SYBASE®
An SAP® Company

新增功能指南

Adaptive Server Enterprise

15.7 ESD #2

文档 ID：DC01057-01-1503-01
最后修订日期：2012 年 6 月

版权所有 © 2012 by Sybase, Inc. 保留所有权利。

本出版物适用于 Sybase 软件及所有后续版本，除非在新版本或技术说明中另有说明。此文档中的信息如有更改，恕不另行通知。此处说明的软件按许可协议提供，其使用和复制必须符合该协议的条款。

仅在定期安排的软件发布日期提供升级。未经 Sybase, Inc. 的事先书面许可，不得以任何形式、任何手段 （电子的、机械的、手工的、光学的或其它手段）复制、传播或翻译本出版物的任何部分。

Sybase 商标可在 the Sybase trademarks page (http://www.sybase.com/detail?id=1011207) 处进行查看。 Sybase 和文中列出的标记均是 Sybase, Inc. 的商标。 ® 表示已在美国注册。

SAP 和此处提及的其它 SAP 产品与服务及其各自的徽标是 SAP AG 在德国和世界各地其它几个国家/地区的商标或注册商标。

Java 和所有基于 Java 的标记都是 Oracle 和/或其分公司在美国和其它国家/地区的商标或注册商标。

Unicode 和 Unicode 徽标是 Unicode, Inc. 的注册商标。

提到的所有其它公司名和产品名均可能是与之相关联的相应公司的商标。

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

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

目录

第 1 章 创建延迟表 1
- 设置数据库级别的延迟表创建 1
- 创建延迟表 1
- 显式实现延迟表 2
- 标识延迟表 2
- 延迟表的回退 3
- 命令在延迟表中的作用方式 3

第 2 章 并发增强 5
- reorg rebuild 5
- 恢复 5
- 限制 6

第 3 章 合并和拆分分区 7
- 可用于拆分或合并的分区方案 7
- 拆分分区 8
- 合并分区 9
- 移动分区 10
- 锁定 11
- 拆分或合并后的分区对索引的影响 11
- 在拆分或合并分区后保持统计信息准确 12

第 4 章 语句高速缓存中查询的最大大小 13

第 5 章 show_cached_plan_in_xml 的增强 15
- 扫描覆盖范围 15
- 工作表 16
- 动态分区排除 16
- 总逻辑 I/O 和总物理 I/O 17

第 6 章 快速记录 bcp 19

第 7 章 增强型并行 create index 21
配置增强型并行 create index 21
使用增强型并行 create index 22
通过 showplan 查看并行 create index 命令 22

第 8 章 预先计算的结果集 25
预先计算结果集的优势 26
配置 Adaptive Server 使用预先计算的结果集 26
创建预先计算的结果集 27
标识预先计算的结果集 27
刷新预先计算的结果集 28
变更预先计算的结果集 30
删除或截断预先计算的结果集 31
配置失效性 32
查询预先计算的结果集 32
重写查询 32
更新统计信息 34
复制预先计算的结果集 34
限制 34

第 9 章 并发 dump database 和 dump transaction 命令 37
将 Adaptive Server 配置为使用并发转储 39
enable concurrent dump tran 39
限制 39

第 10 章 基于散列的更新统计 41
启用基于散列的统计 42
update statistics hashing 42
收集基于散列的统计信息 42
设置分配粒度 44
设置缓冲区管理器内存 44

第 11 章 通过 update statistics 包含进度消息 45
使用 print_progress 参数 45

第 12 章 Dump 和 Load 的增强 47
配置参数 48
enforce dump configuration 48
enable dump history 49
dump history filename 49
使用 dump configuration 49

转储历史记录文件 51
转储标头的增强 52

第 13 章 在不复制数据的情况下从表中删除列 55
限制 55

第 14 章 扩展了最大数据库大小 57

第 15 章 用户定义的优化目标 59
创建用户定义的优化目标 59
为整个服务器范围和会话设置目标 60
目标报告 60

第 16 章 共享查询计划 63

第 17 章 异步初始化数据库 65
将 Adaptive Server 配置为异步创建或修改数据库 65
enable async database init 65
异步创建或修改数据库 66
确定是否存在需要初始化的空间 66
限制 67

第 18 章 行内大对象压缩 69

第 19 章 配置共享内存转储 71
将 Adaptive Server 配置为使用压缩共享内存转储 71
memory dump compression level 71
配置共享内存转储 72

第 1 章　创建延迟表

create table 命令的 with deferred_allocation 参数用于延迟针对表的页分配。延迟表可帮助创建大量表的应用程序，但是只能使用其中的一小部分。在 Adaptive Server® 为表分配页面前，表被称为 “延迟”表。

系统表包括延迟表的条目。这些条目用于创建与延迟表相关联的对象，例如视图、过程、触发器等。

当 Adaptive Server 插入第一行时 （称为实例化表），将为延迟表执行页面分配。在第一次执行 insert 前访问表，例如选择、删除或更新报告空间使用情况的函数，或者在其它表上进行 DML 的过程中执行参照完整性约束，该操作与表为空时的操作一样。即，针对延迟表进行的 select 会生成空结果集。尽管可以在延迟表上创建索引，这些索引的页面分配将延迟，直到 Adaptive Server 对表进行实例化。

设置数据库级别的延迟表创建

使用 deferred table allocation 数据库选项将数据库配置为延迟针对所有后续创建用户表的页分配：

sp_dboption *database_name*, "deferred table allocation", true

无法针对任何系统数据库 （如 master、 sybsystemprocs、 sybsystemdb） 或临时数据库启用 deferred table allocation。

创建延迟表

使用 create table . . . with deferred_allocation 参数创建延迟表：

create table *table_name* .. .with deferred_allocation

例如，要创建名为 im_not_here_yet 的表，则输入：

```
create table im_not_here_yet (
col_1 int,
```

```
col_2 varchar(20)
)
with deferred_allocation
```

要创建延迟表，无需启用 sp_dboption 'deferred table allocation'。

使用 create table . . . with immediate_allocation 创建在启用 sp_dboption 'deferred table allocation' 时不会发生延迟的表。语法为：

create table *table_name* .. .with immediate_allocation

显式实现延迟表

使用 alter table . . . immediate_allocation 显式实现延迟表。语法为：

alter table *table_name* immediate_allocation

实现表后， Adaptive Server 为所有数据和索引分区分配页面。

例如，要实现表 im_not_here_yet，则输入：

```
alter table im_not_here_yet immediate allocation
```

标识延迟表

sp_help 在 object_status 列中存放延迟表的相关信息。本例显示针对 im_not_here_yet 延迟表的部分 sp_help 输出：

```
sp_help im_not_here_yet
Name            Owner  Object_type  Object_status        Create_date
--------------  -----  -----------  -------------------  -------------------
im_not_here_yet    dbo   user table  deferred allocation    Apr 9 2012 2:09PM
```

sysobjects 在 sysstat3 列中存放 0x80 状态位，此状态位表示表已经延迟。

延迟表的回退

如果延迟表的实现属于遭到回退的事务， Adaptive Server 不会回退它为延迟表执行的页分配。

例如：

```
create table im_not_here_yet with deferred_allocation
go
begin tran t1
go
insert into deferred table ...
go
rollback tran t1
```

insert 会实现 im_not_here_yet 表，然后插入一个值。尽管 rollback tran 从表中删除此值，但页分配不会回退，因此，表会保持实现状态，而且不再是延迟表。

命令在延迟表中的作用方式

大多数命令在延迟表中的作用与在空表中相似：

命令	对延迟表的操作
insert	实现表；执行 insert
select	选择 0 行
update	影响 0 行
delete	影响 0 行
alter table	实现表；执行 alter table
drop table	删除表
create view、 trigger 或 procedure	创建视图、触发器或过程
create index	在不分配页的情况下创建索引
drop index	删除索引
reorg 子命令	无
update statistics	无
truncate table	无
dbcc checktable	无
dbcc checkcatalog	跳过延迟表上的索引

第 2 章　并发增强

Adaptive Server 15.7 ESD # 2 及更高版本中的 reorg rebuild 命令含有 online 参数，此参数用于重组数据并执行维护，不会阻塞用户数据。

reorg rebuild

reorg rebuild ... with online 用于在不使数据脱机的情况下重组数据。语法为：

```
reorg rebuild table_name
    [with online]
```

例如，如果要为 titles 表重新构建索引并保持数据联机，请输入：

```
reorg rebuild titiles with online
```

reorg rebuild ... online 包含三个阶段：

- 在阻塞阶段内短时间采用独占表锁，以便设置新的元数据
- 在非阻塞阶段内重组数据并同步并发活动
- 在另一阻塞阶段内再次采用独占表锁来同步剩余并发活动并载入新的元数据

Sybase® 建议您在表的事务装载量相对较低时运行 reorg rebuild ...online。

恢复

reorg rebuild ... online 在单个事务中运行。恢复由 online 参数执行的工作与恢复在不使用 online 参数时执行的工作相似。当回退通过 online 参数执行的工作时，考虑以下问题：

- 实用程序的运行时回退会释放由 online 参数分配的页。
- 崩溃恢复会清空由 online 参数分配的扩充的分配状态，使其可用于其它任务。
- 在高可用性环境中进行节点故障切换恢复时，如果 reorg rebuild ... online 尝试启动物理锁或逻辑锁，它会等待恢复完成后再获得锁。

限制

reorg rebuild ... online 具有以下限制：

- 对其运行 reorg rebuild ... online 的表必须拥有唯一索引。
- 在 reorg rebuild ... with online 运行时，可以对表进行所有 DML 操作（即 select （但不是 select into）、 insert、 update 和 delete）。Adaptive Server 不允许在 reorg rebuild ... online 运行时，对采用所有页锁定方案的表执行会导致页面拆分的插入操作。
- 不能同时对表执行多个 reorg rebuild ... online 实例。
- reorg rebuild ... online 不会实现不得为空的非实现缺省列。

