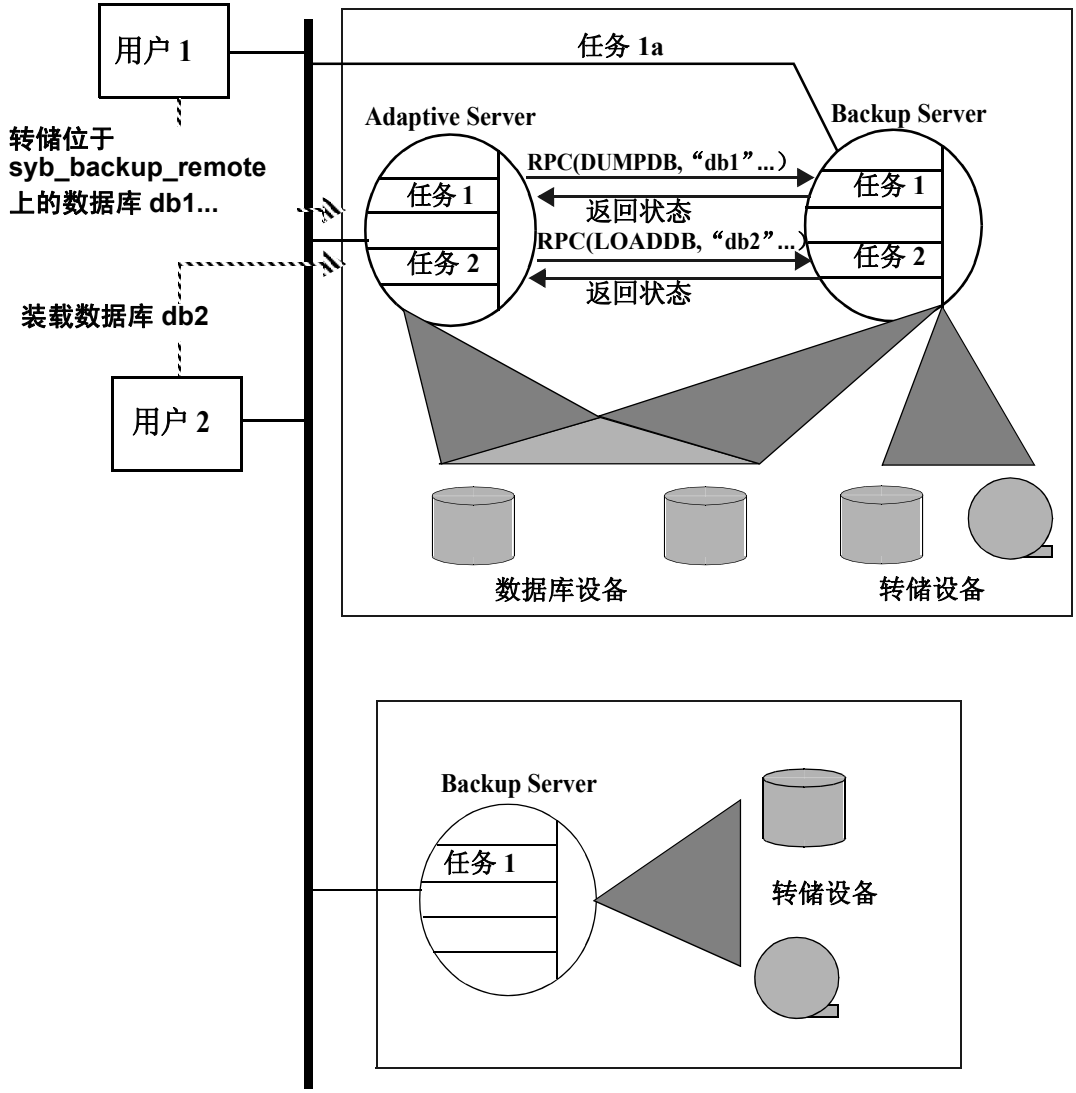
图 12-8 显示同时在两个数据库上执行备份活动的两个用户：

- User1 将数据库 db1 转储到远程 Backup Server。
- User2 从本地 Backup Server 装载数据库 db2。

每名用户都从 Adaptive Server 会话发出适当的转储或装载命令。Adaptive Server 解释该命令并将远程过程调用 (RPC) 发送给 Backup Server。这些调用指明要转储或装载哪些数据库页、要使用哪些转储设备以及其它选项。

执行转储和装载操作时，Adaptive Server 和 Backup Server 使用 RPC 来交换指令和状态消息。 Backup Server （而不是 Adaptive Server）针对 dump 和 load 命令执行所有数据传输。

图 12-8：Adaptive Server 与具有远程 Backup Server 的 Backup Server



当本地 Backup Server 收到用户 1 的转储指令时，它从数据库设备读取指定页，并将它们发送给远程 Backup Server。远程 Backup Server 将数据保存到脱机介质。

同时，本地 Backup Server 执行用户 2 的装载命令，从本地转储设备读取数据并将其写入数据库设备。

与 Backup Server 通信

要使用 dump 和 load 命令， Adaptive Server 必须能够与其 Backup Server 进行通信。下面是一些相关要求：

- Backup Server 必须与 Adaptive Server 在同一计算机上 （或 OpenVMS 的同一集群上）运行。
- Backup Server 必须列在 master..sysservers 表中。安装 Adaptive Server 时 Backup Server 条目 SYB_BACKUP 在 sysservers 中创建。使用 sp_helpserver 可查看此信息。
- Backup Server 必须列在接口文件中。本地 Backup Server 的条目在安装 Adaptive Server 时创建。列在接口文件中的 Backup Server 名称必须与 master..sysservers 中 SYB_BACKUP 条目的列 srvnetname 名称一致。如果已将远程 Backup Server 安装在另一计算机上，则在这两台计算机共享的文件系统中创建 interfaces 文件；或者，将条目复制到本地 interfaces 文件。远程 Backup Server 的名称在这两个 interfaces 文件中必须相同。
- 启动 Backup Server 进程的用户对转储设备必须要有写权限。经常启动 Adaptive Server 和 Backup Server 的 “Sybase”用户可以对数据库设备进行读取和写入操作。
- Adaptive Server 必须被配置为用于远程访问。缺省情况下，安装完的 Adaptive Server 都允许远程访问。请参见第 320 页的 “配置服务器用于远程访问”。

装入新卷

在备份和恢复过程中，可能需要更改磁带卷。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。装入另一卷后，操作员通过在 Adaptive Server 上执行 sp_volchanged 来通知 Backup Server。

在 UNIX 系统中， Backup Server 在磁带容量满后请求更改卷。操作员装入另一磁带，然后执行 sp_volchanged （请参见表 12-2）。

表 12-2：在 UNIX 系统上更换磁带卷

顺序	使用 isql 的操作员	Adaptive Server	Backup Server
1	发出 dump database 命令		
2		向 Backup Server 发出转储请求	
3			接收到来自 Adaptive Server 的转储请求消息 将装入磁带的消息发送给操作员 等待操作员的答复
4	接收到来自 Backup Server 的卷更换请求 装入磁带 执行 sp_volchanged		
5			检查磁带 如果磁带没有问题，则开始转储 磁带已满后，向操作员发出卷更换请求
6	接收到来自 Backup Server 的卷更换请求 装入磁带 执行 sp_volchanged		
7			继续转储 转储完成后，将消息发送给操作员和 Adaptive Server
8	接收到转储已完成的消息 取出磁带并给磁带标号	接收到转储已完成的消息 释放锁 完成 dump database 命令	

启动和停止 Backup Server

大多数 UNIX 系统都使用 startserver 实用程序启动与 Adaptive Server 在同一计算机上的 Backup Server。在 Windows 平台上，从 Sybase Central 启动 Backup Server。有关启动 Backup Server 的信息，请参见所使用平台的配置文档。

使用 shutdown 关闭 Backup Server。请参见 《系统管理指南，卷 1》中的第 11 章 “诊断系统问题”和第 2 章 “系统和可选数据库”以及 《参考手册：命令》。

配置服务器用于远程访问

缺省情况下，当您安装 Adaptive Server 时， remote access 配置参数设置为 1。这将允许 Adaptive Server 执行对 Backup Server 的远程过程调用。

为保证安全，您最好除进行转储和装载操作之时外禁用远程访问。若要禁用远程访问，请使用：

```
sp_configure "allow remote access", 0
```

执行转储或装载之前，请重新启用远程访问：

```
sp_configure "allow remote access", 1
```

allow remote access 是动态的，不要求重新启动 Adaptive Server 使其生效。只有系统安全员才能设置 allow remote access。

选择备份介质

最好选择磁带作为转储设备，因为它们允许数据库和事务日志转储的库保持脱机状态。大的数据库可跨多个磁带卷。在 UNIX 系统上， Backup Server 要求所有转储和装载都使用非回绕磁带设备。

有关支持的转储设备列表，请参见所用平台的配置文档。

保护备份磁带以免被覆盖

tape retention in days 配置参数确定避免备份磁带被覆盖的天数。tape retention in days 的缺省值是 0，这意味着可以立即覆盖备份磁带。

使用 sp_configure 可更改 tape retention in days 的值。新值在下次重新启动 Adaptive Server 后生效：

```
sp_configure "tape retention in days", 14
```

dump database 和 dump transaction 命令都提供了 retaindays 选项，此选项会替换用于该转储的 tape retention in days 值。

转储到文件或磁盘

一般情况下， Sybase 建议您不要转储文件或磁盘。如果包含该文件的磁盘或计算机出现故障，则可能无法恢复数据。在 UNIX 和 PC 系统上，必须能在单个卷中装入整个 master 数据库转储。在这些系统中，如果 master 数据库太大而不能装入单个磁带卷中，则转储到文件或磁盘上是您的唯一选择，除非有另一个能发出 sp_volchanged 请求的 Adaptive Server。

可以将到文件或磁盘的转储复制到磁带以便脱机存储，但是在 Adaptive Server 可以读取这些磁带之前，必须将它们复制回联机文件。如果对磁盘文件所做的转储随后被复制到磁带，则 Backup Server 不能直接读取该转储。

创建本地转储设备的逻辑设备名

如果要转储到本地设备或从本地设备装载 （也就是说，不通过网络备份到远程 Backup Server），则可以通过提供其物理位置或通过指定其逻辑设备名来指定转储设备。在后一种情况下，可能需要在 master 数据库的 sysdevices 系统表中创建逻辑转储设备名称。

注释　如果要转储到远程 Backup Server 或从远程 Backup Server 装载，则必须指定转储设备的绝对路径名。不能使用逻辑设备名。

sysdevices 表存储有关每个数据库和备份设备的信息，包括其 *physical_name*（实际操作系统设备或文件名）和其 *device_name*（或逻辑名，仅在 Adaptive Server 中已知）。在大多数平台中，Adaptive Server 为安装在 sysdevices 中的磁带设备有一个或两个别名。这些设备的物理名是平台的公用磁盘驱动器名；逻辑名为 tapedump1 和 tapedump2。

创建备份脚本和阈值过程时，请使用逻辑名称，而不是物理设备名称，并且如果可能，每次更换备份设备时，都请修改引用实际设备名称的脚本和过程。如果使用逻辑设备名，则只需删除有故障的设备的 sysdevices 条目，并创建将逻辑名与不同物理设备关联的新条目。

注释 确保包括在转储命令中的设备驱动程序选项准确无误。Backup Server 在执行转储命令期间不对包括的任何设备驱动程序选项进行验证。例如，如果包括强制 Backup Server 在使用前回绕磁带的选项，则它始终将磁带回绕到开始处，而不是从转储点读取磁带。

列出当前设备名

若要列出系统的备份设备，请运行：

```
select * from master..sysdevices
  where status = 16 or status = 24
```

若要列出数据库和备份设备的物理名和逻辑名，请使用 sp_helpdevice：

```
sp_helpdevice tapedump1
device_name physical_name
 description
 status cntrltype vdevno vpn_low      vpn_high
 ------ --------- ------------ -------- -------

tapedump1 /dev/nrmt4
tape, 625 MB, dump device
     16         3              0        0   20000
```

添加备份设备

使用 sp_addumpdevice 可以添加备份设备。

若要将现有逻辑设备名用于另一物理设备，请使用 sp_dropdevice 删除该设备，再使用 sp_addumpdevice 增加该设备。例如：

```
sp_dropdevice tapedump2
sp_addumpdevice "tape", tapedump2, "/dev/nrmt8", 625
```

安排用户数据库的备份

开发备份计划中的主要任务是确定备份数据库的频率。备份频率决定在介质出现故障时丢失的工作量。本节提供有关何时转储用户数据库和事务日志的一些指导方针。

安排例行备份

创建每个用户数据库之后立即转储它以提供基点，并且以后按固定的时间表进行。推荐至少要每天备份事务日志，每周备份数据库。许多拥有大型、活动数据库的安装每天转储数据库，并且每半个小时或每小时进行一次事务日志转储。

在没有跨数据库数据修改活动期间，应同时备份互依数据库 （其中存在跨数据库事务、触发器或参照完整性的数据库）。如果其中一个数据库失败并且需要重新装载，则从所有这些同时转储中重新装载它们。

警告！ 在跨数据库约束执行添加、更改或删除操作之后或者在删除包含跨数据库约束的表之后，一定要立即转储这两个数据库。

在其它时间备份数据库

除了定期转储以外，每次升级用户数据库、创建新索引、执行未记录的操作或者运行 dump transaction with no_log 或 dump transaction with truncate_only 命令时，也都要转储数据库。

升级后转储用户数据库

将用户数据库升级到当前版本的 Adaptive Server 后，转储最近升级的数据库，以便创建与当前版本兼容的转储。 dump database 必须在允许执行 dump transaction 之前、在已升级的用户数据库上进行。

创建索引后转储数据库

向表中添加索引时，将在事务日志中记录 create index。而在向索引页填充信息时， Adaptive Server 却不记录这些更改。

如果在您创建完索引后数据库设备出现故障，则使用 load transaction 命令重建索引所用时间可能与使用 create index 命令建立索引所用时间一样多。为避免长时间的延迟，需在数据库的一个表上创建索引后立即转储每个数据库。

进行未记录的操作后转储数据库

Adaptive Server 不在事务日志中增加任何条目 （或者，在执行 bcp 时，增加最少的条目），而是将以下命令的数据直接写入磁盘：

- 未记录的 writetext
- 永久表上的 select into
- 快速批量复制 (bcp) 到一个没有触发器或索引的表中

在发出其中一条命令后，不能恢复对数据库所做的任何更改。为确保这些命令具有可恢复性，应在执行其中任何一条命令前发出 dump database 命令。

在日志被截断后转储数据库

dump transaction with truncate_only 和 dump transaction with no_log 将从日志中删除事务而不进行备份。为确保可恢复性，请在每次由于磁盘空间不足而运行任一命令时转储数据库。这样做之后，才能复制事务日志。请参见第 301 页的 “使用特殊 dump transaction 选项”。

如果 trunc log on chkpt 数据库选项设置为 true，并且事务日志包含 50 个或更多个行，则 Adaptive Server 会在发生自动检查点时截断日志。如果发生这种情况，请转储整个数据库 （而不是事务日志）以确保可恢复性。

安排 *master* 的备份

master 数据库备份用作恢复过程的一部分，以防出现影响 master 数据库的故障。如果没有 master 数据库的当前备份，则可能在需要用户数据库并再次运行它时不得不重建重要的系统表。

在每次更改后转储 *master* 数据库

虽然可以限制 master 中数据库对象的创建，但 create login、 drop login 和 alter login 等命令允许用户在数据库中修改系统表。经常备份 master 数据库以记录这些更改。

在执行影响磁盘、存储、数据库或段的每个命令后，都备份 master 数据库。始终在发出以下任何命令或系统过程后备份 master 数据库：

- disk init、 sp_addumpdevice 或 sp_dropdevice
- 磁盘镜像命令
- 段系统过程 sp_addsegment、 sp_dropsegment 或 sp_extendsegment
- create procedure 或 drop procedure
- sp_logdevice
- sp_configure
- create database 或 alter database

保存脚本和系统表

为进一步进行保护，保存包含所有 disk init、 create database 和 alter database 命令的脚本，并在每次发出这些命令之一后为 sysdatabases、 sysusages 和 sysdevices 表生成书面副本。

您无法使用 dataserver 命令自动恢复这些命令导致的更改。如果您保留脚本 （包含 Transact-SQL 语句的文件），则可以运行它们以重新创建这些更改。或者，您必须针对重新构建的 master 数据库重新发出每个命令。

保留 syslogins 的书面副本。从转储中恢复 master 时，将表的书面副本与当前版本进行比较，以确保用户保持相同的用户 ID。

有关要针对系统表运行的精确查询的信息，请参见 《系统管理指南，卷 1》中的第 2 章 “系统和可选数据库”。

截断 *master* 数据库事务日志

因为 master 数据库事务日志与数据存储在相同数据库设备上，所以不能单独备份其事务日志。不能移动 master 数据库的日志。必须经常使用 dump database 备份 master 数据库。定期使用具有 truncate_only 选项的 dump transaction （例如，每次数据库转储后）清除 master 数据库的事务日志。

避免卷更换和恢复

转储 master 数据库时，确保整个转储都装在单个卷中，除非有多个可以与 Backup Server 进行通信的 Adaptive Server。必须先在单用户模式下启动 Adaptive Server，然后再装载 master 数据库。这不允许进行独立的用户连接以在装载期间响应 Backup Server 的卷更改消息。因为 master 通常比较小，所以将它的备份放到单个磁带卷上通常不成问题。

安排 *model* 数据库的备份

保留 model 数据库的当前数据库转储。每次更改 model 数据库后，都生成一个新备份。如果 model 已损坏并且您没有备份，则必须重新输入所做的全部更改以恢复 model。

截断 *model* 数据库的事务日志

与 master 一样， model 将其事务日志存储在数据所在的同一个数据库设备中。在每次数据库转储后，应始终使用 dump database 备份 model 数据库，并且使用具有 truncate_only 选项的 dump transaction 清除事务日志。

安排 *sybsystemprocs* 数据库的备份

sybsystemprocs 数据库仅存储系统过程。除非对该数据库进行更改，否则要运行 installmaster 脚本来恢复此数据库。

如果更改对某些系统过程的权限，或在 sybsystemprocs 中创建自己的系统过程，则有两个可选的恢复过程：

- 运行 installmaster，然后通过重新创建过程或通过重新执行 grant 和 revoke 命令来重新输入全部更改。
- 在每次更改 sybsystemprocs 后对它进行备份。

有关这两个恢复选项的说明，请参见第 14 章 “恢复系统数据库”

与其它系统数据库一样， sybsystemprocs 将其事务日志存储在数据所在的那一设备中。必须经常使用 dump database 备份 sybsystemprocs 数据库。缺省情况下， trunc log on chkpt 选项在 sybsystemprocs 中被设置为 true (on)，因此，应该不需要截断事务日志。如果更改此数据库选项，请在转储数据库时截断日志。

如果正在 UNIX 系统或 PC 中运行，并且只有一个可以与 Backup Server 通信的 Adaptive Server，则需确保 sybsystemprocs 的整个转储都装在单个转储设备中。发送卷更改信号要求 sp_volchanged，并且当 sybsystemprocs 处于恢复过程中时不能在服务器上使用此过程。

配置 Adaptive Server 以用于同时装载

Adaptive Server 可以同时执行多个 load 和 dump 命令。装载数据库要求有一个 16K 缓冲区来用于每个活动数据库装载。缺省情况下，Adaptive Server 被配置为可同时进行六个装载。要同时执行多项装载，系统管理员可以增加大型 I/O 缓冲区的数量：

```
sp_configure "number of large i/o buffers", 12
```

此参数要求您重新启动 Adaptive Server。这些缓冲区不用于 dump 命令或 load transaction 命令。请参见《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

收集备份统计信息

使用 dump database 可对一个实际的用户数据库生成几个实际备份，使用 dump transaction 可备份事务日志。使用 load database 可恢复数据库，使用 load transaction 可应用连续的事务日志转储。

保留有关每个转储和装载操作执行的时间长度以及所要求的空间量的统计信息。越接近实际的备份情况，所做的预测越有意义。

在制订和测试完备份过程后，将它们写到纸上。确定一个合理的备份时间表并按此时间表进行操作。如果提前制订、记录和测试备份过程，则在发生灾难性事故时您可以更加从容地应对，使数据库恢复联机状态。

第 13 章

备份和恢复用户数据库

经常定期备份是防止由于数据库设备出现故障而损坏数据库的唯一方法。

本章介绍了如何使用 Backup Server 备份和恢复用户数据库。

如果您的节点支持 Tivoli Storage Manager (TSM)，另请参见《将 Backup Server 与 IBM Tivoli Storage Manager 配合使用》。但是，本章中的大多数语法和用法信息都与支持 TSM 的节点相关。

dump database、dump transaction、load database 和 load transaction 命令具有相似的语法。例行转储和装载要求数据库名和至少一个转储设备。这些命令还可包括下列选项：

- compression=，用于将转储文件压缩为本地文件
- at *server_name*，用于指定远程 Backup Server
- density、blocksize 和 capacity，用于指定磁带存储特性
- dumpvolume，用于指定 ANSI 磁带标签的卷名
- file = *file_name*，用于指定要转储到的或要从其装载的文件的名称
- stripe on *stripe_device*，用于指定其它转储设备
- dismount、unload、init 和 retaindays，用于指定磁带的处理操作
- notify，用于指定是将 Backup Server 消息发送到启动转储或装载的 client，还是发送到 operator_console

注释 转储用户数据库时，其数据库选项不会转储，这是因为它们存储在 master 数据库的 sysdatabases 表中。如果将以前转储的数据库装载到其自身之中，则不会出现此问题，原因是 sysdatabases 中描述此数据库的行仍位于 master 中。但是，如果在执行 load database 之前删除数据库，或者在新服务器上装载数据库转储，则无法恢复这些数据库选项。若要恢复用户数据库的映像，您还必须重新创建这些数据库选项。

如果设备上的可用空间不足，无法成功发出 dump transaction 或 dump transaction with truncate_only 命令，请使用 dump transaction with no_log。

请参见《参考手册：命令》。

转储和装载完成百分比

dump 和 load database 在运行时显示完成百分比。dump database 显示所转储的数据库的完成百分比，而 load database 显示为目标数据库装载的百分比。

注释 dump 和 load transaction 命令不显示完成百分比。

例如，如果将 sybsystemprocs 数据库转储到名为 *pubs2.dump* 的文件中，则 Adaptive Server 会显示：

```
dump database sybsystemprocs to "pubs2.dump"
Backup Server session id is:13.  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
/linuxkernel_eng3/Pubs/REL1502/ASE-15_0/bin/pubs2.dump.
Backup Server:6.28.1.1: Dumpfile name 'pubs20805209785  ' section number 1
mounted on disk file '/linuxkernel_eng3/Pubs/REL1502/ASE-
15_0/bin/pubs2.dump'
Backup Server:4.188.1.1: Database pubs2:876 kilobytes (46%) DUMPED.
Backup Server:4.188.1.1: Database pubs2:1122 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 pubs2:1130 kilobytes (100%) DUMPED.
Backup Server:3.42.1.1: DUMP is complete (database pubs2).
```

将 *pubs2.dump* 装载到数据库中时，Adaptive Server 会报告：

```
load database pubs2 from “pubs2.dump”
Backup Server session id is:17.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server:6.28.1.1: Dumpfile name 'pubs20805209785  ' section number 1
mounted on disk file '/linuxkernel_eng3/Pubs/REL1502/ASE-
15_0/bin/pubs2.dump'
Backup Server:4.188.1.1: Database pubs2:1880 kilobytes (45%) LOADED.
Backup Server:4.188.1.1: Database pubs2:4102 kilobytes (100%) LOADED.
Backup Server:4.188.1.1: Database pubs2:4110 kilobytes (100%) LOADED.
Backup Server:3.42.1.1: LOAD is complete (database pubs2).
Started estimating recovery log boundaries for database 'pubs2'.
Database 'pubs2', checkpoint=(1503, 22), first=(1503, 22), last=(1503, 22).
Completed estimating recovery log boundaries for database 'pubs2'.
Started ANALYSIS pass for database 'pubs2'.
Completed ANALYSIS pass for database 'pubs2'.
Started REDO pass for database 'pubs2'.The total number of log records to
process is 1.
Completed REDO pass for database 'pubs2'.
Use the ONLINE DATABASE command to bring this database online; ASE will not
bring it online automatically.
```

对于 dump database，所显示的百分比是估计值，取决于所转储的数据库的总大小。但是，对于 load database，所显示的百分比是根据接收数据库的总大小估计得出的值。例如，如果将 500MB 的数据库转储装载到 100 MB 的数据库中，则完成百分比值是根据 100MB 数据库（而不是 50MB 转储）估计得出的值。

本章剩余部分将提供有关转储和装载命令及卷更改消息中所指定信息的更详细说明。

有关执行转储和装载命令所要求权限的信息，请参见第 277 页的“指定备份的职责”。

指定数据库和转储设备

所有 dump 和 load 命令必须至少包括要转储的或装载的数据库的名称。转储或装载数据（不只是截断事务日志）的命令还必须包括一个转储设备名。

请参见《参考手册：命令》。

指定数据库名的规则

可以将数据库名以文字、局部变量或参数的形式指定给某一存储过程。

如果从转储中装载数据库：

- 此数据库必须存在。可以使用 create database 的 for load 选项创建一个数据库，或通过装载覆盖一个现有数据库。装载数据库始终会覆盖现有数据库中的所有信息。
- 使用的数据库名不必与所转储的数据库的名称相同。例如，您可以转储 pubs2 数据库，创建另一个名为 pubs2_archive 的数据库，然后将转储装载到新数据库中。

警告！ 对于含有其它数据库所引用的主键的数据库，不要更改其名称。如果必须装载来自此类数据库的转储并提供其它名称，应首先从其它数据库删除对该据库的引用。

指定转储设备的规则

指定转储设备时：

- 您可以将转储设备指定为文字、局部变量或存储过程的参数。
- 您无法转储到“空设备”或从“空设备”装载（在 UNIX 平台上为 */dev/null*；不适用于 PC 平台）。
- 当转储到本地设备或从本地设备装载时，可以使用以下任意形式指定转储设备：
  - 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 的当前工作目录解析相对路径名。
- 通过网络转储或装载时：
  - 必须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。
- 如果使用 with standby_access 来转储事务日志，则必须使用 with standby_access 来装载该转储。

示例

下面的示例使用单一磁带设备进行转储和装载。转储和装载不必使用同一设备。

- 在 UNIX 平台上：

```
dump database pubs2 to "/dev/nrmt4"
load database pubs2 from "/dev/nrmt4"
```

- 在 Windows 平台上：

```
dump database pubs2 to "\\.\tape0"
load database pubs2 from "\\.\tape0"
```

也可以转储到一个操作系统文件中。以下示例适用于 Windows：

```
dump database pubs2 to "d:\backups\backup1.dat"
load database pubs2 from "d:\backupbackup1.dat"
```

Backup Server 确定磁带设备

当您发出 dump database 或 dump transaction 命令时，Backup Server 会检查 Adaptive Server 是否知道（在内部提供并支持）指定转储设备的设备类型。如果设备不属于已知类型，则 Backup Server 会检查带配置文件（缺省位置为 *$SYBASE/backup_tape.cfg*）以获取设备配置。

如果发现其配置，则 dump 命令继续执行。

如果在磁带设备配置文件中找不到该配置，则 dump 命令会失败，并发出以下错误消息：

```
Device not found in configuration file.INIT needs to
be specified to configure the device.
```

若要配置设备，请发出带有 init 参数的 dump database 或 dump transaction。通过使用操作系统调用，Backup Server 尝试确定该设备的特性；如果成功，它将设备特性存储在磁带配置文件中。

如果 Backup Server 不能确定转储设备的特性，就将缺省设置为每个磁带存放一个转储。如果这种配置连一个转储文件都无法写入，则该设备不能使用。

Backup Server 的磁带配置仅适用于 UNIX 平台。

磁带设备配置文件

磁带设备配置文件包含仅由 dump 命令使用的磁带设备信息。该文件的格式是每个磁带设备条目占用一行。字段之间用空白或制表符分隔。

仅当 Backup Server 准备向磁带设备配置文件中写入信息时，才会创建该文件。当 Backup Server 首次尝试向此文件中写入信息时，您会看到：

```
Warning, unable to open device configuration file
for reading.Operating system error.No such file or
directory.
```

请忽略此消息。Backup Server 给出此警告，然后创建该文件并向文件中写入配置信息。

如果当用户收到以下错误消息时出现该文件，将需要用户交互，这是所需的唯一用户交互：

```
Device does not match the current
configuration.Please reconfigure this tape device by
removing the configuration file entry and issuing a
dump with the INIT qualifier.
```

这意味着某个设备名的磁带硬件配置已经更改。删除该设备名的条目行，然后按指示发出 dump 命令。

配置文件的缺省路径名为 *$SYBASE/backup_tape.cfg*，您可以通过 Sybase 安装实用程序使用缺省位置来更改该路径。

压缩转储

dump 命令包括两个选项，利用这两个选项，您可以使用 Backup Server 压缩数据库和事务日志，从而减少已存档数据库的空间要求。参数为：

- compression = *compression_level* — 压缩至远程服务器。导致 Backup Server 使用其自己的本机压缩方法。Sybase 建议使用此压缩选项。
- compress::[*compression_level*::]— 压缩至本地文件。导致 Backup Server 调用外部过滤器，支持此选项是为了向后兼容。

compression_level 可以是 0 到 9 之间的某个数字，也可以是 100 或 101。对于一位数的压缩级别，0 表示不压缩，9 表示压缩级别最高。压缩级别 100 和 101 表示压缩比较速、高效，其中压缩级别 100 表示压缩速度较快，101 表示压缩性能较好。

注释　compress::参数不支持压缩级别 100 和 101。

请参见《参考手册：命令》。

利用 dump 命令的 compression= 参数，可以减少已存档数据库的空间要求。使用 Adaptive Server 12.5.2 及更高版本，可以通过 compression= 参数将转储压缩到远程计算机。

如果使用旧的 compress::选项，装载数据库转储时不需要包括压缩级别。但是，可以发出 load with listonly=full 命令以确定进行转储的压缩级别。

如果您使用本机 compression= 选项，则当装载数据库转储时，不需要包括 compression= 选项。

例如，若要将 pubs2 数据库转储到文件“*compress_file*”中，请输入：

```
dump database pubs2 to compress_file...compression=100
```

表 13-1 显示了 pubs2 数据库的压缩级别。这些压缩级别数字仅供参考；根据操作系统级别和配置情况，您的节点的数字可能不同。

表 13-1：pub2 的压缩级别和压缩文件大小

压缩级别	压缩文件大小
级别 1	254K
级别 9	222K
级别 100	324K
级别 101	314K

压缩级别 100 和 101 所占用的 CPU 时间比级别 0 - 9 要少，而且压缩速度更快。然而，使用级别 100 和 101 可能会导致转储文件较大。级别 100 可提供较速的压缩，而级别 101 可提供比较完全的压缩。实际压缩结果取决于文件的内容。

Sybase 建议您根据性能要求选择一组压缩级别。对于占用 CPU 时间不太多的压缩，请使用压缩级别 100 并根据存档空间要求切换至级别 101。对于常规压，请使用压缩级别 6，然后根据性能要求增高或降低级别。

示例

示例 1 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 2 将 pubs2 数据库转储到名为“remotemachine”的远程计算机，并使用压缩级别 100：

```
dump database pubs2 to "/Syb_backup/mydb.db" at remotemachine
with compression ="100"
```

有关 dump database 和 dump transaction 的完整语法信息，请参见《参考手册：命令》。

Backup Server 转储文件和压缩转储

本节介绍与 compress:: 选项相关的转储。它们不适用于首选的 compression= 选项。

使用已存在的存档文件对磁带设备执行 dump database 或 dump transaction 时，Backup Server 将自动检查现有转储存档的标头。如果标头不可读，Backup Server 将假定此文件为有效的非存档文件，并提示您更改该转储存档：

```
Backup Server: 6.52.1.1: OPERATOR:Volume to be overwritten on
'/opt/SYBASE/DUMPS/model.dmp' has unrecognized label data.
Backup Server:6.78.1.1: EXECUTE sp_volchanged
    @session_id = 5,
    @devname = '/opt/SYBASE/DUMPS/model.dmp',
    @action = { 'PROCEED' | 'RETRY' |
'ABORT' },
    @vname = <new_volume_name>
```

因此，如果在不带有 compress:: 选项的情况下对文件执行 dump database 或 dump transaction，使其转储到现有压缩转储档案中，则 Backup Server 将不识别档案的标头信息，因为它是压缩的。

示例

第二个 dump database 报告错误并使用 sp_volchanged 提示您：

```
dump database model to 'compress::model.cmp'
go
dump database model to 'model.cmp'
go
```

为防止出现此错误，请将 with init 选项包含在后续的 dump database 和 dump transaction 命令中：

```
dump database model to 'compress::model.cmp'
go
dump database model to 'model.cmp'
   with init
go
```

装载压缩转储

本节介绍与 compress:: 选项相关的转储。如果使用本机 compression= 选项创建转储，则不需要任何特定的语法即可装载。

如果使用 dump ... compress::来转储数据库或事务日志，则必须使用 load ... compress:: 选项来装载该转储。

load database ... compress::和 load transaction ... compress::的部分语法为：

```
load database database_name
 from compress::stripe_device
…[stripe on compress::stripe_device]…

load transaction database_name
 from compress::stripe_device
…[stripe on compress::stripe_device]…
```

语法中的 *database_name* 表示您存档的数据库，compress::调用已存档数据库或事务日志的解压缩。*archive_name* 是您要装载的已存档数据库或事务日志的完整路径。如果创建转储文件时未包括完整路径，则 Adaptive Server 将在启动 Adaptive Server 的目录中创建转储件。

如果使用 compress::选项，则对于每个转储设备，它必须是 stripe on 子句的一部分。如果您使用 compression= 选项，则在设备列表之后使用它一次。请参见第 348 页的“指定其它转储设备：stripe on 子句”。

注释 不要对 *load* 命令使用 compression_level 变量。

有关 load database 和 load transaction 的完整语法信息，请参见《参考手册：命令》。

指定远程 Backup Server

使用 at *backup_server_name* 子句通过网络将转储和装载请求发送到另一台计算机上运行的 Backup Server。请参见《参考手册：命令》

注释　compress::选项仅适用于本地存档；您不能使用 *backup_server_name* 选项。

对于使用多个磁带设备进行所有备份和装载的单个计算机，通过网络发送转储和装载请求是比较理想的。操作员可以在这些计算机旁准备更换磁带。

下列示例转储到名为 REMOTE_BKP_SERVER 的远程 Backup Server 或从其进行装载：

```
dump database pubs2 to "/dev/nrmt0" at REMOTE_BKP_SERVER
load database pubs2 from "/dev/nrmt0" at REMOTE_BKP_SERVER
```

backup_server_name 必须显示在运行 Adaptive Server 的计算机上的接口文件中，但不需要显示在 sysservers 表中。本地和远程接口文件中的 *backup_server_name* 必须相同。

指定磁带密度、块大小和容量

大多数情况下，Backup Server 使用最适合于您的操作系统的缺省磁带密度和块大小，Sybase 建议您使用这些缺省值。

您可以指定每个转储设备的密度、块大小和容量。也可以在 with 子句中为所有转储设备指定 density、blocksize 和 capacity 选项。为单个磁带设备指定的特性优先于使用 with 子句指定的特性。

请参见《参考手册：命令》

替换缺省密度

dump 和 load 命令将缺省磁带密度用于您的操作系统。多数情况下，这是用于磁带转储的最佳密度。

density 参数对 UNIX 和 PC 平台转储或装载没有影响。

注释 仅在使用 init 磁带处理选项时才指定磁带密度。请参见第 351 页的“转储前重新初始化卷”。

替换缺省块大小

blocksize 参数指定转储设备每次执行 I/O 操作的字节数。缺省情况下，转储和装载命令会为您的操作系统选择最佳块大小。如果可能，应尽量使用这些缺省值。

可使用 blocksize = *number_bytes* 选项替换特定转储设备的缺省块大小。块大小必须至少为一个数据库页（2048 字节），且必须为数据库页大小的整数倍。

对于 UNIX 系统，在装载命令中指定的块大小将被忽略。Backup Server 将使用用来进行转储的块大小。

指定较大的块大小值

如果使用 dump database 或 dump transaction 命令转储到磁带，并且指定的块大小值大于 Backup Server 所确定的设备最大块大小，则转储或装载可能会在某些磁带驱动器上失败。此时会显示一条操作系统错误消息；例如，在 HP 上的 8mm 磁带驱动器上，该错误消息为：

```
Backup Server: 4.141.2.22: [2] The 'write' call
failed for device 'xxx' with error number 22 (Invalid
argument).Refer to your operating system
documentation for further details.
```

指定的块大小不要大于在 *$SYBASE/backup_tape.cfg* 中的磁带设备配置文件中指定的块大小。在磁带设备配置文件中，设备的块大小为所在行的第五个字段。

此错误仅发生在运行磁带自动配置的磁带驱动器上；即这些设备型号在 Backup Server 代码中没有被硬编码。

指定转储命令的磁带容量

对于不能可靠检测到磁带结束标志的 UNIX 平台，必须指明可转储到磁带的千字节数。

如果指定转储设备的物理路径名称，请在转储命令中包括 capacity = *number_kilobytes* 参数。如果指定逻辑转储设备名，Backup Server 会使用 *sysdevices* 表中存储的 size 参数，直到用 capacity = *number_kilobytes* 参数把它替换。

指定的容量必须至少为 5 个数据库页（每页需要 2048 个字节）。Sybase 建议所指定的容量应略低于您设备的额定容量。

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% 的容量用于开销（记录间隙、磁带标志等）。此规则在多数情况下适用，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。

Backup Server 的非回绕磁带功能

非回绕磁带功能自动将磁带定位在有效转储数据的末尾，这使您可以在执行多个转储操作时节省时间。

Backup Server 在每个转储操作结束时编写文件结束标签 EOF3。

磁带操作

执行新的转储时，Backup Server 会进行扫描以获取上次的 EOF3 标签。将保存相关信息并将磁带向前定位到磁带上下一个文件的开头。此位置即为新的附点。

如果找不到 EOF3 标签或者发生任何其它问题，则 Backup Server 会回绕磁带并向前扫描。在这些步骤期间发生的任何错误都不会使转储操作中止，但可导 Backup Server 进行缺省的回绕和扫描行为。如果在回绕和扫描期间错误仍然存在，dump 命令会中止。

转储的版本兼容性

仅当磁带上的标签版本大于或等于 5 时，Backup Server 才激活非回绕逻辑。因此，要激活此逻辑，必须执行含 with init 子句的 dump 命令。如果在标号版本小于 5 的卷上启动 dump without init，系统将提示您更改该卷，转储将在下一卷上开始。多卷转储的标签版本在卷组中间不会更改。

表 13-2 定义为其启用此行为的标签版本。

表 13-2：标签版本兼容性

标签版本	已启用
‘3’	否
‘4’	否
‘5’	是
‘6’	是

指定卷名

使用 with dumpvolume = *volume_name* 选项指定卷名。dump database 和 dump transaction 将该卷名写入 SQL 磁带标签。load database 和 load transaction 检查该标签。如果装载了错误的卷，Backup Server 将生成错误消息。

您可为每个转储设备指定卷名。您还可以在 with 子句中为所有设备指定一个卷名。为各个设备指定的卷名优先于在 with 子句中指定的卷名。

请参见《参考手册：命令》。

从多个卷装载

在您从包含多个转储文件的卷装载数据库转储时，指定转储文件名。如果仅指定数据库名称，则 Backup Server 会将第一个转储文件装载到指定的数据库。例如，此命令将磁带中的第一个转储文件装载到 pubs2 中，无论该转储文件是否包含 pubs2 中的数据均是如此：

```
load database pubs2 from "/dev/rdsk/clt3d0s6"
```

为避免这一问题，应在每次转储或装载数据时指定唯一的转储文件名。若要获取有关给定磁带上的转储文件的信息，请使用 load database 的 listonly = full 选项。

标识转储

在您转储某一数据库或事务日志时，Backup Server 通过并置以下项创建该转储的缺省文件名：

- 数据库名称的后 7 个字符
- 2 位的年份数字
- 一年中的 3 位数日期 (1 - 366)
- 自午夜以来的秒数（采用十六进制）

您可以使用 file = *file_name* 选项替换此缺省值。文件名不得超过 17 个字符，并且必须与所使用操作系统的文件命名约定相符。

您可为每个转储设备指定文件名。您还可以在 with 子句为所有设备指定一个文件名。为各个设备指定的文件名优先于在 with 子句中指定的文件名。

在装载数据库或事务日志时，可以使用 file = *file_name* 子句指定要从包含多个转储的卷中装载哪个转储。

在从多个卷装载转储时，必须指定正确的文件名。

```
dump tran publications
    to "/dev/nrmt3"
load tran publications
    from "/dev/nrmt4"
    with file = "cations930590E100"
```

下面的示例使用用户定义的文件命名约定。15 个字符的文件名 *mydb97jul141800* 标识进行转储的数据库 (mydb)、转储日期（1997 年 7 月 14 日）和转储时间（18:00，即下午 6:00）。在装载前使用 load 命令将磁带前进到 *mydb97jul141800*：

```
dump database mydb
    to "/dev/nrmt3"
    with file = "mydb97jul141800"
load database mydb
    from "/dev/nrmt4"
    with file = "mydb97jul141800"
```

提高转储或装载性能

启动 Backup Server 时，通过为 Backup Server 配置更多共享内存来使用 -m 参数提高 dump 和 load 命令的性能。-m 参数指定了 Backup Server 所用的最大共享内存量。同时还必须配置操作系统以确保 Backup Server 可以使用这么多共享内存。当转储或装载操作完成时，将释放其共享内存段。

注释 仅当尚未达到硬件设置的性能限制时，配置额外的共享内存才可提高转储和装载性能。如果转储到 QIC 之类的慢速磁带设备上，增加 -m 的值可能不会提高性能，但如果转储到 DLT 之类的快速设备上，就可以大大提高性能。

与之前版本的兼容性

转储文件和 Backup Server 之间存在一些兼容性问题。表 13-3 指出当前运行的版本或以前版本的本地 Backup Server 可以装载转储文件格式。

表 13-3：用于本地操作的服务器

	新的转储文件格式	旧的转储文件格式
服务器的当前版本	是	是
服务器的以前的版本	否	是

表 13-4 和表 13-5 指出可由远程 Backup Server 的当前和以前版本装载的转储文件格式。在远程 Backup Server 的情况下，主服务器是与数据库和 Adaptive Server 位于同一计算机的 Backup Server，而辅助服务器是与存档设备位于同一远程计算机上的 Backup Server。

表 13-4 指出在主服务器是 Backup Server 的当前版本时起作用的 load 操作。

表 13-4：新的主服务器版本

	新的转储文件格式	旧的转储文件格式
新的服务器辅助版本	是	是
以前的服务器辅助版本	否	是

表 13-5 指出在主服务器是以前版本时起作用的 load 操作。

表 13-5：以前的主服务器版本

	新的转储文件格式	旧的转储文件格式
新的服务器辅助版本	否	是

	新的转储文件格式	旧的转储文件格式
以前的服务器辅助版本	否	是

以整数格式存储的标签

Backup Server 12.0 和更高版本以整数格式存储分条编号。Backup Server 的早期版本以 ASCII 格式在 HDR1 标签中存储 4 字节分条编号。这些较早版本的 Backup Server 不能装载使用较新转储格式的转储文件。而 Backup Server 12.0 版和更高版本可以读取和写入转储格式的较早版本。

在执行涉及一个或多个远程服务器的转储或装载操作时，如果出现以下情况，操作将中止并显示错误消息：

- 一个或多个远程 Backup Server 的版本早于 12.0，并且数据库被转储到超过 32 个分条中或从超过 32 个分条中装载，或者：
- 装载期间一个或多个 Backup Server 正从中读取的转储文件来自以前版本的格式，并且数据库正从其装载的分条数目大于 32。

配置系统资源

在执行转储和装载前，必须在启动时通过为命令行选项控制的系统资源提供适当的值，配置本地和远程 Backup Server。有关命令行选项的完整列表，请见《实用程序指南》。

如果您的系统资源未被正确配置，转储或装载可能会失败。例如，在使用缺省配置启动的本地和远程 Backup Server 中远程转储到 25 个以上的分条将会失败，这是因为 Backup Server 可以启动的最大网络连接数量（由 -N 选项指定）为 25；但是，缺省情况下，到远程 Backup Server（由 -C 选项指定）的最大服务器连接数量为 30。

若要配置系统以使用更高的分条限制，请设置以下操作系统参数：

- 进程可以附加到的共享内存段的数目
- 共享内存标识符的数目
- 交换空间

如果未正确配置这些参数，在开始向大量分条进行转储时（或者开始从大量分条装载时），操作可能由于系统资源不足而中止。您会收到一条消息，出 Backup Server 无法创建或附加到共享内存段，因此 SYBMULTBUF 进程终止。

设置共享内存使用情况

使用 -m 参数启动 Backup Server 的语法如下：

```
backupserver [-m nnn]
```

其中 *nnn* 是 Backup Server 用于其所有转储或装载会话的最大共享内存数量（以 MB 为单位）。

-m 参数设置共享内存使用率的上限。不过，如果 Backup Server 检测到添加更多的内存并不会提高性能，它可能会使用少于指定值的内存。

Backup Server 通过用 -m 值除以所配置的服务线程数量（-P 参数）来确定可用于每个分条的共享内存数量。

-m 的缺省值是服务线程的数目乘以 1MB。-P 的缺省值是 48，因此缺省最大共享内存使用率是 48MB。但是，只有在所有 48 个服务线程同时处于活动状态的情况下，Backup Server 才会到达此使用率。-P 的最大值是最大服务线程数量，即 12,288。请参见《实用程序指南》。

可用于 Backup Server 的每个分条的共享内存数量与您分配的服务线程数量成比例。如果增加最大服务线程的数量，也必须增加 -m 值以便为每个分条维护相同数量的共享内存。如果您增加了 -P 值，但没有增加 -m 值，则每个分条分配的共享内存量可能会减少到无法处理转储或装载操作的程度。

若要确定将 -m 值增加到多大，请使用以下公式：

（以 MB 计的 -m 值）* 1024/（-P 值）

如果由此公式得出的值小于 128KB，则 Backup Server 将不能启动。

-m 的最小值是 6MB。-m 的最大值取决于对共享内存的操作系统限制。

如果使用具有较多共享内存的 Backup Server 创建转储，并尝试使用具有较少共享内存的 Backup Server 装载转储，则 Backup Server 仅使用可用内存，从而导致载性能降低。

如果装载时每个分条的可用共享内存量小于转储时使用的块大小的两倍，则 Backup Server 将中止该装载，并且显示错误消息。

设置分条的最大数目

Backup Server 可以使用的分条的最大数目受到它可以创建的 Open Server 线程的最大数目的限制。Open Server 对一个应用程序可以创建的线程的数目强制实行大为 12K 的限制。

Backup Server 为每一分条创建一个服务线程。因此，Backup Server 可以转储到或从其装载的本地分条的最大数目是 12,286。

作为一条附加限制，Backup Server 除了使用与错误日志文件、interfaces 文件和其它系统文件关联的文件描述符之外，还为每一分条使用两个文件描述符。但是，操作系统对每个线程可用的文件描述符数量有限制。Open Server 对应用程序可以跟踪的文件描述符数量的限制为 1280。

确定 Backup Server 可以转储到的本地分条的最大近似数目的公式是：

$$\frac{\text{（OS 限制或 Open Server 限制中的较小者）} - 2}{2}$$

用于确定 Backup Server 可以转储到的远程分条的最大适当数量的公式如下：

$$\frac{\text{（OS 限制或 Open Server 限制中的较小者）} - 2}{3}$$

有关文件描述符限制的缺省值和最大值的详细信息，请参见您的操作系统文档。

设置最大网络连接数

Open Server 将可以源自本地 Backup Server 的网络连接的最大数目限制为 9118。因此，Backup Server 在单次转储或装载操作中可以使用的远程分条的最大数目是 9118。

远程 Backup Server 随时可以接受最多 4096 个服务器连接。因此，远程分条到单个远程 Backup Server 的最大数目是 4096。

设置最大服务线程数

Backup Server 的 -P 参数配置 Open Server 创建的服务线程的数目。最大服务线程数量为 12,228。最小值为 6。最大线程数等于可用的最大分条数。如果您在未设置足够高的 -P 值的情况下启动 Backup Server，并且尝试将数据库转储或装载到的分条数量超过了线程数量，则转储或装载操作将会失败。

指定其它转储设备：*stripe on* 子句

分条允许您将多个转储设备用于单个转储或装载命令。使用单独的 stripe on 子句可以指定每个设备的名称（如果需要，还可以指定特性）。

每个转储或装载命令都可以有多个 stripe on 子句。

请参见《参考手册：命令》。

转储到多个设备

Backup Server 将数据库拆分为几个几乎相等的部分，并将每个部分发送到不同的设备。转储在所有设备上同时进行，从而缩短转储单独数据库或事务日所需的时间。因为每个磁带都只存储数据库的一部分，所以不见得必须将新磁带装入特定设备上。

警告！不要将 master 数据库转储到多个磁带设备。从磁带或其它可移动介质装载 master 数据库时，除非拥有另一个可以响应卷更改消息的 Adaptive Server，否则无法更改卷。

从多个设备装载

您可以使用多个设备装载数据库或事务日志。使用多个设备可以减少装载所需时间以及不得不在特定设备上装入多个磁带的可能性。

与转储相比装载所使用的设备较少

即使在转储和装载之间转储设备之一变得不可用，您也可以装载数据库或日志。与转储命令中相比，在装载命令中指定的分条子句较少。

注释 在您通过网络转储和装载时，对于这两个操作必须使用相同数目的驱动器。

以下示例使用三个设备转储一个数据库，但只使用两个设备装载它：

```
dump database pubs2 to "/dev/nrmt0"
    stripe on "/dev/nrmt1"
```

```
    stripe on "/dev/nrmt2"
load database pubs2 from "/dev/nrmt0"
    stripe on "/dev/nrmt1"
```