第 3 章　合并和拆分分区

随着时间的推移，分区的数据分布可能会出现偏差，或者数据最初的分区方式不再满足当前业务要求。使用 alter table 可以合并、拆分或移动分区，从而重新分布数据，并通过分区恢复性能优势。

例如，公司可根据东、南、西、北四个地区来拆分分区，以便更好地访问其数据。拆分分区使得客户代表可以独立于其它地区快速有效地访问其负责地区的客户信息。如果南方区域中的销售有所增长，客户基础已显著扩展，且涉及分区扫描和维护工作的查询较为频繁，将会降低南方分区的速度和效率，从而会失去划分客户数据的意义。在这种情况下，将南方分区中的数据拆分成两个分区 （东南和西南），这样可以恢复其性能而不影响其它分区中的数据。

公司可以合并分区来提高业绩，因为其销售数据分为四个年度季度 – Q1、 Q2、 Q3 和 Q4 分区。在年末，公司将合并年度数据并进行存档。合并分区是一种有效的方法，因为对过去一年销售数据的访问频率并不高，而较旧的数据大多供用户读取而不进行更新。

可用于拆分或合并的分区方案

alter table 用于拆分、合并或移动以下分区方案：

- 域分区
- 列表分区

由于循环分区表中数据的位置是在数据分区中随机分布的，因此无需拆分或合并循环分区。

对于散列分区表，可使用重新分区实用程序来重新分布数据。

拆分分区

使用 alter table ... split partition 参数可将数据重新分布到两个或更多的分区。语法为：

```
alter table table_name
split partition partition_name
into partition_condition_clause
```

其中：

- *partition_name* – 要拆分的分区。
- *partition_condition_clause* – 用来指定源分区数据拆分方式的条件。通常，条件由数值范围或数据范围组成。分区条件应仅涵盖源分区中的全部数据。

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

请参见 《参考手册：命令》。

必须启用 select into/bulkcopy 才能发出 alter table ... split partition。缺省情况下， alter table ... split partition 会在受拆分操作影响的分区表上重新构建局部或全局索引部分。

除重新构建索引的步骤外， alter table ... split partition 操作并不会记录在日志中。Sybase 建议您在运行 alter table ... split partition 命令后执行数据库转储。

以下示例将创建 orders 表，然后拆分其分区以重新分布数据：

```
create table orders (orderid int, amount float,
orderdate datetime)
partition by range (amount)
( P1 values <= (10000) on seg1,
  P2 values <= (50000) on seg2,
  P3 values <= (100000) on seg3,
  P4 values <= (MAX) on seg4)

create clustered index ind_orderid
on orders(orderid) local index (i1 on seg1, i2 on seg2,
i3 on seg3, i4 on seg4)

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

```
alter table orders
split partition P3
into
( P7 values <= (50000) on seg2,
  P8 values <= (100000) on seg3)

alter table orders
split partition P4
into
( P9 values <= (200000),
  P10 values <= (MAX))
```

合并分区

使用 alter table ... merge partition 可对两个或更多允许合并 （即可用于合并）的分区进行数据合并，从而构成一个分区。分区是否可以合并取决于其分区方式：

- 对于列表分区表，任何两个分区都可以合并
- 对于域分区表，分区必须相邻才能合并

语法为：

alter table *table_name*
merge partition {*partition_name* [{, *partition_name*}…]}
into *destination_partition_name* [on *segment_name*]

其中：

- *partition_name* – 要合并的源分区。所有源分区都必须位于同一个段上。
- *destination_partition_name* – 新分区或现有分区。如果 *destination_partition_name* 是现有分区，它不能是您正在合并的任何源分区。

  合并后的目标分区的分区条件派生自所合并的全部源数据分区的分区条件，因此，目标分区包含所合并的源数据分区内的所有数据。例如，对于列表分区表，合并后的分区的新分区条件就是构成源数据分区条件的所有值的并集。

请参见 《参考手册：命令》。

必须启用 select into/bulkcopy 才能发出 alter table ... merge partition。

alter table ... merge partition 将完整记录到日志中。使用事务转储可从服务器故障中恢复。

以下示例将创建 sales 表，然后合并其分区以整合数据：

```
create table sales(salesmanid int, salesdate datetime,
salesregion varchar(10))
partition by range(salesdate)
( Q1 values <= ('31 Mar 2007'),
  Q2 values <= ('30 Jun 2007'),
  Q3 values <= ('30 Sep 2007'),
  Q4 values <= ('31 Dec 2007'))
create index ind_region on sales(salesregion)

alter table sales
merge partition Q3
into Q4

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

移动分区

使用 alter table ... move partition 可将分区 （及其索引）移动到指定段中。语法为：

alter table *table_name*
move partition *partition_name*
to *destination_segment_name*

其中：

- *partition_name* – 要移动的分区。
- *destination_segment_name* – 分区要移动到的新段或现有段。不能指定 “default”作为 *destination_segment_name*。

请参见 《参考手册：命令》。

必须启用 select into/bulkcopy 才能发出 alter table ... move partition。

锁定

在分区的拆分、合并或移动过程中， Adaptive Server 会获取其执行操作的表上的排它锁以及该表对应的系统表条目。

拆分或合并后的分区对索引的影响

对含有索引的表执行 split、merge 或 move partition 时，Adaptive Server 会重新构建所有受影响的索引。

表 3-1：拆分及合并分区对索引的影响

Command	全局非聚簇索引	局部索引	局部聚簇索引	局部非聚簇索引
split partition	重新构建索引	所有受影响的索引分区将重新构建	重新构建所有索引分区	在缺省段上重新构建
merge partition	如果源段和目标段相同，则不会产生影响	所有受影响的索引分区将重新构建	重新构建所有索引分区	在缺省段上重新构建

如果要拆分或合并的表包含单独段的索引，则重新构建的索引所在的段取决于索引类型。

表 3-2：拆分的分区对索引段的影响

索引类型	在拆分或合并操作后
全局非聚簇索引	索引保留在操作前的段上。
局部非聚簇索引	• 新索引分区 （对应于拆分或合并后的目标数据分区）位于在索引级别指定的段上。 • 如果未指定索引段，则索引位于缺省段上。 • 所有未受影响的索引分区 （对应于拆分或合并操作中未涉及的其它数据分区）保留在拆分或合并操作前的段上。
局部聚簇索引	• 新索引分区与相应的目标数据分区位于同一个段上。 • 未受影响的索引分区 （对应于拆分或合并操作中未涉及的其它数据分区）保留在拆分或合并操作前的段上。

在拆分或合并分区后保持统计信息准确

合并或拆分分区会从 systabstats 和 sysstatistics 中删除统计信息。因此，Sybase 建议您在合并或拆分分区后运行 update statistics。

第 4 章　语句高速缓存中查询的最大大小

在 Adaptive Server 15.7 ESD #2 之前的版本中，在语句高速缓存中存储的各个语句都具有 16K 的大小限制，即使将 statement cache size 配置为更大的大小，此限制仍然有效。

在 Adaptive Server 15.7 ESD #2 版本及更高版本中，只需提高 statement cache size 和 max memory 配置参数，即可在语句高速缓存中存储高达 2MB 的 SQL 语句（针对 64 位计算机）。

如果语句高速缓存的 SQL 查询文本小于 16K，则使用 show_cached_text 进行显示。但是，如果 SQL 查询文本大于 16K，即使完整文本在语句高速缓存中可用， show_cached_text 也会截断 SQL 查询文本。

使用 show_cached_text_long 显示大于 16K 的 SQL 查询文本。show_cached_text_long 显示最多 2MB 的 SQL 查询文本。

第 5 章　*show_cached_plan_in_xml* 的增强

Adaptive Server 15.7 ESD #2 在 show_cached_plan_in_xml 的输出中包含以下内容的新信息：

- 索引扫描覆盖范围
- 计划中使用的工作表
- 动态分区排除指示
- 总逻辑 I/O (lio) 和总物理 I/O (pio)

扫描覆盖范围

<scanCoverage> 标记表示查询计划的索引扫描是 “覆盖”还是 “不覆盖”。

有关覆盖索引扫描的详细信息，请参见 《性能和调优系列：锁定和并发控制》中的 “索引”。

示例

```
select show_cached_plan_in_xml(1139220075, 0)

<text>
<![CDATA[
SQL Text:select * from sysobjects group by name]]>
</text>
<IndexScan>
<VA>0</VA>
...
<scanCoverage> NonCovered </scanCoverage>
...
</IndexScan>
```

工作表

<WorkTable> 下的 <wtObjName> 标记提供查询计划所用工作表的名称。<WorkTable> 标记位于创建工作表的运算符下。

示例

```
select show_cached_plan_in_xml(107219961, 0)
go

<text>
<![CDATA[
SQL Text:select distinct c1, c2 from t1, t2 where c1 = d1]]>
</text>
<opTree>
...
    <MergeJoin>
        ...
       <WorkTable>
            <wtObjName>WorkTable2</wtObjName>
    </WorkTable>
    ...
    </MergeJoin>
```

动态分区排除

show_cached_plan_in_xml 在 “分区信息” 部分使用 <dynamicPartitionElimination> 标记显示动态分区排除信息，表示正在排除运行时分区。 show_cached_plan_in_xml 表示在 <eliminatedPartition> 标记下排除编译时分区。

有关分区排除的详细信息，请参见 《性能和调优系列：查询处理和抽象计划》中的 “并行查询处理” 章节。

示例 2

```
select show_cached_plan_in_xml(435436901,0)
go
< query>
   <statementId>435436901</statementId>
   <text>
   <![CDATA[

. . .
```

```
                    <partitionInfo>
                           <partitionCount>3</partitionCount>
                           <eliminatedPartition>1</eliminatedPartition>
                           <eliminatedPartition>3</eliminatedPartition>
                           <dynamicPartitionElimination>No</dynamicPartition
                            Elimination>
                      </partitionInfo>

. . .
```