在装载前两个磁带后，将显示一条消息，通知操作员装载第三个磁带。

也可以将一个数据库转储到多个操作系统文件中。此示例适用于 Windows：

```
dump database pubs2 to "d:\backups\backup1.dat"
    stripe on "d:\backups\backup2.dat"
    stripe on "d:\backups\backup3.dat"
load database pubs2 from "/dev/nrmt0"
stripe on "d:\backups\backup2.dat"
    stripe on "d:\backups\backup3.dat"
```

指定单个设备的特性

为附加到远程 Backup Server 的每个分条设备使用单独的 at *server_name* 子句。如果您没有指定远程 Backup Server 名称，则本地 Backup Server 将搜索本地计算机上的转储设备。如有必要，也可以为单独的分条设备指定单独的磁带设备特性（density、blocksize、capacity、dumpvolume 和 file）。

此示例（在 UNIX 平台上）使用三个转储设备，每个设备都连接到远程 Backup Server REMOTE_BKP_SERVER。

```
dump database pubs2
    to "/dev/nrmt0" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt1" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt2" at REMOTE_BKP_SERVER
```

磁带处理选项

显示在 with 子句中的磁带处理选项应用到所有用于转储或装载的设备，包括：

- nodismount，使磁带可用于其它转储或装载
- unload，用于在转储或装载后倒带和卸载磁带
- retaindays，用于防止文件被覆盖

- init，用于重新初始化磁带，而不是在最后的磁带结束标志后附加转储文件

请参见《参考手册：命令》。

指定是否卸下磁带

在支持逻辑卸下的平台上，在转储或装载操作完成后磁带被卸下。使用 nodismount 选项可保持磁带不被卸下并可用于其它转储或装载。此命令对 UNIX 或 PC 系统不起作用。

回绕磁带

缺省情况下，转储和装载命令都使用 nounload 磁带处理选项。

在 UNIX 系统上，这样做可以防止在转储或装载操作完成后磁带被回绕。这使您可以将其它数据库或日志转储到同一卷，或从该卷装载其它数据库或日。将 unload 选项用于磁带上的最后一个转储以在命令结束时回绕和卸载磁带。

防止转储文件被覆盖

tape retention in days 指定在磁带文件创建后必须经过多少天您才能用其它转储覆盖它。此服务器范围的选项（可以使用 sp_configure 对其进行设置）应用于从一个 Adaptive Server 请求的所有转储。

对于单个数据库或事务日志转储，使用 retaindays = *number_days* 选项替换该 tape retention in days 参数。这个天数必须是正整数或零（如果磁带可以被立即覆盖）。

注释 tape retention in days 和 retaindays 只对磁盘、1/4 英寸盒式磁带和单文件介质有意义。在多文件介质上，Backup Server 只检查第一个文件的有效期。

转储前重新初始化卷

缺省情况下，每个转储都被附加到磁带的最后磁带结束标志之后。磁带卷不被重新初始化。因此，您可以将多个数据库转储到单个卷。新的转储只能加到多卷转储的最后一个卷上。

使用 init 选项可以覆盖任何现有磁带内容。如果您指定 init，则 Backup Server 会重新初始化磁带，但不会检查以下各项：

- ANSI 访问限制
- 尚未到期的文件
- 非 Sybase 数据

缺省值 noinit 检查所有三个条件并发送卷更改提示（如果任何条件成立）。

以下示例初始化两个设备并使用新的事务日志转储覆盖现有内容：

```
dump transaction pubs2
    to "/dev/nrmt0"
    stripe on "/dev/nrmt1"
    with init
```

如果您要将数据库转储到某一操作系统文件中，也可以使用 init 选项覆盖现有文件。下面是一个 Windows 示例：

```
dump transaction pubs2
    to "d:\backups\backup1.dat"
    stripe on "d:\backups\backup2.dat"
    with init
```

请参见《参考手册：命令》，查看有关将多个数据库转储到一个卷的说明。

转储和装载数据库时使用口令保护

使用 dump database 命令的 password 参数防止从未经授权的装载转储数据库。然后，还必须在装载数据库时包括此口令。

请参见《参考手册：命令》。

口令的长度必须为 6 - 30 个字符。

此示例使用口令“bluesky”保护 pubs2 数据库的数据库转储：

```
dump database pubs2 to "/Syb_backup/mydb.db" with passwd = "bluesky"
```

此示例使用同一口令装载数据库转储：

```
load database pubs2 from "/Syb_backup/mydb.db" with passwd = "bluesky"
```

只能在 Adaptive Server 12.5.2 版和更高版本中使用带口令保护的 dump 和 load 命令。如果对 Adaptive Server 12.5.2 版的转储使用口令参数，则在尝试将此转储装载到 Adaptive Server 的早期版本上时，装载会失败。

只能将转储装载到另一台使用相同字符集的服务器中。例如，如果试图将使用 ASCII 字符集的服务器中的转储装载到一台使用非 ASCII 字符集的服务器，由于 ASCII 口令的值不同于非 ASCII 口令的值，因此装载会失败。

用户输入的口令将转换为 Adaptive Server 本地字符集。由于 ASCII 字符在字符集间通常表示相同的值，因此，如果某个用户的口令使用的是 ASCII 字符集，则 dump 和 load 的口令在所有字符集中都可以被识别。

替换缺省的消息显示目标

Backup Server 消息通知操作员何时更改磁带卷以及转储或装载的进度。这些消息的缺省显示目标取决于操作系统是否提供操作员终端功能。

notify 选项（出现在 with 子句中）允许您替换用于转储或装载的缺省消息的显示目标。为使该选项生效，只要 Backup Server 正处于工作状态，Backup Server 从其启动的控制终端或录会话就必须保持活动状态；否则，sp_volchanged 消息将丢失。

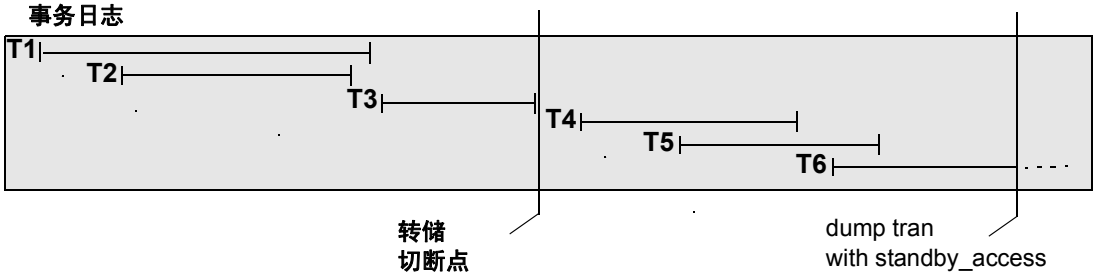
在提供操作员终端功能的操作系统上，卷更改消息始终发送到正运行 Backup Server 的计算机上的操作员终端。使用 notify = client 将其它 Backup Server 消息路由到启动 dump 或 load 请求的终端会话中。

在 UNIX 等系统（这些系统不提供操作员终端功能）上，会将消息发送到启动转储或装载请求的客户端。使用 notify = operator_console 可以将消息传送到启动远程 Backup Server 的终端。

通过 *with standby_access* 使数据库处于联机状态

with standby_access 导致 dump transaction 只转储已完成的事务。它转储事务日志，直到没有活动事务为止。如果您不使用 with standby_access，则将转储整个事务日志，包括所有未结事务的记录。

图 13-1：dump transaction with standby_access 命令的转储切断点



在图 13-1 中，在事务 T1 到 T5 已完成并且事务 T6 仍打开之时发出 dump transaction...with standby_access 命令。因为 T6 仍被打开，所以转储无法包括 T5；并且因为 T5 仍被打开，所以转储无法包括 T4。转储必须在 T3 结束时（在转储切断点之前）停止，此时，它包括已完成的事务 T1 至 T3。

确定何时使用 *with standby_access*

如果您要按顺序装载两个或更多个事务日志，并且希望数据库在各次装载之间保持联机，请使用 dump tran[saction]...with standby_access；例如，如果您拥有一个只读数据库，而该数据库通过从主数据库装载事务转储来获取其数据。如果根据主数据库中的事务使用只读数据库生成每日告，并且主数据库的事务日志在一天结束时转储，则每日操作周期为：

1　在主数据库中：dump tran[saction]...with standby_access

2　在只读数据库中：load tran[saction]...

3 在只读数据库中：online database for standby_access

警告！ 如果某一事务日志包含打开的事务，并且您在转储它时未使用 with standby_access，则 Adaptive Server 将不会允许您装载该日志、使数据库处于联机状态以及装载后面的事务转储。如果您要装载一系列事务转储，则只有在装载 with standby_access 最初生成的转储后或装载整个系列后，才能使数据库处于联机状态。

通过 *with standby_access* 使数据库处于联机状态

online database 命令也包括 with standby_access 选项。使用 for standby_access，可以在通过使用 with standby_access 选项生成的转储装载数据库后使该数据库处于联机状态。

警告！ 如果您尝试将 online database for standby_access 用于未使用 with standby_access 选项转储的事务日志，该命令将失败。

请参见《参考手册：命令》。

获取有关转储文件的信息

如果您不确定磁带的内容，则使用装载命令的 with headeronly 或 with listonly 选项请求该信息。

注释 with headeronly 和 with listonly 均不会在显示报告后装载转储文件。

请求转储标头信息

with headeronly 返回单个文件的标头信息。如果您没有指定文件名，with headeronly 将返回与磁带上的第一个文件有关的信息。

标头指示转储是用于数据库还是用于事务日志、数据库 ID、文件名和进行转储的日期。对于数据库转储，它还显示字符集、排序顺序、页计数和下一对象 ID。对于事务日志转储，标头显示日志中的检查点位置、最早的 begin transaction 记录的位置以及旧的和新的序列日期。

此示例显示磁带上第一个文件的标题信息，然后显示文件 *mydb9229510945* 的标题信息：

```
load database mydb
    from "/dev/nrmt4"
    with headeronly
load database mydb
    from "/dev/nrmt4"
    with headeronly, file = "mydb9229510945"
```

headeronly 的输出看上去类似如下内容：

```
Backup Server session id is:44. Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 4.28.1.1: Dumpfile name 'mydb9232610BC8 ' section number 0001
mounted on device 'backup/SQL_SERVER/mydb.db.dump'
This is a database dump of database ID 5 from Nov 21 1992 7:02PM.
Database contains 1536 pages; checkpoint RID=(Rid pageid = 0x404; row num =
0xa); next object ID=3031; sort order ID=50, status=0; charset ID=1.
```

确定数据库、设备、文件名和日期

with listonly 返回卷上每个转储文件的简要说明。with listonly = full 提供更详细的信息。两个报告都按 SQL 磁带标签进行排序。

load database 命令 with listonly 的输出样本：

```
Backup Server: 4.36.1.1: Device '/dev/nrst0':
File name:'model9320715138  '
Create date & time:Monday, Jul 26, 2007, 23:58:48
Expiration date & time:Monday, Jul 26, 2007, 00:00:00
Database name:'model                          '
```

下面是 with listonly = full 的输出样本：

```
Backup Server: 4.37.1.1: Device '/dev/nrst0':
Label id:'HDR1'
File name:'model9320715138  '
Stripe count:0001
Device typecount:01
Archive volume number:0001
Stripe position:0000
```

```
Generation number:0001
Generation version:00
Create date & time:Monday, Jul 26, 2007, 23:58:48
Expiration date & time:Monday, Jul 26, 2007, 00:00:00
Access code:''
File block count:000000
Sybase id string:
'Sybase  'Reserved:'       '
Backup Server:4.38.1.1: Device '/dev/nrst0':
Label id:'HDR2'
Record format:'F'
Max. bytes/block:55296
Record length:02048
Backup format version:01
Reserved:'   '
Database name:'model                          '
Buffer offset length:00
Reserved:'                                '
```

在列出卷上的所有文件后，Backup Server 发送卷更改请求：

```
Backup Server: 6.30.1.2: Device /dev/nrst0:Volume cataloguing
complete.
Backup Server:6.51.1.1: OPERATOR:Mount the next volume to search.
Backup Server:6.78.1.1: EXECUTE sp_volchanged
@session_id = 5,
          @devname = '/dev/nrst0',
          @action = { 'PROCEED' | 'RETRY' | 'ABORT' },
          @fname = '
```

操作员可以使用 sp_volchanged 装入其它卷并通知卷更改，或者终止所有分条设备的搜索操作。

设备出现故障后复制日志

通常，dump transaction 在复制日志后截断日志的不活动部分。使用 with no_truncate 可以在不截断日志的情况下复制日志。

no_truncate 允许您在保留数据的设备出现故障后复制事务日志。它在 sysdatabases 和 sysindexes 表中使用指针来确定事务日志的物理位置。仅当事务日志位于不同的段上并且可以访问您的 master 数据库时，才可使用 no_truncate。

警告！只有在介质故障使您的数据段不可访问的情况下，才使用 no_truncate。不要针对正在使用的数据库使用 no_truncate。

使用 with no_truncate 复制日志是第 364 页的“恢复数据库：分步指导”中描述的第一步。

您可以将 no_truncate 用于分条转储、磁带初始化和远程 Backup Server：

```
dump transaction mydb
to "/dev/nrmt0" at REMOTE_BKP_SERVER
with init, no_truncate,
notify = "operator_console"
```

截断日志

本节介绍了有关截断日志的问题。

截断不在单独的段上的日志

如果数据库的日志段不在独立于数据段的设备上，则无法先使用 dump transaction 复制日志，然后再将其截断。相反：

1　使用 dump transaction 的特殊的 with truncate_only 选项截断日志，以便它不会用尽空间。因为它不复制任何数据，所以使用 truncate_only 仅要求数据库的名称。

2　使用 dump database 复制整个数据库（包括日志）。

此示例转储数据库 mydb（该数据库的日志段不在独立于数据段的设备上），之后会截断日志：

```
dump database mydb to mydevice
dump transaction mydb with truncate_only
```

在开发环境的早期截断日志

在早期开发环境中，通过创建、删除和重新创建存储过程和触发器以及检查完整性约束来快速填充事务日志。与确保在数据库设备上有足够的空间相，数据恢复的重要性可能较低。

使用 with truncate_only，可以在不创建备份副本的情况下截断事务日志：

```
dump transaction database_name with truncate_only
```

运行 dump transaction with truncate_only 之后，必须先转储数据库，然后才可以运行例行日志转储。

截断没有可用空间的日志

在事务日志非常满时，您可能无法使用常规方法来转储它。如果使用了 dump transaction 或 dump transaction with truncate_only，并且该命令由于日志空间不足而失败，请使用 dump transaction 的 with no_log 选项：

```
dump transaction database_name with no_log
```

此选项截断日志，而不记录转储事务事件。因为此选项不复制任何数据，所以它只要求数据库的名称。

警告！应该将 dump transaction with no_log 用作最后手段，并且在 dump transaction with truncate_only 失败后只使用它一次。如果在输入 dump transaction with no_log 后继续装载数据，日志可能会被完全填满，从而导致随后的任何 dump transaction 命令失败。应使用 alter database 命令为数据库分配额外的空间。

每次使用 dump tran with no_log 都会在 Adaptive Server 错误日志中报告。此外，也会将指示成功或失败的消息发送到错误日志中。no_log 是唯一生成错误日志消息的转储选项。

使用 *with truncate_only* 和 *with no_log* 的风险

with truncate_only 和 with no_log 允许您截断可用空间极其不足的日志。这两个选项都无法恢复自上次例行转储后已提交的事务。

警告！尽早运行 dump database 以确保数据可以恢复。

此示例先截断 mydb 的事务日志，然后再转储数据库：

```
dump transaction mydb
    with no_log
dump database mydb to ...
```

提供足够的日志空间

每次使用 dump transaction...with no_log 都被认为是一种错误并记录在服务器的错误日志中。如果先使用不同于数据段的设备上的日志段创建数据库，编写最后一个机会阈值过程来足够频繁地转储事务日志，然后为日志和数据库分配足够的空间，则不必使用此选项。

但是，某些情况仍可能导致事务日志变得过满，即使使用频繁日志转储也不能缓解。dump transaction 命令通过删除从日志开头到包含未提交的事务记录（通称为最早的活动事务）页的前一页之间的所有页来截断日志。由于 dump transaction 不能截断其它页，因此活动事务未提交的持续时间越长，事务日志中的可用空间就越小。

在具有非常长的事务的应用程序修改具有较小事务日志的数据库中的表时可能发生上述情况，这表明您应该增加日志的大小。当事务长时间保持未提状态时（例如，当隐式 begin transaction 使用链式事务模式时，或者当用户忘记完成事务时），也会发生这种情况。您可以通过查询 syslogshold 系统表来在每个数据库中确定最旧的活动事务。

syslogshold 表

master 数据库中的 syslogshold 表。表中的每行都表示：

- 数据库中最早的活动事务，或
- 数据库日志的 Replication Server 截断点。

数据库在 syslogshold 中可能没有行、具有一个代表上述一项的行，或者具有两个代表上述两项的行。有关 Replication Server 截断点如何影响截断数据库事务日志的信息，请见 Replication Server 文档。

查询 syslogshold 可提供每个数据库中当前状况的快照。由于大多数事务都仅持续短暂的时间，因此查询的结果可能不一致。例如，syslogshold 的第一行中描述的最早活动事务可能会在 Adaptive Server 完成 syslogshold 的查询之前结束。但是，如果 syslogshold 的若干查询经过一段时间后查询数据库的同一行，则该事务可能阻止 dump transaction 截断任何日志空间。

当事务日志达到最后一个机会阈值，并且 dump transaction 无法释放日志中的空间时，可以查询 syslogshold 和 sysindexes 以标识阻止截断的事务。例如：

```
select H.spid, H.name
from master..syslogshold H, threshdb..sysindexes I
where H.dbid = db_id("threshdb")
and I.id = 8
and H.page = I.first

spid    name
------  ------------------------------------
     8  $user_transaction

(1 row affected)
```

此查询使用与 threshdb 数据库中的 syslogs (8) 相关联的对象 ID 将事务日志的第一页与 syslogshold 中最旧的活动事务的第一页匹配。

您也可以查询 master 数据库中的 syslogshold 和 sysprocesses 以标识拥有最旧的活动事务的特定主机和应用程序。例如：

```
select P.hostname, P.hostprocess, P.program_name,
   H.name, H.starttime
from sysprocesses P, syslogshold H
where P.spid = H.spid
and H.spid != 0

hostname hostprocess program_name name                starttime
-------- ----------- ------------ ------------------ ------------------
eagle          15826 isql         $user_transaction  Sep  6 1997 4:29PM
hawk           15859 isql         $user_transaction  Sep  6 1997 5:00PM
condor         15866 isql         $user_transaction  Sep  6 1997 5:08PM

(3 rows affected)
```

使用以上信息，可以通知或关闭拥有最旧活动事务的用户进程，并继续 dump transaction。您也可以在数据库的阈值过程中包括上述类型的查询，作为自动警报机制。例如，您可以决定事务日志应永不达到其最后机会阈值。如果它达到最机会阈值，则最后机会阈值过程 (sp_thresholdaction) 会通过有关阻止事务转储的最旧活动事务的信息提醒您。

注释 事务的初始日志记录可能位于用户日志高速缓存中，在将记录刷新到日志（例如，在检查点之后）之前，该事务不会在 syslogshold 中显示。

请参见《参考手册：表》和第 17 章“使用阈值管理可用空间”。

响应卷更改请求

在 UNIX 和 PC 系统上，使用 sp_volchanged 在已装入正确的卷后通知 Backup Server。

若要使用 sp_volchanged，请登录到任何可与 Backup Server（发出卷更改请求）和 Adaptive Server（启动转储或装载）进行通信的 Adaptive Server。

在覆盖现有转储时，转储到全新磁带时，或者在转储到其内容不可识别的磁带时，Backup Server 会在 ANSI 磁带标签中写入 *vname*。在装载期间，Backup Server 使用 *vname* 确认已装入了正确的磁带。如果您没有指定 *vname*，则 Backup Server 将使用在转储或装载命令中指定的卷名。如果 sp_volchanged 和该命令都没有指定卷名，则 Backup Server 不会在 ANSI 磁带标签中检查此字段。

用于转储的卷更改提示

当您转储数据库或事务日志时，会显示卷更改提示。每个提示都包括可能的操作员操作和适当的 sp_volchanged 响应。

- `Mount the next volume to search.`

  在将某一转储附加到现有卷时，如果 Backup Server 无法找到文件结束标志，它将发出此消息。

操作员可以	通过答复
中止转储	`sp_volchanged session_id, devname, abort`
装入新卷并继续转储	`sp_volchanged session_id, devname, proceed [, fname [, vname]]`

- `Mount the next volume to write.`

  Backup Server 在到达磁带结尾时发出此消息。在以下情况中 Backup Server 会发出此消息：Backup Server 检测到磁带结束标志；转储了 dump 命令的 capacity 参数指定的千字节数目；或者转储了 *sysdevices* 系统表中为设备指定的 high 值。

操作员可以	通过答复
中止转储	`sp_volchanged session_id, devname, abort`
装入下一个卷并继续转储	`sp_volchanged session_id, devname, proceed[, fname[, vname]]`

- `Volume on device devname has restricted access (code access_code).`

  指定 init 选项的转储会覆盖磁带上现有的所有内容。如果尝试在不指定 init 选项的情况下转储到具有 ANSI 访问限制的磁带，则 Backup Server 会发出此消息。

操作员可以	通过答复
中止转储	`sp_volchanged session_id, devname, abort`
装入另一个卷然后重试转储	`sp_volchanged session_id, devname, retry [, fname[, vname]]`
继续转储，覆盖现有的所有内容	`sp_volchanged session_id, devname, proceed[, fname[, vname]]`

- `Volume on device devname is expired and will be overwritten.`

  指定 init 选项的转储会覆盖磁带上现有的所有内容。在转储到单文件介质期间，如果您尚未指定 init 选项并且磁带包含已过有效期的转储，则 Backup Server 将发出此消息。

操作员可以	通过答复
中止转储	`sp_volchanged session_id, devname, abort`
装入另一个卷然后重试转储	`sp_volchanged session_id, session_id, retry[, session_id[, session_id]]`
继续转储，覆盖现有的所有内容	`sp_volchanged session_id, session_id, proceed[, session_id[, session_id]]`

- `Volume to be overwritten on 'devname' has not expired:creation date on this volume is creation_date, expiration date is expiration_date.`

  在单文件介质上，如果您没有指定 init 选项，Backup Server 将检查任何现有转储的有效期。如果转储尚未到期，Backup Server 将发出此消息。

操作员可以	通过答复
中止转储	sp_volchanged *session_id*, *session_id*, abort
装入另一个卷然后重试转储	sp_volchanged *session_id*, *session_id*, retry[, *session_id*[, *session_id*]]
继续转储，覆盖现有的所有内容	sp_volchanged *session_id*, *session_id*, proceed[, *session_id*[, *session_id*]]

- `Volume to be overwritten on 'devname' has unrecognized label data.`

  指定 init 选项的转储会覆盖磁带上现有的所有内容。如果您尝试在不指定 init 选项的情况下转储到新磁带或具有非 Sybase 数据的磁带，Backup Server 将发出此消息。

操作员可以	通过答复
中止转储	sp_volchanged *session_id*, *session_id*, abort
装入另一个卷然后重试转储	sp_volchanged *session_id*, *session_id*, retry[, *session_id*[, *session_id*]]
继续转储，覆盖现有的所有内容	sp_volchanged *session_id*, *session_id*, proceed[, *session_id*[, *session_id*]]

用于装载的卷更改提示

下面是装载期间的卷更改提示和可能的操作员操作：

- `Dumpfile 'fname' section vname found instead of 'fname' section vname.`

  如果 Backup Server 无法在单文件介质上找到指定的文件，它将发出此消息。

操作员可以	通过答复
中止装载	sp_volchanged *session_id*, *session_id*, abort
装入其它卷并尝试装载它	sp_volchanged *session_id*, *session_id*, retry[, *session_id*[, *session_id*]]
将文件装载到当前已装入的卷上，即使它不是指定的文件（不推荐这样做）	sp_volchanged *session_id*, *session_id*, proceed[, *session_id*[, *session_id*]]

- `Mount the next volume to read.`

  当 Backup Server 准备好从多卷转储读取转储文件的下一部分时，它发出此消息。

操作员可以	通过答复
中止装载	sp_volchanged *session_id*, *session_id*, abort
装入下一个卷并继续装载	sp_volchanged *session_id*, *session_id*, proceed[, *session_id*[, *session_id*]]