总逻辑 I/O 和总物理 I/O

<totalLio> 和 <totalPio> 标记表示每项计划的总逻辑 I/O 和总物理 I/O。共有两组 <totalLio> 和 <totalPio> 标记，分别用于表示逻辑 I/O 和物理 I/O 的实际值和估计值。

有关逻辑 I/O 和物理 I/O 的详细信息，请参见 《性能和调优系列：查询处理和抽象计划》中的 “显示查询优化策略和估计值”。

示例

```
select show_cached_plan_in_xml(1123220018, 0)
go
    <text>
           <![CDATA[
                SQL Text:select * from titles]]>
    </text>
<Plan>
<optTree>
. . .


          <est>
                <totalLio>3</totalLio>
                <totalPio>3</totalPio>
            </est>
            <act>
                <totalLio>3</totalLio>
                <totalPio>1</totalPio>
            </act>
. . .
```

第 6 章　快速记录 *bcp*

在 Adaptive Server 15.7 ESD #2 版本及更高版本中，可以快速并完全记录运行的 bcp，以便实现完整的数据恢复。早期版本只能记录页分配。

请参见《实用程序指南》中的第 4 章“使用 bcp 从 Adaptive Server 传出数据或向 Adaptive Server 传送数据”。

第 7 章

增强型并行 create index

在 Adaptive Server 15.7 ESD 2 版本及更高版本中，可以发出并行 create index，以便使用查询执行引擎更加高效地执行命令。

配置增强型并行 *create index*

增强型并行 create index 在缺省情况下处于禁用状态，而且是 enable functionality group 配置参数的一部分。要使 Adaptive Server 使用并行 create index：

1 启用 enable functionality group 配置参数：

```
sp_configure "enable functionality group", 1
```

2 将数据库选项 select into/bulkcopy/pllsort 设置为 true：

```
sp_dboption database_name, "select into", true
```

3 根据硬件环境设置以下配置参数：

- number of worker processes – 设置为所有用户使用的并发执行并行线程的最大数量
- max parallel degree – 设置为单独用户使用的 maximum parallel degree，但不得高于 number of worker processes
- max online engines – Sybase 建议您在配置并行线程时，根据硬件可用性及其它负载配置足够的 number of engines。number of worker processes 的设置一般高于 number of engines

使用增强型并行 create index

经过并行 create index 的配置后， Adaptive Server 可以判断使用并行执行方法执行 create index 是否为最佳选择。如果 Adaptive Server 判断序列查询计划的效率最高，则不会使用并行查询计划。如果 Adaptive Server 判断并行查询计划的效率最高，则在以下情况选择增强型并行查询计划：

- 要为其创建索引的表：
  - 使用仅数据锁定格式，且
  - 未进行分区，且
  - 表不为空
- 正在创建的索引为非聚簇索引，并且
- 索引的前导列至少具有两个不同值

使用 create index ... with consumers = *N* 强制 Adaptive Server 在应使用序列查询计划时使用并行查询计划。例如，在以下情况中，即使表中含有的行过少， Adaptive Server 也会使用并行查询计划：

```
create index i1 on t1(c1, c3) with consumers = 3
```

如果使用 with consumers 强制执行并行 create index，Adaptive Server 不会选择增强型并行查询计划，而会使用低于 15.7 ESD #2 的 Adaptive Server 版本中的并行 create index 查询计划。

通过 showplan 查看并行 *create index* 命令

如果 Adaptive Server 被配置为使用并行 create index 并选择了增强型并行 create index 查询计划，则 showplan 会在 `PLL CREATE INDEX COORDINATOR` 和 `CREATE INDEX` 运算符下显示 create index 命令的相关信息。例如：

```
create index i1 on t5(c1) with consumers = 3
. . .

QUERY PLAN FOR STATEMENT 1 (at line 1).
Executed in parallel by coordinating process and 3 worker processes.

STEP 1
  The type of query is CREATE INDEX.

  5 operator(s) under root
```

```
|ROOT:EMIT Operator (VA = 5)
|
|   |PLL CREATE INDEX COORDINATOR Operator
|   |
|   |   |EXCHANGE Operator (VA = 3) (Merged)
|   |   |Executed in parallel by 3 Producer and 1 Consumer processes.

|   |   |
|   |   |   |EXCHANGE:EMIT Operator (VA = 2)
|   |   |   |
|   |   |   |   |CREATE INDEX Operator
|   |   |   |   |
|   |   |   |   |   |SCAN Operator (VA = 0)
|   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |  t5
|   |   |   |   |   |  Table Scan.
|   |   |   |   |   |  Forward Scan.
|   |   |   |   |   |  Positioning at start of table.
|   |   |   |   |   |  Executed in parallel with a 3-way range repartitioning scan.
|   |   |   |   |   |  Using I/O Size 16 Kbytes for data pages.
|   |   |   |   |   |  With MRU Buffer Replacement Strategy for data pages.
```

第 8 章　预先计算的结果集

预先计算的结果集 (PRS) 是结果经过计算、存储且可供将来使用的视图。配置预先计算的结果集后， Adaptive Server 将对查询进行预先计算，并尝试在后续迭代中使用预先计算的结果。预先计算的结果集也称为物化视图。

从概念上讲，预计算结果集既属于视图 （因为包含存储在系统表中的查询定义），又属于表 （因为包含持久数据）。许多对表执行的操作也可以对预先计算的结果集执行，其中包括创建索引和运行 update statistics。

将 Adaptive Server 配置为使用预先计算的结果集后，优化程序会尝试使用预先计算的结果集自动重写每项查询。不过，优化程序选择的最终计划主要以开销为依据。

优化程序使用预先计算的结果集重写查询时，会判断哪个预先计算的结果集是最佳候选项。如果优化程序选择用预先计算的结果集替换整个查询或查询的一部分，它还会向重写后的查询添加所有必要的补偿 （即，为确保重写后的查询与原始用户查询等效所需的任何谓词）。例如，如果用户查询包含以下连接：

```
c1=c2 and c2=c3 and c3=c4
```

而预先计算的结果集包含以下连接：

```
c1=c2 and c3=c4
```

则使用预先计算结果集的已重写查询必须具有与 `c1=c3` 相似的补偿谓词才能构成等效查询。

与索引一样，预先计算的结果集也存在针对并发 insert、update 和 delete 语句的维护开销。通常，如果定义涉及多个表连接，则预先计算结果集的维护开销要高于索引的维护开销。因此，预先计算的结果集不适用于具有大量并发 insert、 update 和 delete 语句以及基于索引的简单 select 语句的 OLTP。

预先计算结果集的优势

站点能否从预先计算结果集获益取决于其设计方式。尽管您可能希望对尽可能多的查询（尤其是更多的连接）进行预先计算，并且使其可用于多个查询，但预先计算结果集会占用额外的磁盘空间，并且维护开销也较高。可以额外创建索引帮助提高查询性能，不过这样也会产生额外的维护开销。

预先计算结果集最适合频繁执行且开销巨大的查询，例如涉及大量集合与连接操作的查询。用户提交查询后，优化程序会尝试重写查询，以使用现有的预先计算结果集代替基表。

一般情况下，应捕获应用程序的负载，然后根据此负载设计预先计算的结果集。为所有查询及其使用频率创建一份合并连接图，以表明在哪些情况下适合为多个查询使用相同的预先计算结果集，这是一种很好的方法。

在使用前对预先计算的结果集进行测试。如果查询为只读或部分读，则根据查询额外使用的磁盘空间及其填充数据所需的时间来衡量查询的性能增益；如果查询为只读或部分读的组合，则根据吞吐量衡量预先计算结果集的影响。

配置 Adaptive Server 使用预先计算的结果集

创建或变更预先计算的结果集之前，应检验以下会话的 set 参数是否已设置正确：

- set ansinull – on
- set arithabort – on
- set arithignore – off
- set string_rtruncation – on

使用 create precomputed result set 创建预先计算的结果集。要为查询使用预先计算的结果集，应针对会话发出 set materialized_view_optimization 命令。

创建预先计算的结果集

使用 create 命令定义预先计算的结果集。语法为：

```
create {precomputed result set | materialized view}
    prs_name [(alternative_column_name
    [[constraint constraint_name]
        unique (column_name,...)]

    [{immediate | manual } refresh]
    [{populate | nopopulate}]
    [enable | disable]
    [{enable | disable } use in optimization]
    [lock {datarows | datapages | allpages}]
    [on segment_name]
    [partition_clause]

    as query_expression
```

可以在预先计算的结果集中指定以下内容：

- 分区
- 段
- 索引（不允许指定函数索引）
- 唯一键（在为 immediate refresh 创建预先计算的结果集时必须包含唯一键约束）

如果删除基表，则预先计算的结果集将变为 disabled。

请参见《参考手册：命令》。

标识预先计算的结果集

sp_help 在 Object_type 和 object_status 列中包含预先计算结果集的相关信息：

```
sp_help mv1
Name  Owner  Object_type              Object_status
  Create_date
-----  -----  -----------------------  ------------------------------------
  --------------------
mv1   dbo   precomputed result set  immediate, enabled, enabled for QRW
   Apr 10 2012  8:57AM
...
```

sysobjects 指明对象为 type 列中包含 RS 值的预先计算结果集。

刷新预先计算的结果集

预先计算的结果集与作为其构建基础的基表并不一定保持同步，因此，还必须对其进行自动或手动刷新。在创建预先计算的结果集时配置刷新策略，或者以后通过 alter precomputed result set 命令配置：

- 立即刷新 – 预先计算的结果集在用于更新基表的事务中进行更新。这是缺省选项。不过，创建 immediate refresh 的预先计算结果集要求用户拥有定义查询中的所有表。
- 手动刷新 – 预先计算的结果集通过显式 refresh 命令更新。由于系统不会维护手动刷新，因此， Adaptive Server 会将手动刷新所包含的数据视为失效数据 （即便是刚发布 refresh 命令后的数据也是如此），并且只有在查询可接受失效数据时才会选择这些预先计算结果集来重写查询。 refresh 命令在隔离级别 1 或更高级别下执行。

用于手动刷新预先计算结果集的语法如下：

```
refresh {precomputed result set | materialized view}
    [owner_name.]prs_name
```

如果构成预先计算结果集派生基础的任一基表的模式已经更改，或者模式已在删除后重新创建 （即对象 ID 已经更改），则 refresh 命令将无法执行并返回错误，指出必须删除预先计算的结果集，然后重新创建。

只有预先计算结果集的所有者才能使用 refresh 命令。如果用户拥有更新基表的权限，则该用户也可以维护预先计算的结果集。

大多数情况下，优化程序应将预先计算的结果集与 immediate refresh 而非 manual refresh 结合使用来重写查询 （除非将 materialized_view_optimization 设置为 stale）。

如果需要对 insert、update 和 delete 语句的发生时间进行控制，则最好手动刷新预先计算的结果集。在这些命令出现后，对预先计算的结果集进行计划手动刷新，然后使用预先计算的结果集为只读应用程序提供帮助。不过，应注意执行手动刷新所需的时间和额外的磁盘空间，并做出相应的计划。

注释在创建预先计算的结果集后，其所有者可能对基表不具有 select 权限。如果出现这种情况，则手动刷新预先计算结果集这种维护可能会失败，基表的新更改也不会更新到其中。

不能将 refresh 命令作为批处理的一部分执行。

以下示例说明了如何刷新预先计算的结果集：

1　创建表 t1：

```
create table t1 (
c1 int,
c2 int,
c3 char(5))
```

在该表中填入以下数据：

```
 c1          c2          c3
----------- ----------- -----
          1           3  Aagg
          2           8  Xyz
```

2　创建表 t2：

```
create table t2
(a1 int,
a2 int,
a3 char(5))
```

在该表中填入以下数据：

```
 a1          a2          a3
----------- ----------- -----
          1           5  Ghr
          2           1  Gser
          3           6  agfh
```

3　创建预先计算的结果集 prs_1：

```
create precomputed result set prs_1
unique (t1.c1, t2.a2)
as select t1.c1, t2.a2 from t1, t2 where t1.c1=t2.a1
```

prs_1 创建完毕并填充有以下初始行：

```
 c1          a2
----------- -----------
          1           5
          2           1
```

4 如果将 3、7 和“fhi”这几个值插入 t1 中，prs_1 会立即以 3 和 6 这两个值进行更新：

```
 c1          a2
----------- -----------
          1           5
          2           1
          3           6
```

5 如果将 a1 = 2 的行从 t2 中删除， prs_1 会立即以此更改进行更新：

```
 c1          a2
----------- -----------
          1           5
          3           6
```

如果 Adaptive Server 回退基表的更新事务，它也会在同一事务中回退对基表预先计算结果集的立即 update。

变更预先计算的结果集

使用 alter 命令可更改预先计算结果集的策略或属性。语法为：

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

请参见 《参考手册：命令》。

以下示例将 author_prs 预先计算结果集从 manual 变更为 immediate：

```
alter precomputed result set author_prs
immediate refresh
```

alter 会在用户从 manual 更改为 immediate 刷新，或从 disable 更改为 enable 时自动刷新预先计算的结果集。针对 disable use in optimization 变更预先计算结果集可防止预先计算结果集参与以后的查询重写。不过，系统不会对已使用预先计算结果集进行高速缓存的任何计划进行重新编译。

与其它 DDL 命令相似，不能在多语句事务中发出 alter precomputed result set （除非为数据库将 ddl in tran 选项设置为 true）。只有预先计算结果集的所有者才能发出 alter precomputed result set 命令。

删除或变更预先计算结果集所基于的基表或视图后，预先计算结果集将自动变更为 disable。如果对生成预先计算结果集所用的任一基表运行 bcp in 或 select into existing， Adaptive Server 便会将预先计算结果集设置为 disable。

将预先计算结果集变更为 disable（对基表运行 alter precomputed result set 命令或 alter table 命令）后，已进行高速缓存且使用预先计算结果集的任何计划都会在下次执行时重新得到编译。

删除或截断预先计算的结果集

删除预先计算的结果集时，会删除其数据、所有系统表条目以及该预先计算的结果集本身。语法为：

```
drop {precomputed result set | materialized view}
    prs_name
```

只有预先计算结果集的所有者才能发出 drop precomputed result set 命令。请参见 《参考手册：命令》。

以下示例将删除 authors_prs：

```
drop precomputed result set authors_prs
```

使用 truncate 命令可截断预先计算结果集中的数据。 truncate 会在系统表中保留预先计算结果集的定义，从而确保以后可通过 refresh 命令重新填充预先计算的结果集。

截断预先计算的结果集会使其变为禁用状态。如果发出 refresh prs 命令，则 Adaptive Server 会将预先计算的结果集重新变为启用状态。

语法为：

```
truncate {precomputed result set | materialized view}
    prs_name
```

以下示例将截断 author_prs：

```
truncate precomputed result set authors_prs
```

Adaptive Server 会先将 refresh 命令作为 truncate 命令执行，然后再重新计算预先计算的结果集。在极少数情况下， truncate 命令成功执行而重新计算失败，此时，预先计算的结果集将保持禁用状态，用户可以重新发出 refresh 命令。

配置失效性

预先计算的结果集依赖其基表的更新来确保数据最新。如果将预先计算的结果集配置为使用 immediate 更新，则对基表的任何更新都会同时更新预先计算的结果集。这种更新通过基表的更改以增量维护的形式进行。不过，如果将预先计算结果集配置为使用 manual 更新，则数据可能会变为失效，因为只有在运行 refresh 命令后才会进行更新 （在此过程中 Adaptive Server 会重新计算预先计算结果集而不是执行增量维护）。

除非另有规定，否则， Adaptive Server 不会使用失效的预先计算结果集来重写查询。使用 set materialized_view_optimization 可在会话级别指定 Adaptive Server 在优化过程中重写查询时能否使用失效的预先计算结果集。

```
set materialized_view_optimization {disable | fresh | stale}
```

如果希望 Adaptive Server 使用失效的预先计算结果集重写查询：

- 用户必须是失效的预先计算结果集的所有者，并且
- set materialized_view_optimization 必须设置为 stale。

请参见 《参考手册：命令》。

查询预先计算的结果集

Adaptive Server 允许用户从预先计算的结果集中选择信息，但用户不能在预先计算的结果集中插入、更新或删除信息，而是必须在基表中插入、更新或删除信息，然后再刷新预先计算的结果集。

重写查询

查询重写机制根据可用的预先计算结果生成替代计划。替代计划会与优化程序中的其它计划进行竞争， Adaptive Server 从中选择估算开销最低的一项。不过，查询重写机制仅对 select 查询有效；它不考虑重写 insert、 update、 delete 和 select into 查询。

Adaptive Server 可能会重写整个查询以创建等效的预先计算结果集，也可能只重写查询的一部分，具体取决于查询属性以及可用的预先计算结果集。预先计算的结果集必须完全涵盖 Adaptive Server 所重写的查询的逻辑数据集。

例如，您的查询像如下查询那样复杂且涉及多表连接、多个谓词、分组和集合：

```
select t1.col1,t2.col1,t3.col1,
     sum(t1.col3),sum(t2.col3), sum(t3.col3)
from t1, t2, t3
where t1.col1 = t2.col1
     and t2.col1 = t3.col1
     and t1.col2 < 60
     and t1.col1 > 5
     and t1.col2 + t2.col2 < 40
group by t1.col1, t2.col1, t3.col1
```

创建以下预先计算的结果集：

```
create precomputed result set newprs
as
select t1.col1 as p11, t1.col2 as p12, t2.col1 as p21,
     t2.col2 as p22, t3.col1 as p31, t3.col2 as p32,
     sum(t1.col3) as agg_s13,sum(t2.col3) as agg_s23,
     sum(t3.col3) as agg_s33
from t1, t2, t3
where t1.col1 = t2.col1
and t1.col2 < 60
and t1.col2 + t2.col2 < 40
group by t1.col1, t2.col1, t3.col1, t1.col2,
     t2.col2, t3.col2
```

查询重写机制可能会将原始查询变更为以下这种执行起来更为简单且开销也更低的形式：

```
select p11,p21,p31,
     sum(agg_s13),sum(agg_s23),sum(agg_s33)
from newprs
where p21 = p31
     and p11 > 5
group by p11, p21, p31
```

更新统计信息

Adaptive Server 允许用户对预先计算的结果集运行 updates statistics。

复制预先计算的结果集

以下预先计算的结果集命令可供复制：

- create precomputed result set
- alter precomputed result set
- drop precomputed result set
- truncate precomputed result set
- refresh precomputed result set

尽管预先计算的结果集 DDL 可供复制，但预先计算的结果集无法标记为复制。也就是说，与常规表不同， Adaptive Server 不会复制对存储在预先计算结果集中的数据进行的任何维护更改，不管这些更改是从预先计算的结果集 DDL （本身可供复制）发起的，还是作为基表 update 事务的一部分产生的 （从 immediate refresh 策略发起）。

预先计算结果集 DDL 的复制只能在两个具有预先计算结果集功能的 Adaptive Server 间进行。

限制

预先计算的结果集不能包括以下内容：

- 对其它预先计算结果集的引用。不过，如果创建预先计算的结果集时使用 manual refresh 策略，则预先计算的结果集可以引用视图。
- select 列表中的表达式。不过，可以包含作为 group by 列表组成部分的表达式。
- 加密列，或者对自身列数据加密的结果集。
- 对其它数据库中的表或函数的引用。
- 对基表中虚拟计算列的引用。不过，预先计算的结果集可以引用基表中已实现的计算列。