- `Mount the next volume to search.`

  如果 Backup Server 无法在多文件介质上找到指定的文件，它将发出此消息。

操作员可以	通过答复
中止装载	sp_volchanged *session_id*, *session_id*, abort
装入其它卷并继续装载	sp_volchanged *session_id*, *session_id*, proceed[, *session_id*[, *session_id*]]

恢复数据库：分步指导

介质出现故障时的状况因故障原因的不同而异。如果磁盘上仅有一个块损坏，那么，除非您经常运行 dbcc 命令，否则，在损坏发生后，数据库看上去会正常运行一段时间。如果整个磁盘或磁盘控制器损坏。Adaptive Server 将该数据库标记为可疑数据库并显警告消息。如果存储 master 数据库的磁盘出现故障，则用户将无法登录到服务器，已登录的用户将无法执行需要访问 master 中的系统表的任何操作。

当数据库设备出现故障时，Sybase 会建议您执行下列步骤：

1　获取设备上每个数据库的当前日志转储。

2　检查设备上每个数据库 的空间使用情况。

3　在为设备上的所有数据库收集了这些信息之后，删除每个数据库。

4　使用 sp_dropdevice 删除有故障的设备。请参见《参考手册：过程》。

5　使用 disk init 初始化新数据库设备。请参见《系统管理指南，卷 1》中的第 7 章“初始化数据库设备”。

6　重新创建数据库，一次创建一个。

7　将最新的数据库转储装载到每个数据库中。

8　按各个事务日志转储的创建顺序应用这些转储。

这些步骤需要的信息比此处列出的信息更加详细，以下各节对其进行了讨论。

获取事务日志的当前转储

使用 dump transaction with no_truncate 为出故障设备上的每一数据库获取当前事务日志转储。例如，若要获取 mydb 的当前事务日志转储，请输入：

```
dump transaction mydb
to "/dev/nrmt0" at REMOTE_BKP_SERVER
with init, no_truncate,
notify = "operator_console"
```

检查空间使用情况

使用本节中讨论的步骤确定您的数据库使用哪些设备、在每个设备上分配多少空间以及该空间是用于数据、日志还是两者。您可以在重新创建数据库使用这些信息，以确保日志、数据和索引驻留在单独的设备上并保留已创建的任何用户段的作用域。

注释　您也可以使用这些步骤在将数据库转储从一个服务器移到另一个服务器（在同一硬件和软件平台上）时保留段映射。

如果不使用此信息为损坏的数据库重新创建设备分配，则 Adaptive Server 会在 load database 指明差异之后重新映射 sysusages 表。这意味着数据库的系统定义的段和用户定义的段将不再匹配相应的设备分配。sysusages 中的信息不正确可能导致日志与数据存储在相同的设备中，即使数据和日志在恢复前是分开的也是如此。它也可以通过无法预测的方式更改用户定义的段，并且可能会生成无法使用标准 create database 命令创建的数据库。

检查和记录所有损坏的数据库的设备分配：

1 在 master 中，检查损坏的数据库的设备分配和使用情况：

```
select segmap, size from sysusages
    where dbid = db_id("database_name")
```

2 检查查询的输出。每个 segmap 为 3 的行都表示数据分配。每个 segmap 为 4 的行都表示日志分配。较高的值指示用户定义的段；将这些段作为数据分配处理，以保留这些段的作用域。size 列指示数据块的数目。记录每一磁盘区段的顺序、用途和大小。

例如，以下输出来自使用 2K 逻辑页的服务器，它转换为表 13-6 中所述的大小和用途：

```
segmap      size
-------     --------
     3        10240
     3         5120
     4         5120
     8         1024
     4         2048
```

表 13-6：设备分配示例

设备分配	兆字节
数据	20
数据	10
日志	10
数据（用户定义的段）	2
日志	4

注释　如果 segmap 列包含 7，则表示您的数据和日志位于同一设备上，并且您最多只能恢复到最新的数据库转储。不要将 log on 选项用于 create database。只需确保您分配的空间等于或多于 sysusages 报告的总量。

3　为数据库运行 sp_helpdb *database_name*。以下查询列出保存数据和日志的设备：

```
name        db_size owner   dbid    created      status
-------     ------- ------  ----    -----------  -------------
mydb        46.0 MB sa        15    May 26 2005  no_options set

device_fragments  size    usage      created              free kbytes
----------------  -----   --------   ------------         -----------
datadev1          20 MB   data only  June 7 2005 2:05PM         13850
datadev2          10 MB   data only  June 7 2005 2:05PM          8160
datadev3           2 MB   data only  June 7 2005 2:05PM          2040
logdev1           10 MB   log only   June 7 2005 2:05PM  not applicable
logdev2            4 MB   log only   June 7 2005 2:05PM  not applicable
```

删除数据库

在您为出现故障的设备上的所有数据库执行了前面所述的步骤后，使用 drop database 删除每一数据库。

注释　如果其它数据库中的表包含对您尝试删除的数据库中任何表的引用，则必须先使用 alter table 删除参照完整性约束，然后再删除数据库。

如果当您发出 drop database 时，系统由于数据库受损而报告错误，请使用：

```
dbcc dbrepair (mydb, dropdb)
```

如果您使用的是已复制的数据库，请使用 dbcc dbrepair 将转储从以前版本的 Adaptive Server 装载到较新的版本中。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。

请参见《错误消息和故障排除指南》以及《参考手册：命令》。

重新创建数据库

使用以前收集的段信息重新创建每个数据库。

注释 如果您选择了不收集与段使用率有关的信息，则使用 create database...for load 创建大小至少与原始大小相等的新数据库。

1 将 create database 与 for load 选项一起使用。从 sysusages 表中复制数据库的每个行的所有设备段映射和大小，直到第一个日志设备（包括该设备）。以这些行在 sysusages 中出现的顺序使用它们。（sp_helpdb 的结果按设备名的字母顺序排列，而不是按分配顺序排列。）例如，若要重新创建第 366 页的表 13-6 中所示的 mydb 数据库分配，请输入：

```
create database mydb
    on datadev1 = 20,
        datadev2 = 10
log on logdev1 = 10
for load
```

注释 create database...for load 临时锁定新创建的数据库外的用户，并且 load database 将数据库标记为脱机状态以供常规使用。这防止用户在恢复期间执行记录的事务。

2 将 alter database 与 for load 选项一起使用以按顺序重新创建其余的条目。请记住，应像您处置数据分配一样为用户段处置设备分配。

在此示例中，为了在 datadev3 上分配更多的数据空间，在 logdev1 上分配更多的日志空间，使用的命令是：

```
alter database mydb
    on datadev3 = "2M"
log on logdev1= "4M"
for load
```

装载数据库

使用 load database 重装数据库。如果原始数据库在用户定义的段上存储了对象（sysusages 报告 segmap 大于 7），并且您的新设备分配匹配转储的数据库的设备分配，则 Adaptive Server 保留用户段映射。

如果您没有创建新设备分配以匹配转储的数据库的设备分配，则 Adaptive Server 会将段重新映射到可用的设备分配。此重新映射还混合同一物理设备上日志和数据。

注释　如果在装载数据库时发生了其它故障，则 Adaptive Server 将不恢复部分装载的数据库，并且向用户通知这一情况。您必须通过重复执行 load 命令重新开始数据库装载。

装载事务日志

使用 load transaction 以事务日志备份的生成顺序应用事务日志备份。

Adaptive Server 检查每一转储的数据库和事务日志上的时间戳。如果转储以错误顺序装载，或者用户事务在两次装载之间修改了事务日志，则装载将失败。

如果您使用 with standby_access 转储了事务日志，则也必须使用 standby_access 装载数据库。

在您使数据库处于最新状态后，使用 dbcc 命令检查其一致性。

装载事务日志到某个时间点

您可以恢复数据库，一直恢复到其事务日志中某个指定的时间点。为此，请使用 load transaction 的 until_time 选项。例如，在用户无意中删除了某个重要的表的情况下可以使用这一选项；您可以使用 until_time 将对包含该表的数据库进行的更改一直恢复到刚删除该表之前那一时刻的状态。

为了在数据已被损坏后有效使用 until_time，您必须知道发生错误的确切时间。您可以通过在出现错误时发出 select getdate 来找到这一时间。例如，假定用户无意中删除了一个重要的表，然后在几分钟之后您以毫秒为单位获取当前时间：

```
select convert(char(26), getdate(), 109)

--------------------------
Mar 26 2007 12:45:59:650PM
```

在转储包含错误的事务日志并装载最新的数据库转储后，装载在最后转储数据库后创建的事务日志。然后，使用 until_time 装载包含错误的事务日志；在此示例中，大约是在 10 分钟之前：

```
load transaction employees_db
from "/dev/nrmt5"
with until_time = "Mar 26 2007 12:35:59:650PM"
```

在使用 until_time 装载了事务日志之后，Adaptive Server 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，不能在使用 until_time 执行 load transaction 命令后装载之后的事务日志。您必须首先转储该数据库，然后才能转储其它事务日志。

使数据库处于联机状态

将所有事务日志转储应用到数据库之后，使用 online database 使其可供使用。例如，若要使 mydb 数据库联机，请输入：

```
online database mydb
```

复制型数据库

在将复制型数据库升级到 Adaptive Server 的当前版本前，这些数据库必须处于联机状态。但是，在清除日志前，不能使复制型数据库处于联机状态。如您尝试在日志清除之前使复制型数据库联机，则 Adaptive Server 将发出：

```
Database is replicated, but the log is not yet
drained.This database will come online automatically
after the log is drained.
```

在 Replication Server 通过 Log Transfer Manager (LTM) 清除日志后，将自动发出 online database。

有关具有复制型数据库的 Adaptive Server 用户的升级指导，请参考所用平台的安装文档。

装载序列

装载复制型数据库的装载序列是：load database、replicate、load transaction、replicate 等。在装载序列的末尾，发出 online database 以使数据库处于联机状态。由于数据库处于装载序列中而处于脱机状态的数据库不被 Replication Server 自动进入联机状态。

警告！在装载了所有事务日志前不要发出 online database。

从较旧的版本装载数据库转储

在大多数升级期间，所有与服务器相关联的数据库都会自动升级（重要升级 — 比如，从 12.5.x 版升级到 15.5 版 — 要求您运行 preupgrade 和 upgrade 实用程序来完成升级）。

因此，必须首先升级用 Adaptive Server 的以前的版本创建的数据库和事务日志转储，然后才能将它们用于 Adaptive Server 的当前版本。

Adaptive Server 提供自动升级机制（按数据库）来将使用 Backup Sever 创建的数据库或事务日志升级到当前 Adaptive Server 版本，从而使转储兼容以便使用。种机制完全由 Adaptive Server 内部提供，不需要外部程序。它还使用户可以根据需要灵活选择个别的转储进行升级。

但是，自动升级功能不支持这些任务：

- 装载 master 数据库的旧版本。也就是说，如果已将 Adaptive Server 升级到当前版本，则不能装载从其升级的原 master 数据库的转储。
- 安装新的或修改过的存储过程。继续使用 installmaster。

将转储升级到当前版本的 Adaptive Server

将用户数据库或事务日志转储升级到 Adaptive Server 的当前版本：

1　使用 load database 和 load transaction 以装载要被升级的转储。

Adaptive Server 从转储标头确定它要装载哪一版本。在读取了转储标头后，但在 Backup Server 开始装载前，使用 load database 或 load transaction 将数据库标记为脱机。这将使数据库不可用于常规使用（不允许查询和 use *database*），向用户提供对装载序列的更高的控制，以及消除其他用户可能无意中中断装载序列的隐患。

2　在转储已成功装载后，使用 online database 激活升级进程。

注释　在所有事务转储都已装载前，不要发出 online database。

对于从 12.0 版和更高版本装载的转储，online database 会激活升级过程以升级刚装载的转储。在升级成功完成后，Adaptive Server 将数据库置于联机状态，该数据库即可供使用。

对于从当前运行的 Adaptive Server 版本装载的转储，不会激活升级过程。您仍必须发出 online database 才能使数据库联机 — load database 将其标记为脱机。

每个升级步骤都生成一条消息，指出要做的事项。

升级失败将使数据库处于脱机状态，并生成一条消息，指出升级已失败并且用户必须纠正该问题。

请参见《参考手册：命令》。

3 在成功执行 online database 后，请使用 dump database。必须首先转储数据库，然后才允许 dump transaction。在成功进行 dump database 之前，不允许对新创建的或升级的数据库执行 dump transaction。

数据库脱机状态位

数据库脱机状态位表示数据库不可用于一般用途。您可以通过使用 sp_helpdb 确定数据库是否处于脱机状态。如果设置此位，则表示数据库处于脱机状态。

如果 load database 将数据库标记为脱机，则会设置 sysdatabases 表中的状态位，并且将保持该设置状态，直到成功完成 online database。

“数据库脱机”状态位与任何现有状态位一起使用。它扩大以下状态位以提供附加控制：

- 正在恢复

“数据库脱机”状态位覆盖下列状态位：

- 只用于 DBO
- 只读

下列状态位会覆盖“数据库脱机”状态位：

- 已开始升级
- 绕过恢复
- 正在装载
- 未恢复
- 可疑
- 使用未恢复

尽管该数据库无法供常规使用，但您可以在数据库处于脱机状态时使用以下命令：

- dump database 和 dump transaction
- load database 和 load transaction
- alter database on device
- drop database
- online database
- dbcc 诊断（取决于 dbcc 限制）

版本标识符

自动升级功能为 Adaptive Server、数据库和日志记录格式提供版本标识符：

- 配置升级版本 ID — 显示 Adaptive Server 的当前版本；它存储于 sysconfigures 表中。sp_configure 将 Adaptive Server 的当前版本显示为“升级版本”。
- 升级版本指示符 — 显示数据库的当前版本并存储在数据库和转储标头中。Adaptive Server 恢复机制使用该值确定在使数据库可供常规使用之前是否应升该数据库。
- 日志兼容性版本指示符 — 通过在数据库、数据库转储或事务日志转储中显示日志记录的格式来将日志与早期和更高版本的 Adaptive Server 区分开。此常量存储于数据库和转储标头中，由 Adaptive Server 用来检测恢复期间日志记录的格式。

高速缓存绑定和装载数据库

如果您转储一个数据库并用不同的高速缓存绑定将它装载到某一服务器中，则应注意区分用于数据库的高速缓存绑定和用于该数据库中对象的高速缓绑定。您最好将该数据库装载到不同的服务器中以用于调优或开发工作，或者最好装载在您进行转储后从高速缓存绑定已更改的服务器删除的数据。

在恢复后使数据库处于联机状态或在装载后通过 online database 使数据库处于联机状态时，Adaptive Server 为该数据库和数据库对象验证所有高速缓存绑定。如果高速缓存不存在，Adaptive Server 将向错误日志写入一个告，并且 sysattributes 中的绑定被标记为无效。下面是来自错误日志的消息的示例：

```
Cache binding for database '5', object '208003772',
index '3' is being marked invalid in Sysattributes.
```

无效的高速缓存绑定未被删除。如果您创建相同名称的高速缓存并重新启动 Adaptive Server，则绑定被标记为有效并使用该高速缓存。如果不使用同一名称创建高速缓存，则可以将对象绑定到另一高速缓存，或允许它使用缺省高速缓存。

在以下各节（讨论高速缓存绑定主题）中，目标服务器是指正在其中装载数据库的服务器，而原始服务器是指在其中创建转储的服务器。

如有可能，重新创建在目标服务器上与在原始服务器上绑定名称相同的高速缓存。如果为可被移植回原始服务器的性能测试和开发或类似用途使用目数据库，则最好以完全相同的方式配置缓冲池。如果要将目标数据库用于决策支持或运行 dbcc 命令，则可能需要将池配置为允许在 16K 的内存池中增添空间。

数据库和高速缓存绑定

数据库的绑定信息存储在 master..sysattributes 中。在数据库本身中不存储任何与数据库绑定有关的信息。如果您使用 load database 通过绑定到某一高速缓存的现有数据库装载转储，则在发出装载命令前不要删除该数据库，这并不影响该绑定。

如果正装载的数据库已绑定到原始服务器上的某一高速缓存，则您可以：

- 将目标服务器上的数据库绑定到为满足该服务器上的需要而配置的高速缓存，或者
- 为满足目标服务器中应用程序的需要配置目标服务器上缺省数据高速缓存中的缓冲池，并且不将该数据库绑定到指定的数据高速缓存。

数据库对象和高速缓存绑定

对象的绑定信息存储在数据库本身的 sysattributes 表中。如果您频繁将数据库装载到目标服务器，则最简单的解决办法是在目标服务器上配置相同名称的高速缓存。

如果目标服务器不是用与原始服务器上相同名称的高速缓存配置的，则在使数据库处于联机状态后将对象绑定到目标服务器上的适当高速缓存，或者保缺省高速缓存是为满足该服务器上的需要而配置的。

检查高速缓存绑定

使用 sp_helpcache 可显示数据库对象的高速缓存绑定，即使高速缓存绑定无效。

以下 SQL 语句根据用户数据库的 sysattributes 表中的信息重新生成高速缓存绑定命令：

```
/* create a bindcache statement for tables */

select "sp_bindcache "+ char_value + ", "
    + db_name() + ", " + object_name(object)
from sysattributes
where class = 3
    and object_type = "T"

/* create a bindcache statement for indexes */

select "sp_bindcache "+ char_value + ", "
    + db_name() + ", "   + i.name
from sysattributes, sysindexes i
where class = 3
    and object_type = "I"
    and i.indid = convert(tinyint, object_info1)
    and i.id = object
```

跨数据库约束和装载数据库

如果您使用 create table 或 alter database 的 references 约束来跨数据库引用表，则在尝试装载其中一个数据库的转储时可能会遇到问题。

- 如果某一数据库中的表引用一个转储的数据库，而您用不同的名称装载该数据库或装载并非它从其转储的服务器上的数据库，则会导致参照完整性错。若要在您重装数据库时更改它的名称或位置，请在引用数据库中使用 alter database 以在转储该数据库前删除所有外部参照完整性约束。
- 如果装载比引用数据库版本低的被引用数据库的转储，可能会导致一致性问题或数据损坏。作为防范措施，每次添加或删除跨数据库约束或删除包含数据库约束的表时，两个受影响的数据库都应转储。
- 同时转储所有彼此引用的数据库。为了避免发生同步问题，应将两个数据库都置于单用户模式以供转储。在装载数据库时，使这两个数据库同时处于机状态。

在以下情况下，跨数据库约束可能会变得不一致：

- 不以发生的时间顺序装载数据库转储（例如，您首先装载在 1997 年 8 月 13 日创建的转储，再装载在 1997 年 8 月 12 日创建的转储），或者
- 将转储装载到使用新名称的数据库中。

如果不装载数据库转储，则跨数据库约束可能会变得不一致。

要解决此问题：

1 使两个数据库都处于单用户模式。

2 删除不一致的参照约束。

3 使用如下查询检查数据一致性：

```
select foreign_key_col from table1
where foreign_key not in
(select primary_key_col from otherdb..othertable)
```

4 修复任何数据不一致问题。

5 重新创建约束。

第 14 章

恢复系统数据库

恢复系统数据库

系统数据库的恢复过程取决于所使用的数据库和系统中出现的问题。通常，恢复可能包括：

- 使用 load database 装载这些数据库的备份，
- 使用 dataserver、installmaster 和 installmodel 恢复这些数据库的初始状态，或
- 这些任务的组合。

要使系统数据库的恢复尽可能高效地进行：

- 不要在主设备上存储用户数据库或除 master、tempdb、model 和 sybsystemdb 以外的任何其它数据库。
- 始终保存重要系统表的最新打印输出。
- 每次执行初始化数据库设备、创建或变更数据库或添加新的服务器登录名等操作之后，都要备份 master 数据库。

恢复 *master* 数据库

受损的 master 数据库可能是由存储 master 的区域中的介质故障导致，也可能是由数据库的内部损坏导致。如果出现以下情况，将会损坏 master 数据库：

- Adaptive Server 不能启动。
- 出现频繁或破坏性的分段故障错误或输入/输出错误。
- dbcc 在定期检查数据库期间报告损坏。

本节讲述如何恢复 master 数据库以及如何重建主设备。本节假定：

- master 数据库已损坏，或主设备已损坏。
- 您拥有系统表的最新输出，《系统管理指南，卷 1》中的第 2 章“系统及可选数据库”列出了这些输出。
- 主设备只包括 master 数据库、tempdb、model 和 sybsystemdb。
- 您拥有 master 数据库的最新备份，并且，自上次转储 master 以来，您尚未初始化任何设备，也未创建或修改任何数据库。
- 您的服务器使用缺省排序顺序。

《错误消息和故障排除指南》更加完整地介绍了恢复方案。

关于恢复进程

由于 master 数据库和主设备具有中央控制特性，因此需要采用特殊的步骤。master 中的表用于配置和控制所有 Adaptive Server 的功能、数据库和数据设备。恢复进程包括：

- 将主设备重建为第一次安装服务器时的缺省状态
- 将 master 数据库恢复为缺省状态
- 将 master 数据库恢复为上次备份时的状态

在恢复 master 数据库的早期阶段，不能使用系统存储过程。

恢复过程总结

必须遵循以下步骤来恢复损坏的主设备。以下各页更加详细地讨论了每个步骤。

1　查找恢复磁盘、数据库和登录名所需的系统表的书面副本。

2　关闭 Adaptive Server，并使用 dataserver 构建新 master 数据库和主设备。

3　以主恢复方式重新启动 Adaptive Server。

4　在 sysusages 中正确地重建 master 数据库的分配。

5　更新 sysservers 表中 Backup Server 的网络名。

6　检验 Backup Server 以确保其正在运行。

7　使用 load database 装载 master 的最新数据库转储。成功装载 master 后，Adaptive Server 将自动停止。

8　在配置文件中更新 number of devices 配置参数。

9　在单用户模式下重新启动 Adaptive Server。

10　检验 master 的备份是否拥有最新的系统表信息。

11　重新启动 Adaptive Server。

12　如果自上次备份 master 后添加了新的登录名，则检查 syslogins。

13　恢复 model 数据库。

14　将 sysusages 和 sysdatabases 的书面副本与新的联机版本进行比较，针对每个数据库运行 dbcc checkalloc，并检查每个数据库中的重要表。

15　转储 master 数据库。

查找系统表的副本

查找已保存到文件中的系统表的副本：sysdatabases、sysdevices、sysusages、sysloginroles 和 syslogins。在这些过程结束时使用它们，以确保已经完全恢复了系统。

请参见《系统管理指南，卷 1》中的第 2 章“系统及可选数据库”。

建立新的主设备

仅当旧的主设备的损坏无法修复时，才应构建新的主设备。否则，您可以在现有主设备上重新创建 master 和 model 数据库。

重新创建 master 数据库有两个过程：替换主设备，并强制 Adaptive Server 重新创建配置区域。当 master 数据库损坏时替主设备。如果主设备的配置区域也损坏，请强制 Adaptive Server 重新创建配置区域。您可以经常检测配置区域是否由于服务器无法运行而损坏，并生成错误消息来指明该区域已损坏。

以下示例使用 UNIX 的 dataserver 命令。在 Windows 平台上，使用 sqlsrvr 命令。

❖ **替换主设备**

1 用 dataserver -w 选项重建主设备：

```
dataserver -w master
```

注释 dataserver 命令可以包括其它选项，如指定设备路径名、服务器名、接口文件名等等的命令行标志。这些选项列在以正常方式启动此服务器的 *RUN_servername* 文件中。

-w master 选项可使 Adaptive Server 为属于 master 数据库的数据库片段搜索设备。Adaptive Server 找到这些片段后，它会将缺省的 master 数据库写入该空间，然后关闭。

2 用 *RUN_servername* 文件重新启动 Adaptive Server。

❖ **重建配置区域**

如果配置区域损坏，则必须使用 -f 选项强制 Adaptive Server 重建该区域。

注释 dataserver 命令可以包括其它选项，如指定设备路径名、服务器名、接口文件名等等的命令行标志。这些选项列在以正常方式启动此服务器的 *RUN_servername* 文件中。

但有以下限制：

- 如果页大小错误，则您可以指定页大小（例如，`-z8k`）。
- 如果设备大小错误，则您可以指定设备大小（例如， `-b125M`）。

- 磁盘上显示为损坏或当前未分配的所有分配单元都将分配给 master 数据库。

警告！ *不要*尝试使用此过程更改服务器的逻辑页大小。这样做会将设备进一步损坏到无法恢复的程度。所指定的任何设备大小都必须准确无误。dataserver -w 不会更改设备的大小，但是，如果指定的大小太小，则它可能找不到某些数据库的某些部分，如果指定的大小太大，则它将完全失败。

带或不带 -f 的 -w master 选项仅重新创建 master 数据库。磁盘上的所有其它分配单元保持不变，因此，您可以使用 disk refit 恢复数据。

如果整个主设备已损坏（例如，如果磁盘发生故障），请通过使用 dataserver -z*page_size*.. .-b*device_size* 启动 Adaptive Server 来更换整个设备：

1　如果现有主设备不在原始分区上，并且您打算重新使用该设备，请删除旧的主设备文件。

2　使用 dataserver -z*page_size*.. .-b*device_size* 启动服务器。

警告！ 此时，无法使用此命令更改服务器的页大小。该服务器的所有其它设备都使用已配置的页大小，如果您使用不同的页大小，将无法正确恢复。但是，于您要创建新文件，因此可以更改设备大小。