- compute、 compute by、 group by all 或 order by 子句。
- 非确定性函数 （如 getdate）。
- XML。
- 子查询。
- 外连接和半连接。
- 对用户定义函数的调用。
- 派生表。
- 对系统表、临时表或虚设表的引用。
- Union 子句。
- 除唯一键约束外的任何约束。
- 以相同值、空值或非空子句定义的列。
- 缺省值或规则。
- 游标。
- 统计聚合函数。
- text、 image 或 unitext 列。

除上述限制外，使用 immediate refresh 策略的预先计算结果集还不能包含以下内容：

- top、 min、 max 和 avg 命令
- distinct 子句
- 自连接
- 函数
- 对代理表的引用
- 视图引用
- 引用集合结果的 having 子句
- 引用可为空的表达式的 sum 函数

第 9 章　并发 *dump database* 和 *dump transaction* 命令

在 Adaptive Server 15.7 ESD #2 版本及更高版本中，可以并发运行 dump transaction 与 dump database 命令，从而在较长时间内 （比转储策略制定的时间长）降低数据库更新的丢失风险。

dump database 分两个阶段运行：在第一阶段中，将数据库页复制到转储存档中；在第二阶段中，将事务日志的活动部分复制到转储存档中。dump transaction 在将事务日志的活动部分复制到转储存档时仅具有一个阶段。

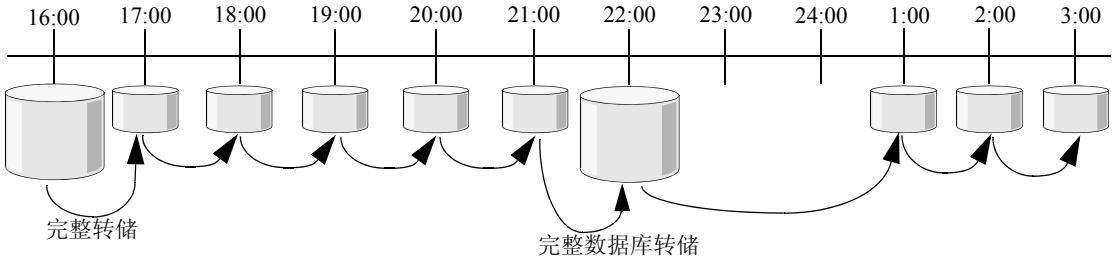
dump database 在复制数据库页 （最长阶段）时允许并发运行 dump transaction，但在复制事务日志的活动部分时不允许并发运行 dump transaction。在复制事务日志的活动部分时， dump transaction 等待 dump database 完成后 （反之亦然）才开始运行。

与 dump database 并发执行的任何 dump transaction （即：在 dump database 开始复制活动日志前完成）均无法装载到该数据库转储之上：事务日志的转储属于之前的装载序列。

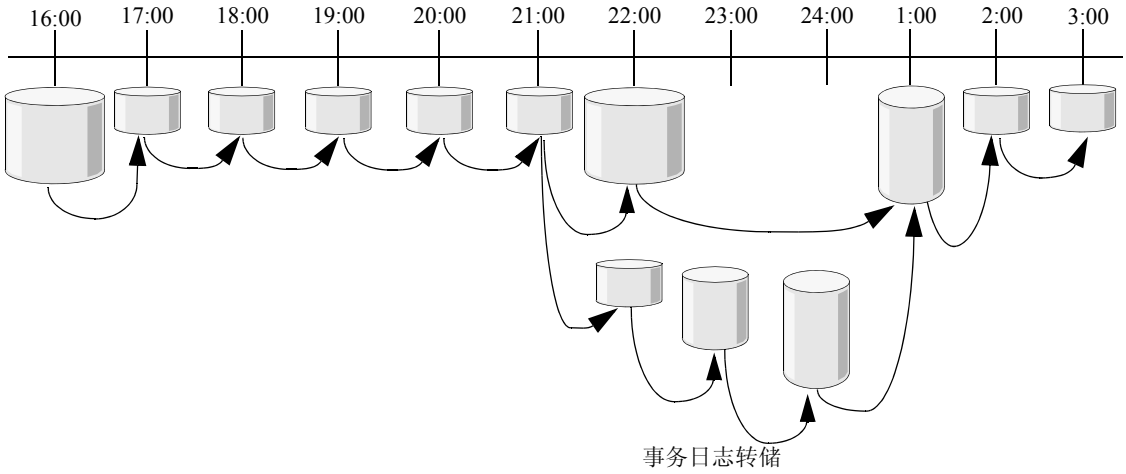
难以判断通过 dump tran 转储的事务日志是在数据库转储之前 （在这种情况下，无需装载）还是在数据库转储之后 （在这种情况下，需要装载）。使用转储历史记录文件，根据需要包含或不包含事务日志，从而解决优先权的问题。请参见第 51 页上的 “转储历史记录文件”。

本例展现典型日常备份策略并使用序列化转储，当 dump database 运行时，数据库在 23:15 发生崩溃，则只能将数据库恢复至 21:00 的状态，超过两小时的数据库更新都会丢失：

序列化转储



然而，并发转储用于在 dump database 运行时继续转储事务日志，因此，在 23:15 发生崩溃时，由于在 22:00 和 23:00 分别进行过事务日志转储，可以将数据库恢复至 23:00 的状态，这样，只有 15 分钟的数据库更新会丢失。



注释dump transaction 在与 dump database 并发运行时不会截断事务日志。

将 Adaptive Server 配置为使用并发转储

通过 enable concurrent dump tran 配置参数使 Adaptive Server 能够执行并发转储。

enable concurrent dump tran

摘要信息	
缺省值	0（关闭）
有效值	0（关闭），1（打开）
状态	动态
显示级别	Comprehensive
要求的角色	系统管理员
配置组	应用程序功能

enable concurrent dump tran 允许或禁止 Adaptive Server 使用并发转储。

enable concurrent dump tran 是 enable functionality 配置参数组的一部分。此组参数的缺省值取决于为 enable functionality group 设置的值。此组中除 enable functionality group 之外的各个配置参数的 DEFAULT 值将设置为与 enable functionality group 相同。也就是说，如果将 enable functionality group 设置为 1，则此组中其它任何配置参数的 DEFAULT 值均为 1。

除 enable functionality group 的值之外，可以在 sp_configure 和 sp_helpconfig 输出中针对 Application Functionality 组中各配置参数忽略 DEFAULT 值。

请参见 《系统管理指南，卷 1》。

限制

针对并发转储存在以下限制：

- 不得将一个 dump database 与另一个 dump database 并发运行。
- 不得将一个 dump transaction 与另一个 dump transaction 并发运行。
- 如果在尚未运行并发数据库转储的情况下启动 dump transaction，则等待 dump transaction 完成后再启动 dump database 命令。

第 10 章　基于散列的更新统计

在 Adaptive Server 15.7 ESD #2 版本中，可以针对次要索引属性和无索引列收集基于散列的统计信息，而不必使用基于排序的统计信息。使用基于散列的统计代替基于排序的统计，可以减少所需扫描次数并避免根据磁盘进行排序，从而提高性能。

基于散列的统计比基于排序的统计更为灵活：

- 基于散列的统计的运行时间比基于排序的统计的运行时间短，这样，便增加了在维护窗口过程中可以完成的工作量。
- 由于基于散列的统计需要的过程高速缓存较小，因此，可在维护窗口外针对仅数据锁定表运行 update statistics，因为 tempdb 缓冲区高速缓存 （通常使用缺省数据高速缓存）通常比过程高速缓存大得多。
- 一般情况下，基于散列的统计不需要较大的 tempdb 磁盘分配空间。如果此前已经增加 tempdb 的大小以容纳因 update statistics 产生的大规模排序，则可重新部署该空间。
- 通过散列运行 update index statistics 和 update all statistics 的速度比通过 with sampling 运行排序的速度快。但 update statistics *table_name*(*col_name*) 例外。

基于散列的统计对唯一列值的数量少于 65536 的列使用低域算法，对唯一列值的数量等于或多于 65536 的列使用高域算法。在这两种算法中，低域散列生成的直方图更为准确，因为 Adaptive Server 使用所有域值的实际计数来创建直方图。然而，高域散列生成的直方图的准确性较低，因为 Adaptive Server 生成的内存中间直方图针对包含 65536 个唯一值的各个块更新。

由于收集基于散列的统计信息比较占用 CPU，因此，您可能需要创建具有 EC3 属性的执行类，向其分配 update statistics 登录名。 Adaptive Server 向 update statistics 维护会话分配较低的优先级，从而降低维护窗口很小或不存在时的影响。

在运行 update statistics 时， Sybase 建议您：

- 使用分区表，以便只有活动分区需要 update statistics 维护
- 使用 datachange 确定何时运行 update statistics

- 在并发性可能成为问题时避免针对所有页锁定表运行 update statistics（因为 update statistics 针对所有页锁定表使用 1 级页锁定，与针对仅数据锁定表的错读相比，此操作的并发性较低）

启用基于散列的统计

使用 update statistics hashing 配置参数使 Adaptive Server 收集基于散列的统计信息。

update statistics hashing

摘要信息	
缺省值	部分
值的范围	可以是以下各项之一： • off – 不散列。 • on – 对所有列使用散列。 • partial – 只对低的唯一计数列使用散列。 • default – 关闭 .
状态	动态
显示级别	Comprehensive
要求的角色	系统管理员
配置组	一般信息

update statistics hashing 使 Adaptive Server 能够收集基于散列的统计信息。

收集基于散列的统计信息

使用以下语法创建基于散列的统计：

```
update index statistics
    table_name [[partition data_partition_name] |
    [ [index_name [partition index_partition_name]]]
```

```
        [using step values]
        [with consumers = consumers] [, sampling=N [percent]]
    [, no_hashing | partial_hashing | hashing]
            [, max_resource_granularity = N [percent]]
            [, histogram_tuning_factor = int ]

    update all statistics
        table_name [partition data_partition_name ]
        [using step values]
        [with consumers = consumers] [, sampling=N [percent]]
    [, no_hashing | partial_hashing | hashing]
            [, max_resource_granularity = N [percent]]
            [, histogram_tuning_factor = int ]

    update statistics
        table_name [[partition data_partition_name]
              [ (col1, col2, ···) | (col1), (col2), ···] |
        [ [index_name [partition index_partition_name]]]
        [using step values]
        [with consumers = consumers] [, sampling=N [percent]]
    [, no_hashing | partial_hashing | hashing]
            [, max_resource_granularity = N [percent]]
            [, histogram_tuning_factor = int ]
```

使用 [no_hashing | partial_hashing | hashing] 确定散列的级别：

- no_hashing – 使用早于 15.7 ESD #2 的 Adaptive Server 版本中的排序算法。
- partial_hashing – Adaptive Server 对低的唯一计数域使用散列。如果 Adaptive Server 遇到超过 65536 阈值的唯一列计数，就会使用排序进行额外扫描。如果在该列上生成的上一直方图指出唯一值的数量大于或等于 65536，也会使用排序。

  这些参数的缺省值是 update statistics hashing 的配置值。

请参见 《参考手册：命令》。

本例为 authors 表收集基于散列的统计信息：

```
update index statistics authors with hashing
```

在 update statistics 命令中显式指定基于散列的统计优先于 update statistics hashing 配置参数的值。在上例中， update statistics ... with hashing 优先于服务器级别的 update statistics hashing 参数。

在收集基于散列的统计信息时，不能使用 consumer 和 sampling 参数。在缺省设置为支持 consumer 和 sampling 参数的基于排序的统计前，Adaptive Server 会尝试使用具有 partial_hashing 的基于低域散列的统计。

基于散列的 update statistics 用于以逗号分隔的列名列表指定列集，其中，列名由括号括起：

update statistics *table_name* (*column1*), (*column2), (column3*), ...

此语法供 Adaptive Server 执行可更新所有列的统计信息的单表扫描。但是，如果采用一对括号括起多列来收集散列统计信息， Adaptive Server 将发出错误消息。

设置分配粒度

使用 histogram tuning factor 配置参数为服务器确定直方图的分配粒度（即梯度数）。

update statistics ... histogram_tuning_factor 参数确定 update statistics 对直方图的分配粒度，这样有助于隔离偏差值，改善等重范围单元的行为。update statistics 用于转换最终直方图使其符合为范围单元配置的梯度数，同时保留所有频率单元。如果最终直方图 update statistics 结果使权重出现偏差，请尝试增加调优因子创建更多等重的范围单元。

设置缓冲区管理器内存

散列会占用大量的 tempdb 缓冲区高速缓存内存。缺省情况下， update statistics 使用为 max resource granularity 配置参数设置的值，即 tempdb 缓冲区高速缓存的可用百分比。

请参见 《系统管理指南，卷 1》中的 “设置配置参数”。

通过 update statistics ... max_resource_granularity 限制使用的缓冲区内存量。如果 Adaptive Server 达到此值，就会选择一列，从中循环利用内存，以便完成对剩余列的散列。对于 Adaptive Server 从中循环利用资源的列，会在使用散列算法的后续扫描中收集其直方图。为避免额外扫描，请根据需要增加 max resource granularity 的值。

第 11 章

通过 update statistics 包含进度消息

Adaptive Server 15.7 ESD #2 版本及更高版本的 update index statistics、update statistics 和 update all statistics 包含 print_progress 参数，该参数允许这些命令显示进度消息。

使用 print_progress 参数

print_progress 的语法如下：

- update index statistics：

  ```
  update index statistics
  table_name [[partition data_partition_name] |
  . . .
  [, print_progress = int]
  ```

- update all statistics：

  ```
  update all statistics
  table_name [partition data_partition_name]
  . . .
  [, print_progress = int]
  ```

- update statistics

  ```
  update statistics
  table_name [[partition data_partition_name]
  . . .
  [, print_progress = int]
  ```

其中：

- 0 – （缺省值）禁用 print_progress，从而不显示任何进度消息
- 1 – 启用 print_progress，从而显示进度消息

本例显示对表 bigtable 运行 update statistics 时的进度消息：

```
update statistics bigtable with print_progress=1
Update Statistics STARTED.
Update Statistics index scan started on index 'bigtable_NC1'.
Update Statistics table scan started on table 'bigtable' for summary statistics.
Update Statistics FINISHED.
```

本例显示对表 bigtable 运行 update index statistics 时的进度消息：

```
update index statistics bigtable with partial_hashing, print_progress=1
Update Statistics STARTED.
Update Statistics index scan started on index 'bigtable_NC1'.
...It is using existing index scan to hash minor column 'a2' (column id = 2).
...Column 'a2' (column id = 2) is moved from hashing to sorting.
Update Statistics table scan started on table 'bigtable' for summary statistics.
Update Statistics table scan started on table 'bigtable'.
...Sorting started for column 'a2' (column id = 2).
Update Statistics FINISHED.
```

本例显示对表 bigtable 运行 update statistics ... with hashing 时的进度消息：

```
update statistics bigtable (a1), (a2), (a3) with hashing, print_progress=1
Update Statistics STARTED.
Update Statistics table scan started on table 'bigtable'.
...Column 'a3' (column id = 3) is picked as hash victim due to limited resource.
Update Statistics table scan started on table 'bigtable'.
```

第 12 章　Dump 和 Load 的增强

Adaptive Server 15.7 ESD #2 中包含 dump 和 load 命令的增强：

- dump configuration 命令用于备份 Adaptive Server 配置文件、转储历史记录文件和集群配置文件。请参见 《参考手册：命令》。
- Adaptive Server 15.7 ESD #2 中引入了用于定义数据库转储的创建选项的转储配置。 Backup Server 随后使用配置执行数据库转储。可以使用：
  - 转储配置创建、修改或列出转储配置，然后将 dump database 或 dump transaction 与转储配置配合使用来执行转储。请参见 《参考手册：命令》。有关 sp_config_dump 的信息，请参见 《参考手册：过程》。
  - enforce dump configuration 配置参数通过转储配置启用转储操作。请参见第 48 页上的 “配置参数”。
  - 表示用户创建转储配置的配置组 “转储配置”。请参见第 49 页上的 “使用 dump configuration”。
- 转储历史记录 – 在 Adaptive Server 15.7 ESD #2 中，可以：
  - 在供 Adaptive Server 稍后将数据库恢复至指定时间点的状态的转储历史记录文件中保留 dump database 和 dump transaction 命令的历史记录。请参见第 51 页上的 “转储历史记录文件”。
  - 读取转储历史记录文件，然后重新生成恢复数据库所需的 SQL 语句的装载序列。使用：

    load database with listonly=load_sql until_time = *datetime*

    有关 load database 扩展的详细信息，请参见 《参考手册：命令》。
  - 使用 sp_dump_history 系统过程清除转储历史记录。

    有关 sp_dump_history 的详细信息，请参见 《参考手册：过程》。
  - 使用 enable dump history 配置参数禁用在每次转储操作结束时对转储历史记录文件进行的缺省更新。
  - 使用 dump history filename 配置参数指定转储历史记录文件的名称。
- dump with listonly 命令在 Adaptive Server 15.7 ESD #2 中有所增强，它为用户提供两种选择。您可以：

- 使用 create_sql 选项列出创建与源数据库采用相同布局的目标数据库所需的 disk init、 sp_cacheconfig、 create database 和 alter database 命令的序列。
- 使用 load_sql 选项通过转储历史记录文件生成将数据库恢复至指定时间点的状态所需的 load database 和 load transaction 命令的列表。

有关 load database 和 load transaction 扩展的详细信息，请参见 《参考手册：命令》。

有关增强转储标头的详细信息，请参见第 52 页上的 “转储标头的增强”。

配置参数

Adaptive Server 包含以下用于转储配置的配置参数。

enforce dump configuration

摘要信息	
缺省值	0 （禁用）
值的范围	0 （禁用）， 1 （启用）
状态	动态
显示级别	Basic
要求的角色	系统管理员
配置组	备份/恢复

enforce dump configuration 确定 Adaptive Server 是否使用转储配置执行数据库转储。

如果启用此命令， Adaptive Server 仅允许通过转储配置执行转储操作。如果 dump 命令具有 blocksize 和 compression 等指定参数，这些指定值不会覆盖由转储配置定义的值。

如果禁用此命令， Adaptive Server 则使用在命令行上指定的参数值并覆盖由转储配置定义的值。

enable dump history

摘要信息	
缺省值	0 （禁用）
值的范围	0 （禁用）， 1 （启用）
状态	动态
显示级别	Basic
要求的角色	系统管理员
配置组	备份/恢复

dump history update 确定在数据库转储操作结束时是否更新转储历史记录文件。

缺省情况下， Adaptive Server 在每次数据库转储后都更新转储历史记录文件。

dump history filename

摘要信息	
缺省值	dumphist
值的范围	
状态	动态
显示级别	Basic
要求的角色	系统管理员
配置组	备份/恢复

dump history filename 指定转储历史记录文件的路径。

使用 dump configuration

dump configuration 配置参数组表示用户创建的以下转储配置：

- stripe directory – 在转储操作过程中文件的存档目录。存档文件通常按以下约定命名：

  database_name.nump_type.date-timestamp.stripeID

- external api name – 要用于转储操作的外部 API（字节流设备）的名称，此名称必须符合以下格式：

  ```
  External API Name::Options
  ```