确定将主设备设置为多大时，请记住，此设备会保留 8KB 作为其配置区域。例如，如果指定的设备大小为 96MB，则会因为没有足够的空间用于完整分配单元，而浪费一些空间。在所需的空间中额外添加 .01MB 以供设备用于此开销。例如，若要为设备使用全部 96MB，请指定 -b96.01M。

在使用此方法重建主设备时，Adaptive Server 会为 master、model、tempdb 和 sybsystemdb 创建新的缺省数据库。这些数据库都尽可能小，以适应您的安装的逻辑页大小。如果想要向这些数据库中装载备份，它们会显得太小。装载这些备份之前，使用 alter database 增加数据库的大小。

无论使用何种方法恢复主设备，master 数据库中的所有用户和口令都将消失。唯一一个有权限的登录名是“sa”，该用户名没有口令。

有关重新创建用户和口令的信息，请参见《实用程序指南》。

在主恢复方式下启动 Adaptive Server

使用 -m（UNIX 和 Windows）选项在主恢复方式下启动 Adaptive Server。

- 在 UNIX 平台上，复制 runserver 文件，并将其命名为 *m_RUN_server_name*。编辑该新文件，在 dataserver 命令行添加参数 -m。然后在主恢复方式下启动服务器：

  ```
  startserver -f m_RUN_server_name
  ```

- 在 Windows 平台上 — 从命令行使用 sqlsrver 命令启动 Adaptive Server。除其它必要的参数外，指定 -m 参数。例如：

```
sqlsrver.exe -dD:\Sybase\DATA\MASTER.dat -sPIANO
-eD:\Sybase\install\errorlog -iD:\Sybase\ini -MD:\Sybase -m
```

有关这些命令的复杂语法，请参见《实用程序指南》。

在主恢复方式下启动 Adaptive Server 时，只允许使用一个用户（系统管理员）的一个登录名。之后立即对 master 数据库执行 dataserver 命令，此时只存在“sa”帐户，其口令为 NULL。

警告！有些节点具有一些自动作业，这些作业可在服务器启动时使用“sa”登录名登录到服务器中。应确保禁用了这些作业。

主恢复方式是必需的，这是因为使用 dataserver 创建的一般 master 数据库与 Adaptive Server 中的实际情况不一致。例如，此数据库中没有数据库设备的任何信息。对 master 数据库的任何操作都可能会使恢复更加复杂而且耗时。

在主恢复方式下启动的 Adaptive Server 被自动配置为允许直接更新系统表。不允许某些其它操作。

警告！不要对系统表进行特定更改，有些更改可致使 Adaptive Server 无法运行。只进行本章中描述的更改，并始终在用户定义的事务中进行更改。

重新创建 *master* 的设备分配

如果您根据上面步骤 2 中描述的过程重新创建了主设备，则您的 master 数据库现在可能太小。为 master 数据库分配更多空间：

1　从 sysusages 的书面副本中，对为 dbid 1（master 数据库的 dbid）显示的 size 值求和。将这些值与当前 master 数据库的大小进行比较。您可以通过发出以下命令来确定它们：

```
select sum(size)
from sysusages
where dbid = 1
```

2　如果当前的 master 数据库太小，请使用 alter database 将它扩大至所需的大小。若要将逻辑页转换为 MB 级页，请使用：

```
select N / (power(2,20) / @@maxpagesize)
```

其中 *N* 是逻辑页的数量。

如果使用 -m master 选项重新编写了 master 数据库，则不必更改 master 数据库的大小。由于 Adaptive Server 已记录了设备上所有数据库使用的分配单元，因此，您应该已有足够的空间装载 master 的转储。

注释　如果没有 sysusages 的书面副本，可以通过尝试进行装载以确定数据库需要再增加多大。如果数据库太小，Adaptive Server 会显示一条错误消息，告诉您还需要使数据库增多大。

早于 15.0 的 Adaptive Server 版本需要执行一系列复杂的步骤来在新的主设备上重新创建分配。现在不再需要此过程了。Adaptive Server 15.0 版及更高版本可动执行这项工作的大部分。

检查 Backup Server *sysservers* 信息

使用空口令以“sa”身份登录服务器。

如果 Backup Server 的网络名称不是 SYB_BACKUP，请更新 sysservers，以便 Adaptive Server 可以与其 Backup Server 进行通信。在接口文件中检查 Backup Server 名称，然后发出：

```
select *
from sysservers
where srvname = "SYB_BACKUP"
```

从此命令的输出中检查 srvnetname。如果与服务器的 Backup Server 的接口文件条目匹配，则转至第 384 页的“检验 Backup Server 是否在运行”。

如果报告的 srvnetname 与接口文件中的 Backup Server 不同，请更新 sysservers。下例将 Backup Server 的网络名更改为 PRODUCTION_BSRV：

```
begin transaction
update sysservers
set srvnetname = "PRODUCTION_BSRV"
where srvname = "SYB_BACKUP"
```

执行此命令，并验证它是否仅修改了一行。重新发出 select 命令，然后核实是否修改了正确的行，该行是否包含正确的值。如果 update 修改了多行，或者修改了不应修改的行，则应发出 rollback transaction 命令，然后尝试再次更新。

如果该命令正确地修改了 Backup Server 的行，则应发出 commit transaction 命令。

检验 Backup Server 是否在运行

在 UNIX 平台上，使用 showserver 命令核实 Backup Server 是否正在运行；如果必要，重新启动 Backup Server。请参见《实用程序指南》中的 showserver 和 startserver。

在 Windows 平台上，安装在本地的 Sybase Central 和 Services Manager 指明 Backup Server 是否在运行。

有关用于启动 Backup Server 的命令，请参见《实用程序指南》。

装载 *master* 的备份

装载 master 数据库的最新备份。

- 例如，在 UNIX 平台上，使用：

  ```
  load database master from "/dev/nrmt4"
  ```

- 在 Windows 平台上，使用：

  ```
  load database master from "\\.\TAPE0"
  ```

有关命令语法的信息，请参见第 13 章“备份和恢复用户数据库”。

在 load database 成功完成之后，Adaptive Server 会关闭。注意在装载过程和关闭过程中是否有错误消息。

更新 *number of devices* 配置参数

仅当使用的数据库设备数比缺省值多时才执行此步骤。否则，转至第 385 页的“在主恢复方式下重新启动 Adaptive Server”。

除非恢复 master 数据库，否则 Adaptive Server 无法使用配置值，因此，请指示 Adaptive Server 在启动时从配置文件中读取适当的 number of devices 参数值。

如果最新配置文件不可用，请编辑配置文件以反映 number of devices 参数的正确值。

编辑 runserver 文件。在 dataserver 或 sqlsrver 命令的末尾添加 -c 参数，以指定配置文件的名称和位置。Adaptive Server 启动时，将从指定的配置文件中读取参数值。

在主恢复方式下重新启动 Adaptive Server

使用 startserver，以主恢复方式重新启动 Adaptive Server（请参见第 382 页的“在主恢复方式下启动 Adaptive Server”）。在恢复过程中注意是否有任何错误消息。

装载 master 的备份时，会使“sa”帐号恢复到先前的状态。如果“sa”帐户有口令，则恢复该口令。如果在进行备份之前使用 sp_locklogin 锁定了此帐户，则“sa”帐户会立即锁定。将帐户与 sa_role 配合使用来执行其余恢复步骤。

检查系统表以检验 *master* 的当前备份

如果在发出最新的 disk init、create database 或 alter database 命令后，已备份了 master 数据库，则 sysusages、sysdatabases 和 sysdevices 的内容会与书面副本匹配。

依据书面副本，检查恢复后服务器中的 sysusages、sysdatabases 和 sysdevices 表。尤其注意以下问题：

- 如果书面副本中的设备有的未包括在已恢复的 sysdevices 中，则自上次备份以来已经添加了设备，并且您必须运行 disk reinit 和 disk refit。请参见第 394 页的“使用 disk reinit 和 disk refit 恢复系统表”。
- 如果书面副本中列出的数据库有的未包括在已恢复的 sysdatabases 表中，则意味着自上次备份 master 以来添加了数据库。您必须运行 disk refit。

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，一定要先阅读《故障排除和错误消息指南》，然后再用任何跟踪标志启动 Adaptive Server。

重新启动 Adaptive Server

以常规（多用户）模式重新启动 Adaptive Server。

恢复服务器用户 ID

检查 syslogins 的书面副本和恢复的 syslogins 表。

- 如果自上次备份 master 以来已经添加了服务器登录名，请重新发出 create login 命令。
- 如果已删除了服务器登录名，则应重新发出 drop login 命令。
- 如果已锁定了服务器帐户，则应重新发出 sp_locklogin 命令。
- 检查由于用户或系统管理员使用 alter login 而引起的其它差别。

确保指派给用户的 suids 正确。数据库中不匹配的 suid 值会导致权限问题，用户可能不能访问表或运行命令。

检查现有 suid 值的有效方法是对用户数据库的每个 sysusers 表执行 union。如果用户有权使用 master，则可以在此过程中包括 master。

例如：

```
select suid, name from master..sysusers
union
select suid, name from sales..sysusers
union
select suid, name from parts..sysusers
union
```

```
select suid, name from accounting..sysusers
```

如果结果列表显示的已跳过的 suid 值介于您在其中恢复登录名的范围内，请为跳过的值添加占位符，然后使用 drop login 删除它们或者使用 sp_locklogin 锁定它们。

恢复 *model* 数据库

恢复 model 数据库：

- 装载 model 的备份（如果您拥有备份）。
- 如果您没有备份，请运行 installmodel 脚本（大多数平台上都提供此脚本）：

```
cd $SYBASE/$SYBASE_ASE/scripts
isql -Usa -Ppassword -Sserver_name < installmodel
```

在 Windows 中：

```
cd $SYBASE/$SYBASE_ASE/scripts
isql -Usa -Ppassword -Sserver_name < instmodl
```

- 重复对 model 所做的任何更改。

检查 Adaptive Server

仔细检查 Adaptive Server：

1　将 sysusages 的书面副本与新的联机版本进行比较。

2　将 sysdatabases 的书面副本与新的联机版本进行比较。

3　针对每个数据库运行 dbcc checkalloc。

4　检查每个数据库中重要的表。

警告！ 如果在 sysusages 中发现差异之处，请与 Sybase 技术支持部门联系。

备份 *master*

完全恢复 master 数据库并运行全部 dbcc 完整性检查后，使用常规转储命令备份此数据库。

恢复 *model* 数据库

本节说明在 model 数据库是唯一需要恢复的数据库时，如何进行恢复。

在不影响 master 的情况下使用 dataserver 恢复 model 数据库。

警告！应首先关闭 Adaptive Server，然后才能使用任何 dataserver 命令。

- 在 UNIX 平台上：

```
dataserver -d /devname  -w model
```

- 在 Windows 中：

```
sqlsrvr -d physicalname -w model
```

如果可以成功发出 use model，则可以使用 load database 从备份恢复 model 数据库。

如果不能使用此数据库：

1 发出上述 dataserver 命令。

2 如果已经更改了 model 的大小，请重新发出 alter database。

3 使用 load database 装载备份。

如果已更改了 model 数据库，且没有备份：

1 发出上述 dataserver 命令。

2 重新发出更改 model 时所发出的所有命令。

恢复 *sybsystemprocs* 数据库

sybsystemprocs 数据库存储那些用于修改和报告系统表的系统过程。如果常规 dbcc 检查报告损坏，而又未保留此数据库的备份，则可使用 *installmaster* 恢复该数据库。如果保留了 sybsystemprocs 的备份，则可使用 load database 命令恢复数据库。

使用 *installmaster* 恢复 *sybsystemprocs*

本节假定您的 sybsystemprocs 数据库存在，但已损坏。如果 sybsystemprocs 不存在，必须使用 create database 创建它。

若要使用 installmaster 恢复 *sybsystemprocs*：

1　查看 sybsystemprocs 当前存储在哪个设备中：

```
select  lstart,
size / (power(2,20)/@@maxpagesize) as 'MB',
d.name as 'device name',
case when segmap = 4 then 'log'
when segmap & 4 = 0 then 'data'
else 'log and data'
end as 'usage'
from sysusages u, sysdevices d
where d.vdevno = u.vdevno
and d.status & 2 = 2
and dbid = db_id('sybsystemprocs')
order by 1
```

结果可能会指明，sybsystemprocs 全部在一个磁盘片段上，并且将日志和数据作为使用内容，但您的布局可能会比此更加复杂。保存此查询的结果以便在以后使用。

2　删除此数据库：

```
drop database sybsystemprocs
```

如果操作成功且设备未损坏，请转到第 3 步。

- 如果 sybsystemprocs 严重损坏，则 drop database 可能会失败。通过删除用于标识数据库的信息来手动删除该数据库：

```
sp_configure ‘allow updates’, 1
go
delete from sysusages
where dbid = db_id('sybsystemprocs')
delete from sysdatabases
where name = 'sybsystemprocs'
go
sp_configure 'allow updates', 0
go
```

- 如果物理磁盘已损坏，请删除该设备：

```
sp_dropdevice name_of_sybsystemprocs_device
```

如果手动删除 sybsystemprocs 后又重新创建了 sybsystemprocs 设备，请使用 shutdown with nowait 关闭 Adaptive Server。如果删除了 sybsystemprocs 设备，并且它不是原始分区，则应删除物理文件。重新启动 Adaptive Server。

3 重新创建 sybsystemprocs 设备。如果删除了 sybsystemprocs 设备，请使用 disk init 创建一个新设备。然后，使用在步骤 1 中保存的结果通过下列方法之一重新创建 sybsystemprocs。

注释 如果计划装载 sybsystemprocs 的备份副本，则可以使用包括 for load 选项的 create database 或 alter database 命令。但是，您必须先使用 load database 装载备份副本，然后再将它用于任何其它目的。

- 如果显示的使用情况全部是日志和数据，请使用以下命令创建一个简单的数据库：

  ```
  create database sybsystemprocs on device_name
  = N
  ```

  其中 *N* 是设备的总大小。您可能会发现，您需要在多个设备上创建新数据库才能获得所需的大小。

- 如果显示的使用内容包含任何日志或数据条目，请使用 create database 和 alter database 重新创建同一布局。可以对一个设备上的连续数据或日志部分进行分组，但应避免日志与数据混合。重新创建第一组数据和日志部分：

  ```
  create database sybsystemprocs
  on device_1 = M
  log on device_2 = N
  ```

  其中，*M* 是第一组数据大小的总和，*N* 是第一组日志大小的总和。对于后续的每个组，请使用 alter database（而不是 create database）重复此操作，以扩大该数据库。

4 运行 *installmaster* 脚本以创建 Sybase 提供的系统过程。

在 UNIX 平台上：

```
isql -Usa -Ppassword -Sserver_name -i
$SYBASE_/$SYBASE_ASE/scripts/installmaster
```

在 Windows 上（从 *%SYBASE%\%SYBASE_ASE%\scripts* 目录中）：

```
isql -Usa -Ppassword -S<server_name> -i instmstr
```

5 如果您的节点添加了任何过程或在 sybsystemprocs 中进行了其它更改，则您必须在新的 sybsystemprocs 数据库中进行相同的更改。

使用 *load database* 恢复 *sybsystemprocs*

如果编写系统过程并将它们存储在 sybsystemprocs 中，则当数据库损坏时，有两种恢复方法：

- 如第 389 页的“使用 installmaster 恢复 sybsystemprocs”第 4 步所述，从 installmaster 恢复数据库。然后，通过重新发出 create procedure 命令来重新创建这些过程。
- 保留数据库的备份，并使用 load database 装载备份。

如果您选择保留数据库备份，请确保能够在一个磁带卷上放下完整的副本或者多个 Adaptive Server 可以与您的 Backup Server 进行通信。如果转储跨多个磁卷，请使用 sp_volchanged（它存储在 sybsystemprocs 中）发出更改卷的命令。在恢复数据库期间不能发出此命令。

- 例如，在 UNIX 平台上，使用：

```
load database sybsystemprocs from "/dev/nrmt4"
```

- 在 Windows 平台上，使用：

```
load database sybsystemprocs from "\\.\TAPE0"
```

如何减小 *tempdb* 的大小

tempdb（临时）数据库可用于存储临时表，还可用于满足其它临时工作存储需求。如果 tempdb 的某些部分位于已损坏的磁盘上，应该首先将 tempdb 减小到缺省大小，然后再将它扩展到任何新设备上。

本节讲述如何将 tempdb 减小到缺省大小（主设备上的 2MB 数据和日志）。

将 *tempdb* 重新设置为缺省大小

开始之前，首先在单用户模式下启动 Adaptive Server，以防止其他用户在您手动更新 sysusages 时更改数据库。

1　在主设备上将 tempdb 增加到 4MB。

2　以系统管理员的身份登录到 Adaptive Server：

3 转储 master 数据库，以便在出错时可从备份恢复：

```
dump database master to "dump_device"
```

其中 *dump_device* 是目标转储设备的名称。

4 根据需要使用 bcp..out 命令将以下主要系统表保存到数据文件中，以帮助 master 数据库恢复：

- master..sysusages
- master..sysdevices
- master..sysdatabases
- master..syslogins
- master..sysconfigures
- master..syscharsets
- master..sysloginroles
- master..sysservers
- master..sysremotelogins
- master..sysresourcelimits
- master..systimeranges

警告！ 该过程仅适用于 tempdb。这是因为每次系统关闭并重新启动，都会重建 tempdb。对其它任何数据库使用该过程将导致数据库损坏。

5 在 master 数据库中，重新配置 Adaptive Server 以允许更改系统目录：

```
sp_configure "allow updates", 1
```

6 显示 sysusages 中目前属于 tempdb 的行，并记下受影响的行数：

```
begin transaction
select * from sysusages
where dbid = db_id('tempdb')
```

db_id 函数返回数据库的 ID 号。在本例中，返回 tempdb 的数据库 ID。

7　将 tempdb 的前 2MB 重新设置为数据和日志，以防止它们分开：

```
update sysusages
set segmap = 7 where dbid = db_id('tempdb')
and lstart = 0
```

8　删除 sysusages 中属于 tempdb 的其它所有行。受影响的行数应该比前面的 select 命令影响的行数少 1。

```
delete sysusages where dbid = db_id('tempdb')
and lstart != 0
```

警告！每次 Adaptive Server 关闭并重新启动，都会将 model 数据库复制到 tempdb。因此，如果 model 数据库大于其缺省大小，请不要减小 tempdb 的大小，否则可能会导致它小于 model。

9　检验 tempdb 是否有类似如下的一个条目：

```
select * from sysusages
where dbid = db_id('tempdb')
```

10　如果信息正确，请转到步骤 10 以提交事务。

如果遇到问题，请通过输入以下命令来取消更改：

```
rollback transaction
```

不要继续执行该过程。检查已执行的步骤，找出问题的原因。

11　完成事务：

```
commit transaction
```

12　重新配置 Adaptive Server，以禁止更改系统目录（Adaptive Server 的正常状态）：

```
sp_configure "allow updates", 0
```

13　立即发出 checkpoint 并关闭 Adaptive Server：

警告！必须首先关闭 Adaptive Server，然后才能再次更改 tempdb 的大小。如果不关闭并重新启动便继续运行，tempdb 会显示严重错误。

```
checkpoint
 go
shutdown
go
```

14　重新启动 Adaptive Server。

使用 *disk reinit* 和 *disk refit* 恢复系统表

如果要从无法反映最新的 disk init 或 create database 以及 alter database 命令的转储恢复 master 数据库，请按照本节中的过程操作，恢复 sysusages、sysdatabases 和 sysdevices 表中的正确信息。

使用 *disk reinit* 恢复 *sysdevices*

如果自上次转储后添加了任何数据库设备，即如果发出了 disk init 命令，则必须使用 disk reinit 向 sysdevices 添加每个新设备。如果保留了最初的 disk init 命令脚本，则应使用这些脚本来确定 disk reinit 的参数（包括 vstart 的初始值）。如果提供的大小过小，或者使用不同的 vstart 值，可能会损坏数据库。

如果未保存 disk init 脚本，请查看最新的 sysdevices 书面副本以确定 disk reinit 的一些正确参数。如果您在原始 disk init 命令中使用了自定义的 vstart，则您还需要知道 vstart 的原始值。

表 14-1 描述了 disk reinit 的参数及其相应的 sysdevices 数据：

表 14-1：使用 sysdevice 确定 disk reinit 参数

disk reinit **参数**	***sysdevices*** **数据**	**注释**
name	name	使用同一名称，尤其是在您拥有任何用于创建或修改数据库或者添加段的脚本时。
physname	*phyname*	完整设备路径。任何相对路径都与服务器的当前工作目录相对。
vdevno	vdevno	选择一个未使用的值。
size	(*high -low*) +1	您必须提供正确的大小信息。

您还可以通过读取 *name*、*physname* 和 *vdevno* 的错误日志来获取有关设备的信息，并使用操作系统命令确定设备的大小。

如果 sybsystemprocs 数据库存储在单独的物理设备上，若 sybsystemprocs 的 disk reinit 命令未列在 sysdevices 中，则应包括该命令。

运行 disk reinit 后，将 sysdevices 表与运行 dataserver 之前制作的副本进行比较。

disk reinit 只能从 master 数据库运行，并且只能由系统管理员运行。这一权限不能转移给其他用户。

请参见《系统管理指南，卷 1》中的第 7 章“初始化数据库设备”或者《参考手册：命令》。

使用 *disk refit* 恢复 *sysusages* 和 *sysdatabase*

如果自上次数据库转储后，添加了数据库设备或者创建或变更了数据库，则应使用 disk refit 重建 sysusages 表和 sysdatabases 表。

disk refit 只能从 master 数据库运行，并且只能由系统管理员运行。这一权限不能转移给其他用户。其语法为：

```
disk refit
```

在 disk refit 重建系统表之后，Adaptive Server 将关闭。在 disk refit 运行时或者在关闭过程中检查输出以确定是否发生了错误。

警告！ 在 disk reinit 命令中提供不准确的信息可能会在更新数据时导致永久性的损坏。运行 disk refit 后，请使用 dbcc 检查 Adaptive Server。

第 15 章

存档数据库访问

存档数据库访问通过将数据库转储（“存档”）视作传统的只读数据库，从而允许数据库管理员验证或选择性地恢复数据库转储中的数据，此类数据也称作“存档数据库”。

与传统数据库不同，存档数据库使用实际的数据库转储作为其主磁盘存储设备，利用最小的传统存储量来表示数据库转储恢复过程中产生的新页或修页。由于数据库转储已包含许多（甚至大多数）数据库页的映像，因此不必使用 Backup Server 将页从存档转换为传统的数据库存储，就可以装载存档据库。因此，装载速度明显快于传统数据库。

概述

存档数据库访问使您可以在数据库转储上直接执行多种操作。

传统数据库装载所需的存储量必须等于或大于源数据库的大小；使用 Backup Server 装载数据库转储涉及到将数据库转储中的页复制到为传统数据库留出存储中。

与此相比，您可以使用最小的传统磁盘存储量来创建存档数据库。在装载存档数据库时，Backup Server 不会复制数据库转储中驻留的页。相反，Adaptive Server 会创建一个映射来表示存档内部页的“逻辑到虚拟”映射关系。这将大大缩短查看数据库转储中的数据所需的时间，并降低装载转储的存储要求

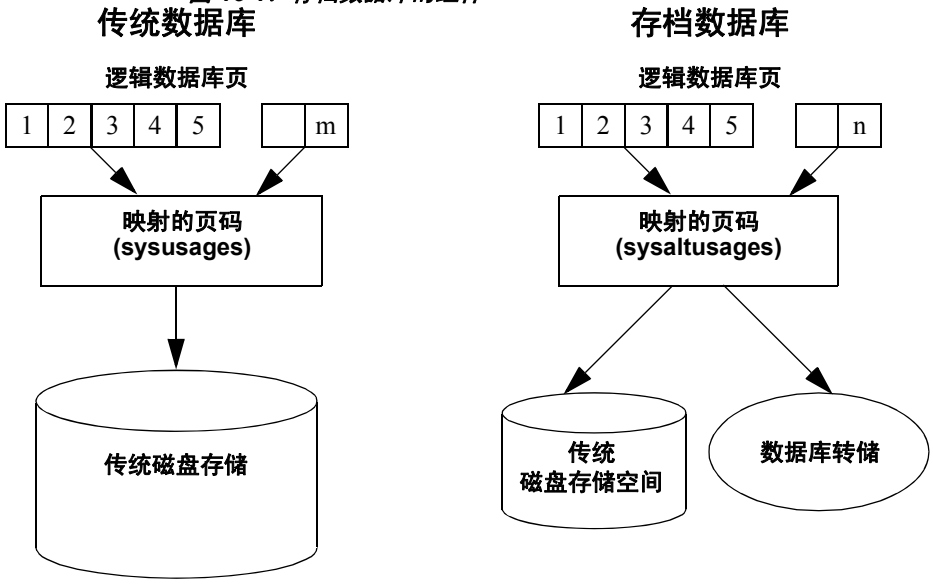
存档数据库不必是原始数据库的完全副本。根据使用 sp_dumpoptimize 转储数据库时使用的优化方式，存档数据库可能被完全填充（数据库中的每一页都存储在数据库转储中），也可能被部分填充（仅分配的页存储在数据库转储中）。

由于数据库转储表现为只读数据库，因此数据库管理员可以使用以下常见工具和技术对它进行查询：

- 对从生产数据库生成的转储的最新副本运行数据库一致性检查。可以将这些检查卸载到其它服务器以避免生产环境中的资源争用。如果资源不成问题则可以在创建存档所用的同一服务器上检查此存档。对存档进行检验能够提供执行恢复操作之前所必需的保证。
- 如果数据库转储的完整性存在问题，那么，通过将其装载到存档数据库中，可以快速测试出操作是否会成功，同时这也是一种很好的工具，可识别出复传统数据库时所应使用的正确数据库转储。
- 数据库转储的对象级别恢复。通过使用 select into 从存档数据库的表中复制要恢复的行，可以恢复丢失的数据。select into 操作可以直接在承载存档数据库的服务器中执行，也可以通过使用“组件集成服务”代理表执行（如果存档数据库与需恢复对象不在同一台服务器上）。

另外，可将事务日志装载到存档数据库中，从而确保执行恢复操作时可以应用同一装载序列。图 15-1 指出了存档数据库与传统数据库结构间的区别。

图 15-1：存档数据库的组件



存档数据库的组件

存档数据库由以下组件构成，它们一起工作时会让人误以为数据库转储的功能与传统数据库的相同：

- 数据库转储（存档）
- 用于存储修改页面区域的可选传统磁盘存储
- 承载 sysaltusages 表的空数据库

数据库转储

只读数据库转储用作大多数未修改页的存储库。

不能对数据库转储进行更改。对转储内的数据所做的任何更改都会存储在修改页面区域中。

Adaptive Server 将数据库转储及其分条看作是只能由存档数据库使用的数据库设备。

修改页面区域

数据库转储反映了数据库在给定时刻的即时快照。表示数据库转储的存档数据库是只读的。不允许执行任何用户事务，但是允许进行一些修改。例如：

- 可以运行恢复以使存档数据库与源数据库保持一致。
- 允许使用执行修复的 dbcc 命令，以便能够恢复表的修复版本。

这些修改和最新分配的数据库页不能存储在数据库转储及其分条内，因此，存档数据库需要一些传统数据库存储。此磁盘空间就称为修改页面区域，可以使用 create archive database 和 alter database 命令分配修改页面区域。

修改页面区域分为两段：

- 一次性更改段存储通过恢复撤消过程修改或分配的任意页，或在任意时间修改或分配的任意日志页。在一次性更改段，每页只有一个条目。
- 永久性更改段存储任意其它页修改或分配。在永久性更改段，每页只有一个条目。

调整修改页面区域的大小时，master 数据库中的 sysusages 行会更新。

sysaltusages 表和空数据库

sysaltusages 是一个 DOL 锁定表，用于将存档数据库中的页码映射到数据库转储及其分条或修改页面区域中的实际页。然而，与传统数据库中的 sysusages 表不同，sysaltusages 不映射数据库中的每个逻辑页，它只映射以下项：

- 已存储在数据库转储中的页
- 已修改并因此重新定位到修改页面区域中的页

请参见《参考手册：表》。

注释 由于 sysaltusages 是行锁定目录，因此您可能需要定期使用 reorg 回收在逻辑上已删除的空间。

空数据库存储 sysaltusages 表。空数据库用于为 sysaltusages 表所在的位置提供灵活性。

空数据库可以是任何数据库（也有一些数据库例外，如 master 和 temporary 数据库）。Sybase 建议您将专用数据库用作空数据库，因为：

- sysaltusages 的大小可能会根据它支持的存档数据库的数目而变化。您不能减小数据库的大小，但是若数据库太大，可以将其删除，并在需要时，重新创建一个较小的数据库。
- 它允许您打开 trunc log on checkpoint 以便您可以自动截断数据库日志。

除了承载 sysaltusages 表之外，此数据库与任何其它数据库类似。您可以使用阈值过程和其它空间管理机制来管理数据库内的空间。

若要指定用作空数据库的数据库，请输入：

```
sp_dboption <db name>, "scratch database", "true"
```

每个存档数据库一次只能指定给一个空数据库，然而多个存档数据库可以使用同一个空数据库。如果您有大量的存档数据库，您可能想要定义多个空据库。

使用存档数据库

在存档数据库上可以执行许多传统的数据库操作。然而，不允许执行会修改数据库的用户定义事务和命令（如 insert、update 和 delete）。

填充的存档数据库与只读数据库类似，且已使用 sp_dboption 应用了 readonly 选项。

对存档数据库访问的 DDLGen 支持

若要为所有存档数据库生成 DDL，请使用扩展过滤器选项“OA”。

```
ddlgen -Uroy -Proy123 -SHARBAR:1955 -TDB -N% -XOA
```

若要为单个存档数据库生成 DDL，请使用常规数据库的语法。例如，若要为存档数据库 archivedb 创建 DDL，请输入：

```
ddlgen -Uroy -Proy123 -SHARBAR:1955 -TDB -Narchivedb
```

配置存档数据库

可以使用 create archive database 创建存档数据库。请参见《参考手册：命令》。

调整修改页面区域的大小

修改页面区域用于存储已修改或新近分配的数据库页。这些页存储在永久性更改段、一次性更改段或者两个段中。

- 一页只能被重新映射到永久性更改区域一次。
- 一页只能被重新映射到一次性更改区域一次。
- 恢复操作将负责对大多数页进行重新映射。
- dbcc checkalloc 也需要分配大量空间。
- 可以使用 alter database 命令增加修改页面区域的大小。然而，若要减少修改页面区域的大小，您必须删除存档数据库并重新创建它。

永久性和一次性更改段在逻辑上不同。永久性更改段由一组 sysusages 片段定义，这些片段中包含到永久性更改段的 segmap。一次性更改段由一组 sysusages 片段定义，这些片段中包含到一次性更改段的 segmap。每个 load tran 命令开始处会放弃一次性更改段。

注释 Adaptive Server 自动管理已修改页区域空间在永久性更改段和一次性更改段间的分配。此自动调整大小完成后，master 数据库中的 sysusages 行会更新。

修改页面区域的最小尺寸取决于数据库中已修改的或新近分配的页的数量。许多这样的页都是通过重做恢复和撤消恢复被修改的。

可以使用 load database with norecovery 命令使修改页的数量最小化，从而使修改页面区域中需要的空间量最小化；然而，这样做会有一些不利的方面。

注释 dbcc checkalloc 会使用修改页面区域中的大量空间，即使您使用 nofix 选项也同样如此。当运行 dbcc checkalloc 时，每个分配页（每 256 页）上都会写入信息。这些分配页的修改存储在修改页面区域中，这意味着，当您使用 dbcc checkalloc 时，您需要大小至少为原始数据库大小的 1/256 的修改页面区域。

如果您在修改页面区域中没有足够的空间，则系统会挂起要求空间的命令，并显示一个类似于下面这样的错误：

```
There is no more space in the modified pages section for
the archive database <database name>.Use the ALTER
DATABASE command to increase the amount of space
available to the database.
```

要增加修改页面区域中的空间，可以：

- 使用 alter database 增加修改页面区域的大小，或者
- 若您不想为修改页面区域分配额外空间，可按 Ctrl+C 以中止当前命令。

注释　不能使用阈值来管理修改页面区域中的空间。

增加分配给修改页面区域的空间量

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恢复操作。

可以在任何时候（而不是仅仅在空间不足时）使用 alter database 增加修改页面区域的大小。

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。

可以使用 load database 命令来实现存档数据库。

注释　Adaptive Server 12.5.4 版和 15.0.2 版中引入了用于存档数据库访问的 load database ... norecovery。不能对传统数据库使用 norecovery。

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。

执行 sp_addumpdevice 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：

- 存档数据库的数据库转储必须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。

使存档数据库联机

可以使用 online database 使存档数据库联机。

online database 执行撤消恢复操作，在此期间，已修改和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。

将事务日志装载到存档数据库中

可以使用 load tran 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，可在装载序列的中间使存档数据库联机，而不会中断装载序列。如果装载传统数据库后使其联机，而没有使用 for standby_access 子句，则不能再装载装载序列中的下一个事务日志。但是，可以在不使用 for standby_access 子句的情况下使存档数据库联机，然后装载装载序列中的下一个事务日志。这样，在装载序列的任何时候都可以执行只读操作，如运行一致性检查。在将事务日志装载到存档数据库时，Adaptive Server 会自动从修改页面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。

删除存档数据库

可以使用 drop database 删除存档数据库。

当删除存档数据库时，将从空数据库的 sysaltusages 表中删除该数据库的所有行。这需要空数据库中的日志空间。

使用存档数据库

本节提供有关可在存档数据库上运行的命令的信息。

将 SQL 命令用于存档数据库

除了已介绍的命令（alter database、load database、online database、drop database 和 load tran）之外，还可以在存档数据库中使用以下 SQL 命令：

- use
- select
- select into — 其中目标数据库不是存档数据库。
- 执行读取的游标操作，包括：
  - declare cursor
  - deallocate cursor
  - open
  - fetch

  不能使用可更新的游标。
- checkpoint — 是受支持的命令。然而，检查点进程不会自动对存档数据库设置检查点。
- execute — 只要存档数据库内允许有任何引用存档数据库的语句，就可以使用。存储过程内部或外部的事务不允许执行 execute 命令。
- lock table
- readtext

注释 不允许执行 DML 命令（包括 insert、update 和 delete），您无法启动用户事务。

将 dbcc 命令用于存档数据库

允许在存档数据库中使用的 dbcc 命令包括：

- checkdb
- checkcatalog

  注释　不支持 checkcatalog 的 fix 版本。

- checktable
- checkindex
- checkalloc
- checkstorage
- indexalloc
- tablealloc
- textalloc

执行 dbcc 命令时，其他用户不能访问存档数据库。如果在执行 dbcc 命令过程中试图访问存档数据库，则会收到一条消息，指出数据库处于单用户模式。

对于处于联机状态或脱机状态下的存档数据库，可以使用以上 dbcc 命令的变体。然而，您仅可以对处于联机状态的存档数据库使用带有 fix 选项的 dbcc 命令。

典型的存档数据库命令序列

以下语法是典型的存档数据库命令序列。

首先，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。

将刚创建的数据库指定为空数据库：

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

创建存档数据库。

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

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

使用 load database 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"
```

使数据库联机：

```
online database archivedb
```

检查存档数据库的一致性。例如：

```
dbcc checkdb(archivedb)
```

使用 load tran 装载事务日志转储，并使用 select into 或 bcp 从存档数据库恢复对象。

```
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
```

存档数据库的压缩转储

要使用存档数据库的压缩转储，您必须：

- 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　利用“compress::”生成的转储不能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。

创建压缩内存池

当 Adaptive Server 读取压缩转储中的页时，它将选择转储中的压缩块，将其解压缩，然后提取所需的页。Adaptive Server 中的解压操作是通过使用特殊内存中的大缓冲区完成的。配置池大小使用的语法是：

sp_configure‘compression memory size’, *size*

这是一个动态配置参数，其大小是以 2KB 页为单位给出的。若大小设置为 0，则不会创建池，并且无法装载压缩转储。

若要确定池的最佳大小，请考虑以下两个因素：

- Backup Server 使用的块 I/O。缺省情况下，此块 I/O 为 64KB，但是可能已使用 dump database 命令中的 with blocksize 选项对其进行了更改。
- 在所有存档数据库内对块进行解压缩的*并发*用户的数目。每个并发用户需要两个缓冲区，并且每个缓冲区的大小应与块 I/O 的大小相同。

采用绝对最小值时，每个存档数据库允许一个并发用户（两个缓冲区）。

升级和降级存档数据库

本节介绍如何升级和降级存档数据库。

升级带有存档数据库的 Adaptive Server

不能升级存档数据库。如果将早期版本 Adaptive Server 中的数据库转储装载到位于更新版本 Adaptive Server 的存档数据库上，则在执行 online database 时，不会在内部升级该数据库。

如果升级包含存档数据库的 Adaptive Server，则会升级除该存档数据库之外的所有数据库。存档数据库保留在早期版本的 Adaptive Server 上。

Sybase 建议您用从已升级的数据库生成的转储来重新装载存档数据库。

有关如何升级 Adaptive Server 的详细信息，请参见针对所用平台的安装指南。

降级带有存档数据库的 Adaptive Server

当降级到不支持存档数据库的 Adaptive Server 版本时，请注意以下几点：

- 如果您必须将包含存档数据库的 Adaptive Server 降级到不支持存档数据库的 Adaptive Server 版本，那么 Sybase 建议您在降级之前，首先删除存档数据库。

  若要消除新的 sysaltusages 表，请在执行降级过程之前，删除空数据库。如果未删除空数据库，sysaltuages 不会引起任何问题。

- Backup Server 15.0 ESD #2 版及更高版本使用了一种新的压缩格式 (with compression = *compression_level*)，以便可以将转储装载到存档数据库中。因此，如果您必须将压缩转储装载到不支持存档数据库访问的 Adaptive Server 版本上，请使用相同版本的 Backup Server 创建并装载压缩数据库转储。早期版本的 Backup Server 不支持压缩数据库转储的新格式。

  若您不压缩就进行降级，则根本不用担心 Backup Server。

压缩转储的兼容性问题

- 不能将利用“compress::”生成的转储装载到存档数据库中。对传统数据库使用此压缩选项不存在任何转储的兼容性问题。
- 利用 with compression = *compression_level* 选项生成的压缩转储的格式在 Backup Server 15.0 ESD #2 中已更改。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　Backup Server 15.0 ESD #2 版及更高版本能够识别 15.0 ESD #2 及早期的压缩格式，因此，可以将 15.0 ESD #2 Backup Server 用于转储和装载。

存档数据库的限制

存档数据库具有以下限制：

- 存档数据库是只读的。
- 执行命令和存储过程以及访问存档数据库中对象所需要的权限与操作利用同一服务器上的相同数据库转储装载的传统数据库所需的权限一样。
- 若在装载存档数据库时使用了 with norecovery，则只有具有 sa_role 的用户或数据库所有者才能访问该数据库。
- 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。
- 如果对整个安装进行迁移，则 sybmigrate 不会迁移存档数据库。
- 只有存档数据库被明确选定要迁移时，sybmigrate 才会迁移该存档数据库。在将存档数据库迁移到目标服务器时，sybmigrate 会在目标服务器上自动创建一个传统数据库（而非存档数据库）。
- 当运行任何导致存档数据库发生更改的命令（如 dbcc 命令）时，存档数据库将自动进入单用户模式。

- 存档数据库仅使用磁盘上的数据库转储或事务日志转储；不支持磁带存储。
- 必须能够在承载存档数据库的服务器上看到数据库转储或事务日志转储。不支持远程转储。
- 如果存档数据库要访问压缩转储，则该转储必须是使用 with compression 选项而不是“compress::”选项创建的。
- 检查点进程不会自动对存档数据库设置检查点。使用 checkpoint 命令对存档数据库设置检查点。
- 不能使用 sp_dbrecovery_order 在数据库恢复序列中指定存档数据库。存档数据库将在最后依据其 *dbid* 顺序进行恢复。
- 在存档数据库中高速缓存页时，缓存的页在内存池中占据的页大小与在服务器上相同。因此，在 2K 的服务器上，页总是高速缓存在 2K 池中。在 16K 的服务器上，页总是高速缓存在 16K 的池中。
- 不能将存档数据库或该数据库内的任何对象绑定到用户定义的高速缓存。存档数据库内的对象缺省为绑定到缺省数据高速缓存。
- disk resize 对于由存档数据库使用且映射到数据库转储或事务日志的任何设备都不起作用。
- disk refit 不会从存档数据库使用的任何设备上重建 master 数据库的 sysusages 条目。这一点适用于转储设备和用于修改页面区域的设备。不过，将保留存档数据库的现有 sysusages 条目。
- 不能复制存档数据库。
- 存档数据库不支持在高可用性服务器上进行故障切换。
- 不能在存档数据库上设定可用空间阈值。

第 16 章

自动扩展数据库

您可以将数据库配置为在空间不足时自动扩展。

利用自动数据库扩展存储过程 sp_dbextend，可以安装用于识别那些具有可用空间的设备的阈值，然后在这些设备上适当地修改数据库以及在其中触发阈值的段。

将数据库设置为自动扩展之后，当数据库增大到它的可用空间阈值时，内部机制就会触发，并按扩展策略指定的空间量来增加数据库大小。自动扩展程会测量绑定到该数据库的所有设备上的剩余空间量。如果在这些设备上有足够的空间，该数据库将继续增长。缺省情况下，如果设备的大小大于 40MB，则数据库的大小将增大 10%。如果数据库的大小小于 40MB，则数据库的大小将增大 4MB。但是，您可以根据您的节点需要指定数据库大小调整限制。

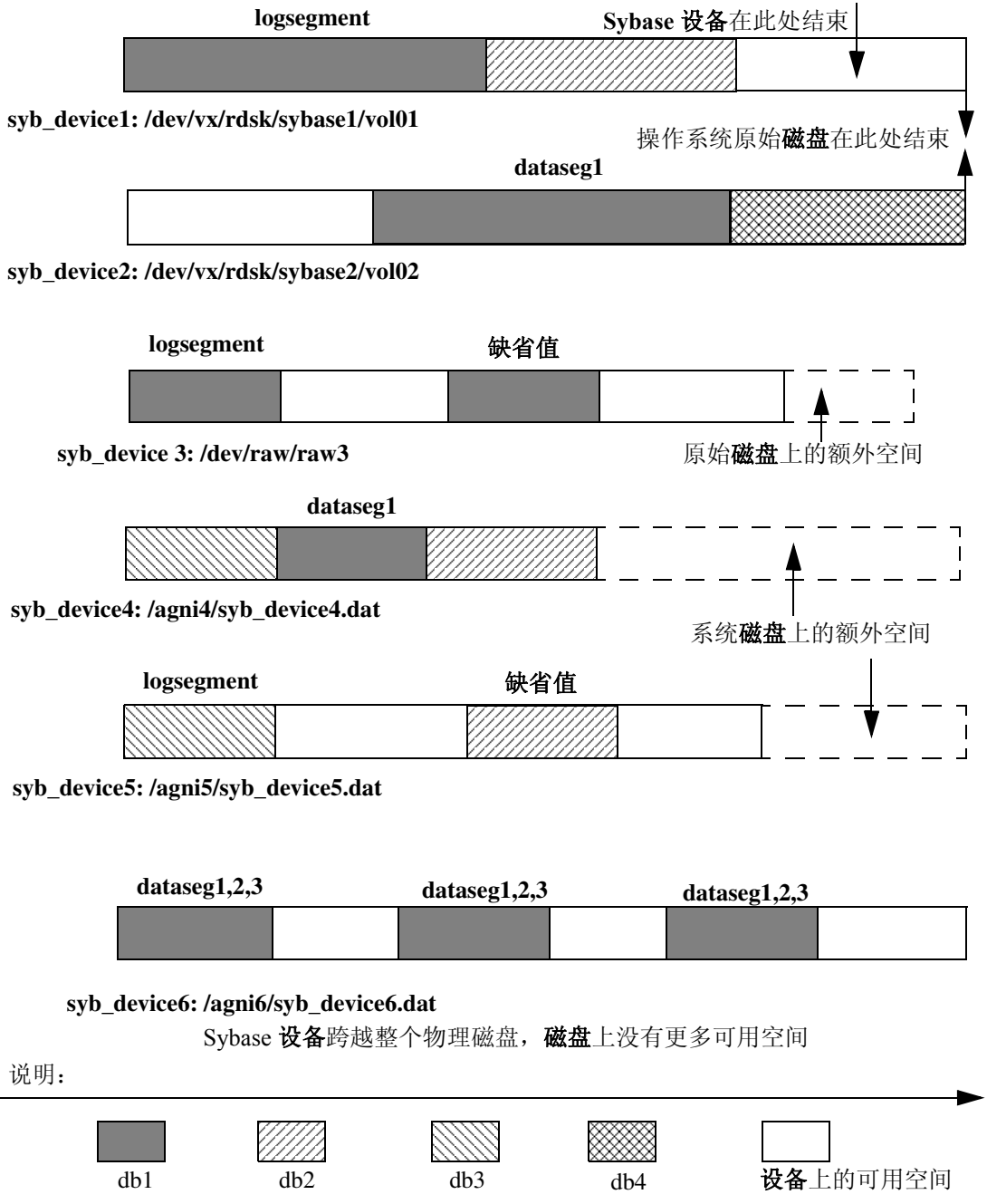
如果有任何设备被配置了扩展功能，接下来这些设备将进行扩展。最后，该数据库将在这些设备上扩展。

此自动扩展进程作为后台任务运行，并在服务器的错误日志中生成信息性消息。

了解磁盘、设备、数据库和段

第 414 页的图 16-1 显示了在执行 disk init、create database 和 alter database 这一系列操作后，在 Adaptive Server 安装中可能存在的物理资源的不同布局。您可以在测试存储过程时使用这些信息设计不同的物理和逻辑空间布局计。

图 16-1：数据库和设备布局



Sybase 设备会占用部分原始磁盘。syb_device2 表示由一个 Sybase 设备完全占用的完整原始磁盘，在该磁盘上，创建了多个数据库。在此原始磁盘 (*/dev/vx/rdsk/sybase2/vol02*) 上，在设备的头部仍有一些空闲空间；如果最初占用此空间的数据库在后来被删除了，就会出现这种情况。

syb_device4 和 syb_device5 显示了文件系统磁盘上 Sybase 设备 */agni4/syb_device4.dat* 和 */agni5/syb_device5.dat* 的布局，其中 Sybase 设备占用了一部分磁盘，但仍为设备（例如操作系统文件）留有增长的空间。

syb_device6 显示了一个完全占用物理磁盘上全部可用空间、但在设备上仍有未使用空间的 Sybase 文件系统磁盘。此空间可用于在该设备上扩展现有数据库。

这些不同的设备说明了用于不同数据库的数据库片段。用于特定数据库的每一设备都具有一个或多个跨该设备的段。

在 */agni6/syb_device6.dat* 中的 syb_device6 中，db1 跨设备的三个单独区段。该设备还属于三个不同的段：即数据段 1 、 2 和 3 。数据库 db1 的 sysusages 中的所有三个条目都与一个包含所有三个段的段映射值一起显示。

但是，在 */dev/raw/raw3* 上的设备 syb_device3 上，该数据库包含设备的两个部分，其中一个部分被以独占方式标记为日志段，而另一个部分被标记为缺省段。假设已经执行初始 create database 命令，则下列 SQL 命令可生成此结果：

```
alter database db1 on syb_device3 = "30M"
alter database db1 log on syb_device3 = "10M" with
      override
```

第一个 alter database 命令创建缺省段的数据库部分，第二个该命令创建 logsegment 的数据库部分，即使这两个区段都位于同一设备上也强制进行替换。在一个数据库上，空间是分别在各个段中进行扩展的。

阈值操作过程

数据库扩展由一组阈值操作过程执行，当可用空间超过为段设置的阈值时，会触发这些阈值操作过程。sp_dbextend 是用于管理指定段或设备的扩展进程的接口。

您可以将自动扩展配置为使用服务器范围的缺省扩展策略运行，也可以针对指定数据库中的各个段自定义自动扩展。您可以在含重要数据的表所在的键段上安装阈值，从而精确控制 Adaptive Server 如何满足不同种类的表的数据空间要求。如果您的节点包含的关键表含大量插入内容，则可以将这些表绑定到特定段，并使用特定于节点的规则来扩展该段。这样，您可以避免在此类键表有很重负载的生产环境中可能发生的损耗。

您不能使用阈值缩减数据库或它的段。

请参见《参考手册：过程》。

安装自动数据库扩展过程

使用 *installdbextend* 脚本安装自动扩展，该脚本将行装载到 master.dbo.sysattributes 中，后者描述数据库或设备中的自动扩展缺省设置。该安装脚本还在 model 和 tempdb 数据库中创建一个控制表。

如果要升级到 Adaptive Server 版本 12.5.1 或更高版本，则必须作为升级进程的一部分单独安装此脚本。

❖ **安装自动数据库扩展**

1 使用 sa_role 权限登录。在 UNIX 平台上，*installdbextend* 位于 *$SYBASE/$SYBASE_ASE/scripts* 中。如果您运行的是 Windows，则它位于 *%SYBASE%/%SYABASE_ASE%/scripts*中

2 在 UNIX 平台上，运行：

```
isql -Usa -P -Sserver_name <$SYBASE/$SYBASE_ASE/scripts/installdbextend
```

在 Windows 平台上，运行：

```
isql -Usa -P -Sserver_name
<%SYBASE%/%SYBASE_ASE%/scripts/installdbextend
```

installdbextend 脚本将阈值操作过程系列和 sp_dbextend 安装在 sybsystemprocs 数据库中。

运行 *sp_dbextend*

sp_dbextend 用于根据节点特定的规则自定义数据库和设备扩展进程。数据库管理员可以配置每个数据库和段进行配对所依据的大小或百分比，并且应该扩展设备。

您还可以通过指定最大大小（超过该大小不能进一步扩展）来限制数据库段或设备的扩展。

您可以在测试模式下使用 sp_dbexpand 来根据所选策略模拟扩展进程。

sp_dbextend 提供了一些用于列出当前设置和删除节点特定的策略规则的方法。

此信息作为新的属性定义存储在 master.dbo.sysattributes 中。

sp_dbextend 接口中的命令选项

sp_dbextend 使用以下语法：

```
sp_dbextend [ command [, arguments...]]
```

其中 command 是下面讨论的选项之一，而 arguments 指定数据库名称、段名、可用空间数量，等等。请参见《参考手册：过程》。

如果不包括参数，则 sp_dbextend 缺省为 help。请参见《参考手册：过程》。

注释　自动扩展过程不会创建新设备，只会修改段当前映射到的现有设备上的数据库和段的大小。

若要停止阈值操作过程，请使用 sp_dropthreshold 清除阈值，或将 sp_dbextend 与 *clear* 选项配合使用。请参见《参考手册：过程》。

验证当前阈值

使用 check 参数可验证各种阈值的当前设置。例如，如果多个段共享同一组设备并且两个段都被设置为自动扩展，或者，如果当前设置为触发 logsegment 上的自动扩展的阈值与用于 logsegment 的当前最后机会阈值太接近，check 将发出警告。在这种情况下，自动阈值不会触发，并且 check 会报告警告。

sp_dbextend 包括一种功能强大的模拟模式，任何具有 sa_role 权限的用户都可以使用该模式模拟顶层阈值操作过程的执行。

- 定义 pubs2 数据库中 logsegment 的扩展策略：

```
sp_dbextend 'set', 'database', pubs2, logsegment,'3M'

sp_dbextend 'set', 'threshold', pubs2, logsegment, '1M'
```

- 若要模拟这些策略的扩展，请执行：

  ```
  sp_dbextend 'simulate', pubs2, logsegment
  ------------------------------
  ```

  在此输入后将显示来自服务器的消息。

  以下示例演示了当数据库 pubs2 的段 logsegment 上的阈值触发一次时将会发生的数据库和磁盘扩展系列：

```
sp_dbextend 'simulate', pubs2, logsegment
---------------
NO REAL WORK WILL BE DONE.
Simulate database / device expansion in a dry-run mode 1 time(s).
These are the series of database/device expansions that would have
happened if the threshold on database'pubs2',
segment 'logsegment' were to fire 1 time(s).