- number of stripes – 要在转储操作过程中使用的分条设备的数量。缺省情况下，仅使用一台分条设备。
- number of retries – 服务器针对非致命错误尝试转储操作的次数，最多 5 次。缺省值为 0。
- block size – 转储设备的块大小，此配置可覆盖设备的缺省块大小。blocksize 必须至少为一个数据库页，也必须是数据库页大小的整数倍。
- compression level – 压缩转储的压缩级别。缺省情况下，禁用压缩。
- retain days – 无法覆盖转储的天数。 Backup Server 要求经过确认才能覆盖未到期的卷。缺省情况下， retaindays 的值为 0，而且可以覆盖转储。
- init – 指定是否必须重新初始化卷。缺省值为 “noinit”。
- verify – 指定将数据页复制到存档时 Backup Server 是否必须对其执行最低限度的标头检查或完整结构行检查。不对全局分配映射 (GAM) 页、对象分配映射 (OAM) 页、分配页、索引、文本或日志页进行结构检查。缺省情况下，在存档过程中不检查数据页。
- notify – Backup Server 的缺省消息目标。可以是以下模式之一：
  - client – 将消息发送到启动 dump 命令的终端。
  - operator_console – 将消息发送到运行 Backup Server 的终端
- remote backup server name – 指定用于转储操作的远程 Backup Server。缺省值为 SYB_BACKUP。

示例

示例 1 包含在 Adaptive Server 配置文件中创建的多项转储配置：

```
[dump configuration :dmp_cfg1]
   stripe_dir = /work/dmp_cfg1_dir
   ext_api = DEFAULT
   num_stripes = 5
   retry = 0
   blocksize = DEFAULT
   compression = 9
   retaindays = DEFAULT
   init = DEFAULT
   verify = DEFAULT
   backup_srv_name = DEFAULT
```

```
[dump configuration :dmp_cfg2]
   stripe_dir = /work/dmp_cfg2_dir
   ext_api = syb_tsm
   num_stripes = DEFAULT
   retry = 3
   blocksize = DEFAULT
   compression = DEFAULT
   retaindays = DEFAULT
   init = DEFAULT
   verify = DEFAULT
   backup_srv_name = SYB_REMOTE
```

转储历史记录文件

Adaptive Server 在转储历史记录文件中维护通过 dump database 和 dump transaction 命令执行的成功和失败备份的历史记录。Adaptive Server 读取转储历史记录文件以恢复数据库，然后生成将数据库恢复至特定时间点的状态所需的 load database 和 load transaction 序列。

每个 Adaptive Server 实例都具有转储历史记录文件，此文件包含所有数据库转储和服务器配置转储的相关信息，无论是否成功。此文件的缺省位置为 -m 启动参数指定的位置，或者 $SYBASE 目录 （如果 -m 没有指定位置）。

使用以下语法备份转储历史记录文件，其中， *file_name* 表示转储历史记录文件的名称：

dump configuration with file = *dump_hist*

转储历史记录文件的缺省名为 *dumphist*。

转储历史记录文件中的每一行表示一项转储记录。将数据库转储到多个分条设备会导致每个分条设备中都存在转储记录。转储记录字段以制表符分隔。

转储记录包含以下项目的相关信息：

- 记录类型
- 数据库 ID
- 数据库名称
- 转储类型
- 转储操作的分条总数

- 远程 Backup Server 名称
- 当前转储序列号 （当前转储的时间戳）
- 之前转储序列号 （之前转储的时间戳）
- 转储创建时间
- 分条名称
- 转储服务器名称
- Adaptive Server 错误数量
- 口令保护信息 （表示备份是否受到口令保护的布尔值）
- 压缩级别
- 最大逻辑页码 （已转储数据库中的最大逻辑页码）
- Status

转储历史记录文件由 Adaptive Server 读取和写入。启动 Adaptive Server 的用户需要拥有针对转储历史记录文件的相应读取和写入权限。

转储标头的增强

Adaptive Server 15.7 ESD #2 版本及更高版本在转储标头中存储了数据库设备的相关信息。此信息以及段映射用于在转储图像的创建过程中生成 disk init、 sp_cacheconfig、 disk init 以及目标数据库的 create database 和 alter database 命令的序列。

转储标头块中各数据库设备的相关信息包括：

- 设备号
- 逻辑名
- 实际设备大小
- 物理路径
- 设备类型
- 数据库设备大小
- 设备选项

内存数据库设备利用高速缓存进行配置，在生成用于创建图像的 disk init 和 sp_cacheconfig 命令时，这些高速缓存的相关信息为必需信息。因此，高速缓存特定信息均存储在转储标头中，而且包括：

- 高速缓存 ID
- 高速缓存名
- 数据库高速缓存大小
- 实际高速缓存大小
- 设备类型
- 高速缓存选项

第 13 章　在不复制数据的情况下从表中删除列

alter table drop column 的 no datacopy 参数用于在不复制数据的情况下从表中删除列，并缩短 alter table drop column 所需的运行时间。

语法为：

```
alter table [[database.][owner].table_name
    {add column_name datatype}
. . . }

    modify column_name
    drop {column_name [, column_name]...
        with exp_row_size=num_bytes
            | transfer table [on | off]}
            | no datacopy
```

no datacopy 不会立即从表中删除列，反而更新系统表，这表示在下次运行 reog rebuild 或另一项数据复制操作时将重新设置受影响行的格式（在下次运行 reorg rebuild 前，不会清空为删除列 （包括大对象）分配的空间）。

本例可在不复制数据的情况下从 titles 表中删除 total_sales 列：

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

限制

不能将 no datacopy 参数用于：

- 已实现或虚拟的计算列
- 加密列
- XML 列
- IDENTITY 列
- Java 列

- 代理表
- 使用以下数据类型的列：
  - 时间戳
  - bit
- 如果表具有以下特点，则不能更改它的锁定方案：
  - 受到 no datacopy 操作的影响
  - 自上次执行 drop column with no datacopy 后尚未对其执行 reorg rebuild 或数据复制操作

  在更改表的锁定方案前，必须运行 reorg rebuild。

第 14 章　扩展了最大数据库大小

Adaptive Server 15.7 ESD #2 及更高版本将逻辑页码从带符号的整数值转变为不带符号的整数值，从而将数据库的最大大小扩展到约 64 太字节。

低于 Adaptive Server 15.7 ESD #2 的版本允许最大数据库的大小约为 32 太字节。

数据库的最大大小取决于其逻辑页的大小：

- 2K 页服务器 = 8TB
- 4K 页服务器 = 16TB
- 8K 页服务器 = 32TB
- 16K 页服务器 = 64TB

注释由于 Adaptive Server 预留 256 个无法分配或使用逻辑页 ID （它们均处于逻辑页范围的上限），因此以上列出的大小会略高于实际可用空间量。这种开销使实际可用空间量降低了每个列出的页大小的逻辑页大小的 256 倍（例如，2K 服务器的实际可用大小为 8TB – (256 x 2K)）。

要扩展数据库的最大大小，需要将以下列的数据类型从 int 更改为 unsigned int：

- sysusages – lstart、 size 和 unreservedpgs
- sysaltusages – lstart 和 size
- syspartitions – firstpage、 rootpage、 dataoampage 和 indoampage
- systabstats – leafcnt、 pagecnt、 emptypgcnt、 warmcachepgcnt、 unusedcnt 和 oampgct
- syslocks – page
- syslogshold – page
- systhresholds – free_space

注释必须将依靠上述列的所有查询的预期结果从 int 更改为 unsigned int。

之后，这些函数将返回 unsigned int 结果而不是 int：

- curunreservedpgs
- used_pages
- data_pages
- reserved_pages
- lct_admin

第 15 章　用户定义的优化目标

Adaptive Server 15.7 ESD #2 及更高版本允许用户创建用户定义的优化目标 （所有活动优化程序条件的集合），这使得用户可以执行以下操作：

- 创建新的优化目标
- 定义一组包含在目标中的活动条件
- 激活服务器、会话、过程和查询级别的目标
- 动态更改目标内容，而不断开客户端会话并重新连接

创建用户定义的优化目标后，可以在服务器级别或为用户会话对其进行调用。

创建用户定义的优化目标

使用 sp_optgoal 可创建用户定义的优化目标。语法为：

```
sp_optgoal "goal_name", "save"
```

其中：

- *goal_name* – 要创建的目标的名称，长度不得超过 12 个字符。
- save – 如果目标尚不存在，则创建目标

请参见 《参考手册：过程》。

示例

以下示例将创建名为 goal_1571 的目标，即：

1　将优化级别设置为 ase157ga

2　将优化目标设置为 allrows_mix

3　启用散列连接

4　启用 CR # 123456 的优化条件

5　禁用 CR # 234234 的优化条件：

```
set plan optlevel ase157ga
set plan optgoal allrows_mix
```

```
set hash_join 1
set CR123456 1
set CR234234 0
go
execute sp_optgoal "goal_1571", "save"
go
```

为整个服务器范围和会话设置目标

使用 sp_configure 'optimization goal' 参数可设置适用于整个服务器范围的目标。语法为：

sp_configure 'optimization goal',1,'*goal_name*'

例如，如果要为服务器设置 goal_1571，应输入：

```
sp_configure 'goal',1,'goal_1571'
```

使用 set 可为当前会话或整个服务器范围设置目标。语法为：

set plan optgoal *goal_name*

例如，如果要为当前会话设置 goal_1571，应输入：

```
set plan optgoal goal_1571
```

以下示例在抽象计划的查询级别使用 *goal_name*：

```
select count(*) from tab1,tab2
PLAN '(use optgoal goal_1571)'
go
```

目标报告

sp_optgoal 'show','goal_name' 可报告由名为 *goal_name* 的目标激活的所有单个条件。例如：

```
sp_optgoal 'goal_1571', 'show'
```

sp_optgoal @@optgoal, 'show' 可报告当前目标设置：

```
sp_optgoal @@optgoal, 'show'
name
-------------------------------------------------------------------------------
```

```
distinct_sorted
distinct_sorting
distinct_hashing
group_sorted
group_hashing
nl_join
merge_join
append_union_all
merge_union_all
merge_union_distinct
hash_union_distinct
opportunistic_distinct_view
parallel_query
order_sorting
store_index
replicated_partition
ndex_union
streaming_sort
nary_nl_join
alternative_greedy_search
cr562947:OPTLEVEL EXCEPTION SEE CR - allow cursor table scans
data_page_prefetch_costing:clustered row bias added
mru_buffer_costing:wash size buffer limit for MRU
cr546125:implicitly updatable cursor non-unique index scan
cr545771:improves multi-table outer-join and semi-join costing
cr545653:avoid inner table buffer estimate starvation
cr545585:covered iscan CPU costing too expensive
cr545379:disallow reformatting on user forced index scan
cr545180:avoid reformat with no sargs if useful index exists
cr545059:reduce usage of buffer manager optimization sorts
cr544485:mark subquery join predicates with distinct view as sargs
cr534175:compute GROUP BY worktables in nested subqueries only once when possi
ble
cr531199:increases the number of useful nested loop join plans considered
cr500736:supports nocase sortorder columns in mergejoin and hashjoin keys
cr487450:improves DISTINCT costing of multi-table outer joins and/or semi-joins
cr467566:allow abstract plans and statement cache to work together
cr497066:infer the nullability of isnull() by looking at its parameters
cr421607:support NULL=NULL merge and hash join keys
cr552795:eliminate duplicate rows during reformatting when they're not needed
imdb_costing:0 PIO costing for scans for in-memory database
allow_minmax:allow local session to consider MINMAX optimization
cr646220:enable better store index key generation with correlated predicate
```

第 16 章　共享查询计划

Adaptive Server 15.7 ESD #2 及更高版本允许用户共享查询计划，因而不需要 Adaptive Server 创建或重新编译与现有计划相同的查询计划。共享查询计划从并发系统下的主查询计划复制而来。

通过 sp_configure 可以使 Adaptive Server 使用共享查询计划。语法为：

```
sp_configure 'enable plan sharing', 1
```

enable plan sharing 是 enable functionality group 的一部分。有关设置此组构成参数的信息，请参见 《系统管理指南，卷 1》。

当 Adaptive Server 共享而不是重新使用或重新编译查询计划时，您会发现性能有所改进。当将主查询计划固定到高速缓存中且当 Adaptive Server 使用其共享查询计划时，您会发现过程高速缓存内存的使用情况出现了轻微的变化。

要使查询计划成为可共享状态：

- 它必须是轻量过程 (LWP)，或者：
  - 用于动态 SQL （通常用于 ODBC/JDBC 预准备语句），或：
  - 因语句高速缓存而产生，此时限定语句会转换为 LWP 以供重复使用
- 它是 lava 查询计划
- 不得包含 instead-of 触发器
- 只能包含以下内容：
  - declare、 insert、 delete、 update、 merge 或 select 语句
  - 单个语句，在第一个语句为 declare 语句的情况下，查询计划可以包含两个语句
  - 串行查询计划
  - 不能访问引用 Java 对象的计划

show_cached_plan_in_xml 函数在 <planSharing> 元素中描述计划共享。

- shareable – 计划可以共享。
- notShareable – 计划不可共享。
- primary – 主计划 （共享其副本）。
- shared – 共享的计划 （主计划的副本）。

第 17 章　异步初始化数据库

alter database 和 create database 命令的 async_init 参数用于在使用数据库时对其异步初始化。即：在创建或修改数据库后，数据库立即可用，不必等到数据库初始化完成时。初始化过程对用户透明。

任何使用尚未初始化的数据库页的任务都将对页所在的分配单元执行初始化操作。

异步初始化由通过 create database 或 alter database 命令启动的服务任务执行。当其重新启动时， Adaptive Server 将自动启动新的服务任务，该任务用于完成初始化操作。在集群环境中，如果运行服务任务的实例失败或关闭，协调实例将启动新的服务任务来完成初始化。

将 Adaptive Server 配置为异步创建或修改数据库

enable async database init 配置参数确定 Adaptive Server 是否异步创建或修改数据库。

enable async database init

摘要信息	
缺省值	0 （关闭）
有效值	0 （关闭）， 1 （打开）
状态	动态
显示级别	Comprehensive
要求的角色	系统管理员
配置组	SQL Server 管理

enable async database init 确保所有 create database 和 alter database 命令在缺省情况下都异步初始化数据库。

异步创建或修改数据库

以下语法用于异步创建数据库：

```
create [temporary] database database_name
    [on {default | database_device} [= size]
. . .
    [with {override
    | default_location = "pathname" [,[no]async_init] }
    [for {load | proxy_update}]
```

noasync_init 表示已经同步初始化数据库。

以下语法用于异步修改数据库：

```
alter database database_name
    [on {default | database_device } [= size]
. . .
    [with override [,[no]async_init]]
    [for load]
    [for proxy_update]
```

noasync_init 表示正在扩展数据库且 Adaptive Server 同步初始化扩展空间。

对 create 或 alter database 使用 [no]async_init 参数可覆盖 enable async database init 的设置。

确定是否存在需要初始化的空间

Adaptive Server 在使数据库可用前同步部分数据和日志段，使初始化程序在需要使用数据库空间的任何命令前进行初始化。但是，您有时可能注意到，在 Adaptive Server 忙于进行空间初始化时，针对数据库正常运行的命令的性能会受到影响。因为对于所需空间尚未初始化的命令，必须完成空间初始化才能继续运行。

已初始化空间的相关信息均存储在 sysattributes 中。

要确定数据库中是否存在尚未初始化的空间 （例如，如果初始化程序提前终止，导致剩余数据库未完成初始化），请发出类似下面语句的查询语句：

```
select lstart=object_info1, size=object_info2, segmap=object_info3
from master..sysattributes where class=42 and object=db_id("mydb")
lstart           size             segmap
 -----------     -----------  -----------
        1536         3584000            3
        5120           51200            4
```

如果此次查询返回一行或多行，则数据库中存在尚未初始化的空间（在此次查询中，为 mydb 数据库）。此次查询并不表明异步初始化服务任务是否正在运行，仅表明该任务尚未完成（如果已经完成，结果集应不含行）。

使用类似于下面语句的查询语句确定初始化程序是否正在针对特定数据库运行（在此次查询中，为 test 数据库）：

```
select spid from sysprocesses
where dbid=db_id("test") and cmd="CRDB AUINIT"
 spid
------
    22
```

如果异步初始化服务任务正在运行， Adaptive Server 将以下消息写入错误日志：

```
Asynchronous initialization of database 'database_name'
has completed.
```

如果异步初始化服务任务提前停止， Adaptive Server 将以下消息写入错误日志：

```
Asynchronous database initialization terminated
prematurely for database '%.*s'.Use DBCC
DBREPAIR(%.*s, async_database_init, start) to restart
it if required as uninitialized pages will incur a small
performance penalty when they are first referenced.
```

限制

- 即使显式使用 async_init 参数，仍然无法异步初始化以下数据库：
  - 所有系统数据库
  - 所有临时数据库，包括系统数据库和用户数据库
  - 存档数据库
  - 代理数据库
  - 通过 for load 选项创建的任何数据库
- 不得在仍处于初始化过程的数据库中运行以下命令：
  - unmount database
  - alter database ... log off

- 在初始化过程中，可以将数据库置于单用户模式。但是，当数据库处于单用户模式时，初始化程序不会运行，在数据库退出单用户模式后，该程序会自动重启并继续初始化操作。

注释您可能注意到，在异步初始化服务任务运行时，使用正在初始化的数据库中空间的 DML 的性能会受到轻微影响。

第 18 章　行内大对象压缩

Adaptive Server 15.7 ESD #2 版本及更高版本支持行内大对象 (LOB) 压缩。请参见 《压缩用户指南》。

Adaptive Server 在以下情况中使用行内 LOB 压缩：

- 表经过隐式或显式行压缩或页压缩，而且
- 表中的任何行内大对象列都经过隐式或显式 LOB 压缩。

第 19 章

配置共享内存转储

使用 memory dump compression level 配置参数将 Adaptive Server 配置为压缩共享内存转储文件。

将 Adaptive Server 配置为使用压缩共享内存转储

启用 memory dump compression level 可明显减小由 Adaptive Server 生成的共享内存转储文件的大小。

memory dump compression level

摘要信息	
缺省值	0
有效值	1 – 9
状态	动态
显示级别	Comprehensive
要求的角色	系统管理员
配置组	诊断

memory dump compression level 控制共享内存转储的压缩级别。压缩级别的范围为 0（不压缩）到 9（最高级压缩）。压缩速度与转储的压缩量成反比。压缩级别越低， Adaptive Server 对转储的压缩速度就越快，但压缩文件的大小也会越大。

配置共享内存转储

使用 sp_shmdumpconfig 配置共享内存转储。语法为：

sp_shmdumpconfig "*action*", *type*, *value*, *max_dumps*, *dump_dir*, *dump_file*, *option1*, *option2*, *option3*, *option4*, *option5*

action 参数确定 Adaptive Server 处理转储的方式。请参见 《参考手册：命令》。

注释创建共享内存转储文件的目的是协助 Sybase 客户支持部门分析 Adaptive Server 出现的问题。请仅在 Sybase 客户支持部门的指导下使用 sp_shmdumpconfig。

本例发出没有参数的 sp_shmdumpconfig 以便显示共享内存转储的当前配置：

```
sp_shmdumpconfig
Configured Shared Memory Dump Conditions
----------------------------------------

   Defaults   ---
     Maximum Dumps:          1
     Halt Engines:           Halt
     Cluster:                Local
     Page Cache:             Omit
     Procedure Cache:        Include
     Unused Space:           Omit
     Dump Directory:         $SYBASE
     Dump File Name:         Generated File Name
     Estimated File Size:    100 MB

Current number of conditions:0
Maximum number of conditions:10

Configurable Shared Memory Dump Configuration Settings
------------------------------------------------------
Dump on conditions:1
Number of dump threads:1
Include errorlog in dump file:1
Merge parallel files after dump:1

Server Memory Allocation
Procedure Cache  Data Caches  Server  Memory  Total Memory
--------------  -----------  -------------  ------------
          16 MB         9 MB           85 MB        108 MB
```

本例将 Adaptive Server 配置为在遇到信号 11 （即分段错误）时执行共享内存转储：

```
sp_shmdumpconfig "add", signal, 11,1,"dump_dir"
```