Threshold fires:Iteration:1.
==============================

Threshold action procedure 'sp_dbxt_extend_db' fired in db 'pubs2' on
segment 'logsegment'.

Space left:512 logical pages ('1M').

ALTER DATABASE pubs2 log on pubs2_data = '3.0M'
-- Segment:logsegment

Database 'pubs2' was altered by total size '3M' for
segment 'logsegment'.
Summary of device/database sizes after 1 simulated extensions:
================================================
devicename initial size final size
---------- ------------ ----------
pubs2_data 20.0M        20.0M

(1 row affected)

Database 'pubs2', segment 'logsegment' would be altered from an initial
size of '4M' by '3M' for a resultant total size of '7M'.
```

```
To actually expand the database manually for this threshold, issue:
sp_dbextend 'execute', 'pubs2','logsegment', '1'

(return status = 0)
```

若要为此阈值手动扩展数据库，请执行：

```
sp_dbextend 'execute', 'pubs2', 'logsegment'
--------------
```

此示例说明，如果阈值在此级别触发，则 alter database 命令会作用于 logsegment 的 pubs2_data 设备：

```
sp_dbextend 'execute', pubs2, logsegment

Threshold fires:Iteration:1.
================================
Threshold action procedure 'sp_dbxt_extend_db' fired in db 'pubs2' on
segment 'logsegment'.Space left:512 logical pages ('1M').

ALTER DATABASE pubs2 log on pubs2_data = '3.0M' -- Segment:logsegment
Extending database by 1536 pages (3.0 megabytes) on disk pubs2_data
Warning:The database 'pubs2' is using an unsafe virtual device
'pubs2_data'.The recovery of this database can not be guaranteed.

Warning:Using ALTER DATABASE to extend the log segment will cause user
thresholds on the log segment within 128 pages of the last chance
threshold to be disabled.

Database 'pubs2' was altered by total size '3M' for segment 'logsegment'.

(return status = 0)
```

- 若要模拟当针对特定段连续 <*n*> 次触发阈值时实际发生的情况，请发出同一命令，并指定迭代次数：

  ```
  sp_dbextend 'simulate', pubs2, logsegment, 5
  -----------------
  ```

  若要将该数据库扩展五次，请输入：

  ```
  sp_dbextend 'execute', 'pubs2', 'logsegment', 5
  ------------------
  ```

  如果执行此命令，您将会看到，连续五次触发阈值的输出将数据库置于一系列 alter database 操作之中，之后是一个或多个 disk resize 操作，最后是针对指定设备的 alter database。

对数据库进行自动扩展设置

执行此过程，在数据库中设置不同的段以执行自动扩展。本节使用 pubs2 数据库。

并非下述所有步骤都必须执行。例如，您可能不会选择为各个设备设置 *growby* 或 *maxsize* 并且会选择仅为这些设备使用系统缺省策略。

❖ **设置数据库**

1 创建数据库：

```
create database pubs2 on pubs2_data = "10m" log on pubs2_log = "5m"
```

2 为 *pubs2_data* 设备将 *growby* 和 maxsize 策略分别设置为 10MB 和 512MB。您可以在任何数据库中设置这些策略。输入：

```
exec sp_dbextend 'set', 'device', pubs2_data, '10m', '512m'
```

3 设备的系统缺省 *growby* 策略是 10%。您可以修改此系统缺省值，选择适当的 growby 值，而不为 *pubs2_log* 设备设置新策略。pubs2_log 然后以此比率扩展。输入：

```
exec sp_dbextend 'modify', 'device', 'default', 'growby', '3m'
```

4 设置缺省段的 *growby* 速率，但不要指定最大大小。输入：

```
exec sp_dbextend 'set', 'database', pubs2, 'default', '5m'
```

缺省段上的 *growby* 比率可能不同于该段驻留的设备上的增长率。*growby* 控制该段在用尽可用空间后的扩展率，并且只在扩展该段时使用。

5 为 *logsegment* 设置 *growby* 和 maxsize 变量：

```
exec sp_dbextend 'set', 'database', pubs2, 'logsegment', '4m', '100m'
```

6 检查为 pubs2 数据库中不同的段设立的策略：

```
exec sp_dbextend 'list', 'database', pubs2
```

7 检查 pubs2 所跨的各种设备中的策略。用于 *devicename* 的模式指示符（“%”）选用全部这些设备：

```
exec sp_dbextend 'list', 'device', "pubs2%"
```

8　为 pubs2 中的缺省段以及 logsegments 段安装扩展阈值。这将设置并启用扩展进程，并且允许您选择据此触发扩展进程的可用空间阈值。输入：

```
use pubs2
-----------------------------------------------------------------

    exec sp_dbextend 'set', 'threshold', pubs2, 'default',   '4m'
    exec sp_dbextend   'set', 'threshold',  pubs2,  'logsegment', '3m'
```

9　检查由上述命令安装的阈值。

```
exec sp_dbextend list, 'threshold'

segment name free pages  free pages (KB) threshold procedure status
------------ ----------- --------------- ------------------- -----------
default          2048    4096              sp_dbxt_extend_db enabled
logsegment        160  320           sp_thresholdaction lastchance
logsegment       1536  3072          sp_dbxt_extend_db      enabled
Log segment free space currently is 2548 logical pages (5096K).
(1 row affected, return status = 0)
```

在此输出中，sp_dbxt_extend_db 是推动在运行期执行扩展进程的阈值过程。扩展阈值当前在缺省段和 logsegment 段上均处于启用状态。

10　使用 simulate 查看扩展：

```
exec sp_dbextend 'simulate', pubs2, logsegment
exec sp_dbextend 'simulate', pubs2, 'default', '5'
```

11　根据需要使用 modify 更改策略：

```
exec sp_dbextend 'modify', 'database', pubs2, logsegment, 'growby','10m'
```

12　若要暂时对特定段禁用扩展，请使用 disable：

```
 exec sp_dbextend 'disable', 'database', pubs2, logsegment
```

13　检查数据库和设备的扩展策略的状态：

```
exec sp_dbextend list, 'database'
name         segment    item    value status
------------ ---------- ------- ----- --------
server-wide (n/a)       (n/a)   (n/a) enabled
default     (all)       growby  10%   enabled
pubs2       default     growby  5m    enabled
pubs2       logsegment growby  10m   disabled
pubs2       logsegment maxsize 100m  disabled
(1 row affected, return status = 0)
```

禁用状态表示当前针对 pubs2 中的 logsegment 禁用扩展进程。

```
exec sp_dbextend list, 'device'
name           segment item   value status
-------------- ------- ------ ----- -------
server-wide    (n/a)   (n/a)  (n/a) enabled
default        (n/a)   growby 3m    enabled
mypubs2_data_0 (n/a)   growby 10m   enabled
mypubs2_data_1 (n/a)   growby 100m  enabled
mypubs2_log_1  (n/a)   growby 20m   enabled
mypubs2_log_2  (n/a)   growby 30m   enabled
(1 row affected, return status = 0)
```

14 使用 enable 可以重新启用扩展进程：

```
exec sp_dbextend 'enable', 'database', pubs2, logsegment
```

约束和限制

设置阈值时具有下面一些约束和限制：

- 在一个或多个数据库中的多个段上安装阈值过程时，将按阈值触发的顺序执行扩展。如果 abort tran on log full 对于 logsegment 是禁用的，则任务将一直等待，直到 logsegment 的阈值过程按照计划改变了数据库。
- 在非日志段中，即使在超过可用空间阈值后任务也继续处理，而阈值过程保留在队列中。这可能导致在数据段中出现“空间不足”错误。请设计您的值，以便在数据库中有足够的空间来完成任务。
- 如果许多阈值过程同时触发，则过程高速缓存会变得超载。在安装了大量数据库、许多段和许多阈值操作过程的安装中更容易发生上述情况。
- 如果 tempdb 中的空间严重不足，并且其它操作需要 tempdb 资源，则甚至在尝试纠正这一情况时阈值过程也可能失败。应确保使用足够大的可用空间量（至少为 2MB,）安装 tempdb 中的阈值过程，以避免发生此问题。

您可能需要更改转储和装载过程以管理用于确定数据库和设备如何扩展的节点特定策略。

转储数据库不会传输存储在 master.dbo.sysattributes 中的信息，因此，如果您使用转储和装载将数据库从源服务器迁移到目标服务器，则必须手动迁移在 sysattributes 数据库中编码为数据的全部节点特定策略。有以下两种可能的解决方法：

- 通过从 master.dbo.sysattributes 上为类编号是 19 的条目定义的视图使用 bcp out，您可以手工从 master.dbo.sysattributes 提取数据，然后使用 bcp in 将这些数据装载到目标服务器中。这要求跨两台服务器的两个数据库具有相同的段 ID。
- 您也可以使用 Sybase Central 的 ddlgen 功能来重新生成重新创建策略规则所需的 sp_dbextend set 调用，具体方法是在目标服务器上运行 ddlgen。但是，不能使用 ddlgen 过程管理跨服务器的重命名逻辑设备。您必须手动在目标服务器中重命名设备。

这些限制不会导致失败：

- 您可以在数据库启用了 sp_dboption 'no free space acctg' 时在非日志段上安装阈值操作（请参见《参考手册：过程》）。该选项只表示不执行任何数据库扩展，因为在禁用此选项后不触发阈值操作。将此选项保留为启用状态将生成一个警告消息。
- Sybase 还建议您在发生扩展时定期转储 master 数据库，以便可以在若干次扩展之后发生故障时重新创建 master 数据库。
- Sybase 建议您不要在任何系统数据库上安装这些通用阈值过程，尤其是不要在 master 数据库上安装，因为修改 master 数据库的空间使用率要求进行一些特殊处理（请参见《参考手册：命令》）。
- 不能使用阈值缩减数据库或段。

第 17 章

使用阈值管理可用空间

当创建或变更数据库时，可为数据库的数据和日志段分配有限的空间量。当创建对象和插入数据时，数据库中可用空间量将减少。

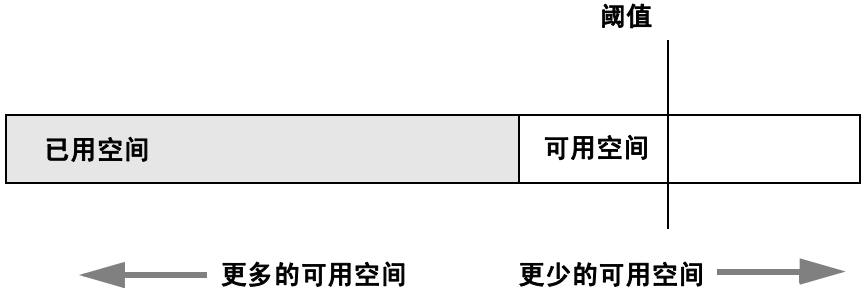
使用最后机会阈值监控可用空间

包括 master 在内的所有数据库都有一个**最后机会阈值**。该阈值是对备份事务日志所需的可用日志页数的估计值。为日志段分配更多空间时，Adaptive Server 会自动调整最后机会阈值。

当日志段中的可用空间量低于最后机会阈值时，Adaptive Server 将自动执行名为 sp_thresholdaction 的特殊存储过程。（可以使用 sp_modifythreshold 指定一个不同的最后机会阈值过程。）

图 17-1 显示一个具有最后机会阈值的日志段。阴影区域代表已用日志空间；无阴影区域代表可用日志空间。此时，日志段尚未达到最后机会阈值。

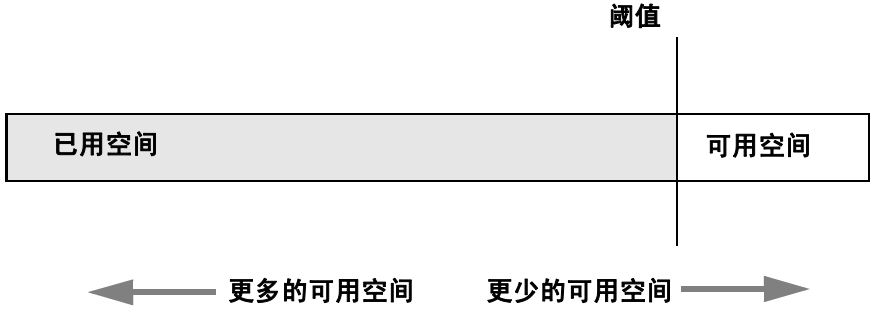
图 17-1：具有最后机会阈值的日志段



达到阈值

随着用户执行事务，可用日志空间量逐渐减少。当可用空间量达到最后机会阈值时，Adaptive Server 将执行 sp_thresholdaction：

图 17-2：达到最后机会阈值时执行 sp_thresholdaction



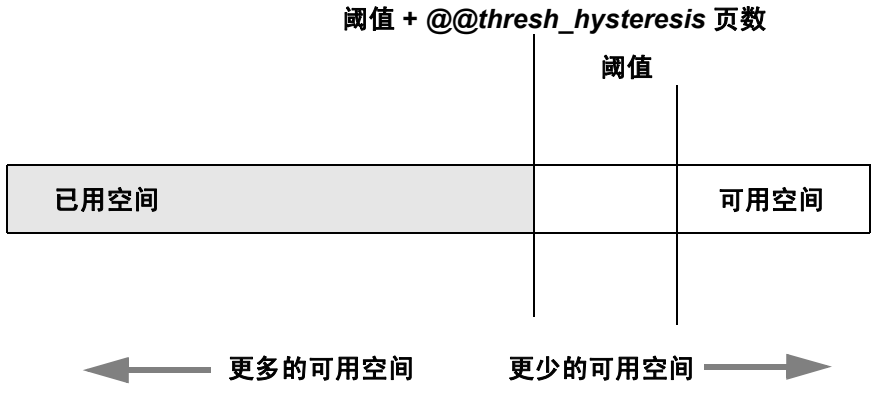
控制执行 *sp_thresholdaction* 的频率

Adaptive Server 使用一个停滞值，即全局变量 *@@thresh_hysteresis*，来控制阈值对可用空间变化的敏感程度。

一个阈值在执行其对应的阈值过程之后即会失效，在段中的可用空间量超过阈值的页数达到 *@@thresh_hysteresis* 之前，该阈值保持不活动状态。这样可防止阈值为响应可用空间的微小变化而重复执行对应的阈值过程。不能更改 *@@thresh_hysteresis* 的值。

例如，当图 17-2 中的阈值执行 sp_thresholdaction 后，它将失效。在图 17-3 中，当可用空间量按 *@@thresh_hysteresis* 设置的值增大后，将重新激活阈值：

图 17-3：可用空间的增加幅度必须是 @@thresh_hysteresis 才能重新激活阈值



回退记录和最后机会阈值

Adaptive Server 11.9 和更高版本的事务日志中包括**回退记录**。每当回退某个事务时，将记录下回退记录。服务器会保留足够的空间来为一个打开的事务的每一个更新记录下回退记录。如果事务成功完成，则不会记录回退记录，同时会释放为它们所保留的空间。

在长时间运行的事务中，回退记录可保留大量空间。

若要检查 syslogs 使用的空间，请运行 sp_spaceused：

```
sp_spaceused syslogs
```

输出为：

```
name            total_pages     free_pages      used_pages      reserved_pages
--------------- --------------- --------------- --------------- --------------
syslogs                5632            1179            3783            670
```

dbcc checktable(syslogs) 生成类似的输出：

```
Checking syslogs:Logical pagesize is 2048 bytes
The total number of data pages in this table is 3761.
*** NOTICE:Space used on the log segment is 3783 pages (7.39 Mbytes), 67.17%.
*** NOTICE:Space reserved on the log segment is 670 pages (1.31 Mbytes), 11.90%.
*** NOTICE:Space free on the log segment is 1179 pages (2.30 Mbytes), 20.93%.
```

如果在似乎有足够空间时引发了事务日志的最后机会阈值，则可能是为回退保留的空间导致了此问题。请参见第 428 页的“确定回退记录的当前空间”。

计算回退记录的空间

若要计算为包含回退记录而应使事务日志增加的空间量，应估计：

- 事务日志中很可能属于已回退事务的更新记录的数目。
- 在任何时刻，事务日志中可能属于打开的事务的更新记录的最大数量

更新记录更改时间戳值，并且包含对数据页、索引页、分配页等的更改。

每个回退记录都要求大约 60 字节的空间，即一页的 3%。因此，计算事务记录中包含的回退记录 (RR) 的公式为：

增加的空间（页数）=（已记录的 RR 数 + 打开的更新数）X 3/100

可能还需要增加日志空间以补偿回退记录对最后机会阈值和用户定义的阈值的影响，如以下各节所述。

使用 lct_admin 确定可用日志空间

使用 logsegment_freepages 可以确定专用日志段所拥有的可用空间量。

若要查看 pubs2 数据库日志段的可用页数，请输入：

```
select lct_admin("logsegment_freepages", 4)
---------------------
                    79
```

确定回退记录的当前空间

若要确定数据库当前为回退保留的页数，请发出带 reserved_for_rollbacks 参数的 lct_admin。

返回的页数是为回退记录保留的尚未分配的数量。

例如，若要确定为 pubs2 数据库（其 dbid 为 5）中的回退保留的页数，请发出：

```
select lct_admin("reserved_for_rollbacks", 5)
```

请参见《参考手册：命令》。

回退记录对最后机会阈值的影响

使用回退记录的 Adaptive Server 必须为最后机会阈值保留额外空间。由于已记录的回退记录占用空间，并存在为打开的事务可能产生的回退记录保留的空间，因此可能较快达到最后机会阈值。

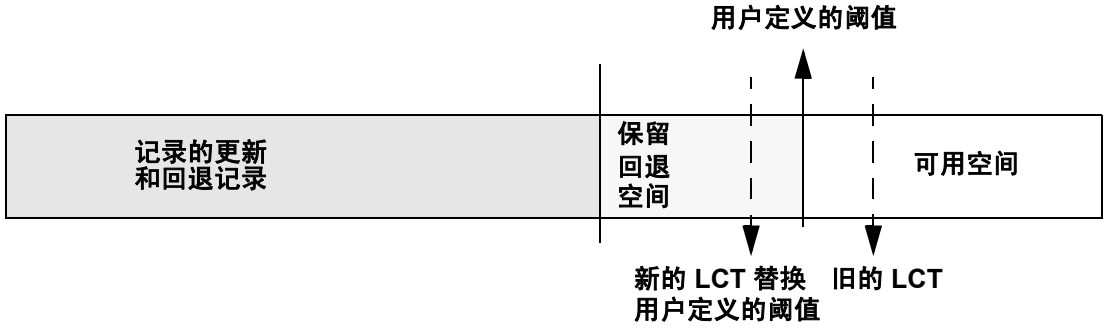
用户定义的阈值

因为回退记录占据事务日志中的额外空间，所以在用户定义的阈值完成转储后，可用空间少于不使用回退记录的 Adaptive Server 版本中的可用空间。然而，由于最后机会阈值增加而导致的用于转储的空间减少量可能多于由保留给打开的事务的回退记录的空间补偿的空间。

可以使用用户定义的阈值启动 dump transaction。设置此阈值，以便在达到最后机会阈值之前和日志中所有打开的事务挂起之前，有足够的空间完成转储。

在使用混合日志和数据的数据库中，最后机会阈值动态变化，并可以将最后机会阈值的值自动配置为小于用户定义的阈值。如果进行了自动配置，则会禁用用户定义的阈值，并且在达到用户定义的阈值之前将引发最后机会阈值，如图 17-4 中所示：

图 17-4：在达到用户定义的阈值之前引发 LCT



如果为最后机会阈值设置的值大于用户定义的阈值（例如，如果将最后机会阈值重新配置为图 17-4 中“旧 LCT”的值），则会重新启用用户定义的阈值。

在具有单独日志段的数据库中，日志有专用的空间量，并且最后机会阈值是静态的。用户定义的阈值不受最后机会阈值的影响。

共享的日志段和数据段的最后机会阈值和用户日志高速缓存

Adaptive Server 中的每个数据库都有最后机会阈值，且所有数据库都允许在用户日志高速缓存中缓冲事务。创建具有共享日志段和数据段的数据库时，将根据 model 数据库的大小确定该数据库的最后机会阈值。一旦增加数据并开始记录活动，就会立即根据可用空间和当前打开的事务动态地重新计算最后机会阈值。具有独立日志段和数据段的数据库的最后机会阈值是根据日志段大小确定的，它不会动态变化。

若要获取任何数据库的当前最后机会阈值，请使用 lct_admin 的 reserve 参数，并指定 0 个日志页：

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

数据库的最后机会阈值将存储在 systhresholds 表中，而且还可以通过 sp_helpthreshold 进行访问。然而：

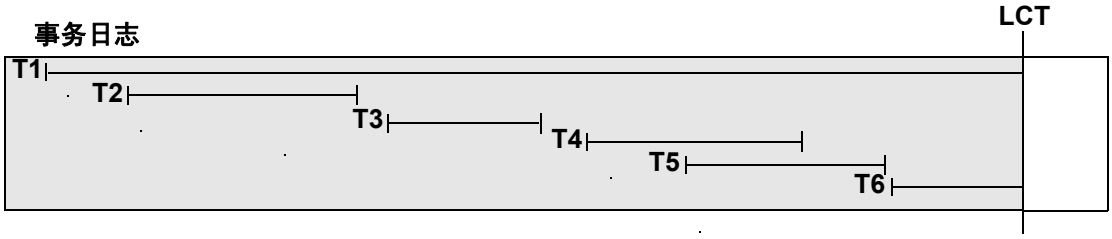
- sp_helpthreshold 返回用户定义阈值和其它数据，以及最后机会阈值的最新值。如果只需得到当前的最后机会阈值，使用 lct_admin 更简便。两个值中的任何一个都可产生最后机会阈值的最新值。
- 对于带共享日志段和数据段的数据库，systhresholds 中的最后机会阈值可能不是当前的最后机会阈值。

使用 *lct_admin abort* 中止挂起的事务

事务日志达到最后机会阈值时，所有打开的事务都将被挂起。一般通过转储事务日志创建空间，因为这将从日志的开始之处移去已提交的事务。然而，如果日志开始处有一个或多个事务仍为打开状态，则会阻止事务日志的转储。

使用 lct_admin abort 终止阻止事务日志转储的挂起的事务。因为终止事务的操作可将该事务关闭，所以转储可以继续进行。图 17-5 显示了使用 lct_admin abort 的一种可能情况：事务日志已经达到其 LCT，且打开的事务 T1 和 T6 被挂起。因为 T1 位于日志的开始处，所以它使得转储不能删除已关闭的事务 T2 到 T5，从而不能为后面的日志记录创建空间。使用 lct_admin abort 终止 T1 将允许您关闭 T1，以便转储可以从日志中清除事务 T1 到 T5。

图 17-5：何时使用 lct_admin abort



❖ **中止挂起的事务**

必须首先确定事务的 ID 才能中止事务。

1　使用以下查询查找已经达到最后机会阈值的事务日志中最早打开的事务的 spid：

```
use master
 go
select dbid, spid from syslogshold
where dbid = db_id("name_of_database")
```

例如，若要查找在 pubs2 数据库上最早运行的事务：

```
select dbid, spid from syslogshold
where dbid = db_id ("pubs2")
dbid   spid
------ ------
     7      1
```

2　若要终止最早的事务，请输入启动此事务的进程的进程 ID (spid)，这还将终止日志中属于该指定进程的任何其它挂起的事务。

例如，如果进程 83 存有挂起日志中最早打开的事务，若要终止此事务，请输入：

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

这也将终止同一事务日志中属于进程 83 的任何其它打开的事务。

若要终止日志中所有打开的事务，请输入：

```
select lct_admin("abort", 0, 12)
```

请参见《参考手册：命令》。

为 master 数据库的事务日志增加空间

达到 master 数据库中的最后机会阈值后，可以使用 alter database 为 master 数据库的事务日志增加空间。通过激活在日志中挂起的事务，可允许在服务器中进行更多的活动。然而，当 master 事务日志达到最后机会阈值时，不能在其它数据库中使用 alter database 进行更改。所以，如果 master 数据库和另一个数据库都达到了各自的最后机会阈值，则必须先使用 alter database 为 master 数据库增加日志空间，然后再次使用该命令为另一个数据库增加日志空间。

自动中止或挂起进程

按照设计，最后机会阈值将留出足够的可用日志空间来记录 dump transaction 命令。可能没有足够的空间记录数据库的其它用户事务。

达到最后机会阈值时，Adaptive Server 将挂起用户进程并显示以下消息：

```
Space available in the log segment has fallen critically
low in database 'mydb'. All future modifications to this
database will be suspended until the log is successfully
dumped and space becomes available.
```

现在仅可以执行事务日志中未记录的命令（select 或 readtext）和释放额外日志空间可能需要的命令（dump transaction、dump database 和 alter database）。

使用 *abort tran on log full* 中止事务

若要将最后机会阈值配置为自动中止打开的事务而不是将其挂起，请输入：

```
sp_dboption database_name "abort tran on log full", true
```

如果从 Adaptive Server 的以前版本升级，则新升级的服务器将保留 abort tran on log full 设置。

唤醒挂起的进程

在 dump transaction 释放足够的日志空间后，自动唤醒并完成挂起的进程。如果 writetext 或 select into 导致未记录上次备份以来对数据库的更改，可以运行 dump tran with truncate_only，该命令甚至可以在存在未记录的写入操作的情况下运行。如果日志空间严重不足，以至于 dump tran with truncate_only 失败，则可以运行 dump tran with no_log。但是，dump tran with no_log 仅用于紧急状态，并且仅在不得以的情况下才运行。

在事务转储完成且事务已成功从日志挂起状态中释放后，系统管理员和数据库所有者可以转储数据库。

如果这没有释放足够的空间以唤醒挂起的进程，则可能需要使用 alter database 的 log on 选项增大事务日志的大小。

除了注销命令外，还可以使用 lct_admin("abort", *spid*)，这种方法可能比注销连接更可取，因为您可能希望保持连接。请参见《参考手册：构件块》。

如果有系统管理员权限，则可以使用 sp_who 确定哪些进程处于日志挂起状态，然后使用 kill 命令唤醒休眠的进程。

添加、更改和删除阈值

数据库所有者或系统管理员可以创建额外的阈值来监控数据库中任何段上的可用空间。这些额外的阈值称为**可用空间阈值**。每个数据库最多可以有 256 个阈值，其中包括最后机会阈值。

sp_addthreshold、sp_modifythreshold 和 sp_dropthreshold 用于创建、更改和删除阈值。所有这些过程都需要指定当前数据库的名称。

显示现有阈值的有关信息

使用 sp_helpthreshold 获取数据库中所有阈值的有关信息。使用 sp_helpthreshold *segment_name* 可以获取特定段上阈值的有关信息。

以下示例显示数据库 default 段上阈值的相关信息。因为“default”是保留字，所以必须用引号将其引起。sp_helpthreshold 的输出表明：在该段的 200 页上设置有一个阈值。“last chance”列中的 0 表示这是可用空间阈值而不是最后机会阈值：

```
sp_helpthreshold "default"

segment name   free pages   last chance?   threshold procedure
------------   ----------   ------------   ------------------
default        200                     0   space_dataseg

(1 row affected, return status = 0)
```

阈值和系统表

系统表 systhresholds 含有阈值的有关信息；sp_helpthreshold 利用此表。除了段名、可用页、最后机会状态和阈值过程名的信息之外，此表还记录创建阈值的用户的服务器用户 ID 以及创建阈值时用户具有的角色。

Adaptive Server 从 curunreservedpgs 获取有关段中剩余的可用空间量的信息以及是否激活阈值的信息。

添加可用空间阈值

可以使用 sp_addthreshold 创建可用空间阈值。请参见《参考手册：过程》。

当段上的可用空间量低于阈值时，一个内部 Adaptive Server 进程将执行相关联的过程。此进程具有的权限为：创建阈值的用户执行 sp_addthreshold 时拥有的权限减去创建阈值后已撤消的任何权限。

阈值可以在同一个数据库、另一个用户数据库、sybsystemprocs 或 master 中执行一个过程。阈值也可以调用 Open Server 上的远程过程。创建阈值时， sp_addthreshold 不检验阈值过程是否存在。

更改或指定新的可用空间阈值

使用 sp_modifythreshold 可以将可用空间阈值与新阈值过程、可用空间值或段关联起来，或者更改与最后机会阈值相关联的过程的名称。sp_modifythreshold 删除现有阈值并创建替代此值的新阈值。请参见《参考手册：过程》。

例如，若要在段上的可用空间下降到 175 页（而不是 200 页）以下时执行阈值过程，请输入：

```
sp_modifythreshold mydb, "default", 200, NULL, 175
```

在此示例中，NULL 充当占位符，这样 *new_free_space* 正好位于参数列表中的正确位置。阈值过程的名称不变。

sp_modifythreshold 要求指定与最后机会阈值关联的可用页数。可以使用 sp_helpthreshold 确定此值。

修改阈值者成为新阈值的所有者。当段上的可用空间量降到低于阈值时，Adaptive Server 将使用以下权限执行阈值过程：所有者执行 sp_modifythreshold 时的权限减去运行该命令后已撤消的权限。

以下示例显示有关最后机会阈值的信息，然后指定当达到阈值时执行新的过程 sp_new_thresh_proc：

```
sp_helpthreshold logsegment
```

```
segment name    free pages    last chance?    threshold procedure
------------    ----------    ------------    -------------------
logsegment      40                        1    sp_thresholdaction

(1 row affected, return status = 0)
```

```
sp_modifythreshold mydb, logsegment, 40,
sp_new_thresh_proc
```

删除阈值

可以使用 sp_dropthreshold 从段中删除可用空间阈值。请参见《参考手册：过程》。

dbname 必须指定当前数据库的名称。必须同时指定段名和可用页数，因为一个特定段上可以有多个阈值。例如：

```
sp_dropthreshold mydb, "default", 200
```

为日志段创建可用空间阈值

如果达到最后机会阈值，在释放足够的日志空间之前将中止或挂起所有事务。在生产环境中，这可能对用户造成重大影响。在日志段上增加正确放置的可用空间阈值，可使达到最后机会阈值的可能性最小。

其它阈值应当经常转储事务日志，以保证极少达到最后机会阈值。转储事务日志的频率不应该高到需要装载太多的磁带才能恢复数据库。

本节帮助您确定第二个日志阈值的最佳位置。该方法的起始步骤是添加一个阈值，其 *free_space* 值设置为日志大小的 45%，然后根据自己节点的空间使用情况调整此阈值。

❖ **以日志大小的 45% 添加日志阈值**

使用以下过程添加日志阈值，该阈值的 *free_space* 值设置在日志大小的 45% 处。

1 确定日志大小（以页为单位）：

```
select sum(size)
from master..sysusages
where dbid = db_id("database_name")
and (segmap & 4) = 4
```

2 使用 sp_addthreshold 添加一个新阈值，其 *free_space* 值设置为 45%。例如，如果日志的容量为 2048 页，则添加一个阈值，其 *free_space* 值为 922 页：

```
sp_addthreshold mydb, logsegment, 922, thresh_proc
```

3 创建将事务日志转储到相应设备的简单阈值过程。请参见第 440 页的“创建阈值过程”。

测试和调整新阈值

使用 dump transaction 确保事务日志大小低于总空间的 55%。然后测试新阈值：

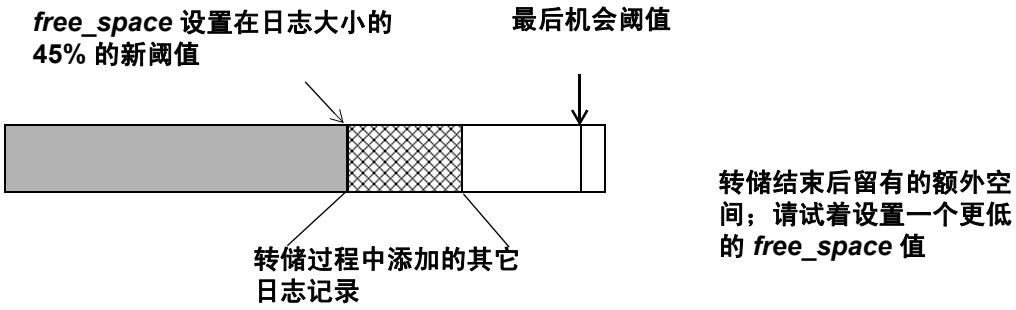
1 通过模拟例程操作填写事务日志。使用以设定的速率执行典型事务的自动脚本。

达到可用空间阈值 45% 时，阈值过程将转储事务日志。因为不是最后机会阈值，所以不会挂起或中止事务；在转储过程中日志将继续增大。

2 在转储进行过程中，使用 sp_helpsegment 监控日志段上的空间使用情况。在转储完成前一刻记录事务日志的最大大小。

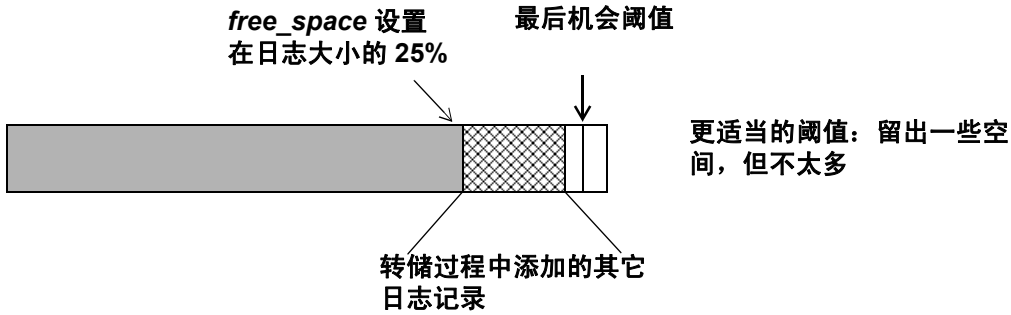
3　如果当转储完成后日志中留有大量空间，可能不需要这么早地转储事务日志，如图 17-6 中所示：

图 17-6：具有 45% 附加阈值的事务日志



尝试等到只剩余日志空间的 25% 时再转储，如图 17-7 中所示：

图 17-7：移动阈值使转储后留下较少的可用空间

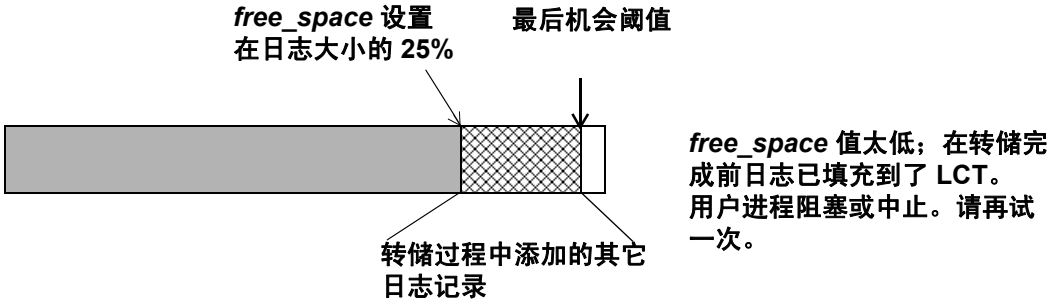


使用 sp_modifythreshold 将 *free_space* 值调整为日志大小的 25%。例如：

```
sp_modifythreshold mydb, logsegment, 512,
    thresh_proc
```

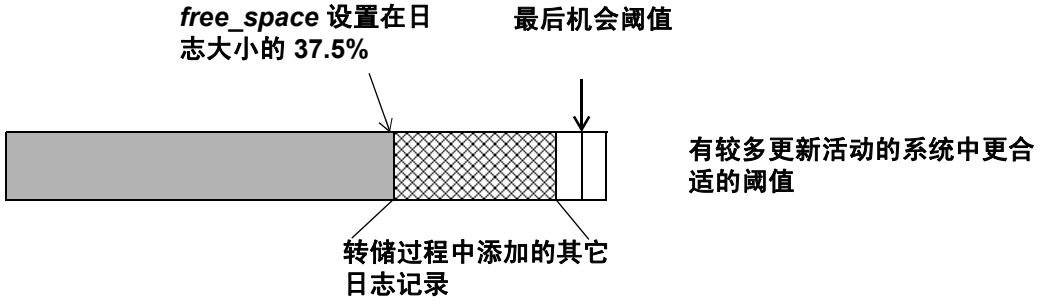
4 转储事务日志并测试新的 *free_space* 值。如果在完成转储之前就达到了最后机会阈值，则表明您没有及早开始 dump transaction，如图 17-8 中所示：

图 17-8：其它日志阈值未能及时开始转储



25% 的可用空间不够。尝试当日志有 37.5% 的可用空间时开始转储事务，如图 17-9 中所示：

图 17-9：移动阈值以留下足够的可用空间来完成转储



使用 sp_modifythreshold 将 *free_space* 值更改为日志容量的 37.5%。例如：

```
sp_modifythreshold mydb, logsegment, 768,
    thresh_proc
```

在其它段上创建其它阈值

在数据段和日志段上都可以创建可用空间阈值。例如，可以在用于存储表和索引的缺省段上创建可用空间阈值。您还要创建关联的存储过程以便在 default 段上的空间下降到此阈值以下时，可在错误日志中输出消息。如果监控错误日志是否包含这些消息，则可在用户遇到问题之前给数据库设备增加空间。

以下示例在 mydb 的 default 段上创建可用空间阈值。当此段上的可用空间下降到 200 页以下时，Adaptive Server 将执行称为 space_dataseg 的阈值过程：

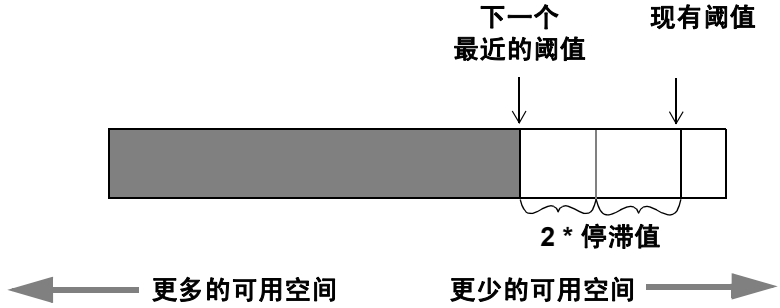
```
sp_addthreshold mydb, "default", 200, space_dataseg
```

请参见第 440 页的“创建阈值过程”。

确定阈值放置位置

每个新阈值必须与下一个最近的阈值相距至少 *@@thresh_hysteresis* 值的两倍空间，如图 17-10 中所示：

图 17-10：确定在何处放置阈值



若要查看数据库的停滞值，请使用：

```
select @@thresh_hysteresis
```

在本例中，段有一个设置为 100 页的阈值，而且数据库的停滞值是 64 页。下一个阈值必须至少为 100 + (2 * 64) 页，即 228 页。

```
select @@thresh_hysteresis

------------
          64

sp_addthreshold mydb, user_log_dev, 228,
    sp_thresholdaction
```

创建阈值过程

Sybase 不提供阈值过程；请自行创建阈值过程以确保它们适合自己节点的需要。

建议阈值过程中的操作包括写入服务器的错误日志和转储事务日志（以便增加日志空间量）。还可以执行对 Open Server 或 XP Server 的远程过程调用。例如，如果在 sp_thresholdaction 中包括以下命令，则该命令在 Open Server 上执行名为 mail_me 的过程：

```
exec openserv...mail_me @dbname, @segment
```

请参见《Transact-SQL 用户指南》。

本节提供关于编写阈值过程的一些准则，还有两个示例过程。

声明过程参数

Adaptive Server 向阈值过程传递四个参数：

- *@dbname*，varchar(30)，其中包含数据库的名称
- *@segmentname*，varchar(30)，其中包含段名
- *@space_left*，int，其中包含阈值的剩余空间值
- *@status*，int，对于最后机会阈值，此参数的值为 1；对于其它阈值，此参数的值为 0

按位置而不是按名称传递这些参数。过程可为这些参数使用其它名称，但必须按以上顺序以及以上数据类型声明这些参数。

生成错误日志消息

在过程开始的几句中包括一条 print 语句，以便在错误日志中记录数据库名、段名和阈值大小。如果过程不包含 print 或 raiserror 语句，错误日志将不包含阈值事件的任何记录。

执行阈值过程的进程是一个内部 Adaptive Server 进程。该进程没有关联的用户终端或网络连接。如果在终端会话过程中通过直接执行阈值过程（即，使用 execute *procedure_name*）测试阈值过程，则可在屏幕上看到来自 print 和 raiserror 消息的输出。如果因为达到阈值而执行相同过程，包含数据和时间的消息将记录在错误日志中。

例如，如果 sp_thresholdaction 包含以下语句：

```
print "LOG DUMP:log for '%1!' dumped", @dbname
```

Adaptive Server 会将以下消息写入错误日志：

```
00: 92/09/04 15:44:23.04 server:background task message:LOG DUMP:log for
'pubs2' dumped
```

转储事务日志

如果 sp_thresholdaction 过程包含 dump transaction 命令，则 Adaptive Server 会将日志转储到过程中指定的设备上。dump transaction 将自日志开头起删除包含未提交的事务记录的页之前的所有页，它通过这种方式来截断事务日志。

当有足够日志空间时，挂起的事务被唤醒。如果中止事务而不是挂起事务，则用户必须重新提交这些事务。

如果 sp_thresholdaction 由于未记录的写入状态而失败，则可以发出 dump tran with no_log 作为替代项。

通常，Sybase 建议不要转储到磁盘，尤其不要转储到同一台计算机的磁盘上或转储到与数据库磁盘相同的磁盘控制器上。然而，因为由阈值启动的转储可能在任何时刻发生，所以最好转储到磁盘后将得到的文件复制到脱机的介质上。（必须将这些文件复制回磁盘才能重新装载这些文件。）

选择取决于：

- 是否已使专用转储设备联机、已装载并已准备好接收转储的数据
- 在数据库可用时，是否有操作员可装入磁带卷
- 事务日志的大小
- 事务率
- 转储数据库和事务日志的定期时间安排
- 可用磁盘空间
- 其它特定于节点的转储资源和约束

一个简单的阈值过程

以下是一个转储事务日志并向错误日志输出消息的简单过程。因为此过程使用一个变量 (*@dbname*) 表示数据库名称，所以此过程可以用于 Adaptive Server 中的所有数据库：

```
create procedure sp_thresholdaction
    @dbname varchar(30),
    @segmentname varchar(30),
    @free_space int,
    @status int
as
dump transaction @dbname
    to tapedump1
print "LOG DUMP: ' %1!' for ' %2!' dumped",
        @segmentname, @dbname
```

一个更复杂的过程

根据传递给过程的参数值，以下阈值过程执行不同的操作。该过程的条件逻辑使得可与日志段和数据段一起使用该过程。

该过程的操作有：

- 如果是由于达到日志的最后机会阈值而调用该过程，则输出“LOG FULL”消息。最后机会阈值的状态位是 1，而所有其它阈值的状态位是 0。只有对于最后机会阈值，测试 if (@status&1) = 1 才返回“true”值。
- 确认所提供的段名为日志段。即使日志段的名称已更改，日志段的段 ID 也始终为 2。
- 输出有关事务日志的“前”和“后”大小信息。如果日志未明显减小，则表明长时间运行的事务导致日志填满。
- 输出事务日志转储开始和停止的时间，这有助于收集有关转储持续时间的数据。
- 如果阈值不在日志段上，则在错误日志中输出消息。此消息给出数据库名、段名和阈值大小，使您了解数据库的数据段已填满。

```
create procedure sp_thresholdaction
     @dbname           varchar(30),
     @segmentname      varchar(30),
     @space_left       int,
     @status           int
as
```

```
declare @devname varchar(100),
           @before_size int,
           @after_size int,
           @before_time datetime,
           @after_time datetime,
           @error int

/*
** if this is a last-chance threshold, print a LOG FULL msg
** @status is 1 for last-chance thresholds,0 for all others
*/
if (@status&1) = 1
begin
      print "LOG FULL: database ' %1!' ", @dbname
end

/*
** if the segment is the logsegment, dump the log
** log segment is always "2" in syssegments
*/
if @segmentname = (select name from syssegments
               where segment = 2)
begin
     /* get the time and log size
     ** just before the dump starts
     */
     select  @before_time = getdate(),
      @before_size = reserved_pages(db_id(), object_id("syslogs"))
      print "LOG DUMP: database ' %1!' , threshold ' %2!' ",
        @dbname, @space_left

      select @devname = "/backup/" + @dbname + "_" +
        convert(char(8), getdate(),4) + "_" +
        convert(char(8), getdate(), 8)

      dump transaction @dbname to @devname

      /* error checking */
      select @error = @@error
      if @error != 0
      begin
           print "LOG DUMP ERROR: %1!", @error
      end
/* get size of log and time after dump */
    select  @after_time = getdate(),
      @after_size = reserved_pages(db_id(), object_id("syslogs"))
```

```
        /* print messages to error log */
        print "LOG DUMPED TO: device ' %1!", @devname
        print "LOG DUMP PAGES: Before: ' %1!' , After ' %2!' ",
             @before_size, @after_size
        print "LOG DUMP TIME: %1!, %2!", @before_time, @after_time end
          /* end of ' if segment = 2' section */
        else
         /* this is a data segment, print a message */
        begin
        print "THRESHOLD WARNING: database ' %1!' , segment ' %2!' at '
%3!' pages", @dbname, @segmentname, @space_left
end
```

决定在何处放置阈值过程

尽管可以为每个阈值创建单独的过程来转储事务日志，但创建可以由所有日志段阈值执行的一个阈值过程更简便。当段上可用空间量下降到阈值以下时，Adaptive Server 将读取受影响的数据库中的 systhresholds 表，获取关联的存储过程的名称。该存储过程可以是下列之一：

- 对 Open Server 的远程过程调用
- 用数据库名限定的过程名（例如，sybsystemprocs.dbo.sp_thresholdaction）
- 非限定的过程名

如果过程名不包括数据库限定符，则 Adaptive Server 将在发生空间不足的数据库中进行查找。如果在其中找不到该过程，而且过程名以字符“sp_”开头，则 Adaptive Server 将在 sybsystemprocs 数据库中查找该过程，然后在 master 数据库中查找该过程。

如果 Adaptive Server 找不到该阈值过程，或不能执行该阈值过程，则会在错误日志中输出消息。

禁用数据段的可用空间计数

使用 sp_dboption 的 no free space acctg 选项，然后使用 checkpoint 命令，可以禁用非日志段上的可用空间计数。不能禁用日志段上的可用空间计数。

如果禁用可用空间计数，则只有日志段上的阈值监控空间使用情况；当达到这些阈值时，并不执行数据段上的阈值过程。禁用可用空间计数可缩短恢复时间，因为除日志段之外，在恢复过程中不再重新计算任何段的可用空间量。

以下示例关闭 production 数据库的可用空间计数：

```
sp_dboption production,
     "no free space acctg", true
```

警告！如果禁用可用空间计数，系统过程将无法提供有关空间分配的准确信息。

禁用数据段可用空间计数后，即使将 no free space acctg 设置为 false，计数也可能会不准确。若要强制 Adaptive Server 重新计算，请发出 shutdown with nowait，然后重新启动 Adaptive Server。这可能会增加恢复时间。

索引

符号

英文

A

B

C

D

F

G

H

J

K

L

M

N

P

Q

R

S

T

W

X

Y

Z